Declaration on Plagiarism

Assignment Submission Form

This form must be filled in and completed by the student(s) submitting an assignment

Name(s): Matthew Nolan, Michael O'Hara
Programme: CASE 4
Module Code: CA4006
Assignment Title: Concurrency Assignment 2
Submission Date: 13/04/2020
Module Coordinator: Rob Brennan

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

I/We have read and understood the referencing guidelines found at

http://www.dcu.ie/info/regulations/plagiarism.shtml ,

https://www4.dcu.ie/students/az/plagiarism and/or recommended in the assignment guidelines.

Name(s): Michael O'Hara, Matthew Nolan

Date: 13/04/2020

Installation and User Guide:

In order to be able to run our code you need to have python installed on your machine and you would have to install flask and requests. These can be installed using “pip install flask” and “pip install requests” respectively.

The next thing you need to do is make sure all the paths in the code (specific paths mentioned further on) are all pointing at directories on your local machine.

Then depending on which mode you want to run the application you must call the following commands:

Locally:

“python flaskserver.py” in the directory where the files are. This will start the flask server and wait for clients to connect.

“python client.py” needs to be run in a separate command line window and this will start the client and try to connect to the localhost on your machine.

Once the client is running and connected to the server, you will then be prompted with commands to enter to do specific requests

PythonAnywhere:

To connect to python anywhere make sure this line in client.py is uncommented:

Design Documentation:

For our design we decided the programming language we would use would be python. We decided to use this language as we are both most comfortable with since we have been using it since 1st year. We decided that we would implement a REST architectural design using the micro web framework, flask. This would act as a server for our application. We would host the application on Python Anywhere which would allow us to run the server on a dedicated machine.We decided on flask as from version 1.0 onwards Flask was designed to be able to handle multiple requests. This would allow for multiple clients to connect at the same time.

We had number of classes in our set up:

- Flaskserver.py
- ChunkGenerator.py
- Client.py

Server File:

The flask server file is responsible for starting the flask server which allows clients to connect to it. It is also responsible for trying to take in the files that a client wishes to upload and saving them which in turn makes them available for the client to request chunks of metadata from.

Client File:

The client file is responsible for connecting to the server and letting the clients make requests. We decided that the client should be able to see a list of files that have been uploaded to the server and then using that list they can decide which file they would like to get a chunk of metadata from. It also allows the client to upload a text file of their choosing in order to get chunks of metadata from as well.

Chunk Generator File:

This file is responsible for creating the chunks of metadata requested by the clients. This allows the users to request a few different types of chunk. As it was not directly stated in the assignment spec *we made the assumption* that the client may want to generate a random chunk from the file the uploaded based on no other previous specifications, therefore being completely random. This file also handles the other chunking options such as chunking by letter. The file is also ultimately responsible for generating the output file after completing the chunking of the source text file.

Different modes of running the client and server:

We also included an option in our application set the destination for where it will be run to either be locally on your own machine via multiple command line windows, with the changing of a few settings within the following files:

Flaskserver.py = GENERATE_FOLDER, UPLOAD_FOLDER

ChunkGenerator.py = GENERATE_FOLDER, UPLOAD_FOLDER

Client.py = flaskUrl , download_path

These need to be changed in order to match the paths on your local machine.

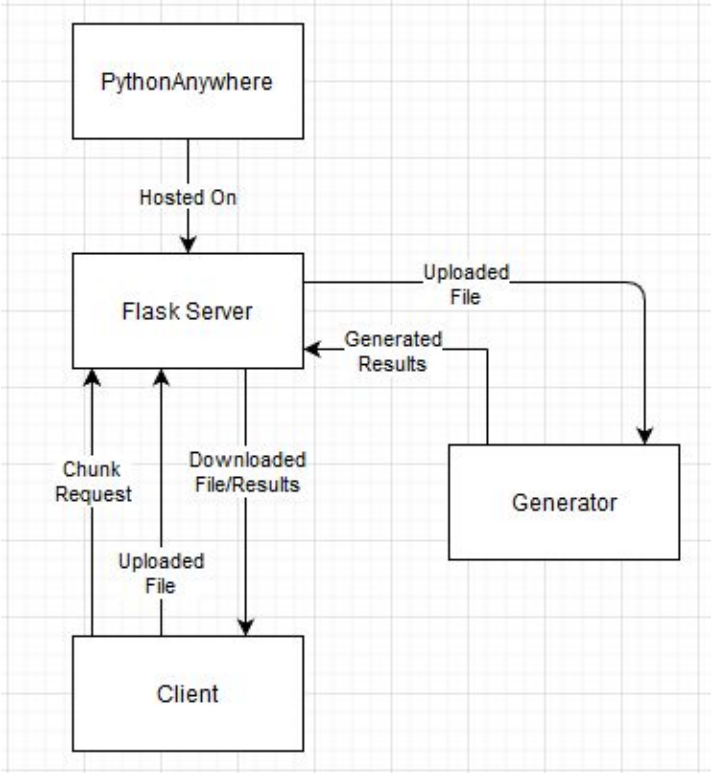
Note: default.txt(file from loop) is needed in upload folder

The other option for running our application is to allow the server to connect to python anywhere which is the website which we are using to host the application which allows for multiple clients to connect at once while all being run from different machines.

We felt that having two separate running modes for the application was best as it allows for more flexibility in the scenario of running it. If the client does not have access to the internet the server can be run locally and multiple instances of the client window can be opened to simulate many client connections. But if the client can connect to the internet then it can

connect to the pythonanywhere.com server and the many clients can connect that and use it that way also.

UML :



Dependencies:

PythonAnywhere: This where the server is hosted and allows us to keep it running. Only the server will interact with this.

FlaskServer: This interacts with the client and the generator.The server takes in a request from the client, it can also receive a file that the client uploads. The server then passes the file that the client uploads to the generator and then gets the results of the request back .

Generator:

Generator gets passed the uploaded file from the server and performs the desired request. The request may be to create random chunks of data from the default file on the server or chunks of data from a file which was uploaded by the client. It will then return the results that it has generated to the server to be passed to the client.

Client:

The client interacts with the server. The client is able to upload new files to the server and have the server save them. The client can also make requests for chunks to be made from the files which have already been uploaded to the server.