

<Name-of-Software-Application>

# **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 | 11/14/2021 | Kathryn McNeil | A summary how Draw It or Lose It works is given. A solution to their problem of expanding onto the web as well as multiple platforms is also explained. The UML diagram is explained, going into detail of how the game will be programmed and work in more technical terms. There is a chart comparing the pros and cons of various operating systems. This explains the benefits and drawbacks each one has. After that I suggest which OS would be best for the company and why. I explain the basic architecture of the OS as well as its storage and memory management. I also explain how it will work with other networks, as well as its security. |

**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 would like to take their Android only app, Draw It or Lose It, and create a web-based game that can serve on multiple platforms. The game should support one or more teams playing at the same time, as well as multiple players per team. The game must track names, have a large library of stock drawings, and a timer. The library will be used to create a complete drawing after 30 seconds. During this time, whoever’s turn it is guesses what the drawing is. A timer is necessary to make sure the drawing is complete after 30 seconds. If the team does not guess after 30 seconds another timer for 15 seconds will run to allow other teams to guess.

To create a web-based game that serves multiple platforms a progressive web app (PWA) should be created. This can be accessed from one’s web browser, as well as through the Google Play Store and Microsoft Store. Accessing it through the store allows the PWA to be downloaded so it can perform like a mobile app. Because the app is for Android it is most likely made with Java, Kotlin, or a C++ language. All of these languages can be used to create web apps making the switch from Android OS to a web browser fairly easy.

[Design Constraints](#_2et92p0)

One design constraint is that The Gaming Room would like their web-based game to run on multiple platforms. As stated above, creating a progressive web app will solve this issue. By creating a web app only one code base needs to be created that can then be accessed through a web browser, and as of recent through the Google Play and Microsoft store as well. The app can be accessed through multiple platforms, this means that support will be needed for the different platforms. This can cause problems because if there is an issue it may be difficult to determine whether the problem lays with the device or the web app itself. The web app may be more difficult to track usage patterns for, potentially impacting marketing. The app may also be more difficult to advertise especially in apple platforms because the Apple store does not support web apps. This means the company will need to put consideration into the marketing of the web app while it is being designed.

Progressive web apps are not true apps, this means they are limited in both functionality and speed. Draw It or Lose It will not be able to perform the same way its native Android app counterpart does. For example, if the company wanted to allow people to upload profile pictures from their phone it may be harder to do so if the web app cannot access the phones camera. Another example is that the web app relies on internet, whereas the Android app is downloaded. The web aps speed is affected by internet speed, whereas the Android app may not. This may prove frustrating, especially if someone is attempting to play using cellular data rather than Wi-Fi. Because the web app will not perform the same as the Android app in speed and functionality this is a design constraint to be considered. During planning it is also important to consider how the web app will be different from the original Android app.

Another design constraint is security. Web apps can be more susceptible to security threats, and the language it is written in can also make it more prone to viruses and hacking. For example, C++ is a complex language that when not written properly can have both memory and security issues. If the Android app is written in this, or it is being considered for the web app it may be worth it to look into other languages. The language the web app is written in is very important as it will have a direct impact on user experience and interaction with the app. Determining what language will be used can affect how the end product works and feels, making this a design constraint. If the web app is prone to viruses and bugs this will make the web app frustrating to use and drive people away.

## [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)

Looking at the UML diagram we can see we have a super class called “Entity”. There are three classes that inherit from “Entity”- “Game”, “Team”, and “Player”. We can see that these three as well as “GameService” are all associated. Each line connecting “GameService”, “Game”, “Team”, and “Player” has a “0…\*”. This means there can be no instances of these, or there can be a large amount of them. This allows Draw It or Lose It to have 0 games going, or many of them. It also means that there can be 0 to many teams and 0 to many players. We also have a “ProgramDriver” class that uses and sends messages to the “SingletonTester” class. We can see that these two classes are not connected to any of the other classes, therefore their information is separate from the rest of the classes and not inherited by any of them. There can only be one instance of the singleton class at a time, this means the game must create unique games. This prevents data from one game getting mixed up with another. The diagram shows that the game can take a name and id, and with that create a game with teams and players.

