

MAY 6, 2024

## Second Network Programming Homework

إعداد الطالبتين: آية وليد مجبور 2809 آلاء إبراهيم عدرة 2638

إشراف الدكتور: مهند عيسى

السنة الخامسة

هندسة الاتصالات والالكترونيات

## Question 1: Bank ATM Application with TCP Server/Client and Multi-threading

Server code:

```
import socket
import threading

ACCOUNTS = {
    "Aya": 25000,
    "Alaa": 3000
}

def handle_client(conn, addr):
    print("New connection from {}".format(addr))
    try:
        account_id = conn.recv(1024).decode()
        if account_id not in ACCOUNTS:
            conn.sendall("Invalid account ID".encode())
            return

        balance = ACCOUNTS[account_id]
        conn.sendall("Welcome to the bank ATM! Your current balance is:
{}".format(balance).encode())

        while True:
            choice = conn.recv(1024).decode()
            if choice == "1":
                conn.sendall("Your balance is:
{}".format(balance).encode())
            elif choice == "2":
                amount = int(conn.recv(1024).decode())
                balance += amount
                ACCOUNTS[account_id] = balance
                conn.sendall("Deposit successful. Your new balance is:
{}".format(balance).encode())
            elif choice == "3":
                amount = int(conn.recv(1024).decode())
                if amount > balance:
                    conn.sendall("Insufficient funds".encode())
                else:
                    balance -= amount
                    ACCOUNTS[account_id] = balance
                    conn.sendall("Withdrawal successful. Your new balance
is: {}".format(balance).encode())
            elif choice == "4":
```

```

        conn.sendall("Thank you for using the bank ATM. Your final
balance is: {}".format(balance).encode())
        break
    else:
        conn.sendall("Invalid choice".encode())

except Exception as e:
    print("Error: {}".format(e))
finally:
    conn.close()

def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(("0.0.0.0", 7777))
    server_socket.listen(5)
    print("Server is listening on localhost:7777")

    while True:
        conn, addr = server_socket.accept()
        client_thread = threading.Thread(target=handle_client, args=(conn,
addr))
        client_thread.start()

if __name__ == "__main__":
    main()

```

قمنا باستيراد المكتبتين socket و threading لبناء المقابس وحتى نجعل السيرفر يعمل بنفس الوقت مع عدة مستخدمين وتخدمهم جميعاً. تم بناء وإنشاء الحسابات بتعريف القاموس ACCOUNTS الذي يحوي اسم صاحب الحساب والمبلغ المالي الذي يملكه حالياً. السيرفر يعمل على العنوان: 0.0.0.0 وتم ربطه مع رقم المنفذ 8000، طبعاً السيرفر يتعامل مع أي عنوان IP ضمن التطبيق. عن طريق اشتقاق الغرض client\_thread من الصنف Thread نستطيع التعامل مع العملاء بنفس الوقت، وذلك بإمرار التابع الذي يتعامل مع اتصالات العملاء وبارامترات. التابع handle\_client(conn, addr) يأخذ بارمترين هما: عنوان العميل الذي يتصل مع السيرفر ومقبس العميل. يطبع التابع رسالة إضافة الاتصال مع العميل الجديد ويطبع بجانبها عنوانه. نقوم باستقبال اسم صاحب الحساب عبر المتحول account\_id والتحقق منه إن كان

موجود أم لا. لأجل كل عميل هناك العديد من الخيارات للعمليات مثل: عرض إجمالي النقود في الحساب، أو إضافة مبلغ مالي على الحساب، أو سحب مبلغ.

**Client1 code:**

```
import socket

def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(("127.0.0.1", 7777))

    print("Enter your account ID:")
    account_id = input()
    client_socket.sendall(account_id.encode())

    response = client_socket.recv(1024).decode()
    print(response)

    while True:
        print("\nChoose an option:")
        print("1. Check balance")
        print("2. Deposit money")
        print("3. Withdraw money")
        print("4. Exit")
        choice = input()
        client_socket.sendall(choice.encode())

        if choice == "1":
            balance = client_socket.recv(1024).decode()
            print(balance)
        elif choice == "2":
            print("Enter the amount to deposit:")
            amount = int(input())
            client_socket.sendall(str(amount).encode())
            response = client_socket.recv(1024).decode()
            print(response)
        elif choice == "3":
            print("Enter the amount to withdraw:")
            amount = int(input())
            client_socket.sendall(str(amount).encode())
            response = client_socket.recv(1024).decode()
            print(response)
        elif choice == "4":
            response = client_socket.recv(1024).decode()
```

```

        print(response)
        break
    else:
        response = client_socket.recv(1024).decode()
        print(response)

    client_socket.close()

if __name__ == "__main__":
    main()

```

**Client2 code:**

```

import socket

def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(("127.0.0.2", 7777))

    print("Enter your account ID:")
    account_id = input()
    client_socket.sendall(account_id.encode())

    response = client_socket.recv(1024).decode()
    print(response)

    while True:
        print("\nChoose an option:")
        print("1. Check balance")
        print("2. Deposit money")
        print("3. Withdraw money")
        print("4. Exit")
        choice = input()
        client_socket.sendall(choice.encode())

        if choice == "1":
            balance = client_socket.recv(1024).decode()
            print(balance)
        elif choice == "2":
            print("Enter the amount to deposit:")
            amount = int(input())
            client_socket.sendall(str(amount).encode())
            response = client_socket.recv(1024).decode()
            print(response)
        elif choice == "3":
            print("Enter the amount to withdraw:")

