File Sharing Artefact

By Rian Rutherford - *RR262471; 6th February 2023*

1,438 total words

Problem

152 words

There are many cloud storage solutions available: Dropbox [6], Google Drive [12], and OneDrive [16]. They all provide permission access levels, but they don't provide file locking. This is a problem when you want to prevent two or more users accidentlly working on the same binary file at the same time.

For word documents this normally isn't a problem because they have a built-in editor that allows more than 1 user to edit at a time. But it is a problem for unsupported formats like FBX, PNG, and PSD.

There are systems that do support file locking. Those being Software Configuration Management [22] [14] solutions like Plastic [5], Perforce Helix TeamHub [17], and Subversion [3]. But each one of these has their own problems.

- Plastic is $7-$23 per user per month depending on the plan.
- Helix TeamHub from Perforce is $23+ per user per year (varies depending on the plan).
- Subversion is old and doesn't have any good user interfaces.

Artefact Solution

392 words

I plan to make a client and server solution that that can be privately hosted by anyone. It's a file storage and syncing solution like Cloud Storage, but with file locking which is commonly found in Software Configuration Management (also known as SCM) systems.

The user will sync/connect a folder to a server via the command line from the client. Once connected to the server, that folder on their computer will mirror every change that's made on the server.

When a user saves any changes made to a file, it is automatically synced to the server and all connected clients. If a user locks a file, then anyone who saves changes to the locked file will have their changes rejected by the server and their unauthorized changes reverted locally.

Artefact Client Features:

- Syncing of file & folder directory with the server the client is connected to.
- When a file is saved, the changes are instantly synced to the server.
- Files can be locked if not already locked.
- Files can be unlocked by the client who locked them.

Artefact Server Features:

- Files are stored on the server.
- When a changed file is synced to the server, the server broadcasts the changes to all connected clients.
- If a file is locked by a client, then any new changes made by a different client will be rejected by the server.
- Optional: Clients can force override of locked files.
- Optional: The server can be configured to have lifetimes for how long files will remain locked before they unlock automatically.
- Optional: When a file is synced to the server, it's automatically locked and assumes the user will continue to make edits.

Legend - Optional means a feature that would be nice to have, but is outside of scope and will only be added provided there is still time remaining.

The optional features are more challenging to implement, but would make a great addition if I get the time. I used Perforce during one of my recent GAMJAMs [15]. I didn't have any of the optional features, as suggested above, available in Perforce which caused problems. People are forgetful, they forget to lock files and they forget to unlock them. By making locking automatic on saves and giving it specified lifetimes, even if people forget their files are still appropriately locked and unlocked for them most of the time.

Artefact Defence

185 words

My artefact serves a purpose of being a free, self hosted, network file sharing solution that can will run on almost any computer. It operates like normal file storage allowing it to be used with a wide array of applications.

For people that want a simple home Network-Attached Storage (NAS) [4] [23], this will allow them to repurpose their old phone, laptop, or desktop instead of buying a NAS that costs them £150 to £4k [2].

This also provides people & businesses the option to build their own NAS systems from new or old hardware without relying on on expensive software or vendor locking hardware & software.

