**7CCSMGPR GROUP PROJECT**

**FINAL REPORT**

28TH March 2019

Lucky Group:

* Robinson, Thomas James 1880679
* Choi, Daeun 1821307
* Wu, Zihao 1836287
* Fu, Yixin 1862819
* Sarosi, Marcell 1542196
* Zhu, Tianning 1817456

Contents

1. Introduction
2. Context
3. Achievements
4. Review
5. Requirements and Design
6. Implementation
   1. Web Application
   2. Mobile Application
7. Team Work
8. Evaluation
9. Peer Assessment

1. Introduction

i. Context

For the last 12 weeks Lucky Group have been developing a Multi-Host File Synchroniser which is accessible on mobile and web platforms. The want for ‘multi-host’ file systems has increased dramatically with the development of the web, allowing people to collaborate on complex projects means that there is a greater pool of knowledge contributing, which in turn leads to greater work being produced and in less time. As well as allowing more people to contribute to a project a multi-host system also allows people to work on their projects from wherever they are as long as they have an internet connection. This further aids the ability of home-based teleworking, which has been found to increase productivity and decrease job attrition by over 50% (Bloom, Liang, Roberts, & Ying, 2015). As multi-host systems allow remote access to files and as companies are looking to decrease overhead at every point this kind of system, which increases flexibility for employers and employees, is seriously increasing the possibility of removing the need for a physical office location which is one reason they are becoming so popular.

One of the main aspects of working as a team, especially if working remotely, is consistency. Each user must be working from the latest version of the document and be updated with new content as and when it is needed, this is where the synchronisation becomes important. Synchronisation means every user on the system can see where the project is currently at, also allowing management to make more informed decisions to keep the project on track. Synchronisation is the key to allowing multiple users access the same document at the same time without corrupting the master file when the two versions are uploaded. Systems do this by scanning documents to see the changes then merge them, depending on the rules set when the system is created additions and deletions will be made to the master document as each user uploads their version. There are a few ways this can be executed which depend on the user’s needs for the system, these methods will be discussed later in the report.

This is where is problem lies, there needs to be a way in which users can be confident that they have data that is consistent with the rest of their team and that when they make changes to the project they will transferred to the rest of their team without any errors or deletion of any others work that is still relevant to the system.

ii. Summary of Achievements

As a group we have achieved the completion of a mobile application and a website that will allow users to upload and download documents to and from their own disk space. After downloading a file from the website / app the user can edit the document in whatever way they please. After they have finished editing the file can be re-uploaded under the same file name, at this point the system (API?) will identify any changes and update the file on the system with the latest version of the document. If there is a conflict of two users trying to upload at different times, the most recent upload will take precedence over the older upload, given they have the same access level. These files will then be available to see in after the user has logged into the system, where they will be presented with all the files on the system, this is where the cycle of downloading an uploading may occur.

Along with the completion of the synchroniser we have also created two reports, of which this is one, and given two presentations documenting the implementation and organisation of the project.

Review

I will now discuss some of the file sharing systems that are currently on the market. These sites are owned by some of the largest technology companies which shows how valuable this kind of system has become with all of the systems having hundreds of millions of users. Many of these systems have free access up to a certain point of data storage, when this point is reached there will be a fee charged to gain access to more space on their database. In this review I will be looking at the positives and negatives in these systems and suggesting ways in which they could be utilised for our system. The systems I am reviewing are mainly aimed at average consumers instead of corporate customers as this is what our system will be directed towards

The first system to be reviewed is Google Drive, owned and created by Google, which has over one billion users (reported at Google conference, reported by (Tech Crunch, 2018)). Google Drive offers online and offline platforms in the form of a website, applications for desktop users on Windows and macOS and applications for mobile platforms including Android and iOS smartphones and tablets. This platform allows you to store any file you want up to 5 TB per file, with only the first 15 GB being free you must subscribe to Google One to receive additional space (up to 30 TB). Drive also offers editing capabilities directly within the system by including other Google products on the platform. All information (Google, 2019).

