Python Code Documentation

Abstraction :

Client and generator :

import socket

import tqdm

import os

#Device's ip address

SERVER\_HOST = "192.168.1.159"

SERVER\_PORT = 5001

#Recieve x-amount bytes each time

BUFFER\_SIZE = 2404404

SEPERATOR="<SEPERATOR>"

#create the server socket

#TCP

s = socket.socket()

#bind the socket to our local adress

s.bind((SERVER\_HOST, SERVER\_PORT))

#enabling our servet to accept connections

#5 here is the number of unaccepted connections that the system will allow before refusin new connections

s.listen(5)

print(f"[\*] Listening as {SERVER\_HOST}: {SERVER\_PORT}")

#accept connection fi there is any

client\_socket, address = s.accept()

#if below code is executed, that means the sender is connected

print(f"[+] {address} is connected.")

#recieve the file infos

#recieve using client socket, not server socket

received = client\_socket.recv(BUFFER\_SIZE).decode()

filename, filesize = received.split(SEPERATOR)

#remove absolute path if there is

filename = os.path.basename(filename)

#convert to integer

filesize = int(filesize)

#start receiving the file from the socket and writing to the file stream

progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B", unit\_scale=True, unit\_divisor=1024)

with open(filename, "wb") as f:

while True:

#read 1024 bytes from the socket (recieve)

bytes\_read = client\_socket.recv(BUFFER\_SIZE)

if not bytes\_read:

# nothing is received file transmitting is done

break

#write to the file the bytes we just recieved

f.write(bytes\_read)

#update the progress bar

progress.update(len(bytes\_read))

#close the client socket

client\_socket.close()

#close the server socet

s.close()

Client Code :

#importing librarieis

import tqdm

import os

import psycopg2 as db

from faker import Faker

import socket

#Generating fake data for the data base

fake = Faker()

data1 = []

for r in range(5000):

data1.append()

data\_for\_db=tuple(data1)

#Making database connection

conn = db.connect

cur=conn.cursor()

query = "insert into users () Values()"

cur.executemany(query,data\_for\_db)

conn.commit()

with open('DataBase.csv', 'w') as file:

cur.copy\_to(file,'users', sep = ',')

conn.close()

#Raspberry PI connection

SEPARATOR = "<SEPERATOR>"

BUFFER\_SIZE = 2404404

#host = '172,18,3,176'

host = '192.168.1.159'

port = 5001

filename = "DataBase.csv"

filesize = os.path.getsize(filename)

s = socket.socket()

print(f"[+] Connecting to {host}:{port}")

s.connect((host, port))

print("[+] Connected.")

s.send(f"{filename}{SEPARATOR}{filesize}".encode())

progress = tqdm.tqdm(range(filesize), f"Sending {filename}", unit="B", unit\_scale=True, unit\_divisor=1024)

with open(filename, "rb") as f:

while True:

bytes\_read = f.read(BUFFER\_SIZE)

if not bytes\_read:

break

s.sendall(bytes\_read)

progress.update(len(bytes\_read))

s.close

Introduction :

Hello there hope you are having the best of times while reading in this documentation we will talk about our server client code that includes data base and data sending from the client to the server.

Problem : the problem here is the need for backup data.

Solution : creating a server and client to send data from the client to the server in case of an emergency to save it all.

Methods :

Firstly we started by importing the need libraries for this project and the libraries are the following : tqdm, os, psycopg2, faker, csv, and socket. Each one of them has a job the tqdm to give a loading bar for the sending , the os library creates and removes files from directories, the psycopg2 is used to connect to the database, the faker library to generate fake data, and finally the csv library to save the files as csv

Secondly we started generating the fake data we needed using faker to fill in the data base we created before connecting it to pg admin and to save all the data to a csv file, here we used the following libraries : psycopg2, faker, and csv.

After that we created the connection between the server and the client the client being being the user that created the data for pgAdmin and saved on csv file to send it later to the server so it can be stored and saved as backup in any case of emergency that could occur, here we used the following libraries : tdqm, os, socket.

Finally we tried using apache nifi to query the csv file we created.

Results:

Sadly the code is not 100% finished there is still a slight problem with the file sending for some reason that we are trying to resolve as the time of typing this documentation, but the good news is that we can connect the server and the client and communicate our problem falls on finding the appropriate port and ip for sending a receiving files it is possible through the tcp ones but we couldn’t really find one while searching, also sadly we didn’t have enough time to perfect the apache nifi process although we really hoped to do so but we were accompanied by an immense amount of errors that accrued in almost every step and took a huge portion of our time to fix but at the end code is partially still running at least and the nifi pipe line is there without anny apparant errors other than the csv file not giving a header which would not make our query and put file statement work.

Future work :

Our future plan consists of fixing the nifi and fixing the sending data problems so the project would be ready to run in the real world.

Thank you sincerely for your time. Hopefully we were up to standard in the work we were able to do in our time period , our best regards.

Documentation done by :

Tareq Aburajab , 2110025

Ali otaibi, 2110053