



الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الكهربائية والميكانيكية

قسم : هندسة الاتصالات والإلكترونيات

سنة خامسة ... برمجة شبكات : الوظيفة 2

الوظيفة الثانية .. برمجة شبكات

إعداد :

محمد تيسير عسكر .. 2775

إشراف :

د. مهند عيسى

Question 1: TCP Server/Client Quiz App with Multi-threading?

As an improvement to previous first homework, build a TCP server and client quiz application using Python. The server should handle multiple client connections simultaneously using multi-threading. The application should allow clients to connect, participate in a quiz, and receive their quiz scores upon completion.

Requirements:

- A. The server should be able to handle multiple client connections concurrently.
- B. The quiz should consist of a set of pre-defined questions stored on the server.
- C. Each client should connect to the server and receive the quiz questions.
- D. Clients should send their answers to the server.
- E. The server should keep track of the scores for each client.
- F. At the end of the quiz, the server should send the final scores to each client.

Guidelines:

- Use Python's socket module “don't use 3rd-party packages”.
- Implement multi-threading to handle multiple client connections concurrently.
- Store the quiz questions and correct answers on the server side.

Notes:

- Write brief report describing the design choices you made and any challenges faced during implementation.
- You can make a [TCP Server/Client of your choice](#), such as Bank ATM, Chat application, or any other appropriate application that fulfil all requirements.

Solution :

Server_Code :

```
1// import socket , threading

2// q = open("quiz.txt","r")

3// qe = []

4// for i in range(20):

5// qe.append(q.readline()[0:-3])

6// Check_Answers = ["'9'", "'6'", "'15'", "'30'", "'150'", "'8'", "'24'", "'1000'", "'60'", "'80'",
"'1200'", "'90'", "'100'", "'2000'", "'99'", "'88'", "'625'", "'2500'", "'36'", "'180'"]

7// So = socket.socket()

8// So.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

9// So.bind(('127.0.0.1',7777))

10// So.listen(5)
```

```
11// def handle_Users (i ,UserSocket,UserAdd):  
  
    12// result = 0  
  
    13// print(f'+++ Accepted User_{i} and his Address is : {UserAdd}')  
    14// UserSocket.send(f'{qe}'.encode())  
  
    15// Ans = ''  
  
    16// while True:  
  
        17// Ans=UserSocket.recv(4096).decode()  
  
        18// if Ans :  
  
            19// break  
  
    20// User_Answers = Ans[1:len(Ans)-1].split(',')  
  
    21// For j in range(20) :  
  
        22// if User_Answers[j] == Check_Answers[j]:  
  
            23// result +=0.5  
  
    24// UserSocket.send(f'Your Result is : {result} // Thank You'.encode())  
  
    25// UserSocket.close()  
  
    26// print(f'Done from User {i} and His result is {result}')  
27// i = 1  
  
28// while True :  
  
    29// print('The Server is Weating for new Users .....')  
  
    30// us , uadd = So.accept()  
  
    31// user = threading.Thread(target=handle_Users,args=(i, us, uadd))  
  
    32// user.start()  
  
    33// i +=1
```

explanation:

- السطور 1-3 : عملية تضمين ل Socket && threading , فتح ملف الأسئلة بامتداد txt وتخزينه في المتحول q , تعريف مصفوفة الأسئلة وفي البداية تكون فارغة .
- السطور 4-5 : عبارة عن حلقة بداخلها عملية قراءة للأسئلة من الملف سطر سطر وبكل سطر يتم اقتطاع جزء وهو الجزء الذي يتم قراءة الرمز \n وهو النزول إلى سطر جديد حيث يتم قراءة هذا الجزء وبالتالي تم اقتطاعه عند عملية قراءة الأسئلة في كل مرة من دورات الحلقة , حيث يتم قراءة السطر وتخزينه في مصفوفة الأسئلة التي اسمها qe .
- السطور 6-10 : تخزين الأجوبة الصحيحة في مصفوفة تدعى Check_Answers التي سيتم استخدامها فيما بعد في عملية التحقق , ومن ثم عملية إنشاء Tcp Socket من أجل Server من الكائن Socket , السطر الثامن عبارة عن تعليمة مهمتها في حال حدوث خطأ عند الخروج من السيرفر بشكل غير نظامي فإنه سوف يظهر مشاكل وبالتالي هذه التعليمة معالجة هذه المشاكل وتفاديها حيث يتم اسناد القيمة 1 لها , عملية ربط ل Socket الذي تم عملها ب Address و port من أجل عملية الاتصال ل Server , ثم تعليمة عدد المستخدمين المسموح بهم من أجل الانتظار حتى يتم تخديمهم .
- السطور 11-14 : تم تعريف Method اسمها handle_Users وتم تمرير البارامترات التالية : i عداد من أجل أن يحسب ترتيب المستخدمين وعددهم المتصلين مع Server و UserSocket وهي Socket الخاصة بالمستخدم من أجل عملية تبادل البيانات مع Server و UserAdd وهو عنوان المستخدم المتصل مع ال Server , ثم ضمن التابع تم تعريف متحول Result الذي سوف يخزن النتيجة النهائية للمستخدم , ثم تعليمة طباعة أنه تم الاتصال مع مستخدم ويعرض Address الخاص بالمستخدم المتصل , عند الاتصال يتم إرسال المصفوفة التي تحوي الأسئلة إلى هذا المستخدم ...وظيفة هذا التابع بشكل عام أنه عند كل عملية اتصال من المستخدم يتم تخصيص thread خاصة بهذا المستخدم ويقوم التابع handle_Users بالتعامل مع هذا المستخدم وتخديمه عبر ال thread المخصص لهذا المستخدم
- السطور 15 – 19 : تعريف متحول اسمه Ans فارغ , ثم حلقة وما تقوم به أنه عند كل لفة تستقبل رسالة من المستخدم وتخزنه في المتحول Ans **إذا كان هناك رسالة** ومن ثم شرط انه في حال كان هناك رسالة يتم كسر الحلقة والخروج منها .. هنا فكرة الحلقة انتظار المستخدم ليرسل الأجوبة وتخزينها في المتحول Ans
- السطر 20 : عند استقبال الأجوبة من المستخدمة تكون عبارة عن list لكن مكون من سلسلة محارف أي عبارة عن Str على شكل list لذا كما نرى يتم اقتطاع الأقواس ومن ثم فصل هذه السلسلة المحرفية ب علامة “;” وبالتالي تتحول الرسالة إلى مصفوفة أي نوعها List ومن ثم يتم تخزينها في المتحول User_Answers
- السطور 21-23 : عبارة عن حلقة مهمتها تتأكد من أن الأجوبة المرسله من المستخدم صحيحة وذلك من خلال عملية مقارنة في المصفوفة User_Answers والمصفوفة Check_Answers المخزن بها الإجابات الصحيحة وعند تحقق شرط المساواة يتم زيادة المتحول Result الذي يخزن العلامة النهائية للمستخدم (في هذه الأثناء يكون المستخدم في حلقة انتظار استقبال العلامة النهائية من ال Server كما سنرى تاليا) تتم هذه العملية في كل دورة من دورات الحلقة
- السطور 24-26 : عملية إرسال العلامة النهائية للمستخدم على ال Socket الخاصة به ومن ثم إغلاق الجلسة مع هذا المستخدم أي إغلاق Socket الخاص به ومن ثم طباعة جملة "انتهاء من المستخدم وعلامته هي : " وذلك على جانب Server وليس المستخدم
- السطور 27-33 : بعد شرح الخوارزمية الذي يقوم عليها هذا التابع , يتم تنفيذ البرنامج الرئيسي الآن .. بداية تعريف متحول i وهو عداد للمستخدمين كما ذكرنا سابقا , ثم حلقة لانهاية ينتظر بها ال Server اتصال من المستخدم كما هو موضح في تعليمة الطباعة , عند اتصال مستخدم يتم تخزين Socket&&Address الخاصة بهذا المستخدم في متحولين , ومن ثم عملية انشاء Thread خاصة بهذا المستخدم حيث كما نرى نشأت من الكائن Thread كائن ونمرر له الخوارزمية التي سوف تخدم هذا المستخدم وهو في حالتنا التابع handle_Users وتم تمرير بارمترات هذا المستخدم للتابع من أجل تخديمه وهي i && Address && Socket , ثم تعليمة تنفذ هذا ال Thread من أجل البدء بالعمل , ومن ثم زيادة العداد بمقدار 1 ... هنا عند كل عملية اتصال لمستخدم يتم انشاء Thread خاص به بشكل منفصل عن أي Thread خاصة بمستخدم آخر وبالتالي يكون ال Server قادر على تخدم عدة مستخدمين بشكل متوازي وهذه هي فكرة Multi Threading ..

