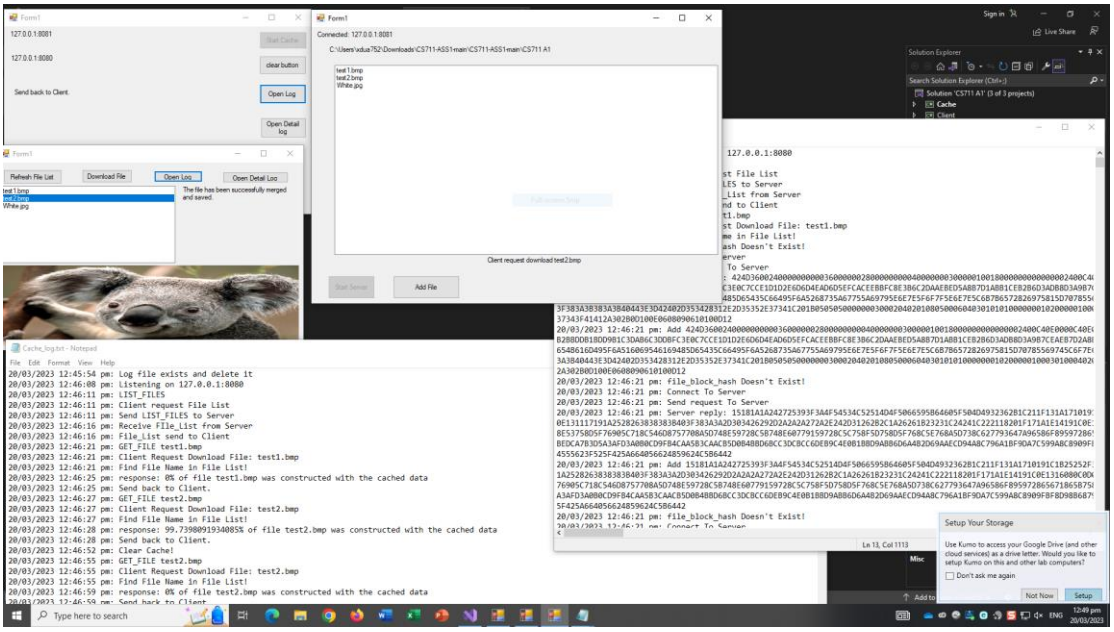
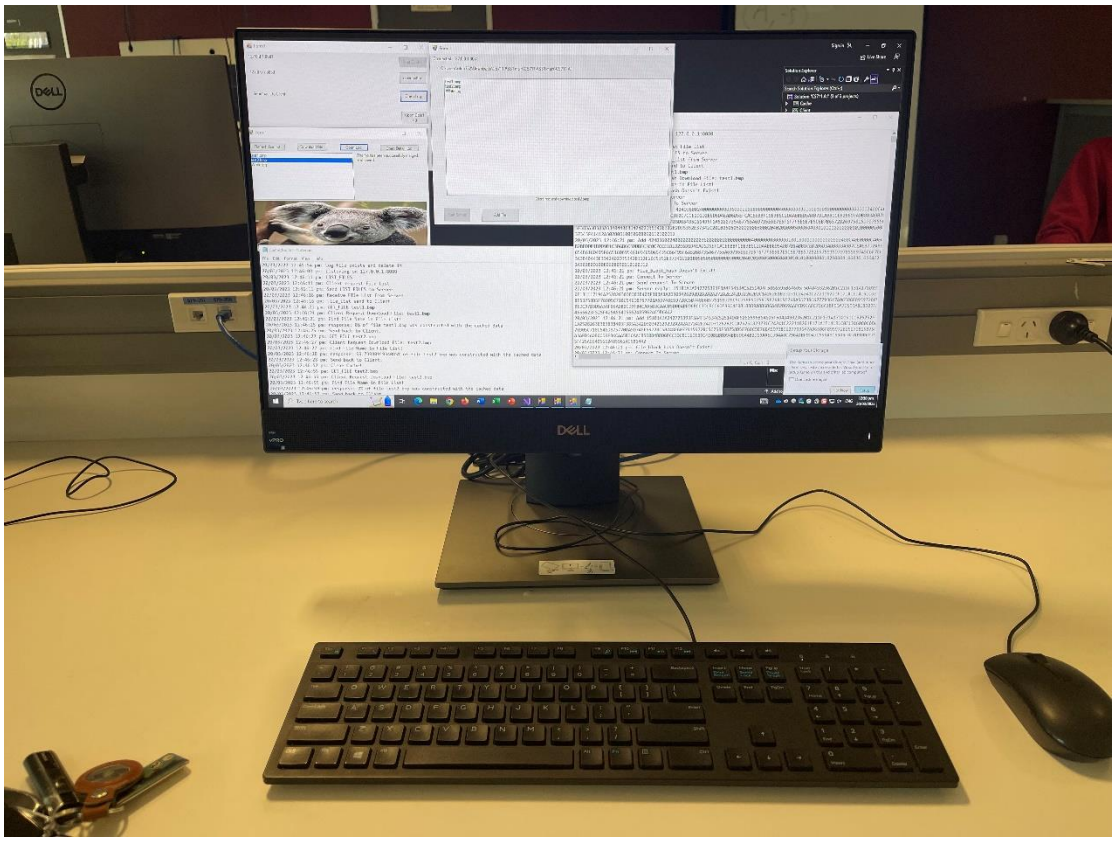
COMPSCI 711 Ass1 Report