The most interesting feature of Google Drive, that could be implemented onto our system is the ability to directly edit files on the platform without having to download the file to your local disk space. This feature speeds up the process of making simple adjustments to the file as you do not have to go through the upload and download process, this also means that users save room on their local disk and on their bandwidth as the document may be a large file that would affect other users on the same network. One disadvantage of using Drive is that it does not offer offline applications for Linux systems and Windows phone, although these only account for around 2% (Net Market Share, 2019) and 1% (Statista, 2018) of the average consumer market[[1]](#footnote-1) share respectively, there will still be a large amount of users that will not be able to effectively handle offline file sharing.

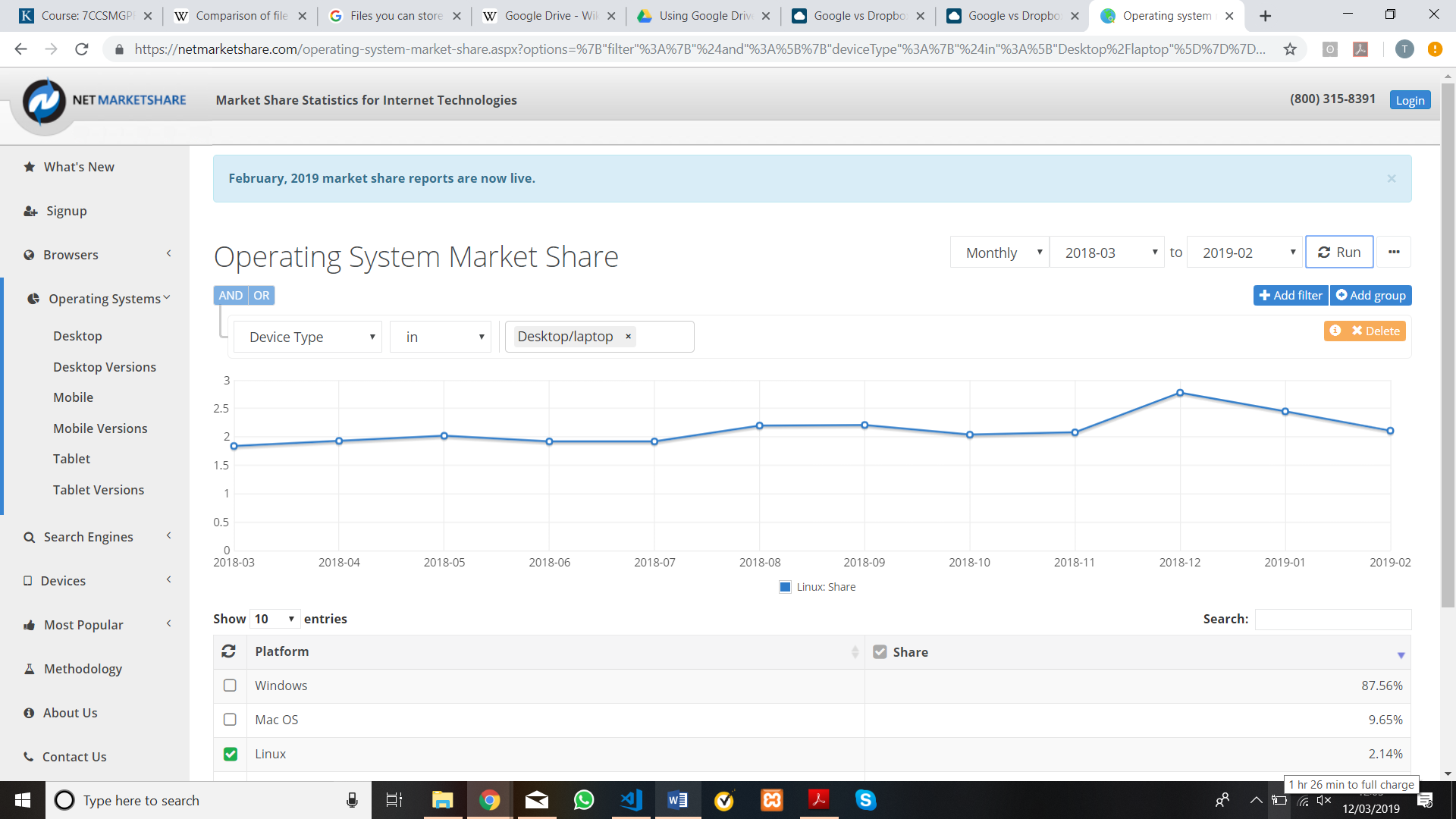
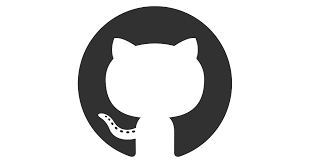