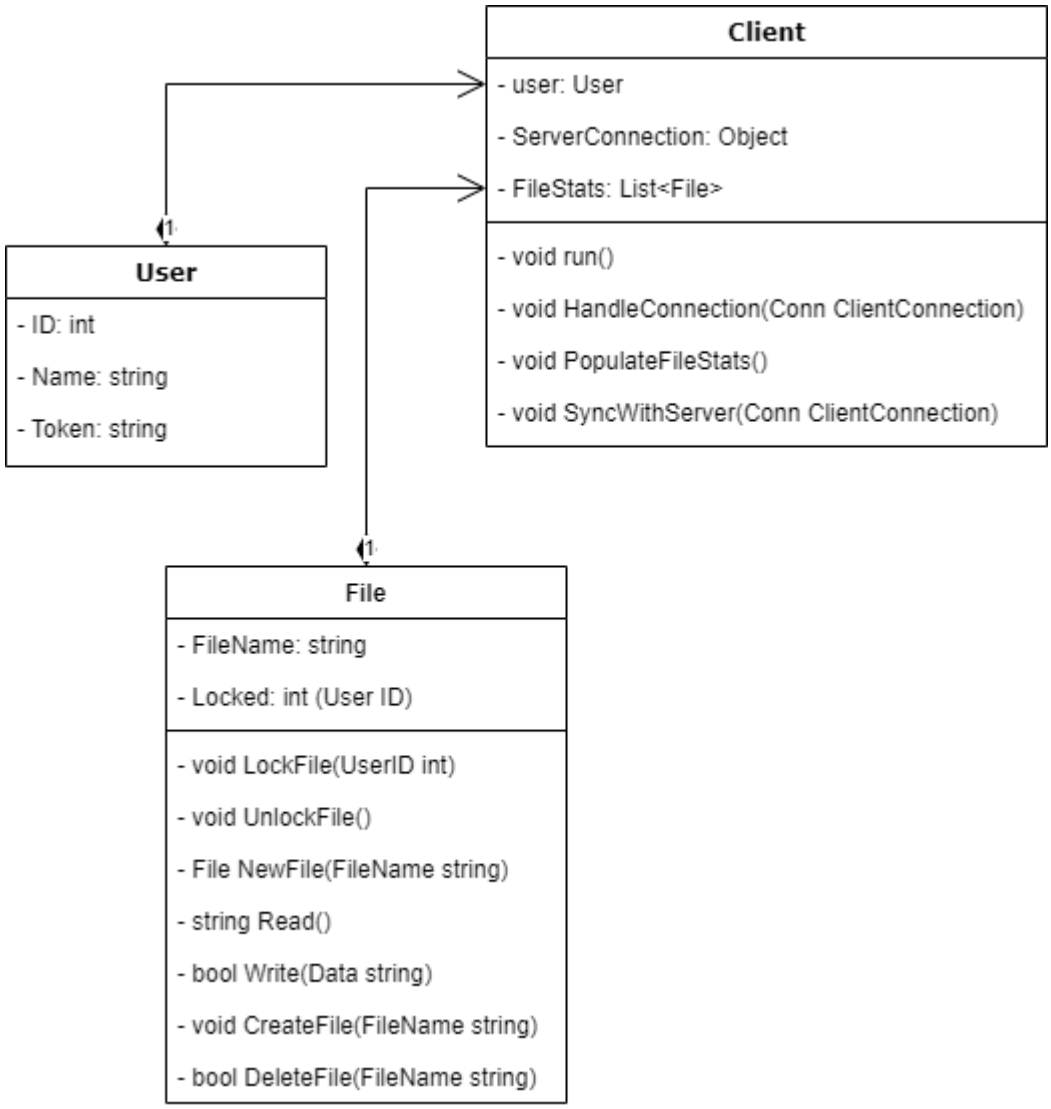
The simple nature of the artefact will also mean that it should be easy to mod and integrate into other pipelines and software's.

During the recent COVID pandemic many business went remote and started using Cloud services [10]. In my opinion cloud and internet services as a whole are very young in the remote collaboration sector, because until recently most companies had no reason to facilitate staff working offsite, and so the technical issues with doing so haven't been thoroughly explored yet.

Artefact Architecture

65 words

Client Design

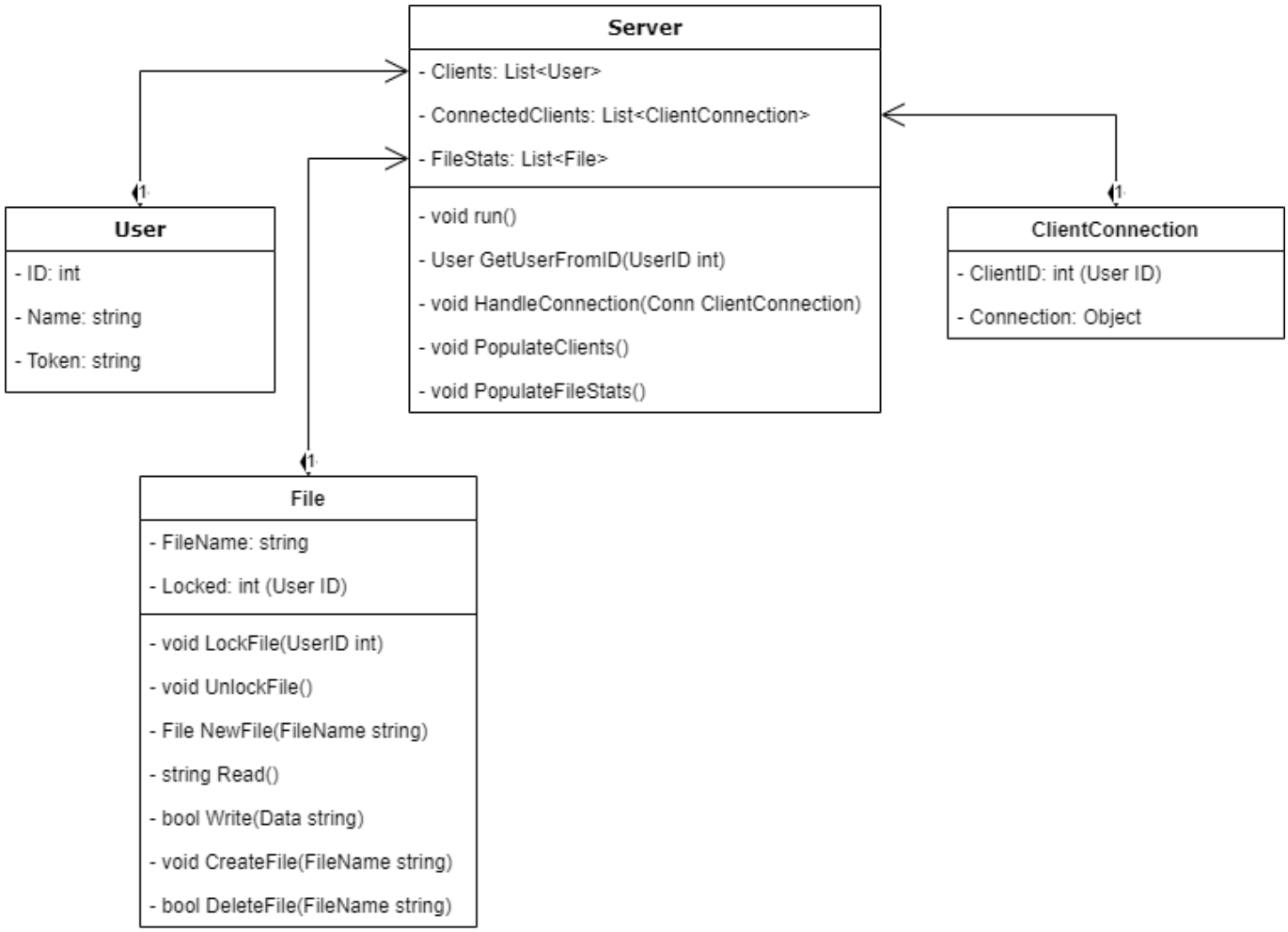


Client Internet Packet Specification

Required Fields:

1. User ID - *The users public ID so it can be identified*.
2. User Token - *Used to validate the user owns the account*.
3. Request Type - *Decides how the sent package is treated*.
4. Package - *Data to be used by the server*.

Server Design



Server Internet Packet Specification

Required Fields:

1. Request Type - *Decides how the sent package is treated*.
2. Package - *Data to be used by the client*.

Development Plan

364 words

I plan on doing a simple test on Python 3 [18], Rust [19], and Go [11] to see which is better suited to developing this artefact. Python is easy to use and has a lot of supporting libraries as seen by how popular it is [9]. Go has great support for making server software because that is what it was made for [13]. Rust has a low overhead and if suitable would enable the software to run more efficiently and on more systems like phones because of the lower hardware requirements [19].

