Problem Statement

With web services or distributed objects **in any programming language you like** you should develop a client-server application implementing a flexible text file chunking and generation system with the following specification:

1. The client and server must be capable of running on 2 distinct machines connected by a network (the server can itself be distributed, there can be many clients).
2. The server should be able to split up and create metadata (statistics) about text files in the word list format of the example supplied on loop[1] (line number, word, word, word count). For example

```
10152    anxious    anxious    168
10153    bacon    bacon    168
10154    collaborative    collaborative    168
10155    columns    columns    168
10156    consumed    consumed    168
```

3. The server should provide the following functionality to a client:
   - The ability to specify a chunk of analysis for the file based on grouping/filtering words by their first letter(s) e.g. chunk 1 is all words beginning a-c, chunk 2 is all words d-z.
   - (Metadata) Provide a list of the chunks in the file, the number of words in each chunk, the total count of all works in that chunk, the count of the number of occurrences of all letters in that chunk.
   - Generate an output file based on the chunk specification that has the same words as the chunk specification randomly repeated the number of times shown the word count. E.g. a chunk based on letters a-b in the example above would generate a file with the word anxious 168 times and the word bacon 168 times.
   - Estimate the time to generate an output file under current server load.
   - Manage a queue of output files to be generated.
   - Ability to download an output file.
4. The server should be able to process requests from many clients concurrently and they should all be able to see the same queue of chunk file creation requests.
5. The server should protect itself against filling its own disk and be able to simulate what happens when this occurs.
6. The client should be able to:
   - Display the queue of chunks being created.
   - Cancel the generation of an output file.
   - Configure a filter for a chunk specification and display the metadata about it.
   - Upload a new text file to the server.
   - Download an output file for local storage.
   - Be able to run in a test mode where many client requests are automatically generated with random delays between them.
   - **Deal with an unreliable server or network**.

Hint: you could use RMI to implement a basic file chunking service.

[1] Further files available here https://wortschatz.uni-leipzig.de/en/download/

Instructions & Deadline:

The assignment is a two person project, worth 15% of the overall module mark. The choice of implementation languages is up to you. Java will always be acceptable. **You may not use a database in your design**. If you use a distributed applications framework to provide fault tolerance, distribution and load balancing then you will be expected to have a much higher degree of reliability and performance evidence in your report.

In addition to building the system it is important that you profile it for scalability and performance – how well does it deal with larger files? How about more clients?

A project demo video (4 mins max) will be required and a random number of students will have group interviews via zoom.

Please also write a short (5 page max) report with:

- Design documentation which will get you some partial credit. Make sure to describe your concurrency strategies and failure handling.
- A plot/figure and a description of a simple concurrency or scalability study you did that shows how the system performs with many clients/requests (2 pages max).

You have to submit all your source files and documentation as well as any details needed for compiling or running it. Some students may be asked to build their code over zoom and talk through the code and design.

Please submit the above via Loop with a completed plagiarism declaration form which is signed by both of you by Monday 13th April 2020.

A Word on Continuous Assessment

A resit is available for all components of this module.

Students who fail the module on the first sitting must resit each component (CA or exam) that they failed. In particular, students who fail the CA must resit the CA.

A student may not resit a component that has been passed and their marks for the component are carried forward.

Marks for failed components will be set to zero in ITS before the resit.

Good Luck!