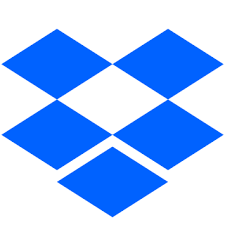
Table 1 - Linux Market Share for the Last 12 Months

Another disadvantage to Drive is the way that that you share files with other users, by sending a link to the file or folder that you wish to allow the receiver to edit. This is a disadvantage as they have not implemented an option to have a password on the links, there is also no expiry dates or download limits. These all mean that the security of the shared file / folder could be easily compromised.

The next system under review will be GitHub, which is now owned and ran by Microsoft after acquisition in 2018 (Warren, 2018), which has over 32 million users (GitHub, 2019), making it the largest host of source code in the world[[2]](#footnote-2) (Gousios, Vasilescu, Serebrenik, & Zaidman, 2014). GitHub Desktop is the application that allows you to perform GitHub operations without accessing their website, you can also perform all the GitHub features directly in the command line with third party software installed on your system e.g. Git BASH or Git for Windows. There is no official application on either iOS or Android, although there are apps to remove notifications e.g. GitHawk for GitHub. There is no set disk quotas on GitHub, within reason, stating on their website “Keeping repositories small ensures that our servers are fast and downloads are quick for our users.” (GitHub, 2019). GitHub is a more complex file sharing system which is actually known as a Git repository service which is used for sharing projects involving computer code as it offers all the distributed version control and source management functionality of Git.

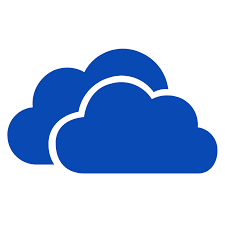
The main advantages that GitHub has involve its distributed version control. Distributed version control means that you, or anyone who has access to the repository, can update your file from any location, allowing for collaboration. Git also allows you to compare changes to a previous version, so if the program no longer works properly you can compare the changes that were made to the previous version to rectify or revert to the previous version. These features mean that GitHub is a prime contender for file sharing if the file includes computer code. The other main advantage for GitHub is the branching feature, this feature allows you to make changes to the document in isolation then merge it with the main branch when you’re sure it works correctly. This means that one user can work on the project at the same time without affecting any other users work, then when they merge to the main line the changes will be implemented independently.

One disadvantage of GitHub is the learning curve expected when first using GitHub. It isn’t as intuitive to use as the other multi-host file synchronisers discussed in this review, with many users claiming it is aimed more at programmers which means it doesn’t communicate with layman’s so well, meaning many people use the CLI over the GUI. This is something that this project will try to avoid by having a clean and simple GUI.

Dropbox will be the next system under review, owned and ran by Dropbox, Inc, which has over 500 million users (Dropbox, 2016). Dropbox offers offline applications for all three major desktop operating systems and mobile applications for Android, iOS and Windows Phone. Dropbox can also be set up to have a special folder on the user’s computer of which can be synchronised with Dropbox servers and other computers that have Dropbox installed as well as smartphones with the mobile applications. Dropbox comes with free storage of up to 2 GB, at this point you can upgrade to a paid personal plan which includes 2 TB or a paid plan ‘For Teams’ which have either 3 TB or an unlimited amount of space (Dropbox, 2019). Dropbox only has one native application built into the system, Dropbox Paper, which is essentially for note taking. However, Dropbox does allow for third party apps to connect adding extra functionality.