1. **Which option you have implemented.**
   Option 2
2. **Clearly indicate whether you have tested your programs on a lab machine.**
   I went to the Compsci Lab in the 303 building for testing, and everything is normal.





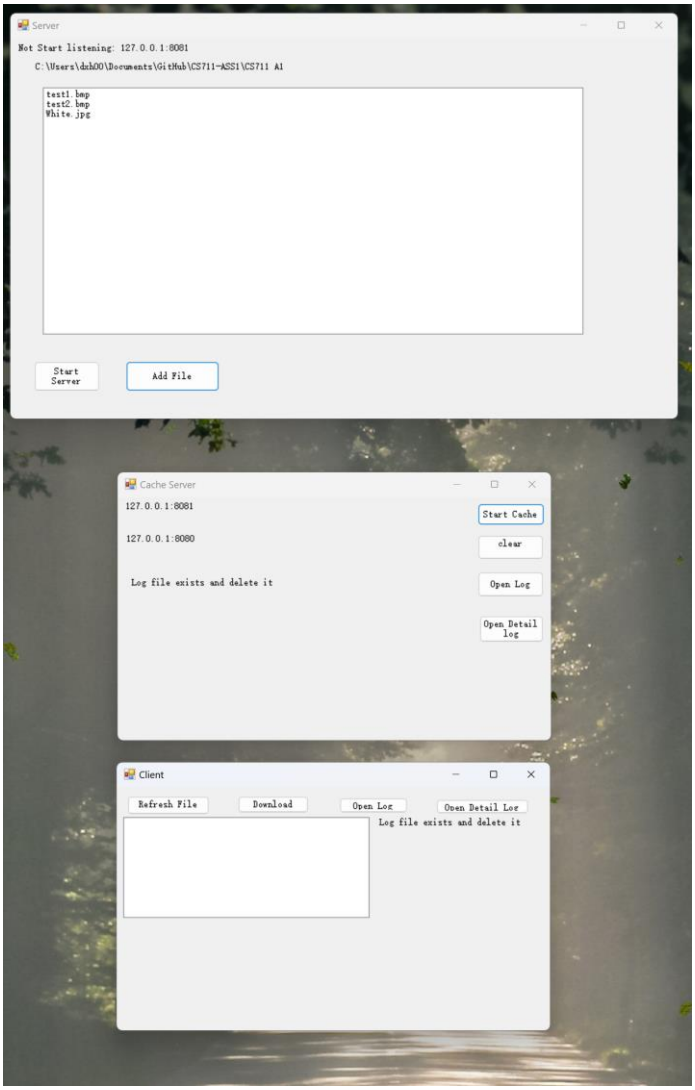
3. Instructions on how to run your programs.

I used Rider to create a solution that includes three projects: Server, Cache, and Client. However, it seems that Visual Studio cannot run all three projects at once.
So there are Three ways to run them.

