الجمهورية العربية السورية

Syrian Arab Republic

اللاذقية-جامعـــة تشريـــن

Lattakia - Tishreen University

كلية الهندسة الكهربائية والميكانيكية قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة برمجة شبكات

Name:_____الياس هيثم ملحم_____, Number:_____2086_____, إعادة عملي

Submitted To GitHub_____ : ___

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.

The server code:

```python
import socket
import threading

# Predefined bank accounts (for simplicity)
accounts = {
    "1001": {"password": "pass1", "balance": 5000},
    "1002": {"password": "pass2", "balance": 3000},
    "1003": {"password": "pass3", "balance": 7000},
}

# Client handling function
def handle_client(client_socket, client_address):
    try:
        # Authentication
        client_socket.send(b"Enter account number: ")
        account_number = client_socket.recv(1024).decode().strip()
        client_socket.send(b"Enter password: ")
        password = client_socket.recv(1024).decode().strip()

        if account_number in accounts and accounts[account_number]["password"] == password:
            client_socket.send(b"Authentication successful\n")
            while True:
                # Menu
                client_socket.send(b"\n1. Check Balance\n2. Deposit Money\n3. Withdraw Money\n4. Exit\nChoose an option: ")
                option = client_socket.recv(1024).decode().strip()

                if option == '1':
                    # Check Balance
                    balance = accounts[account_number]["balance"]
                    client_socket.send(f"Your balance is: {balance}\n".encode())
                elif option == '2':
                    # Deposit Money
                    client_socket.send(b"Enter amount to deposit: ")
                    amount = float(client_socket.recv(1024).decode().strip())
                    accounts[account_number]["balance"] += amount
                    client_socket.send(b"Deposit successful\n")
                elif option == '3':
                    # Withdraw Money
                    client_socket.send(b"Enter amount to withdraw: ")
                    amount = float(client_socket.recv(1024).decode().strip())
                    if amount <= accounts[account_number]["balance"]:
                        accounts[account_number]["balance"] -= amount
                        client_socket.send(b"Withdrawal successful\n")
                    else:
                        client_socket.send(b"Insufficient balance\n")
                elif option == '4':
                    # Exit
                    final_balance = accounts[account_number]["balance"]
                    client_socket.send(f"Your final balance is: {final_balance}\n".encode())
                    break
                else:
                    client_socket.send(b"Invalid option\n")
        else:
            client_socket.send(b"Authentication failed\n")
    finally:
        client_socket.close()

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("0.0.0.0", 9999))
    server.listen(5)
    print("Server started and listening on port 9999")

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

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

The client code:

```python
import socket

def start_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(("127.0.0.1", 9999))

    while True:
        response = client.recv(4096)
        if not response:
            break
        print(response.decode(), end="")

        # Sending user input to server
        user_input = input()
        client.send(user_input.encode())
        if user_input == '4':  # Exit option
            break

    client.close()

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

الخرج:

خرج السيرفر

```
Server started and listening on port 9999
Accepted connection from ('127.0.0.1', 61881)
Accepted connection from ('127.0.0.1', 61891)
```

خرج ال client 1

```
Enter account number: 1001
Enter password: PASS1
Authentication failed

PS C:\Users\Karam\Desktop\alias2>
```

خرج ال client 2

```
Enter account number: 1002
Enter password: pass2
Authentication successful


1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Choose an option: 1
Your balance is: 3000


1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Choose an option: 2
Enter amount to deposit: 1500
Deposit successful


1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Choose an option: 1
Your balance is: 4500.0
```

يقوم السيرفر بالاستماع على البورت 9999 ويمكن ان يعمل مع 5 عملاء بنفس الوقت

يطلب من العميل بداية ادخال رقم الحساب وكلمة المرور للتحقق من هوية المستخدم

حيث يوجد قاعدة بيانات ضمن السيرفر تحوي على اسم المستخدم وكلمة المرور ورصيده ضمن البنك

وعند نجاح عملية التحقق يدخل السيرفر حلقة لا نهائية في حال اراد الزبون سحب او إيداع او التحقق من رصيده

لحين اختيار الزبون انهاء العملية

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.

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

if __name__ == '__main__':
    app.run(debug=True)
```

تم انشاء السيرفر المحلي باستخدام ال flask و عند التشغيل يعمل على العنوان http://127.0.0.1:5000

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Page</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Welcome to the Home Page</h1>
        <p>This is a simple website built with Flask.</p>
        <div>
            <img src="{{url_for('static', filename='FLASK.png')}}" />
        </div>
        <a href="{{ url_for('about') }}" class="btn btn-primary">About</a>
        <a href="{{ url_for('contact') }}" class="btn btn-secondary">Contact</a>
    </div>
</body>
</html>
```