For Dropbox, the main advantage of this system lies with its file sharing capabilities. Although you must have the file on your local disk as well as on the cloud service, meaning you do not save any disk space on your local file, Dropbox makes use of block level file transfer. This essentially means the only time the whole file is transmitted to the server is the initial upload, with the future edits being uploaded in 4 MB blocks, this means that files should sync faster.

A disadvantage of using Dropbox compared to other systems is the fact that you must use third party applications to edit any documents, although they have Office integration, there are still some office applications that are missing. Another disadvantage is the amount of space and functionality that is granted with the free membership. Although, the pricing, storage and app integration isn’t something that we will be including in this project.

The last system I will be reviewing is OneDrive, which is ran by Microsoft, again has hundreds of millions of users. Offering desktop applications for Windows and Mac users as well as mobile applications for Android and iOS, but none for Linux and Windows Phone users. OneDrive also allows for a folder to be set up in the users file explorer in which you can directly transfer files to the cloud service and this comes as a default on Windows computers. OneDrive comes with 5 GB of free storage, which can be upgraded to 50 GB of storage without any other subscriptions, at this point there are only ‘premium’ accounts. These accounts include other Office 365 applications and offer either 1 TB or 6 TB (for 6 users at 1 TB per person) (Microsoft, 2019). As this is a Microsoft service, Online Office is readily available, as well as OneNote, which is Microsoft’s answer to Dropbox Paper. Any other third-party application integration is harder to come by, although Office does cover most of the essential tasks that you could want to complete without downloading the file.

A strength that OneDrive has is its file sharing operations, this has a clean and simple mechanism to share and collaborate on projects with other people. With password and expiry date capabilities, you can be sure that the intended recipients are the only person/people that can read the documents. OneDrive also makes use of block level file transfer, although not at the same rate as Dropbox, it still makes for faster upload speeds

One thing that I have found from personal use of OneDrive is that when the space is full in your OneDrive account there are persistent notifications prompting you to upgrade your account, with the synchronisation a default on Windows systems, this becomes full quickly and this has led me into using another service for cloud storage. Other users have complained about OneDrive struggling to deal with files with hundreds of sub folders and subsequently a large number of files[[3]](#footnote-3), meaning OneDrive would not be suitable for people with large amounts of data to transfer or for companies to hold their data on

iCloud

Requirements & Design

The first requirement we discussed as a group was whether we wanted the user to manually upload the file to the system or for there to be a synchronisation update of the file on the system every X minutes, without any action from the user. We came to the decision to keep the uploading of the files to the user, the main reason we decided to use this method is because it will provide less opportunity for conflicts to occur, meaning that when the file is overwritten there is less chance of your work being deleted. If the files were being deleted as they were being written it could cause confusion for the author. This method also doesn’t require a constant connection with the server, allowing for offline work to be completed without exceptions being called. With less connection to the server required, this is also a benefit for the rest of the network as there is more bandwidth space for other users to perform their tasks. This also means that if a user was to lose connection with the server and close their local version without saving (with them assuming a version had been uploaded to the system), they would lose all their work up to the last upload point or all the work created since opening the file.

This is one of the main strategic decisions that shaped the outcome and design of the project as this would require less UI for the user to interact with and potentially include a file in the user’s desktop similar to the way that OneDrive and Dropbox do. This would also place a lot of work on the server side of the system, with our experience as coders not being in server-side programming, we decided to keep most of the work being in the user’s application. This decision was made unanimously within our group, with everyone agreeing that this method was the best way for us to proceed.

The next decision we made for the system was to handle what would happen should a conflict of uploads occur; this choice was made at a similar time to the first strategic decision as when deciding what should happen when there was a conflict changed depending on whether there was a constant connection with the server or the manual upload that we settled on. A conflict can be defined as when two users who are working from the same version of a file locally then proceed to make different changes to the document and then upload. If a conflict occurs in our system a pop-up will prompt you to decide between **replacing the file** in the system or **renaming the file** with an additional identifier attached to the file name. This works by using the ‘rsync’ algorithm, this will be further explained in the implementation section.

