Name: بانه بلال ابراهيم , Number: 2826 , Submitted To GitHub:_____

Second Network Programming Homework

Question 1:

Bank ATM Application with TCP Server/Client and Multi-threading Project Description: Build a TCP server and client Bank ATM application using Python. The server should handle multiple client connections simultaneously using multi-threading. The application should allow clients to connect, perform banking operations (such as check balance, deposit, and withdraw), and receive their updated account status upon completion.

Requirements:

A. The server should be able to handle multiple client connections concurrently.

B. The server should maintain a set of pre-defined bank accounts with balances.

C. Each client should connect to the server and authenticate with their account details.

D. Clients should be able to perform banking operations: check balance, deposit money, and withdraw money.

E. The server should keep track of the account balances for each client.

F. At the end of the session, the server should send the final account balance to each client.

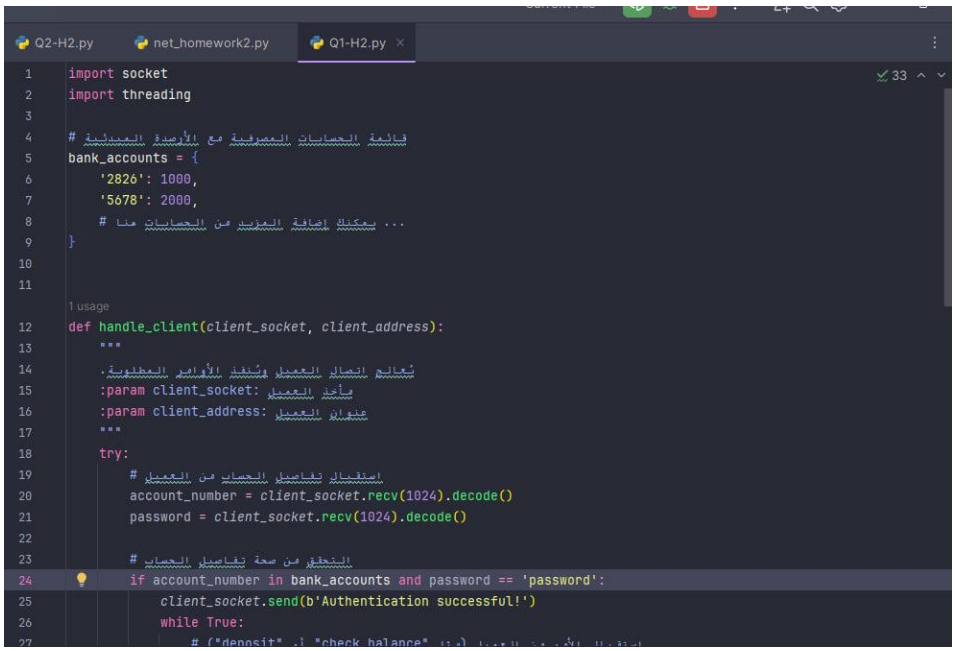
Guidelines: • Use Python's socket module without third-party packages. • Implement multi-threading to handle multiple client connections concurrently. • Store the account details and balances on the server side.

Notes: • Write a brief report describing the design choices you made and any challenges faced during implementation. • You can choose to create a TCP Server/Client Bank ATM application or any other appropriate application that fulfills all requirements.

الحل:

هذا الكود يُعالج اتصالات العملاء مع الخادم ويُنفذ الأوامر المصرفية المطلوبة

Commented [BI1]:



```python
import socket
import threading

# قائمة الحسابات المصرفية مع الأرصدة المبدئية
bank_accounts = {
    '2826': 1000,
    '5678': 2000,
    # يمكنك إضافة المزيد من الحسابات هنا ...
}


1 usage
def handle_client(client_socket, client_address):
    """
    .تُعالج اتصال العميل وتُنفذ الأوامر المطلوبة
    :param client_socket: مأخذ العميل
    :param client_address: عنوان العميل
    """
    try:
        # استقبال تفاصيل الحساب من العميل
        account_number = client_socket.recv(1024).decode()
        password = client_socket.recv(1024).decode()

        # التحقق من صحة تفاصيل الحساب
        if account_number in bank_accounts and password == 'password':
            client_socket.send(b'Authentication successful!')
            while True:
                # ("deposit" أو "check_balance" مثل) استقبال الأمر من العميل
                command = client_socket.recv(1024).decode()
                if command == 'check_balance':
                    balance = bank_accounts[account_number]
                    client_socket.send(f'Your balance: {balance}'.encode())
                elif command == 'deposit':
                    amount = float(client_socket.recv(1024).decode())
                    bank_accounts[account_number] += amount
                    client_socket.send(b'Deposit successful!')
                # يمكنك إضافة المزيد من الأوامر هنا ...
        else:
            client_socket.send(b'Authentication failed!')
    except Exception as e:
        print(f"Error handling client {client_address}: {e}")
    finally:
        client_socket.close()
```

```python
1 usage
def main():
    server_ip = '127.0.0.1'
    server_port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_ip, server_port))
    server_socket.listen(5)
    print(f"Listening on {server_ip}:{server_port}...")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Accepted connection from {client_address}")
        client_thread = threading.Thread(target=handle_client, args=(client_socket, client_address))
        client_thread.start()


if __name__ == "__main__":
    main()
```

خطوات عمل الكود:

1 .تعريف قائمة الحسابات المصرفية:

- تم تعريف قائمة bank_accounts التي تحتوي على أرقام حسابات مصرفية مع الأرصدة المبدئية لكل حساب.