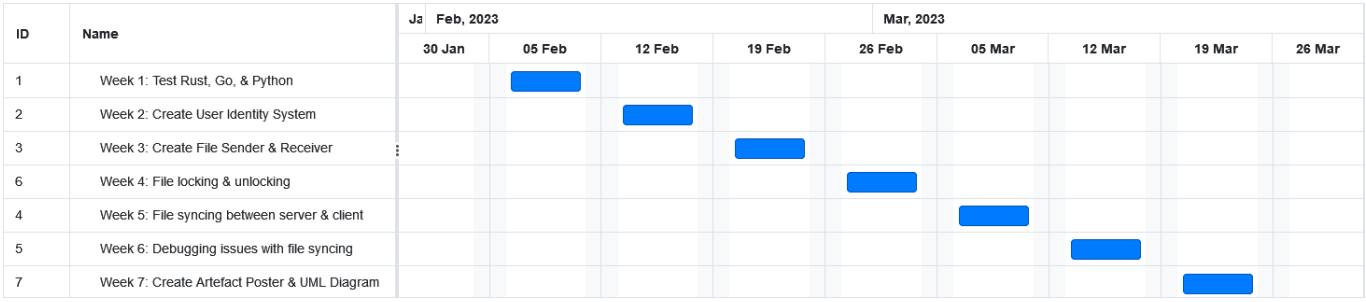
Each of the three languages will go through these tests:

1. Make a TCP client & server to test the difficulty of coding client & server.
2. Make a concurrent/threaded console application to test the difficulty of implementing concurrency into the artefact.
3. Parse JSON or serialized data to test the difficult of transmuting data to bytes for sending and receiving over the internet.

4. Make a script that reads & writes to a file to test the difficulty of doing so.
5. Test the difficulty of reading input from the console.

The language that provides the best ease of use between all 5 tests will be used to develop the artefact. If there is a draw between any of the languages then they're favoured in order: Rust over Go, and Go over Python 3. The winning language will be used to develop the artefact because it will have proven suitable for this use case.

For developing the artefact I have created a Gantt Chart to estimate how long it will take to make.



Weekly Breakdown:

1. Week 1 is where I test and decide the appropriate programming language for the artefact.
2. Week 2 is when I create backend systems for identifying users connected to a server.
3. Week 3 is when I create the base code that will handle sending files between the client & server.
4. Week 4 is when I add file locking based on user identities.
5. Week 5 is when I functionality for syncing files between client & server.
6. Week 6 is reserved for debugging the artefact.
7. Week 7 is when the artefact architecture UML diagram & technical poster are created.

Practice Research

280 words

While planning out a demo, artefact, or MVP it is very productive to use a Gantt Chart or plan out exactly what you will do because it allows for full focus on developing the system without any interruptions.

Planning everything out has a drawback though, it doesn't incorporate customer feedback into the design of the product or service. The industry standard for developing software is to use Agile based or Agile inspired development. The first two principles of Agile are about the customer [1], that's how important they are to the development of a system.

Once my artefact is complete I plan on using Extreme Programming [20] methodology for the development cycle of the product/service. Extreme Programming (XP) is like Agile, but it has a specific order to things [21]. XP integrates the consumer into the development cycle of the product or service.

Using XP I will add, remove, and change features of the system to match the consumer's needs. Business, individuals going GAMJAMs, and indie game development studios will be the target consumers that will be used to inform the design choices of the system to turn it from an artefact to a stable product & service that can be used in industry and by individuals.

Immediate candidates are the local Games Academy [8] and Falmouth Launchpad [7]. At the Games Academy they teach students video game development and use version control, which makes them a target audience. Falmouth Launchpad supports start-ups, who use a wide array of cloud based services to start and run their businesses. Launchpad start-ups are a target audience because they use services like Google Drive, and OneDrive both of which don't support file locking - *the artefacts key feature*.

