

Creative Technology Solutions

# **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)3-4

[**System Architecture View**](#_ilbxbyevv6b6)4

[**Domain Model**](#_8h2ehzxfam4o)4-5

[**Evaluation**](#_2o15spng8stw)5-11

[**Recommendations**](#_m8aleynsvzvc)12-14

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 10/18/2020 | Jon Frodin | This is the third version of this summary. The revisions make recommendations and extrapolate on the various strengths and weaknesses of operating platforms. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is looking to develop a web-based version of a gaming application. This application must permit multiple teams each with multiple players assigned to them. The team names and player names also must be unique to avoid confusion or errors. Also, only one instance of the game can exist in memory at any given time. The game should work across platforms (Windows, Linux, Mac, etc.) and allow teams to have players from various computer backgrounds (Team1, for example, could have a player using a Windows PC and a player using a Mac). Since it is a web-based gaming app, it is unclear whether it needs to work in browsers or simply connect to web servers for hosting the game. This Design Template will cover an application rather than something working in browsers. The game should also work on mobile devices to attract the broadest user base possible. The Gaming Room wants to modify the structure of the existing game to be web-based and multi-platform. The game entails watching images render at a steady rate which are fully complete at 30 seconds. The user is guessing the phrase, title, or thing the application is rendering from a large library of stock drawings.

## [Design Constraints](#_2et92p0)

The application architecture is going to be changing. The standalone Android application will be moving from an app hosted on an individual Android phone where it is downloaded and installed to running primarily via a web server and pushed to clients via a browser. This alters the entire makeup of the game and changes the core of how the game operates.

A second design constraint is storage and memory. Since the user will not be drawing anything anymore, and the application will rely on rendering images from a library, that library needs to be hosted and controlled. The Gaming Room will need an extensive library of images in order to make the game fun long-term, so the images are likely to take up a good deal of space when they are stored and they are also likely to take up a good amount of bandwidth when people are playing the game.

A third design constraint is security. The Gaming Room will need to secure this application. They not only need to watch out for the personal information of the users, but also the integrity of the image library. They wouldn’t want, for example, someone gaining access to their server image library and uploading thousands of copyrighted images that they have not licensed because then they could be sued for each individual violation. They also wouldn’t want violent or graphic images being uploaded to their game, scandalizing the company and the users. They will need well-defined roles with authentication procedures for any portion of the program that edits or stores anything in the databases.

A fourth design constraint is to have multiple players and for those players to all have different names and ID’s. This is very similar to the previous two constraints because they operate almost the same. They are unique, though, in how they interact. This will be accomplished by running a check every time a player is created against the current list of players and failing to add any duplicates. Making all of these members of a class can simplify the process.

The above constraints make development a bit more complicated, but they also make the games more secure and give The Gaming Room the ability to keep their information clean and their game user-friendly. The design constraints make it so that we have to build multiple lists to hold the data and different mutators for that data. This also makes the program more efficient overall, though. This also means that we have to focus on an object-oriented model to achieve the needed inheritance and other features.

A fifth design constraint is for this application to work across platforms. That may mean that several models of it have to be built. If it is designed for Mac and iOS, then Swift or Xcode is needed. If it is designed for Linux or Android, then Java is needed. If it is designed for PC, then .NET or C# may be necessary. However, since this is going to be a web-based game, we need to focus on what browsers are the most popular and therefore in need of the most support. This would expand the timeline for development because the major methods would have to be translated several times to work on other platforms.

A sixth constraint is that the teams need to be able to work together in real time. This means that the server environment has to be quick and it has to be able to accommodate as many users as may want to use the game at a given moment. Auto-scaling may be needed as well as something like Docker to manage the allocation of resources to not overload the server.

## [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 UML class diagram below shows how the classes are going to interact together. As we can see, the Program Driver class uses the Singleton Tester class and the Program Driver class houses the main() method which is what actually runs the game. We can see that the Program Driver class has a usage relationship with the Singleton Tester class.

Then we have the relationships of the other classes which relate to each other in different ways. The Entity class is the base class for all of the items. This houses the private elements of “id” and “name” which, as outlined above, need to be unique to the type of entity (game, player, or name). This class also has the get methods for name and id as well as the default method for displaying an element (toString()).