2. دالة:`handle_client`

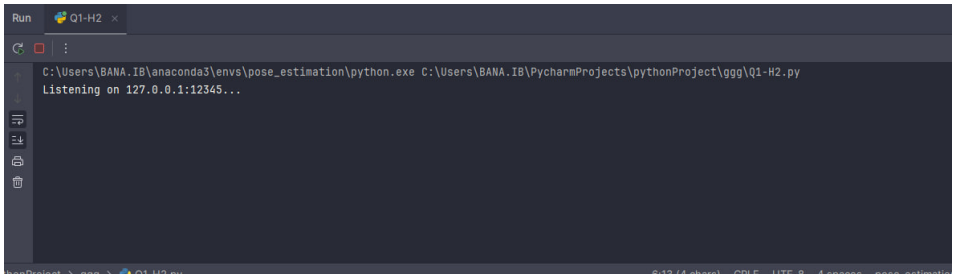
- هذه الدالة تُعالج اتصال العميل وتنفذ الأوامر المطلوبة.
- تستقبل مأخذ العميل (client_socket) وعنوان العميل.(client_address)
- تبدأ بقراءة تفاصيل الحساب من العميل (رقم الحساب وكلمة المرور).
- تقوم بالتحقق من صحة تفاصيل الحساب:
- إذا كان رقم الحساب موجودًا في قائمة bank_accounts وكانت كلمة المرور هي"password" ، يتم إرسال رسالة "Authentication successful!" إلى العميل.
- إلا إذا فشل التحقق، يتم إرسال رسالة."Authentication failed!"
- بعد التحقق، يستمر في استقبال الأوامر من العميل (مثل "check_balance" أو "deposit") وينفذها.

- إذا كانت الأمر"check_balance" ، يُظهر رصيد الحساب للعميل.
- إذا كانت الأمر"deposit" ، يتم إيداع مبلغ معين في الحساب.
- في حالة حدوث أي استثناء، يتم طباعة رسالة خطأ.

3.الدالة الرئيسية:`main`

- تعيد تعريف معلومات الخادم (عنوان IP ومنفذ الاستماع).
- تقوم بإنشاء مأخذ الخادم وتبدأ في الاستماع للاتصالات الواردة.
- عند قبول اتصال من عميل، يتم إنشاء موضوع جديد لمعالجة العميل باستخدام دالة.handle_client

ويكون الخرج عند التنفيذ:



Question 2:

Simple Website Project with Python Flask Framework (you have choice to use Django or any Other Deferent Useful Python Project “from provide Project Links”)

Create a simple website with multiple pages using Flask, HTML, CSS, and Bootstrap. The website should demonstrate your understanding of web design principles .

Requirements :

G. Set up a local web server using XAMPP, IIS, or Python's built-in server (using Flask) .

H. Apply CSS and Bootstrap to style the website and make it visually appealing .

I. Ensure that the website is responsive and displays correctly on different screen sizes

J. Implement basic server-side functionality using Flask to handle website features

بالاستفادة من الرابط التالي "المرفق بالسؤال"

Python Projects – Beginner to Advanced

https://www.geeksforgeeks.org/python-projects-beginner-to-advanced/

Program to extract frames using OpenCV:

الكود التالي يقوم باستخراج الإطارات من مقطع فيديو باستخدام مكتبة OpenCV في لغة Python :

Q2-H2.py | net_homework2.py | Q1-H2.py

```
import cv2

1 usage
def FrameCapture(path):
    vidObj = cv2.VideoCapture(path)
    count = 0

    while True:
        success, image = vidObj.read()
        if not success:
            break

        cv2.imwrite("frame%d.jpg" % count, image)
        count += 1

        print('تم قراءة إطار جديد:', success)

    vidObj.release()

if __name__ == '__main__':
    video_path = "C:\\Users\\BANA.IB\\Desktop\\Bana Documents\\Short-Video-Maker_Lead.mp4"
    FrameCapture(video_path)
```

خطوات الكود:

1 . قراءة الفيديو:

- يتم استخدام الدالة cv2.VideoCapture() لفتح ملف الفيديو. قمت تمرير مسار الملف كوسيط لهذه الدالة.

- في الكود الخاص بي تم استخدام المسار "C:\\Users\\BANA.IB\\Desktop\\Bana Documents\\your_video.mp4" كمثال. (ملاحظة: يجب أن تستبدل "your_video.mp4" بالمسار الفعلي لملف الفيديو الذي ترغب في استخدامه.)

2 . استخراج الإطارات:

- يتم استخدام حلقة while لقراءة الإطارات من الفيديو.
- الدالة ()captured.read تُقرأ إطارًا من الفيديو في كل تكرار للحلقة. إذا تمت قراءة الإطار بنجاح، يكون ret قيمة True.
- يتم حفظ الإطارات كملفات صور باستخدام.()cv2.imwrite

وتكون نتيجة تنفيذ الكود: