Computer Systems 2A AC21009

Peer Assessment:

|  |  |
| --- | --- |
| Name | Amount Contributed |
| Idris Mahamdi | 100% |
| Lucas Khan | 0% |
| Jonathan Mather | 0% |

Starting this project, I was unsure how to start, I knew what a git repository was but was unsure how to tackle this project. I my first thought was linking my project with GitHub as it already has the structure, I started doing this but then came across that features such as making a new repository was only available by using their website and not remotely.

After weighing up pros and cons of using GitHub or using the local machine I came to the conclusion doing the project de-centralised would allow me to implement more features, but it would be harder to implement.

Changing what I had done to a de-centralised project cause me to get off-track, with hours spent confused on how to approach the project. With my group members not responding,

I sat hours a side to research how to combat doing it locally.

It then cognized to me the commands that I could use for remotely for GitHub were similar to what can be used locally.

This allowed me to comfortably implement ‘Create a repository’ and ‘Add/update a repository’ this was the point I decided to allow users to add/update one file to the repository or add/update all files within the repository and also moving files into their repository within the script.

Moving on to ‘Checking in and out a file’ this proved challenging to implement. I started by using the ‘git checkout’ command, I struggled to understand the command, I researched other git command. Git tags got me close, I was aware they weren’t made for this purpose though, thought I could manged doing these, it worked sometimes, it was unstable.

It then sprung to mind that log files could be a solution; At first, I was going to store all the information in one log file this would be inefficient as the script would loop throughout a possible massive file. Instead it was more efficient creating a logfile folder within the repository where files about 24bytes big were stored. This was efficient and reliable.

At this point it was irritating as when testing the script, I sometimes selected the wrong option at menu. It concurred to me the end-user would also get frustrated. Due to this I implemented a back feature, whereat most points within the script the user can type ‘back’ and go back a page. At first a stack structure seemed the direction to go in, but in this case, it was more efficient making everything into a function then calling the function which the user was on beforehand. Another frustration was that the script exited after completing a task, to combat this problem the user would be taking back to the main menu with an exit option at the main menu.

Moving to advance features including backing up, this was more challenging than I thought, needing a place to store backups and unique names for each backup. At first the names of backup repositories included the name of the original repo, finished with a number which was incremented by a log file. This was also stored within the original repository folder. This was messy and inefficient.

Quickly changing to the names of backup folder being named after which included the original repository with the current date and time (with seconds). The backup folder would now held at the directory above.

At first, I was going to automatically backup the repository, I realised this could cause a lot wasted storage, I settled on letting users to manually backup the repository whereas it automatically backed up single files every time they get checked out

**Word Count: 600**