Connected to the Entity by an inheritance relationship (meaning each of these inherit from Entity), are Game, Team, and Player classes. This means that every Game, Team, or Player element must have a name and an id because that is the minimum that makes up an Entity. Unique elements are within each as needed, and they have internal relationships which will be gone over shortly.

A class that is related to Game, but not a child of Entity, also exists. This is called the Game Service class. This is our Singleton, so only one instance of this can exist at any given time. This holds the getters used by the other classes (except for the id and name), and supplies the id numbers to the games, players, and teams. This Game Service class has a 0-to-many relationship with the Game class. This means that it can hold many instances of a Game without needing multiple instances of a Game Service.

Then, also via a 0-to-many relationship, we have the connection between Game and Team classes. Each game can have multiple teams and they will be stored in a List in the Game class. The exact same relationship exists between Team and Player if you switch out the names. In practice, this means that the Game class holds the list of teams (which makes sense because each game can have different teams), and each Team class holds the list of players. None of these inherit from the others, but they do have relationships to each other.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is an operating system that requires licensing (not open-source). It is based on UNIX OS. Mac charges licensing fees for each machine running Mac OS X. This means that every device that requires the OS to support the game as a web server will require a separate license and specific hardware. Mac computers can be made into Mac servers, but the cost is high.[[1]](#footnote-1) Mac OS controls access to files using group-based permissions which can be set by an administrator. This can be done via the Finder or via the Terminal.[[2]](#footnote-2) Mac does support LDAP for Directory controls. This standard is seen as fast and efficient.[[3]](#footnote-3) A big drawback of Mac is that it does not support Docker and would, instead, use virtual machines.[[4]](#footnote-4) Apple doesn’t really make servers, or cloud infrastructure. They do use AWS and Google services to implement iCloud, though, and do have Cloud services.[[5]](#footnote-5) Mac is extremely easy to integrate and that is mostly due to the fact that there are tools built in to the server application. | Linux is an open-source and free operating system. That helps significantly when thinking about the overall cost of a server operating system. Linux is based on UNIX OS and, since it is open-source, has a vast community of developers who offer free and helpful tips on setup and configuration. Linux protects files with role or user-based permissions and is widely considered to be the most secure. Root level access is for administrators, and everything else can be broken up to groups, others, and users.[[6]](#footnote-6) Linux does provide LDAP capabilities for user information and authentication.[[7]](#footnote-7) Linux is compatible with Docker, which compartmentalizes server resources more efficiently than virtual machines. Mac is not compatible with Docker at this time, which gives Linux an edge here.[[8]](#footnote-8) Linux will have some work involved with integration because it is meant to be tinkered with and tailored to your specific needs. Linux does offer cloud support with CloudLinux OS+.[[9]](#footnote-9) | Windows is a proprietary operating system put out by Microsoft. There is a licensing fee for each copy of Windows needed and the Windows Server OS which seems to start at $501 per device.[[10]](#footnote-10) Windows is not based on UNIX. Windows also limits the Client Access Licenses (CALS) per account.[[11]](#footnote-11) Windows uses role or user-based permissions for file access.[[12]](#footnote-12) Windows uses Active Directory which does have LDAP capabilities. Windows is compatible with Docker, which, as mentioned previously, is more efficient than a virtual machine environment common with servers.[[13]](#footnote-13) Windows is very easy to launch and integrate because it is so widespread. Also, Microsoft offers extensive support for the Windows servers that are out there. Microsoft, through Azure, offers cloud services for their servers.[[14]](#footnote-14) | I do not know if it is possible to use a mobile device as a server for other computers or mobile devices to connect to. If it is, then it would be incredibly taxing on the hardware to support even a small number of games at a time. I was able to read that Apple devices, using GameKit, essentially nominate the device with the best specifications to be the host when dealing with real-time applications, but the device is not actually a server in the traditional sense of the word. |