**"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 a up and coming operating system for web hosting. Because it is only more recently becoming popular for this purpose it has limited options for web hosting. It is east to make changes with, as well as safe which limits security issues. On the downside it does cost money to use, one has to pay an OS X license cost to use it because it is not open source. It is also limited in the hardware that can be used, this operating system is limited to Apple products only. | Linux is an open-source OS, this means it is mostly free with no license costs. It is also a very safe OS because many people are constantly looking at the code. Operating systems that are not open source cannot have the same amount of eyes on the code causing some bugs to slip through. It is very flexible in hardware; one does not need specific brand of computer to use it. Linux is prone to having less frequent updates, however, as well as sometimes slow migration periods. | Windows is a very commonly used OS, as well as fairly intuitive. There is a license cost making it more expensive than Linux, however, it is the most popular on the market right now. It is also not compatible with Linux, another popular OS. Windows is known for having security issues which is a drawback. It also has minimum CPU and RAM requirements that also adds to cost. | Programming for mobile devices is not as versatile. They are typically Android or Apple. Creating a web app does not require one to write the program for an Android or an Apple as it can be opened on either platform. The format one wants for the web app in is a consideration. A web app designed for a desktop may not perform optimally in a mobile device and vice versa. Mobile devices are very common so they may help the company reach a wider audience. Probably not best to only cater to mobile devices as a web app cannot be downloaded in an Apple phone. |
| **Client Side** | Because OS X is not open source it costs more money than other operating systems to use. Because of this, it also requires individuals who are a bit more specialized in using this system. Because it is not open source it is harder to study and learn to use, this also means it costs a bit more money finding and hiring individuals who do specialize in this OS. They also need to know the Apple language, Swift. Programmers will need to use Apple products; this may be a cost for the company if they need to provide Apple products. Because this OS is specially tailored to Apple products it does not translate to other platforms that easily. This is yet another cost. | Because it is open-source Linux is a very cost-effective OS. Due to there being no license costs for use it is also cheaper than both Windows and Mac. There is also a lot of technical support to be found in forums making it slightly easier to use. There is a learning curve, however, so people with slightly more experience may be necessary. As stated above, there is no specific hardware that needs to be used so the company can choose whatever their preferred computers are. | Windows is not the cheapest option to use. It not only has a license cost, but it also has minimum requirements for what type of computer can be used. It is not compatible with Linux and extra steps may need to be taken to make it compatible with Apple which is a draw back. Due to its poor security extra time needs to be put in planning to make the web app more secure. It is very common and intuitive, however, so it is easy to find people proficient in this OS. This also makes it rather quick to build on. | Planning needs to go into developing for mobile devices. A tablet has different formatting than a phone and certain design aspects may not translate well. It also may be difficult to make something compatible for all the different mobile devices. It may be expensive because if developing using Apple or Windows these both have license costs. May be time consuming planning something that will be compatible for all mobile devices. |
| **Development Tools** | Apple is a C based language, this includes C, C++, and Apples own language Swift. The IDE XCode supports swift, and an extension can also make it supported in Visual Studio Code. This IDE also supports C languages. A Mac computer is necessary for using and building on this OS | Linux is C based, but it also uses PHP, Perl, Ruby, and Python for web hosting. There are many IDEs available such as Atom, PyCharm, RubyMine, etc. Any preferred computer can be used. | Windows uses C, JavaScript, and Visual Basics. Microsoft Visual Studio is the most common IDE for this OS. The computers that can be used with this OS do have CPU and RAM requirements which may need to be purchased by the company. | The languages used to make mobile apps are Swift and Java. HTML, JavaScript, and CSS can be used to make a web app which will still open on a mobile device. There are many IDEs available including, but not limited to, Visual Studio Cide, Android Web Developer, XCode, etc. |

## 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**: Linux is a good choice for The Gamin Room to expand Draw It or Lose It to other computing environments. It is very common and can be used by any platform including but not limited to Windows, Mac, and Android. It is open source making it a very economical choice, and it is also secure limiting security threats and hacking problems. Linux allows software that is not created and maintained by just one company, unlike Mac and Windows. For a company just starting to expand into the Web market like ours Linux is the more economical choice. Because it is secure that also means less resources need to be used to keep information secure.
2. **Operating Systems Architectures**: Linux has a modular kernel created with object-oriented programming. The kernel is layered above hardware, so it calls upon a computers terminals and disks to work. This kernel acts for users by retrieving system hardware. It also controls memory, and the scheduling of processes. The kernel is kept efficient by storing only necessary memory and adding functionality only when it is required. To maintain efficiency the kernel dynamically loads modules. This means modules are loaded when their functionality is needed by the system and removed only when specifically told to do so. This prevents rapid loading and unloading of modules which could slow down the system. Outside of the kernel is the shell, Linux’s command line interpreter. This serves to give commands to the kernel from the user as well as execute commands. This can also execute programs such as applications which are the most outer facing layer of the Linux operating system. This layer is where application programs reside.
3. **Storage Management**: Linux has an interface called System Storage Manager. This tool simplifies complicated systems by handling specific storage technology methods such as create and remove volumes and pools. Technologies such as btrfs can be used by the interface to make managing storage simpler. Not only can btrfs manage large amount of storage in subsystems, but it can also detect and repair errors that are stored in the data on disk.
4. **Memory Management**: By using memory management techniques, the large library of photos Draw It or Lose It will need can be managed more efficiently. Memory management allows the photos to be called upon in the correct order with minimum delays. By using a System Storage Manager, files can be created and mounted with only one command, whereas without it, it would take five. This simplification means less interaction between commands and therefore more efficient management.
5. **Distributed Systems and Networks**: Creating a web app will allow Draw It or Lose It to be used on various platforms. It also means that the program can communicate between various platforms through internet. Because web apps rely on internet that means for friends to play virtually, they need to be connected to either Wi-Fi or cellular data. If someone has an internet outage this also means they will not be able to access the game. Because the web app will mostly reside in web servers (unless downloaded) players will not even be able to load the game without internet. If it is downloaded, they may be able to open the app, but they will not be able to play with other people. This also means connectivity is important to the web app. Care needs to be taken in development to create an efficient web app. If it runs slow anyways than someone with poor connectivity may find the web app unusable. This may affect how the game performs in a few ways. Perhaps as the timer is going someone tries to enter a guess, maybe the app will not be able to process the guess but still keep the timer going. This may cause someone to incorrectly lose a round. Another problem that may occur with poor connectivity is the display of images. It may freeze on one image for a long-time causing user frustration.
6. **Security**: Linux is one of the more secure operating systems. This means threats to users’ information will be limited. Tools such as brtfs scan for errors and can even repair some that are stored on the disk. This means it is always checking for bugs. Because Linux is open source this also means bugs that could compromise security are found relatively quickly. Security-Enhanced Linux is a kernel that also boosts the security of Linux. Careful consideration should also be put into the programming of the web app to minimize security threats. Poorly written code is more susceptible to threats, hiring professionals who are competent can help prevent this.

