



الْجُمْهُورِيَّةُ الْعَرَبِيَّةُ السُّورِيَّةُ
وِزَارَةُ التَّعْلِيمِ الْعَالِي - جَامِعَةُ تَشْرِين
كَلِيَّةُ الْهَنْدَسَةِ الْمِيكَانِيكِيَّةِ وَالْكَهْرَبَائِيَّةِ
قِسْمُ هَنْدَسَةِ الْاِتِّصَالَاتِ وَالْاِلِكْتْرُونِيَّاتِ
السَّنَةُ الدِّرَاسِيَّةُ الْخَامِسَةُ 2023-2024

Network Programming

Homework No2

إِعْدَادُ الطَّالِبِ:

مُحَمَّدُ مَحْسَنُ التَّزْه

الرَّقْمُ الْجَامِعِيُّ 2482

إِشْرَافُ الدُّكْتُورِ : مَهْدُ عَيْسَى

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.

Solution / Code:

```
server.py
C: > Users > LCT > Desktop > homework2 > server.py > ...

1 import socket
2 import threading
3 # bank accounts
4 details = {'Mohammed': {'password': '555', 'balance': 2000}, 'Ali': {'password': '444', 'balance': 1000}, 'Ahmad'
5 }
6 answer = ["A", "B", "C", "D"]
7
8
9 def handle_Clients(socketCleint, addr):
10     # accsess client
11     socketCleint.send(b"Enter your name : ")
12     name = socketCleint.recv(1024).decode().strip()
13     socketCleint.send(b"Enter your Password : ")
14     password = socketCleint.recv(1024).decode().strip()
15     if name in details and details[name]['password'] == password:
16         socketCleint.send(b"acccess successful!\n")
17     else:
18         socketCleint.send(b"wrong details please try later !\n")
19         socketCleint.close()
20     return
21
22 while True:
23     socketCleint.send(b"what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit\n")
24     process = socketCleint.recv(1024).decode().strip()
25     if process in answer :
26         # Operation-1
27         if process == 'A':
28             socketCleint.send(f"Your balance is: ${details[name]['balance']}\n".encode())
29
30         # Operation-2
31         elif process == 'B':
32             socketCleint.send(b"how much you want to deposit ? ")
```

```
server.py
C: > Users > LCT > Desktop > homework2 > server.py > handle_Clients
32     socketCleint.send(b"how much you want to deposit: ")
33     money = int(socketCleint.recv(1024).decode().strip())
34     details[name]['balance'] += money
35     socketCleint.send(b"successful!\n")
36
37     # Operation-3
38     elif (variable) money: int
39         much you want to withdraw: ")
40     money = int(socketCleint.recv(1024).decode().strip())
41     if details[name]['balance'] >= money :
42         details[name]['balance'] -= money
43         socketCleint.send(b"successful!\n")
44     else:
45         socketCleint.send(b"you don't have enough money !\n")
46
47     # Operation-4
48     elif process == 'D':
49         TotalBalance = details[name]['balance']
50         socketCleint.send(f"Total-balance for client ${addr} is : ${TotalBalance}\n".encode())
51         socketCleint.close()
52         break
53
54     else:
55         socketCleint.send(b"please ty agin\n")
56
57 i = 1
58 while True:
59     server_S = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
60     server_S.bind(('localhost',9999))
61     server_S.listen(5)
62     socketCleint, addr = server_S.accept()
63     print(f"Accepted connection from {addr}")
64     thread = threading.Thread(target=handle_Clients, args=(socketCleint,addr))
65     thread.start()
```

Explanation:

هذا الكود يقوم بإنشاء خادم TCP يعمل على منفذ 9999 ويستطيع معالجة اتصالات متعددة من العملاء (أو الزبائن) بشكل متزامن باستخدام التعدد الخيطي (Multi-threading). هدف التطبيق هو تطبيق نظام أساسي لصراف آلي بنكي يُمكن العملاء من القيام بعمليات مصرفية مثل التحقق من الرصيد، الإيداع، والسحب.

الكود يتكون من الأجزاء الرئيسية التالية:

1. تفاصيل الحسابات البنكية: هي قاموس في Python يحتوي على حسابات المستخدمين مع كلمات المرور والأرصدة المرتبطة بهم.

2. دالة معالجة العملاء (handle Clients): وهي تتولى التفاعل مع العميل:

- يتم طلب اسم المستخدم وكلمة المرور.
- يتحقق من صحتها وإذا كانت تفاصيل الدخول خطأ، يغلق الاتصال.
- إذا كانت صحيحة، تعرض على العميل قائمة بالعمليات المتاحة وتنفذها استجابةً

لإدخالات العميل:

- التحقق من الرصيد: يعرض الرصيد الحالي للعميل.
- الإيداع: يستفسر عن مبلغ الإيداع ويضيفه إلى رصيد العميل.
- السحب: يستفسر عن مبلغ السحب ويخصمه من رصيد العميل إذا توفر رصيد كاف.