| **Client Side** | Mac clients have two major browsers, both going by the same name: Safari. Despite the uptick in popularity for Mac in recent years, Safari topped out in 2019 at 3.5% of the browser traffic on the internet.[[15]](#footnote-15) Safari is a pretty standard browser. It works much the same way as Firefox does and provides many of the same features. One thing to look out for is the JavaScript toggle. If our game uses JavaScript on the front end, users may have a difficult time with this feature. | Linux devices may not be the most common out there, but the use of the flagship Linux browser definitely is. The overall high in 2019 for internet traffic through Firefox was 10.2%.[[16]](#footnote-16) Firefox is portable, modern, and user-friendly but it does have some features like auto-play blocking which can create a problem for development.[[17]](#footnote-17) Browser extensions like ad-blockers and anti-tracking plugins like Ghostery could also pose problems with some functionality depending on the revenue method. | Microsoft is falling behind in the desktop client realm in recent years. The two major browsers, Internet Explorer and Microsoft Edge combined accounted for a high of just 4% of internet traffic in 2019.[[18]](#footnote-18) Internet Explorer is not popular, but it is still used. It’s not a secret that it’s on the way out, Microsoft even acknowledges that it’s over and they want people to switch to Edge.[[19]](#footnote-19) Since it is so unpopular and not supported even by the maker, I would suggest focusing on Edge. Edge does rely on extensions to handle things that were previously done by ActiveX.[[20]](#footnote-20) | Here, I thought, was the best place to mention Google Chrome. There are other mobile browsers, but Chrome has become a powerhouse on desktop and mobile. Making up 81.8% of all internet traffic[[21]](#footnote-21), and 88.3% of Android traffic and 3.9% of iPhone traffic.[[22]](#footnote-22) A big difference in developing for these clients is integrating the use of touch controls instead of standard I/O devices. Auto-rotate is another thing to take into consideration. There are many other considerations like cache in Safari, JavaScript toggles in Safari mobile, and expired resources in Chrome mobile.[[23]](#footnote-23) |
| **Development Tools** | Swift would be our primary development language for Mac. Xcode would be our primary IDE for Mac, though we could use Eclipse as well if a developer were more familiar with them. Eclipse does have a Swift plugin called Tifig.[[24]](#footnote-24) Xcode, the default, is extensive, but only available on Mac OS. So, if we used it, we would really have to commit. Any development done outside of it would be done from a non-Mac. Xcode supports C, C++, Obj-C, Obj-C++, Java, and many more. Xcode would provide many benefits when testing displays for mobile. Xcode itself is free but can cost to upload to the App stores, which should not impact this project.[[25]](#footnote-25) | Linux Development tools are widespread. They are also mostly free. I would look to Eclipse, AWS Cloud9, or Visual Studio. The site will likely be written primarily in Java since it was an Android application previously. All of the above environments support Java. There is no “official” IDE for Linux. Atom and Sublime also look like they would be good IDE options for Linux.[[26]](#footnote-26) This diversity would mean that developers could choose the IDE they are most comfortable with, or that the team could choose one and standardize it. | IDE environments for Windows would be Visual Studio, Visual Studio Code, or Eclipse. They all have different benefits and drawbacks. Visual Studio is very extensive but has several expanded sets that require expensive add-ons. The good news is that there is a free version available of both Visual Studio and Visual Studio Code. Eclipse is also free and heavily used on Windows. One drawback to Windows builds is that you won’t be testing, most likely, on the default option (Edge) because it is one of the least popular clients. | You do not develop or use development tools *on* a mobile OS. You do use them *for* mobile operating systems, like GameKit for iOS or Android SDK for Android. No tools need to be evaluated for use on the mobile platforms. |