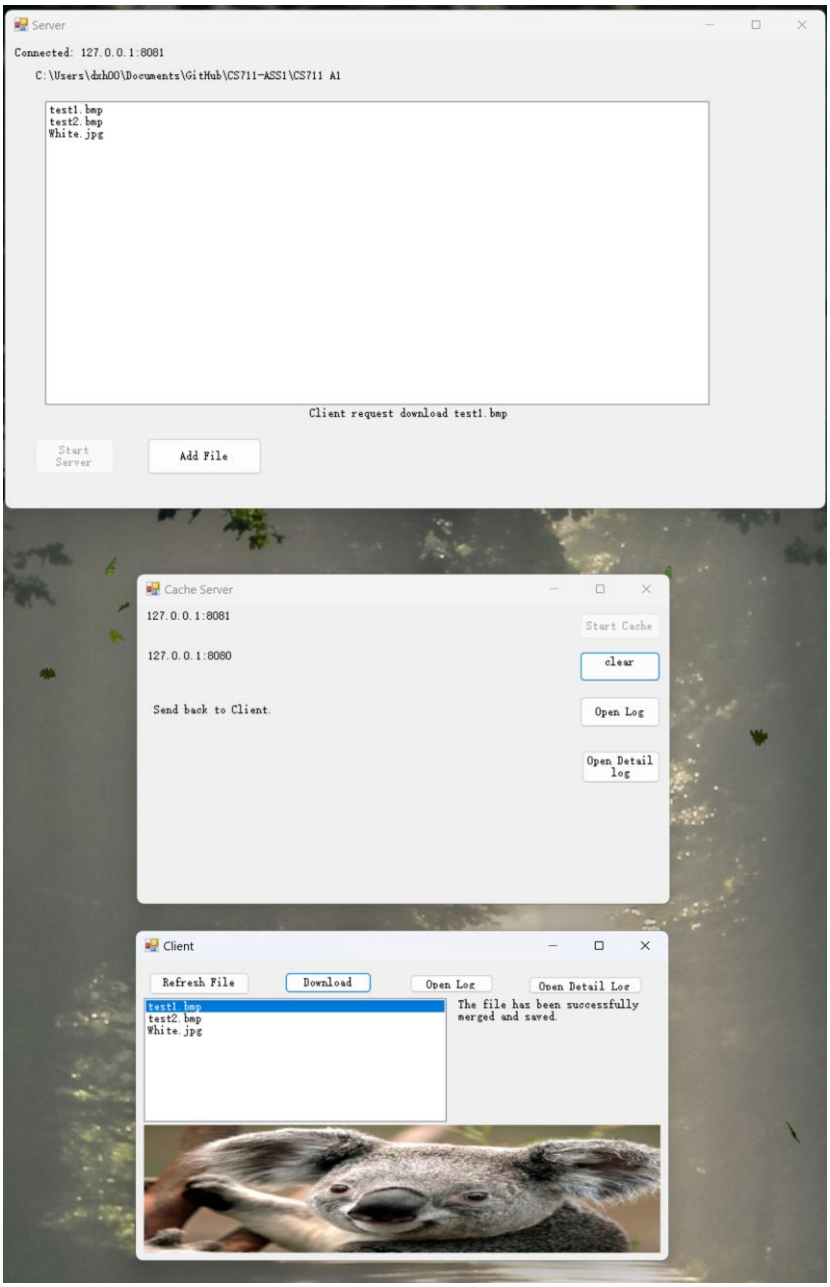
The First way is Double-click Run.bat

The second way is to download Rider and open the CS711 A1.sln file in the solution directory, then use Rider to run Server, Cache, and Client separately.

The Third way is to open the solution directory and separately open .\Cache\bin\Debug\Cache.exe, .\Client\bin\Debug\Client.exe, and .\Server\bin\Debug\Server.exe. This method does not require installing Rider, but if you need to view my code, you may also need to open Visual Studio.



To run all three apps, you need to click 'Start Server' on the Server app, 'Start Cache' on the Cache app, and finally 'Refresh File' on the Client app. Then, available files will appear on the Client app. You can select a file and click 'download' to begin the download process.



Both the Client and Cache apps have two buttons for opening stored log files. 'Open Log' opens a log file that records basic information, while 'Open Detail Log' opens a more detailed log file that includes the hash value and 16-digit hexadecimal string of the file blocks. The Cache app also has a 'Clear' button for clearing the cache.

To view the downloaded files, you can click on the image in the Client app, and the files will be stored in the Solution directory \Client\bin\Debug.

4. **Describe the techniques that you use to determine which portions of a file need to be downloaded from the server.**

I use a method to calculate the hash value of each file block and compare it with the cache to determine which file blocks need to be download. When the Client initially requests the File List, the Server divides each file into many small blocks use Rabin_Karp,

The file block average size is 2048, calculates the hash value of each block, and sends the hash values, file names, and the starting index value of each block to the Cache. The Cache stores this information as a value named 'deserializedListOfDictionaries' and forwards it to the Client, which then lists the available files.

When a user downloads a file, the Cache checks all of the file blocks' hash values in the 'deserializedListOfDictionaries' have already been stored in the cache. If so, the Cache adds the hexadecimal string of that file block to the list that needs to be sent to the Client. If not, the Cache downloads the missing file blocks from the Server based on the file name and starting index value, saves them in the cache, and then adds them to the list that needs to be sent to the Client. The entire process is asynchronous, so there are no errors in the file block positions. Once all the file blocks are added to the list, the Cache sends them to the Client, which parses and combines them into a single file.

To view the downloaded files, you can click on the image in the Client app, and the files will be stored in the Solution directory \Client\bin\Debug.

5. **Consider the techniques that you need to use if you have chosen to implement option 1. Compare and contrast the techniques used in option 1 and 2 in terms of the response time perceived by the user, the amount of network bandwidth that can be saved, and the amount of computation being carried out by the cache and the origin server.**

   If the file requested by the Client contains file blocks that are already stored in the Cache, Option 2 will download the file faster than Option 1, saving network bandwidth and reducing the computational load on the original server. However, it will increase the computational load on the cache server. If the file blocks are not in the Cache, Option 2 may be slower than Option 1 because it needs to retrieve all the file blocks from the Cache. This will increase network bandwidth usage, as well as the load on both the cache and original servers.