Should you choose to replace the file, what is currently in the database will be permanently deleted and that version of the file will only be held on the local disk of said uploader. We decided to use this method of syncing the files as this allows for more control over the master copy, this is as if someone if unsure if they should be updating the master file they can save it under a new name. This file can then be reviewed by the team of people working on the project who can then at this point decide if they want to commit the changes to the master copy. Although this method may lead to an untidier database with multiple copies of the largely similar files, this can quickly be fixed with efficient team work to update the master copy. Another reason we wanted to use this method was that it would be easier to implement a ‘manager’ profile with user access levels. This would allow the system too to have an omniscient figure that can ensure that only what is required goes into the latest version of the system and nothing pivotal is deleted.

Again, this is one of the defining features of our system that deeply affected the rest of the design decisions. If we had decided to use the timed synchronisation method with this strategy, there could be constant stream of uploading new files should there be a conflict with another user working on the file at the same time. This would be an inefficient use of space on the database and would also further increase the strain on the connection as uploading a whole new file every X minutes would take up a lot of bandwidth. There was a discussion within the group about the type of synchronisation that we wanted to implement, with the other option being the master copy being updated with the just the latest changes, should a conflict arise the changes would be merged into one document on the system with a mix of the changes implemented on the document. But having decided on using the manual uploading the majority of us agreed that this was the most effective way of handling conflicts as this gives the team more control over what the master copy contains.

When you first load our application, you will be presented with a login page. At this point you can either login with a previously registered account, if you don’t already have an account you will have to register with a valid email and matching passwords. We decided to include user login so each user can have a page which they can see all the files they have uploaded, this also means that if someone edits a document already on the system there will be a signature attached to this edit, allowing people to see who changed the document and when. Having user accounts also increases the security of the system as only authorised user will be able to access the database, it also allows us to potentially implement locks on certain documents meaning an author could be able to decide on specific users that can edit a document. User accounts also mean that implementation of a manager profile can be set at the start of the process, giving them different abilities to average users.

This decision allowed us to have a more traditional flow of a file sharer and increase security but the design of the system stayed largely the same if there wasn’t a user login at the start of the system. The only difference in design would be depending on if the users have different access levels where one user could have a higher priority when it came to replace the master copy on the database. This level of user could also receive a notification or an email every time someone makes changes to a document that they originally uploaded. The decision to have a login page was unanimously agreed from the start of the project, the use of access levels was also agreed but we decided the implementation of this system was to have a low priority as this is not vital to the core components of the system.

Users of this system will be able to upload any document type to the system; this process occurs on the ‘UPLOAD’ page which is accessible from the main menu when the user has passed the login page. There will be an option to either drag the file directly from the file location open in another program or search their files from a button on the application. When uploading an item, you will be able to add certain footnotes to give other people looking at the item a more information about what is in the folder / document.

As we have decided to allow any document type to be uploaded to the system there will not be any function to edit documents directly on the system, as there is a vast number of file types that could be uploaded it wouldn’t be possible to implement something that could edit all types of files on this system. The decision not include online editing was agreed within the group to be a low priority and the concentration was to be placed on the core functionality of the system, something that this does not fall under. This works by using a HTTP POST request.

One of the main features of our system, considering we aren’t using within-app editing, is the download function. A user will be able browse all the files they have uploaded and any files that have been made available to see by all the users on the system. When they find the file, they want to download they will be able to inspect the document (read the information provided by uploader), then decide if they want to continue with downloading the item. The user will then be able to DECIDE where they want the file to be saved to. This works by using a HTTP GET request.

The ability to only see the documents you have uploaded and ones that have been marked public allows a user to have a system that is like GitHub in the sense that when a user is happy with the version they have been working on privately they can reupload to replace the master copy.

There will also be the option for a user to delete a file from the system if they want to, a user may only delete a file that they have uploaded or edited as a user may accidentally delete another user’s work with a similar title. This may also be implemented with the access levels, meaning that if a user has a ‘manager’ account they may be able to delete all the files in the system as they are trusted to know what is and isn’t important. This works by using a HTTP DELETE request.

We have designed the database to include 3 tables; User, Password and File. The User table will contain a list of usernames, the Password will contain a lift of passwords, the file table will contain information about the files on the system, this includes; size of file and name of file. Other file information is held on the SERVER, here you can see when the file was last updated as well as who uploaded the file. At the users end they can affect what is on the database by changing the name of the file and the file size will update according to what is in the file. This information has a bearing on how our synchronisation works, when deciding which file is to take precedence on the system it will look at what the most recent edited file and which file is the largest. This works by using a HTTP PUT[[4]](#footnote-4) request.

Implementation

* Describe most significant part of implementation
* Include code fragments
* Testing
* Performance issues

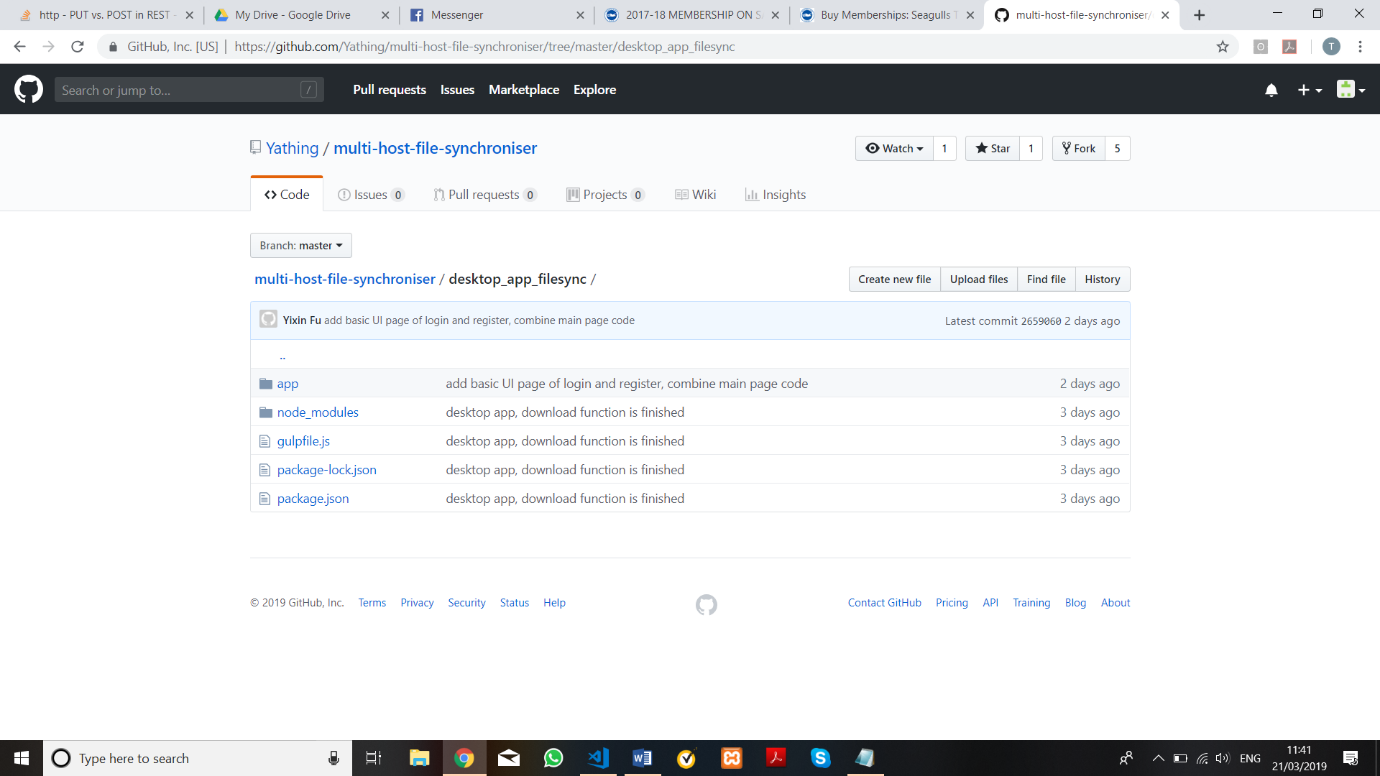
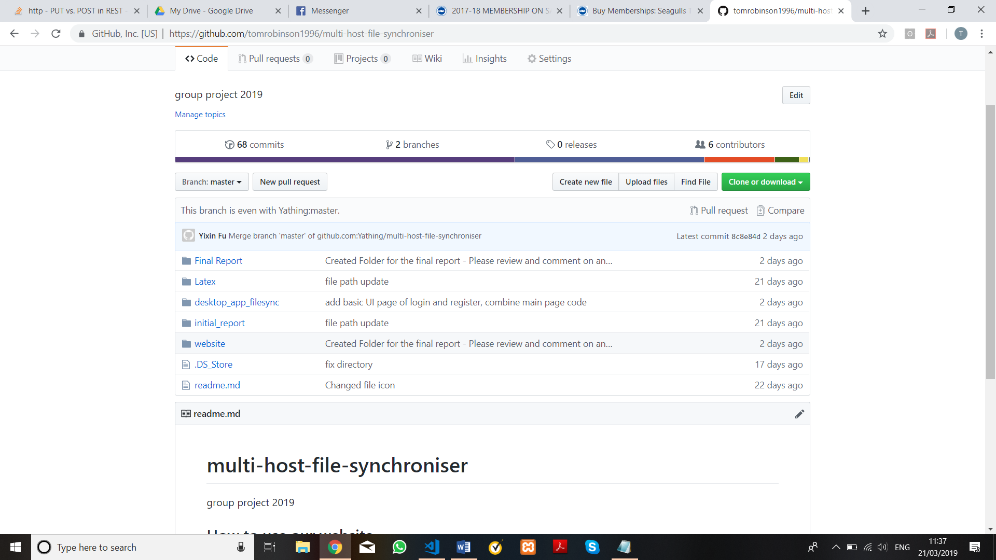
Team Work