## Recommendations

1. **Operating Platform**: I would recommend the Linux operating platform. In my research, I have found that Linux appears to be the best choice for servers. Linux is commonly believed to be more secure than systems like Windows or Mac OS.[[27]](#footnote-27) Security was a major concern for the client and Linux offers the needed security roles to allow the client the greatest level of access control for their system. One aspect of this security is the privileges. Linux isolates the root privileges where systems like Windows give administrator access (the equivalent) by default to users when they first set up the operating system.[[28]](#footnote-28) Linux is also the least taxing on the hardware. Some distributions require as little as 256 MB of RAM to run while Windows needs a minimum of 1GB.[[29]](#footnote-29) In practice, both need more than that, especially with the graphical interface, but Linux has lower overhead than Windows. Microsoft recommends 4 GB while Canonical (the makers of Ubuntu distribution) recommend only 2 GB of RAM.[[30]](#footnote-30) Lastly, Linux is free and open-source while Windows is licensed and can be quite expensive when running several copies.
2. **Operating Systems Architectures**: Linux, like all operating systems, is based on a kernel. Linux has what is called a monolithic kernel design.[[31]](#footnote-31) A monolithic kernel, some say, becomes a bit overly bloated because so much is being handled by the kernel. Given the minimum systems requirements for the most popular Linux distributions (Debian, Ubuntu, etc.), I would say that bloat is not really a problem. This is especially true when compared to Microsoft Windows or Mac OS X.[[32]](#footnote-32) The kernel runs all of the hardware and manages all calls to hardware related processes. The operating system then runs the shell. The shell is the layer of the operating system that offers either a graphical or command-line interface where a user can tell the computer what to do.[[33]](#footnote-33) Bash should be the default on most Linux machines, but the user can change the default shell to things like Zsh or Ksh.[[34]](#footnote-34) Applications run on the shell and commands that require kernel access are passed through the shell when needed. User space is segregated from kernel space and root users can access that kernel space.
3. **Storage Management**: Storage management is not very taxing on this project. The specifications call for storage of files amounting to approximately 2 GB of image files plus the code for the program. Making the storage accessible is another matter, though. That will require something with a good deal of RAM if it is supporting a significant number of users. That will be addressed in the memory section. Using the internal storage of a Linux server could be ideal for this setup because it will definitely have the storage capacity and would not require any additional hardware or licensing. However, AWS cloud services offers solutions that provide enough storage for this application with server access.[[35]](#footnote-35) Storage would need to be readily accessible on the website since the project is converting an Android application to a web application accessible via a browser. This would also mean that The Game Room would want something with rare outages and with backups in place to ensure the integrity of the files. AWS is known for being extremely reliable and even has tools built in for building games called GameLift.[[36]](#footnote-36)
4. **Memory Management**: Memory management needs to be monitored closely with this project. Using Linux on the server will already help with that. As stated above, Linux can be run on machines using very little RAM. Compared to Windows and Mac, Linux provides the most RAM to the rest of the system. The RAM is what matters when it comes to memory management since RAM is where items are stored temporarily as they are being used. This game calls for rendering images over the course of 30 seconds. That means that these images will be loaded into the RAM and slowly rendered for the users while the game is running. Here, I think that a solution like Docker would be ideal. When RAM is needed, it helps to have your servers not constrained by virtual machine requirements. Virtual machines siphon off a chunk of RAM and dedicate it to a specific instance. This has to be at least enough to run the OS and the program. Docker enables the operating system to run once for multiple sessions of the application. This saves memory and improves service and efficiency, and is currently only available for Linux and Windows.[[37]](#footnote-37) This is done using what they call “containers” and deploying the app within those containers. Then, memory can be allocated dynamically so that user A, for example, is not always using 512 MB of RAM even if the active process is only using something like 128 MB. This was one of the reasons for recommending Linux over Mac and Windows.
5. **Distributed Systems and Networks**: Due to the complicated nature of distributed systems[[38]](#footnote-38) and networks, I would recommend Amazon Web Services (AWS) for the distribution and network. AWS offers reliable connections as they are one of the most powerful web server provider in the country. This would make connectivity largely subject to the user’s internet provider or mobile provider, thereby eliminating many unknowns. Outages are rare and it should provide a very reliable connection. The cost of AWS can add up since it is able to scale automatically based on demand. That can be offset by an increase in expertise. Amazon has extensive customer support and, due to its prevalence, many developers are familiar with the way AWS works and how to set up their systems to work together. Communication between platforms needs to follow specific protocols and this can get quite complicated. A slow connection on one device needs to be accounted for when running a real-time game on the internet. One slow connection could throw the whole game off, if not handled carefully.
6. **Security**: Protecting user information is paramount. This should be done on a local server if possible, rather than on the distributed systems to prevent unauthorized access. Member access is one of the reasons I recommended Linux. Even if this information is stored on the primary server, Linux allows the administrator to build in protections. Windows, by default, gives users the option to run things as administrator. Linux protects files with three layers of permissions based on user and role. Furthermore, the programmers need to work in access controls with the way the game is coded as well to have multiple checks for whether the user is ADMIN or USER for example. This can protect data if someone logs in as a user and then runs some sort of exploit to attempt to access user data without being logged in as an admin. The ease of use of security controls for Linux is superior to both Mac and Windows. Since the clients will primarily be web browsers, specific measures need to be taken to prevent plugins or other tools from accessing any user data. Giving read-only permission to any user, for example, can greatly reduce any potential unauthorized access. On Linux, I believe, this should be chmod 644 with the file names or recursively to make it so only the owner can write or execute the file. If execution is needed for the client, then the permission should be chmod 755 or 655. On Windows and Mac, protecting files is a bit trickier. Windows uses attrib but that only sets read/hidden attributes on a single file and Mac uses icacls which does set the rights for individuals and groups. Mac does basically the same thing but in a more complicated way, while Windows doesn’t really seem to do it in general.[[39]](#footnote-39) Linux is the strongest, I believe, for finely-tailored security.