- الخروج: يعرض إجمالي الرصيد ويغلق الاتصال.

3. الكود الرئيسي: الذي ينشئ الخادم ويبدأ بالاستماع للاتصالات الواردة. عندما يتصل عميل، يُطبع عنوانه ويتم إنشاء خيط جديد لمعالجة الاتصال بحيث لا يؤثر على القدرة على قبول اتصالات جديدة.

Code Agent_1:

```
server.py ● Agent-1.py X
C: > Users > LCT > Desktop > homework2 > Agent-1.py > ...
1  import socket
2  Agent = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
3  Agent.connect(('127.0.0.1', 8888))
4  incoming = ''
5  while True:
6      incoming = Agent.recv(1024).decode()
7      if incoming == '':
8          break
9      print(incoming, end="")
10     Input = input()
11     Agent.send(Input.encode())
12     if Input == "D" :
13         print(incoming, end="")
14
```

Code Agent_2:

```
server.py ● Agent-1.py Agent-2.py X
C: > Users > LCT > Desktop > homework2 > Agent-2.py > ...
1  import socket
2  Agent = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
3  Agent.connect(('127.0.0.1', 8888))
4  incoming = ''
5  while True:
6      incoming = Agent.recv(1024).decode()
7      if incoming == '':
8          break
9      print(incoming, end="")
10     Input = input()
11     Agent.send(Input.encode())
12     if Input == "D" :
13         print(incoming, end="")
14
```

Explanation:

هذا الكود يمثل جهة العميل في تطبيق شبكة يستخدم بروتوكول TCP. إليك شرح مختصر لكل جزء:

1. استيراد موديول socket: يتم استيراد الموديول الذي يسمح بإجراء العمليات على الشبكة.
2. إنشاء السوكيت (Socket): يتم إنشاء سوكيت جديد باستخدام (AF_INET) IPv4 وتحديد أن طبيعة الاتصال ستكون بروتوكول (SOCK_STREAM) TCP.
3. إنشاء الاتصال: يقوم الكود بمحاولة الاتصال بالخادم الذي يعمل على الجهاز المحلي (127.0.0.1) والمنفذ رقم 9999.
4. حلقة لاستقبال البيانات:
 - يستمر العميل في تلقي البيانات من الخادم شريطة ألا يكون السلسلة الواردة فارغة.
 - إذا تلقى سلسلة فارغة، يفهم العميل أن الاتصال قد انتهى ويقوم بالخروج من الحلقة.
 - يتم طباعة البيانات الواردة ليتم رؤيتها من قبل المستخدم دون نهاية السطر (end="") لتتمكن من إدخال البيانات في نفس السطر.
 - يقوم المستخدم بإدخال الرد ويتم إرسال هذه البيانات مرة أخرى إلى الخادم مُشفرة بتسويق UTF-8.
5. شرط لإغلاق الاتصال: إذا كان الإدخال الذي يقوم به المستخدم هو الحرف "D"، فيفترض الكود أن هذا هو إشارة لإنهاء الجلسة فور طباعة الردود السابقة، ومع ذلك، يبدو أن هناك خطأ ما في الكود لأن المتغير incoming لن يتم تحديثه قبل الطباعة في هذا الشرط، ولا يُغلق السوكيت بشكل واضح.

RESULT:

The screenshot displays two command prompts in VS Code. The left window, titled 'Administrator: Command Prompt - python Agent-2.py', shows the execution of 'Agent-2.py' with a connection error. The right window, titled 'Administrator: Command Prompt - python Agent-1.py', shows the execution of 'Agent-1.py' with a connection error. A red box highlights the 'Terminal' pane at the bottom, showing connection logs for '127.0.0.1'.

Client_2

Client_1

عملية اتصال الزبون الثاني بالسيرفر

عملية اتصال الزبون الاول بالسيرفر

Agent-1:

The screenshot shows a command prompt running 'Agent-1.py'. The user enters 'Mohammed' and '555', and the program displays a menu with options A (Check Balance), B (Deposit), C (Withdraw), and D (Exit). The user selects B and enters '2000', then C and enters '1000'. The final output shows the total balance for client '127.0.0.1' as \$3000.

```
C:\Users\LCT\Desktop\homework2>python Agent-1.py
Enter your name : Mohammed
Enter your Password : 555
access successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $2000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
B
how much you want to deposit ? 2000
successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $4000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
C
how much you want to withdraw: 1000
successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $3000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
D
Your balance is: $3000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
Total-balance for client $('127.0.0.1', 53872) is : $3000
```