```

```

        amount = int(input())
        client_socket.sendall(str(amount).encode())
        response = client_socket.recv(1024).decode()
        print(response)
    elif choice == "4":
        response = client_socket.recv(1024).decode()
        print(response)
        break
    else:
        response = client_socket.recv(1024).decode()
        print(response)

    client_socket.close()

if __name__ == "__main__":
    main()

```

تشغيل السيرفر والزيائن:

```

$ python server-code.py
Server is listening on localhost:7777
New connection from ('127.0.0.1', 3725)
New connection from ('127.0.0.1', 3732)

```

```

● $ python client1-code.py
Enter your account ID:
Aya
Welcome to the bank ATM! Your current balance is: 25000

Choose an option:
1. Check balance
2. Deposit money
3. Withdraw money
4. Exit
1
Your balance is: 25000

Choose an option:
1. Check balance
2. Deposit money
3. Withdraw money
4. Exit
3
Enter the amount to withdraw:
200
Withdrawal successful. Your new balance is: 24800

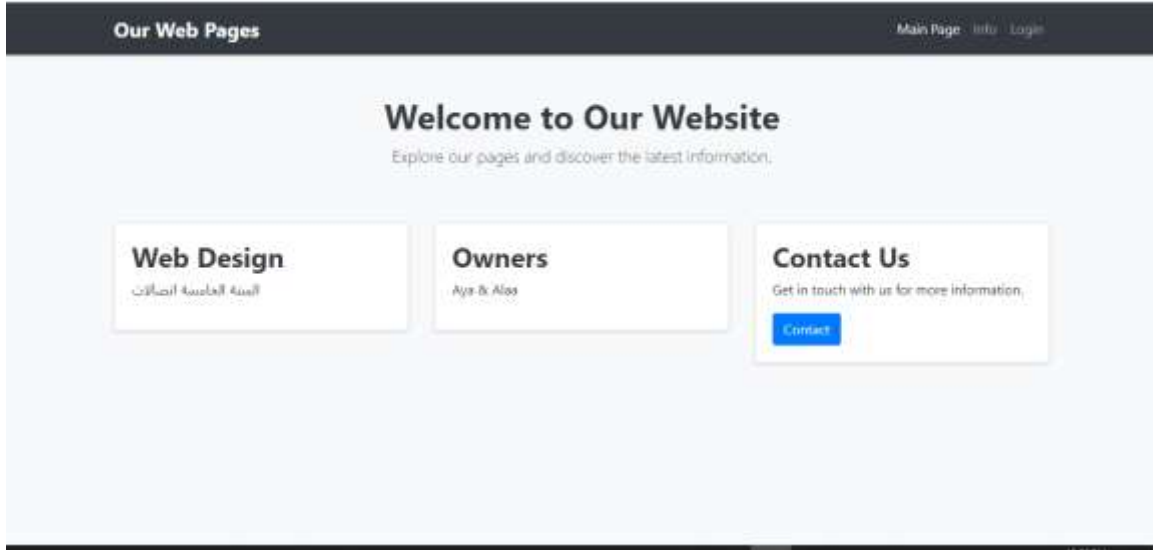
```

```
Enter your account ID:
Alaa
Welcome to the bank ATM! Your current balance is: 3000

Choose an option:
1. Check balance
2. Deposit money
3. Withdraw money
4. Exit
2
Enter the amount to deposit:
2000
Deposit successful. Your new balance is: 5000

Choose an option:
1. Check balance
2. Deposit money
3. Withdraw money
4. Exit
4
Thank you for using the bank ATM. Your final balance is: 5000
```

## Question 2: Web Site Using Flask Server:



The screenshot shows a contact form with three input fields: "Name:", "Email:", and "Message:". The "Name:" field is a single-line text input. The "Email:" field is a single-line text input. The "Message:" field is a multi-line text area. To the right of the "Name:" field, there is a Google Translate widget with buttons for "English" and "Arabic", and a "Google Translate" button. Below the form fields is a red "Submit" button.

برنامج سيرفر الفلاسك الذي يستضيف (يشغل) الموقع:



```

from flask import Flask, render_template

app = Flask(__name__)

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

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

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

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

```

1. from flask import Flask, render\_template:

هذا السطر يقوم بتحميل مكتبة Flask و Flask من Flask، هو إطار عمل لبناء تطبيقات الويب باستخدام Python، بينما يستخدم لتقديم قوالب HTML.

2. app = Flask(\_\_name\_\_):

ينشئ هذا السطر تطبيق Flask ويخزنه في المتغير app.

3. @app.route('/'):

هذا الديكوراتور يعرف مسار العنوان / للتطبيق. عندما يتلقى التطبيق طلبًا إلى هذا العنوان، سيتم تنفيذ الوظيفة التالية.

4. def home():

يعرف هذا السطر وظيفة باسم home التي ستتم تنفيذها عند طلب / من المستخدم.

5. return render\_template('index.html'):

هذا السطر يُرجع نتيجة HTML مقترنة بـ قالب index.html ، حيث يتم تقديم هذا القالب للمستخدم.

6. يتبع نفس النمط لوظائف info و login حيث يُخصص كل منها مسارًا محددًا ووظيفة للتنفيذ عند الطلب.

7. `if __name__ == '__main__':`

هذا السطر يتأكد مما إذا كان البرنامج يتم تشغيله مباشرة (بدلاً من استدعائه كمكتبة)، وفي هذه الحالة، يقوم بتشغيل التطبيق بوضع التصحيح (debug mode).