1. https://www.macworld.com/article/3025092/4-reasons-why-you-should-run-your-own-mac-server-and-4-reasons-why-you-shouldnt.html [↑](#footnote-ref-1)
2. https://www.cnet.com/news/understanding-default-permissions-in-os-x/ [↑](#footnote-ref-2)
3. https://stackoverflow.com/questions/663402/what-are-the-differences-between-ldap-and-active-directory#:~:text=LDAP%20%28Lightweight%20Directory%20Access%20Protocol%29%20is%20an%20application,protocols%20you%20can%20use%20to%20talk%20to%20it. [↑](#footnote-ref-3)
4. https://www.docker.com/resources/what-container [↑](#footnote-ref-4)
5. https://www.macworld.com/article/3274584/where-does-apple-stores-your-icloud-data.html [↑](#footnote-ref-5)
6. https://www.geeksforgeeks.org/permissions-in-linux/ [↑](#footnote-ref-6)
7. https://www.karmak.org/archive/2003/02/ldap/ldap-linux.htm [↑](#footnote-ref-7)
8. https://www.docker.com/resources/what-container [↑](#footnote-ref-8)
9. https://www.cloudlinux.com/ [↑](#footnote-ref-9)
10. https://www.microsoft.com/en-us/windows-server/pricing [↑](#footnote-ref-10)
11. https://www.microsoft.com/en-us/licensing/product-licensing/client-access-license [↑](#footnote-ref-11)
12. https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/access-control [↑](#footnote-ref-12)
13. https://www.docker.com/resources/what-container [↑](#footnote-ref-13)
14. https://azure.microsoft.com/en-us/free/virtual-network/search/?&OCID=AID2100131\_SEM\_X3qXjAAAAKdeqQVy:20201005034828:s&msclkid=89323a0bc71d13d83cc4657150999e23&ef\_id=X3qXjAAAAKdeqQVy:20201005034828:s&dclid=CJmq9tfFnOwCFWEXrQYdT-sNmA [↑](#footnote-ref-14)
15. https://www.w3schools.com/browsers/#:~:text=%20%20%201%20IE%20%3D%20Microsoft%20Internet,Navigator%20%28identified%20as%20Mozilla%20after%202006%29%20More%20 [↑](#footnote-ref-15)
16. https://www.w3schools.com/browsers/#:~:text=%20%20%201%20IE%20%3D%20Microsoft%20Internet,Navigator%20%28identified%20as%20Mozilla%20after%202006%29%20More%20 [↑](#footnote-ref-16)
17. https://www.mozilla.org/en-US/firefox/browsers/compare/safari/ [↑](#footnote-ref-17)
18. https://www.w3schools.com/browsers/#:~:text=%20%20%201%20IE%20%3D%20Microsoft%20Internet,Navigator%20%28identified%20as%20Mozilla%20after%202006%29%20More%20 [↑](#footnote-ref-18)
19. https://techcommunity.microsoft.com/t5/windows-it-pro-blog/the-perils-of-using-internet-explorer-as-your-default-browser/ba-p/331732 [↑](#footnote-ref-19)
20. https://community.spiceworks.com/products/63858-microsoft-edge [↑](#footnote-ref-20)
21. https://www.w3schools.com/browsers/#:~:text=%20%20%201%20IE%20%3D%20Microsoft%20Internet,Navigator%20%28identified%20as%20Mozilla%20after%202006%29%20More%20 [↑](#footnote-ref-21)
22. https://www.zdnet.com/article/which-browser-is-most-popular-on-each-major-operating-system/ [↑](#footnote-ref-22)
