ECE 5650: Project 2

****

Colin Phlypo

Robert Kruzel

October 17, 2017

**Introduction**

The purpose of this project was to use low level socket programming for a client and a server program to interface with each other with the goal of creating a database automatically from a user inputted text file. The user calls several commands and is met with errors when commands are entered in correctly. There is also verbose mode to follow along with what the program is doing in the background. Both programs are run using the command prompt.

**Source Code**

Refer to .py attachment or code below.

# ECE 5650: Project 2

# Names: Robert Kruzel, Colin Phlypo

# Program is interpreted correctly.

# Program does not give correct results.

import socket

import sys

class Node: # Customer Record Class

def \_\_init\_\_(self, customerID, customerFirstName, customerLastName, customerPhone):

self.id = customerID

self.first = customerFirstName

self.last = customerLastName

self.phone = customerPhone

def display(self):

# display record

return 'Customer record: ' + str(self.id) + ' ; ' + self.first + ' ; ' + self.last + ' ; ' + self.phone

class customerDB: # Customer Database Class

def \_\_init\_\_(self):

self.db = []

def display(self): # display all records

allRecords = ''

for record in self.db:

allRecords = allRecords + record.display() + '\n'

if (allRecords == ''):

return 'Database is empty!'

else:

return allRecords

def insert(self, record): # insert a record

self.db.append(record)

return 'Operation was completed successfully.'

def remove(self, customerID): # remove a record

found = False

for record in self.db:

if record.id == customerID:

self.db.remove(record)

found = True

break

if found:

return 'Operation was completed successfully.'

else:

return 'ERROR: No match was found!'

def search(self, last): # search for all records with the specified last name

found = False

matchingRecords = ''

for record in self.db:

if (record.last == last):

matchingRecords = matchingRecords + record.display()

if (matchingRecords == ''):

return 'ERROR: No match was found!'

else:

return matchingRecords

def show(self, customerID): # show record with specified ID

found = False

for record in self.db:

if (record.id == customerID):

return record.display()

found = True

break

if not found:

return 'ERROR: No match was found!'

nextCustomerID = 0

cDB = customerDB() # create customer database

HOST = "192.168.1.106"

PORT = 8889

servSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

servSocket.bind((HOST, PORT))

servSocket.listen(5) # change this based on file size??? ref. instructions

serverRunning = True

while serverRunning:

connection, address = servSocket.accept() # blocks here til connection is made

clientActive = True # inner loop exit condition

verboseMode = False # flag for verbose

while clientActive:

mOutSucc = "Operation was completed successfully."

mOutErr = "ERROR: The operation is not supported!"

message = connection.recv(1024)

command = message.split()

if command[0] != "--verbose command-line":

if command[0] == "load":

fileName = command[1]

extDB = open(fileName, "r") # opens external file to edit DB

nextCustomerID = 0 # resets cDB

for line in extDB: # for entire file

lineSplit = line.split() # split the data

nextCustomerID = nextCustomerID + 1 # first spot

record = Node(nextCustomerID, lineSplit[1], lineSplit[2], lineSplit[3]) # inserts data

mOut = customerDB.insert(cDB, record)

connection.send(mOut)

extDB.close()

elif command[0] == "download":

fileName = command[1]

extDB = open("fileName", "r+") # open for read/write

tempextDB = extDB.read()

while tempextDB:

conn.send(tempextDB) # send to client

tempextDB = extDB.read()

extDB.close()

elif command[0] == "change":

fileName = command[1]

newLine = command[2] + command[3] + command[4] + command[5] # combining string for new line

extDB = open("fileName", "r")

oldLine = extDB.readline(int(command[2])) # finding what content to replace

with open("fileName", "w") as newextDB: # opens new file to write to

with open("fileName") as oldextDB: # opens old to read from

for line in oldextDB:

newextDB.write(line.replace(oldLine, newLine)) # replaces desired line

# recalling load command to enter new data

nextCustomerID = 0 # resets cDB

for line in extDB: # for entire file

lineSplit = line.split() # split the data

nextCustomerID = nextCustomerID + 1 # first spot

record = Node(nextCustomerID, lineSplit[1], lineSplit[2], lineSplit[3]) # inserts data

mOut = customerDB.insert(cDB, record)

connection.send(mOut)

extDB.close()

elif command[0] == "display":

mOut = customerDB.display(cDB)

connection.send(mOut)

elif command[0] == "show":

try:

showById = int(command[1])

mOut = customerDB.show(cDB, int(command[1]))

except ValueError:

mOut = "Invalid value. customerID must only contain numbers"

connection.send(mOut)

elif command[0] == "search":

mOut = customerDB.search(cDB, str(command[1]))

connection.send(mOut)

elif command[0] == "insert":

nextCustomerID = nextCustomerID + 1

record = Node(nextCustomerID, command[1], command[2], command[3])

mOut = customerDB.insert(cDB, record)

connection.send(mOut)

elif command[0] == "remove":

try:

removeById = int(command[1])

mOut = customerDB.remove(cDB, removeById)

except ValueError:

mOut = "Invalid value. customerID must only contain numbers"

connection.send(mOut)

elif command[0] == "exit":

mOut = "Connection Closed"

connection.send(mOut)

clientActive = False

serverRunning = False

else:

connection.send(mOutErr)

servSocket.close() # end of program

**client**

import socket

HOST = "96.27.14.81"

PORT = 8889

clientSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

clientSocket.connect((HOST, PORT))

while True:

inputFromKeyboard = raw\_input('Enter command: ') # get command from user

clientSocket.send(inputFromKeyboard) # send command to server

splitInput = inputFromKeyboard.split()

if (inputFromKeyboard == 'exit'): # If input from user is 'exit', close connection

break

elif (splitInput[0] == 'download'):

tempOldFile = splitInput[1]

with open('tempOldFile' , 'w') as oldFile:

newFile = clientSocket.recv()

if not newFile:

break

oldFile.write(newFile)

oldFile.close()

# print 'waiting for response from server...'

receivedMessage = clientSocket.recv(1024) # Get reply from server

# print 'server response received: '

print receivedMessage # Print the reply on the screen

clientSocket.close()

**Testing Procedure**

The testing procedure for this project involved opening two command windows, one for the server and one for the client. Refer to screen shots below for how our program behaved.

**Conclusion**

This project taught a great deal about having a thoughtful approach to solving the issues at hand. The project taught how to interface between the client and server programs in a challenging way.