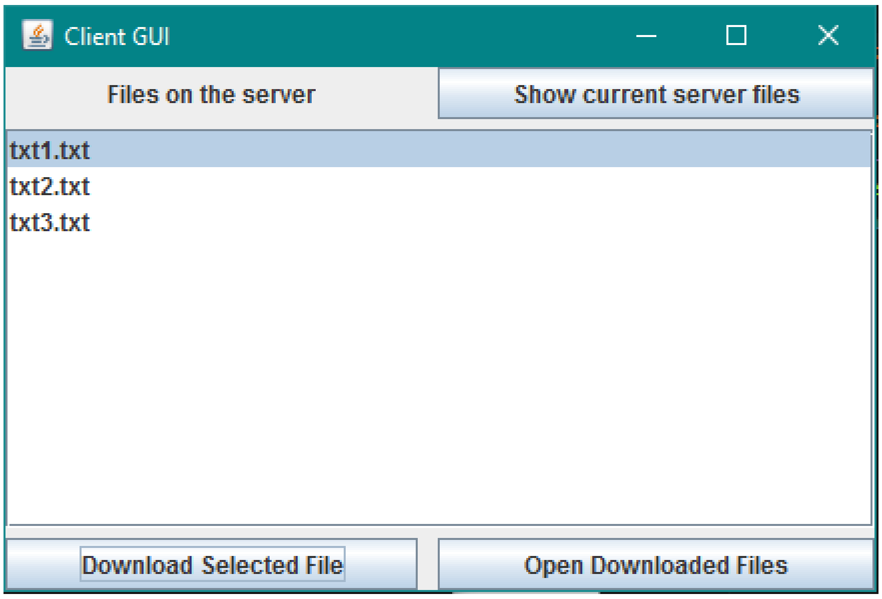
711 Report
Joshua Jones, 6453502

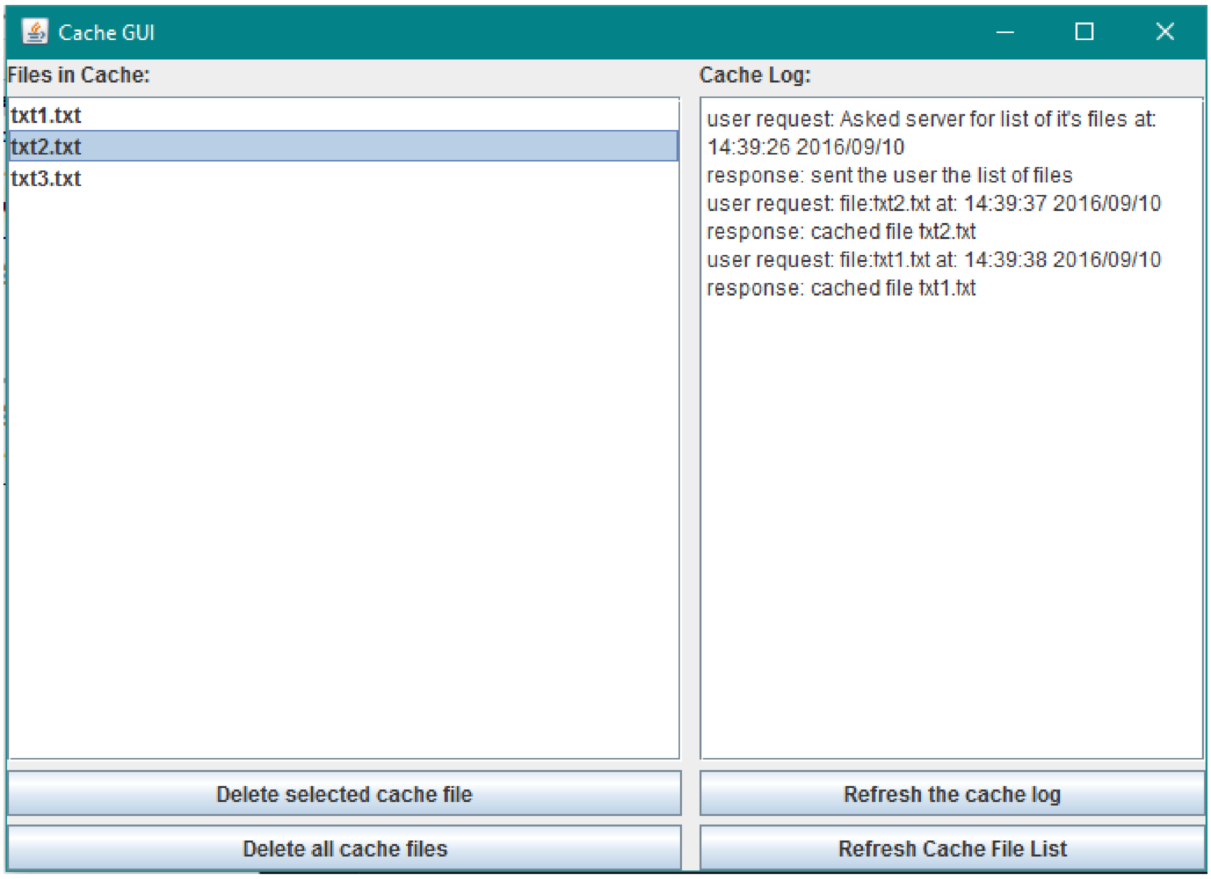
I have tested my programs on the machines at the labs. In order to run my programs, you just need to open the executable jar files. When this happens, each of the client, cache and server will create a folder which will be used to store all of the files for the respective program if the folder does not already exist. Any files which you wish to be placed on the 'server' should be put into the folder ServerFiles which is created when the jar is executed. You need to ensure that each of the jar files has been run before performing any operations in either the client or cache GUIs.