23. https://techbeacon.com/app-dev-testing/what-software-testers-need-know-about-browser-differences [↑](#footnote-ref-23)
24. https://marketplace.eclipse.org/category/free-tagging/swift-programming#:~:text=Tifig%20is%20an%20%5B%26Eclipse%26%5D-based%20IDE%20for%20the%20%5B%26Swift%26%5D,...%20General%20Purpose%20Tools%2C%20IDE%2C%20Editor%2C%20Programming%20Languages. [↑](#footnote-ref-24)
25. https://www.quora.com/How-much-does-Xcode-cost-Does-the-price-vary-depending-on-how-you-plan-to-use-it [↑](#footnote-ref-25)
26. http://www.linuxandubuntu.com/home/8-best-ides-or-code-editors-for-linux#:~:text=Gedit%20is%20a%20Linux%20IDE%20that%20comes%20pre-installed,environment%20by%20installing%20plugins%20and%20configuring%20existing%20settings. [↑](#footnote-ref-26)
27. https://www.pcworld.com/article/202452/why\_linux\_is\_more\_secure\_than\_windows.html [↑](#footnote-ref-27)
28. *Ibid.* § 1. Privileges [↑](#footnote-ref-28)
29. https://fossbytes.com/best-lightweight-linux-distros/#:~:text=Best%20Lightweight%20And%20Fast%20Linux%20Distros%20For%20Old,4.%20Ubuntu%20MATE.%205%205.%20Lubuntu.%20More%20items [↑](#footnote-ref-29)
30. https://www.fossmint.com/linux-vs-windows-ram-usage/#:~:text=Well%20we%20can%20look%20at%20system%20requirements%20that,popular%20Linux%20Version%29%20Canonical%2C%20recommends%202GB%20of%20RAM. [↑](#footnote-ref-30)
31. https://www.techopedia.com/definition/27003/monolithic-kernel [↑](#footnote-ref-31)
32. https://news.softpedia.com/news/the-amount-of-bloatware-in-windows-10-is-just-ridiculous-524716.shtml [↑](#footnote-ref-32)
33. https://www.elprocus.com/linux-operating-system/#:~:text=%20Linux%20System%20Architecture%20%201%20The%20kernel,to%20do%20individual%2C%20and%20specialized-level%20tasks.%20More%20 [↑](#footnote-ref-33)
34. https://www.tecmint.com/different-types-of-linux-shells/#:~:text=5 Most Frequently Used Open Source Shells for,Shell. 4 4. Zsh Shell. 5 5. Fish. [↑](#footnote-ref-34)
35. https://aws.amazon.com/ [↑](#footnote-ref-35)
36. https://aws.amazon.com/pricing/ [↑](#footnote-ref-36)
37. https://www.docker.com/resources/what-container [↑](#footnote-ref-37)
38. https://d1.awsstatic.com/builderslibrary/pdfs/challenges-with-distributed-systems.pdf#:~:text=Developing%20distributed%20utility%20computing%20services%2C%20such%20as%20reliable,forms%20of%20computing%20because%20of%20two%20interrelated%20problems. [↑](#footnote-ref-38)
39. https://superuser.com/questions/106181/equivalent-of-chmod-to-change-file-permissions-in-windows#:~:text=There%20is%20nothing%20called%20chmod%20in%20windows%20because,objects.%20%28But%20they%20are%20more%20towards%20global%20properties.%29 [↑](#footnote-ref-39)