Overall, the team worked well together with no major conflicts of personality or opinion in the way to handle the project, this can be attributed to a few reasons. One of the main reasons was that everyone wanted to achieve the same goal, as we soon came to realise this project is very large and complicated, which meant that no one person could complete the task successfully without the help from other members. This led us to agree to meet every week, enabling us to discuss what we tasks needed completing in the next week as well as discussing the overall aims for the project. It was in one of these meetings towards the end of project one member expressed their concern of using the ‘rsync’ algorithm with the web application. After hearing his reasoning, we decided to change from a website to a desktop application, this was a big change for both the front and back ends of the system and had we not worked efficiently as a team we would not have completed the project in the capacity we did. Although the functionality of the system is not as great as we would have originally liked, the synchronisation works as planned which we felt as a group was the most core feature of the system.

The main reason I think the team worked well was that if someone didn’t understand any part of the project, from GitHub to the logic of our system, someone was readily available to help the person understand what they were struggling with. This could be over Facebook Messenger or in a face to face meeting, which are two of the tools we used to facilitate group work. There were a few occasions near the start of the project in which Yixin took time out of her day to meet with people to help them to get to grips with using GitHub and multiple occasions where Zack met with members to help them understand the logic of the system and help them adapt their code to suit the logic.

The main tool we used for version control was a GitHub repository, as explained earlier GitHub has the best features to deal with computer code due to its ability for identifying changes in multiple user’s code and merge these into the master branch without much input from a ‘leader’ profile. Although there was a learning curve when starting up with GitHub, which was eased by working as a team, the whole group and the project has benefited from the features that GitHub offers over the other file sharing systems.

Figure - A view inside our GitHub Repository



When it came to write the final report for the project there was a shared Google Drive folder set up for people to collaborate on the word document, this was set up due to the online editing capabilities and being able to see the changes to the document in real time. As the original document is written in Microsoft Word, this allowed the team member converting the document into Latex, to do so whenever the file was updated, another example of the team working well together.