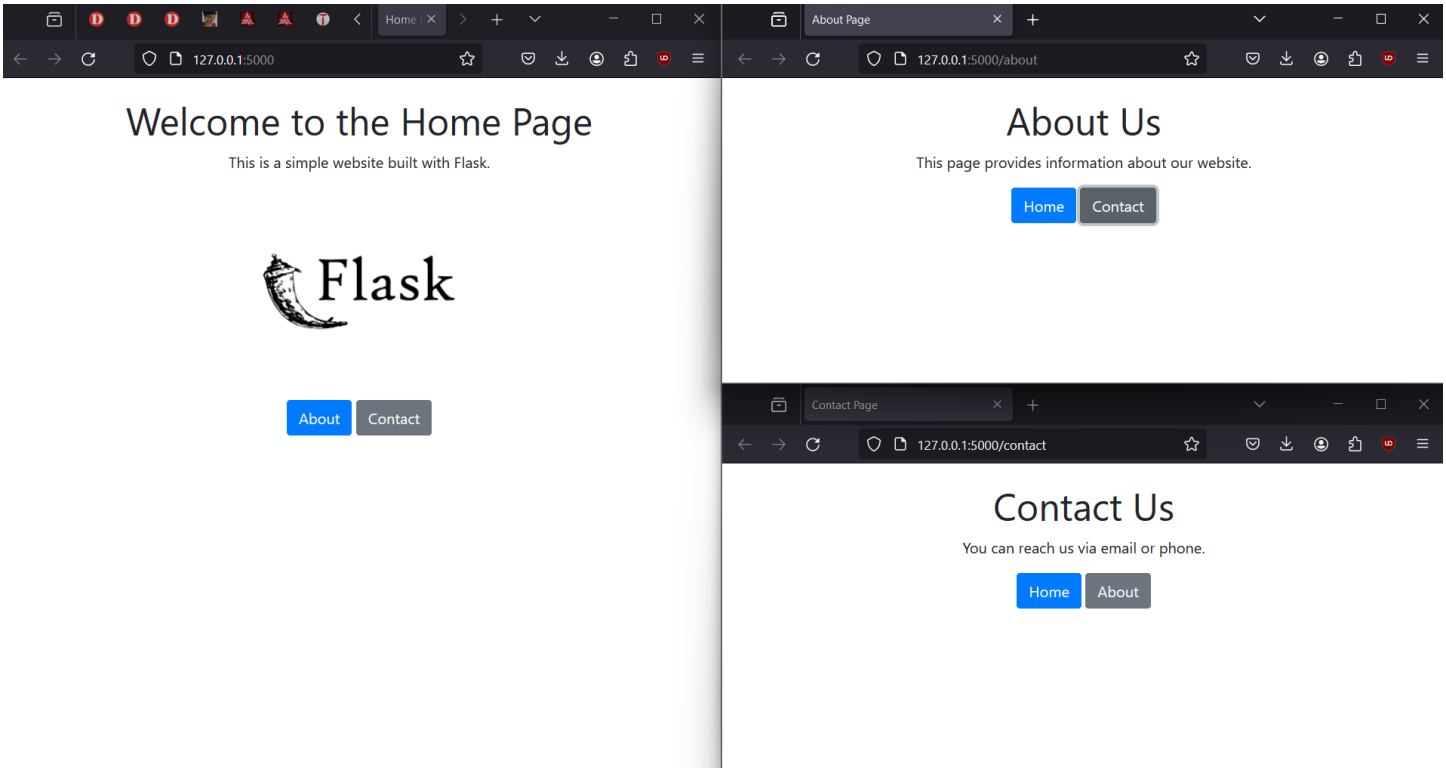
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Page</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Contact Us</h1>
        <p>You can reach us via email or phone.</p>
        <a href="{{ url_for('home') }}" class="btn btn-primary">Home</a>
        <a href="{{ url_for('about') }}" class="btn btn-secondary">About</a>
    </div>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Page</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>About Us</h1>
        <p>This page provides information about our website.</p>
        <a href="{{ url_for('home') }}" class="btn btn-primary">Home</a>
        <a href="{{ url_for('contact') }}" class="btn btn-secondary">Contact</a>
    </div>
</body>
</html>
```

تم انشاء ثلاث صفحات بسيطة و استخدام كل من ال bootstrap و ال css و هي عبارة عن صفحات فارغة تحوي نص رئيسي و زر للتنقل بينها

```
 * Serving Flask app 'from flask import Flask, render_template'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 918-840-294
```

تم استخدام الصور و النصوص فقط



نتيجة التنقل بين الصفحات يقوم السيرفر بتسجيل تاريخ الدخول و في حال وجود أي مشكلة ضمن الصفحة

```
 * Serving Flask app 'from flask import Flask, render_template'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 918-840-294
127.0.0.1 - - [04/Jun/2024 22:40:43] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2024 22:40:43] "GET /static/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2024 22:40:43] "GET /static/FLASK.png HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2024 22:40:51] "GET /about HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2024 22:40:51] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [04/Jun/2024 22:41:02] "GET /contact HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2024 22:41:02] "GET /static/styles.css HTTP/1.1" 304 -
```