I have included screenshots of the GUI for the client:



Each of the components on the GUI should work as follows:
When you click the show current server files button, it asks the server for which files it current holds and then displays them in the box. Pressing the Download selected file button downloads whichever file is currently selected in the screen, a file is selected by clicking on it. When you click on the Open downloaded files button, it attempts to open the selected file, however if the selected file has not been downloaded then it will display a message saying so.

GUI for the cache:



txt1.txt
txt2.txt
txt3.txt

```
user request: Asked server for list of it's files at:
14:39:26 2016/09/10
response: sent the user the list of files
user request: file:txt2.txt at: 14:39:37 2016/09/10
response: cached file txt2.txt
user request: file:txt1.txt at: 14:39:38 2016/09/10
response: cached file txt1.txt
```

Each of these components work as follows:
The text field on the left displays all of the files which are currently being cached, however this does not live update as new files are cached. In order to update the text field to show all of the currently cached files, press the Refresh cache file list button in the bottom right of the GUI. The text field on the right shows all of the cache logs such as polling the server for which files it currently has, downloading new files/returning cached files and showing any deletions from the cache. This box also does not live update as the logs are created, in order to view new logs, you need to press the Refresh the cache log button in the bottom right of the GUI. In the bottom left of the GUI there are two buttons. The first button, Delete selected cache file, deletes from the cache whichever file is currently selected in the text field. A file is selected by clicking on it in the text field. The second button, Delete all cache files, deletes all files which are contained in the cache and ensures that there are no remaining files being cached until a new download request is sent.

I did not make a GUI for the server application as it was not required in the brief.

For part 2, I did not complete the coding, however I will describe the technique for determining which portions of a file need to be downloaded which I would have used. The technique works by first splitting up a file into small chunks e.g. of size 2kb or so. These chunks are then condensed by encoding the contents with some sort of has function such as an MD5 hash which produces a 128bit digest. The cache would split up the whole file into smaller chunks in this way, and then send the results of the hash to the server. The server can then compare these hashes sent by the cache to the hashes of the file it is currently storing. If it sees that one of the hashes is different, then this means that the chunks which is related to this hash has different contents in the cache than in the server. The server will then find all such chunks which need to be updated in the cache and it will reply by sending only these chunks, thereby saving on bandwidth since only the parts of the file which are changed need to be sent to the cache to be replaced. We can further increase the efficiency of this method by realising that generally changes in a file will be in one or more areas clumped together i.e. a set of consecutive chunks is likely to be affected by a change to a file and not just one. Also when one chunk has had any alterations done to it, all following chunks will have been changed since the existing chunks have been 'shifted along'. Then we can take this into account by using a Rabin function which can be used to partition a file into blocks with dynamic sizes. By doing this, we can make the blocks surrounding the changed part of the file larger so that we can limit the number of blocks which have been changed to only a few around where the changes have taken place. For example, if a few characters have been inserted into a file, then we can increase the size of the block which they are now in so that it also contains the old characters which used to be in the fixed size block, or if this becomes too large then we can also make the following block slightly bigger to contain some of the spill-over bytes. This means we can limit any changes in a file to a small amount of blocks and so we can further minimise the amount of data that we need to send from the server to the cache and also save time.

Using this function would increase the response time that the user experiences in a few situations. One such situation is when the user requests to download a file which has been partially changed since the last time they requested it, this is because the cache will only have to download the parts of the file which have been changed whereas in part 1 the cache would have to download the whole file if the user requested to download the changed version. However, if there have been no changes to the file since it was last requested then there will be no difference between using the technique and not since the cache will just return the same file without requiring any extra data transfer from the server. Also if there have been a lot of changes since the last time the file was requested then there will not be much difference between using the partial file caching technique and not since the cache will still have to update the majority of the file. So the fewer changes that are made, the bigger the time gain we get from using this technique. The bandwidth saving also follows in the same vain since if we can get away with transferring fewer parts of the file then we will save on bandwidth. However, when using this technique, both the cache and the server will carry out a lot more computation since they both have to work out which parts of the file are new and need to be transferred from the server to the cache. This will also take up some time since computation can take a while, however the time saved through transferring less data will make up for the time taken to do the calculations and so overall there will be a benefit.