Another tool used was Facebook Messenger, this was used for asking questions, updating team members of progress, requesting help, etc. A useful tool in Facebook messenger is the polling feature, this was used on multiple occasions, in the example it is being used to decide on a day for a meeting.

Evaluation

* What worked / didn’t work
* Achievement vs plan
* Changes to plan
* Identify weaknesses & strengths
* Potential future work

Peer Assessment

Here is the breakdown of points allocation, agreed by each member of the group:



# References

Bloom, N., Liang, J., Roberts, J., & Ying, Z. J. (2015). DOES WORKING FROM HOME WORK? EVIDENCE FROM A CHINESE EXPERIMENT\*. Retrieved from https://nbloom.people.stanford.edu/sites/g/files/sbiybj4746/f/wfh.pdf

Dropbox. (2016). *Celebrating half a billion users*. Retrieved from Dropbox blog: https://blog.dropbox.com/topics/company/500-million

Dropbox. (2019). *The right Dropbox for you*. Retrieved from Dropbox: https://www.dropbox.com/plans?oqa=acc\_plan\_upgrade

GitHub. (2019, April). *User Search*. Retrieved from GitHub: https://github.com/search?q=type:user&type=Users

GitHub. (2019). *What is my disk quota?* Retrieved from GitHub.Help: https://help.github.com/en/articles/what-is-my-disk-quota

Google. (2019). *Drive Features*. Retrieved from https://www.google.com/drive/using-drive/

Gousios, G., Vasilescu, B., Serebrenik, A., & Zaidman, A. (2014). Lean GHTorrent: GitHub Data on Demand. *MSR 2014 Proceedings of the 11th Working Conference on Mining Software Repositories*, 384 -387.

Microsoft. (2019). *One Drive Plans*. Retrieved from One Drive: https://onedrive.live.com/about/en-gb/plans/

Net Market Share. (2019). *Operating System Market Share*. Retrieved from https://netmarketshare.com

One Drive Community. (2018). *onedrive web always stalls/hangs after uploading few files - hangs on "uploading \*\*\* items" for ever (win 10 chrome and Edge and IE)*. Retrieved from One Drive User Voice: https://onedrive.uservoice.com/forums/913528-onedrive-on-the-web/suggestions/33208225-onedrive-web-always-stalls-hangs-after-uploading-f

Price, D. (2018). *The True Market Shares of Windows vs. Linux Compared*. Retrieved from Make Use Of: https://www.makeuseof.com/tag/linux-market-share/

Smith, C. (2019, March). *20 Amazing GitHub Statistics and Facts 2019 | By the Numbers*. Retrieved from DMR: https://expandedramblings.com/index.php/github-statistics/

Statista. (2018). *Market share of Windows Phone in the United Kingdom (UK) from December 2011 to September 2018*. Retrieved from https://www.statista.com/statistics/271255/windows-phone-market-share-in-the-united-kingdom-uk/

Tech Crunch. (2018). *Google Drive will hit a billion users this week*. Retrieved from https://techcrunch.com/2018/07/25/google-drive-will-hit-a-billion-users-this-week/?guccounter=1&guce\_referrer\_us=aHR0cHM6Ly93d3cudGhldmVyZ2UuY29tLw&guce\_referrer\_cs=7TGZhIYb4figcc6zWPN7Bw

Tutorials Point. (2019). *HTTP Requests*. Retrieved from Tutorials Point: https://www.tutorialspoint.com/http/http\_requests.htm

Warren, T. (2018). *Microsoft completes GitHub acquisition*. Retrieved from The Verge: https://www.msn.com/en-us/news/technology/microsoft-completes-github-acquisition/ar-BBOVVOT

1. Although Linux only has about 2% of consumer market share it is one of the most popular operating systems in the world due to powering Chromebooks, Web Servers, Public Cloud etc (Price, 2018) [↑](#footnote-ref-1)
2. More GitHub facts available (Smith, 2019) [↑](#footnote-ref-2)
3. One example of this (One Drive Community, 2018) [↑](#footnote-ref-3)
4. Information on HTTP requests available from Tutorials Point (Tutorials Point, 2019) [↑](#footnote-ref-4)