Bibliography:

Brewster, C. (n.d.). *Web app development in 2021: Everything you need to know*. Trio Developers. Retrieved November 13, 2021, from https://trio.dev/blog/web-app-development.

Digital.com Staff. (2021, October 25). *5 Best Mac hosts for die-hard apple fans in 2020*. Digital.com. Retrieved November 13, 2021, from https://digital.com/best-web-hosting/mac/.

Hauck, T. (2020, October 23). *Configuring Key Linux Components*. DistributedNetworks(Operating Systems and Security). Retrieved November 13, 2021, from https://www.distributednetworks.com/configure-linux-components/module2/modular-monolithic-kernels.php.

H, T. (2017). *Pros and cons of mobile websites and mobile apps*. RubyGarage. Retrieved November 13, 2021, from https://rubygarage.org/blog/mobile-app-vs-mobile-website.

Hussey, E. (2019, May 21). *Web apps vs. native apps: Advantages & disadvantages*. jācapps. Retrieved November 13, 2021, from https://jacapps.com/web-apps-vs-native-apps-advantages-disadvantages/.

Lewis, D. (2021, June 20). *MacOS review: The pros and cons*. Konsyse. Retrieved November 13, 2021, from https://www.konsyse.com/articles/macos-review-the-pros-and-cons/.

Little, C. (2019, November 22). *7 challenges in web application design and development: Tiller Digital*. Tiller Digital. Retrieved November 13, 2021, from https://tillerdigital.com/blog/7-challenges-in-web-application-development/.

MacPherson, L. (2019, November 12). *The Pros and cons of building a mobile app vs. a web app*. Designli Blog. Retrieved November 13, 2021, from https://designli.co/blog/the-pros-and-cons-of-building-a-mobile-app-vs-a-web-app/.

MSEdgeTeam. (n.d.). *Overview of progressive web apps (pwas) -* . Microsoft Edge Development | Microsoft Docs. Retrieved November 13, 2021, from https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/.

NA. (n.d.). *btrfs*. btrfs Wiki. Retrieved November 13, 2021, from https://btrfs.wiki.kernel.org/index.php/Main\_Page.

NA. (n.d.). *Chapter 16. system storage manager (SSM)* . Red Hat . Retrieved November 13, 2021, from https://access.redhat.com/documentation/en-us/red\_hat\_enterprise\_linux/7/html/storage\_administration\_guide/ch-ssm.

NA. (2017, December 6). *Pros and cons of linux web hosting - right for you? |*. TBWHS. Retrieved November 13, 2021, from https://blog.tbwhs.com/pros-and-cons-of-linux-web-hosting/.

NA. (2019, June 20). *Pros & Cons of Windows Hosing servers |*. TBWHS. Retrieved November 13, 2021, from https://blog.tbwhs.com/pros-cons-of-windows-hosting-servers/.

Spdload. (2021, October 22). *How to develop a web application? [types, steps, challenges]*. SpdLoad. Retrieved November 13, 2021, from https://spdload.com/blog/how-to-develop-a-web-application/.

Walburg, M. (2020, December 2). *Mobile app VS Mobile website: Comparison Pros and Cons*. BinarApps. Retrieved November 13, 2021, from https://binarapps.com/mobile-app-vs-mobile-website-comparison-pros-and-cons/.

Williams, M. (2021, July 9). *Linux vs. windows: How to pick the best server OS for your website*. PCMAG. Retrieved November 13, 2021, from https://www.pcmag.com/how-to/linux-vs-windows-how-to-pick-the-best-server-os-for-your-website.