Bibliography

[1] AGILE MANIFESTO. 2001. *Principles behind the Agile Manifesto*. Available at: https://agilemanifesto.org/principles.html [accessed 06 February 2023].

[2] AMAZON. *Network Attached Storage*. Available at: https://www.amazon.co.uk/network-attached-storage/b?ie=UTF8&node=430553031 [accessed at 06 February 2023].

[3] APACHE. 2000. *Subversion*. Available at: https://subversion.apache.org/ [accessed 05 February 2023].

[4] BIGELOW, Stephen J, Ben LUTKEVICH, and Garry KRANZ. 2022. *What is network-attached storage (NAS)? A complete guide*. Available at: https://www.techtarget.com/searchstorage/definition/network-attached-storage [accessed at 06 February 2023].

[5] CODICE SOFTWARE. 2006. *PlasticSCM*. Available at: https://www.plasticscm.com/ [accessed 05 February 2023].

[6] DROPBOX. 2008. *Dropbox*. Available at: https://www.dropbox.com/ [accessed 05 February 2023].

[7] FALMOUTH UNIVERSITY. 2014. Available at: https://www.falmouth.ac.uk/launchpad [accessed at 06 February 2023].

[8] FALMOUTH UNIVERSITY. *Games Academy*. Available at: https://www.falmouth.ac.uk/departments/games-academy [accessed at 06 February 2023].

[9] GITHUB. 2022. *The top programming languages*. Available at: https://octoverse.github.com/2022/top-programming-languages [accessed 05 February 2023].

[10] GOKARNA Mayank. 2021. *Reasons behind growing adoption of Cloud after Covid-19 Pandemic and Challenges ahead*. Available at: https://arxiv.org/ftp/arxiv/papers/2103/2103.00176.pdf [accessed at 06 February 2023].

[11] GOOGLE. 2009. *Go*. Available at: https://go.dev/ [accessed 05 February 2023].

[12] GOOGLE. 2012. *Google Drive*. Available at: https://www.google.com/intl/en-GB/drive/download/ [accessed 05 February 2023].

[13] GOOGLE. 2019. *Go for Cloud & Network Services*. Available at: https://go.dev/solutions/cloud [accessed 05 February 2023].

[14] GURU99. Available at: https://www.guru99.com/software-configuration-management-tutorial.html [accessed 05 February 2023].

[15] LORDUNDERWORLD. 2022. *Knight Watch*. Available at: https://lordunderworld.itch.io/knight-watch [accessed 05 February 2023].

[16] MICROSOFT. 2007. *OneDrive*. Available at: https://www.microsoft.com/en/microsoft-365/onedrive/online-cloud-storage [accessed 05 February 2023].

[17] PERFORCE. *Helix TeamHub*. Available at: https://www.perforce.com/products/helix-teamhub/pricing [accessed 05 February 2023].

[18] PYTHON. 1991. *Python*. Available at: https://www.python.org/ [accessed 05 February 2023].

[19] RUST. 2010. *Rust*. Available at: https://www.rust-lang.org/ [accessed 05 February 2023].

[20] WELLS, Don. 1996. *Extreme Programming: A gentle introduction*. Available at: http://www.extremeprogramming.org/ [accessed at 06 February 2023].

[21] WELLS, Don. 2000. *Extreme Programming Project*. Available at: http://www.extremeprogramming.org/map/project.html [accessed at 06 February 2023].

[22] WIKIPEDIA. Available at: https://en.wikipedia.org/wiki/Software_configuration_management [accessed 05 February 2023].

[23] WIKIPEDIA. *Network-attached storage*. Available at: https://en.wikipedia.org/wiki/Network-attached_storage [accessed at 06 February 2023].

Tools

List of software & services used to make this document:

1. Visual Studio Code (VSCode)
2. Markdown PDF - *VSCode Extension*
3. Online Gantt
4. diagrams.net - *used to make UML diagrams*