Agent-2:

```
Administrator: Command Prompt - python Agent-2.py
C:\Users\LCT\Desktop\homework2>python Agent-2.py
Enter your name : Ali
Enter your Password : 444
access successful!

what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $1000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
B
how much you want to deposit ? 2000
successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $3000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
C
how much you want to withdraw: 1000
successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $2000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
D
Your balance is: $2000
Total-balance for client $('127.0.0.1', 55838) is : $2000
```

• عند عملية التنفيذ لدينا اربع احرف لها وظائف محددة عند ادخالها في الكود وهي :

❖ A فحص الرصيد

❖ B الایداع

❖ C السحب

❖ D اغلاق

• لقد قمت في الكود السابق بادخال عمليات فحص الرصيد في حسابي ومن ثم الایداع

ومن ثم اعادة فحص الرصيد ومن ثم السحب ومن ثم فحص الرصيد ومن ثم

الاغلاق .

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”)

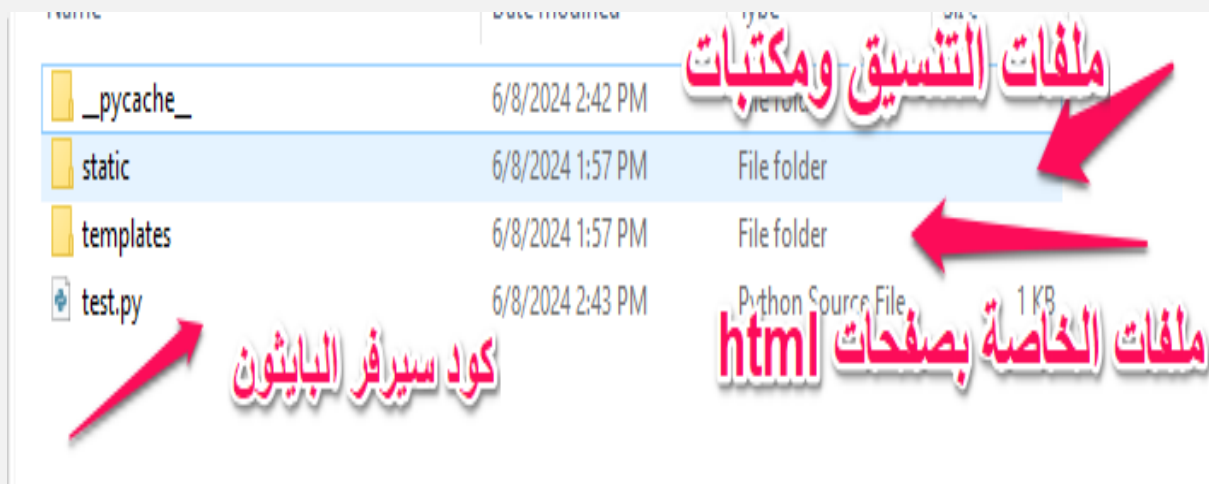
Create a simple website with multiple pages using Flask, HTML, CSS, and Bootstrap. The website should demonstrate your understanding of web design principles.

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.

Solution / Code:

Data structure:



Server code:

```
test.py x
C:\Users\LCT\Desktop> flask test.py
1 from flask import Flask , render_template
2 app = Flask(__name__)
3
4 @app.route("/")
5 def Home():
6     return render_template("Home.html")
7
8 @app.route("/main-page.html")
9 def HomeA():
10    return render_template("Home.html")
11
12
13 @app.route("/About-Us.html")
14 def About():
15    return render_template("About us.html")
16
17
18 if __name__ == "__main__" :
19    app.run(debug=True , port=9000)
20
```

استخدام مكتبات خاصة لتفج ملفات Html بالإضافة إلى Flask

بناء جذور الموقع

كود التشغيل

Lunch code server:

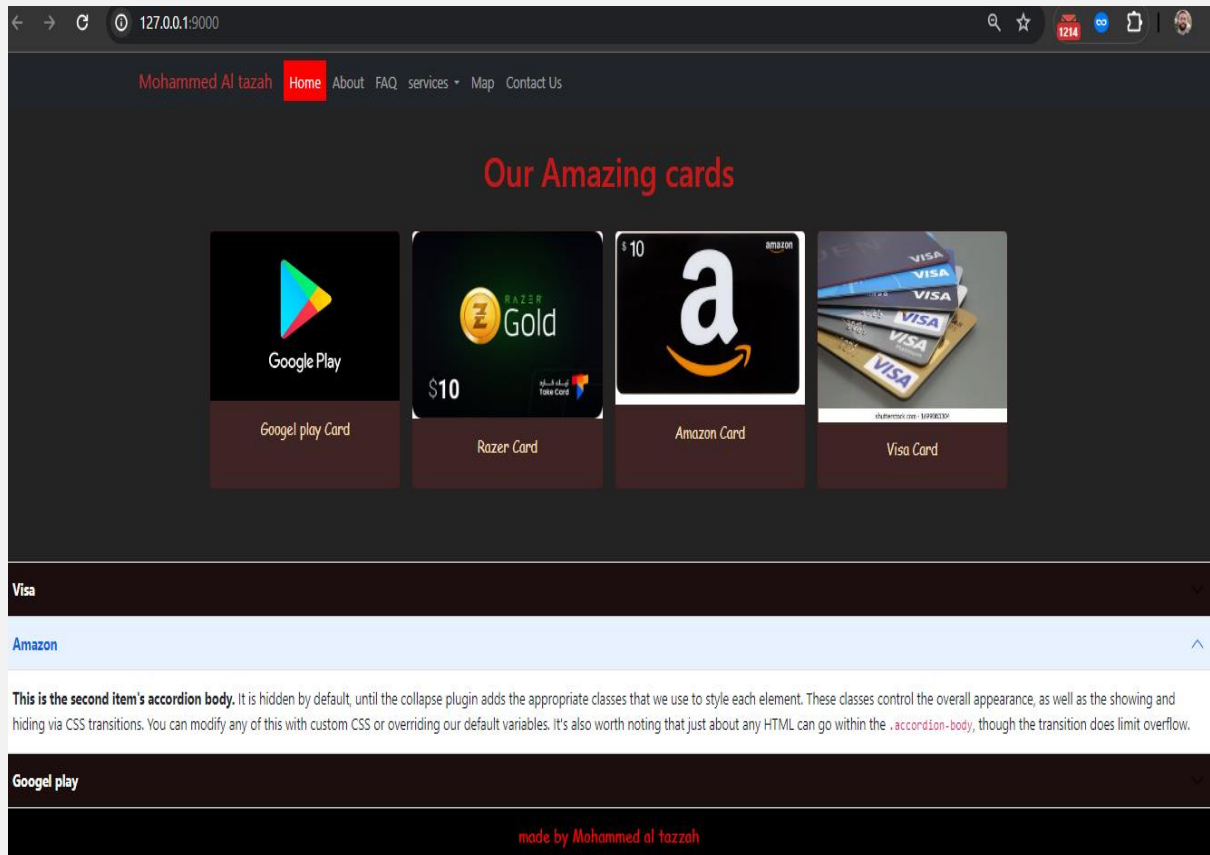
```
Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 877-755-764
* Running on http://127.0.0.1:9000/ (Press CTRL+C to quit)
27.0.0.1 - - [08/Jun/2024 14:56:16] "GET / HTTP/1.1" 200 -
27.0.0.1 - - [08/Jun/2024 14:56:16] "GET / HTTP/1.1" 200 -
27.0.0.1 - - [08/Jun/2024 14:56:16] "GET / HTTP/1.1" 200 -
```

عنوان الموقع والمنفذ الذي يعمل عليه

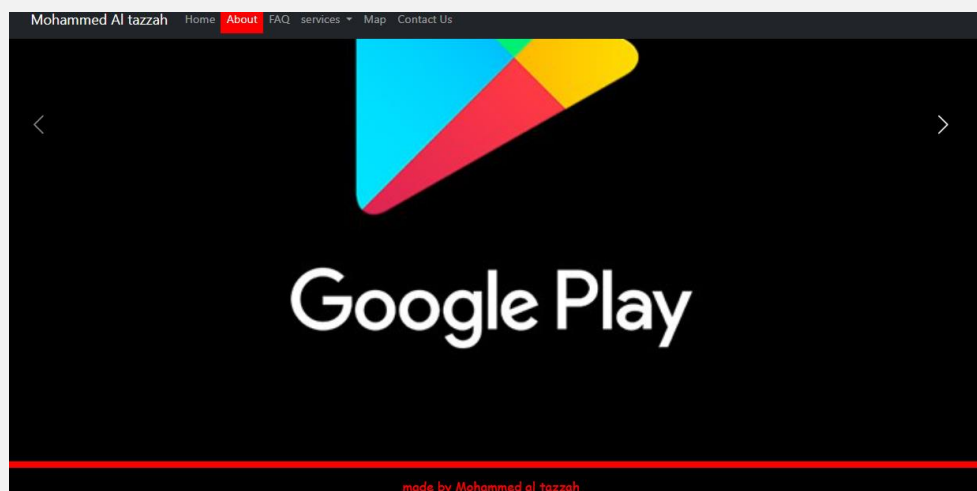
الملفات المحملة ونجاح تحميلها

Web Site :

Home-page



About-Us:



The End