User_Code :

```
1// import socket

2// Answers = []

3// result = ''

4// Cs = socket.socket()

5// Cs.connect(('127.0.0.1',7777) )
```

```

6// msg = Cs.recv(1024).decode()

7// qe = msg[1:len(msg)-1].split(',')

8// for i in range(20) :

9// if i ==19 :

10// Answers.append(input(f'{qe[i]}'))

11// done = input('Are you Finish ? Y/N ')

12// if done == 'Y':

13// Cs.sendto(f'{Answers}'.encode(), ('127.0.0.1',7777))

14// while True:

15// result = Cs.recv(1024).decode()

16// if result :

17// print(result)

18// Cs.close()

19// Break

20// break

21// else :

22// print('You have to try later...!')

23// Cs.close()

24// Break

25// Answers.append(input(f'{qe[i]}'))

26// print('Done.....!')

```

explanation:

- السطور 1-3: عملية تضمين ل Socket Object , ومن ثم تعريف مصفوفة الأجوبة التي سوف تخزن بها إجابات المستخدم , وتم ثم تعريف متحول التي سوف يخزن بها علامة المستخدم النهائية المستقبلية من Server
- السطور 4-5 : عملية إنشاء Socket خاصة بالمستخدم ومن ثم الاتصال بال Server على العنوان والبورت الخاص بال Server كما هو موضح بالكود
- السطر 6 : تم تعريف متحول اسمه msg الذي بمجرد الاتصال مع ال Server سوف يرسل نموذج الأسئلة للمستخدم وسوف تخزن في المتغي الذي اسمه msg

- السطر 7 : عند استقبال نموذج الأسئلة من Server تكون عبارة عن list لكن مكونة من سلسلة محارف أي عبارة عن Str على شكل list لذا كما نرى يتم اقتطاع الأقواس ومن ثم فصل هذه السلسلة المحرفية ب علامة “;” وبالتالي تتحول الرسالة إلى مصفوفة أي نوعها List ومن ثم يتم تخزينها في المتحول qe وبالتالي يكون تم تخزين الأسئلة المرسلة من ال Server
- شرح ما تبقى من السطور : تم إنشاء حلقة مهمتها الأساسية طرح الأسئلة المستقبلية من ال Server والمخزنة في المتحول qe حيث يتم طرح الأسئلة سؤال سؤال كحقل الإدخال والجواب المدخل من قبل المستخدم يتم تخزينه في مصفوفة الأجوبة واسمها هو Answers , لكن تم وضع شرط في بداية الحلقة ومهمتها ما يلي : إذا تم الوصول لآخر دورة يتم طرح السؤال الأخير من نموذج الأسئلة ومن ثم يضيف سؤال أخير وهو إذا كان المستخدم قد انتهى من الحل , **إذا كان جواب الشرط** هو Y يتم إرسال مصفوفة الأجوبة المخزن بها أجوبة المستخدم إلى جهة ال Server ثم يدخل في حلقة لانهاية وظيفتها انتظار العلامة من ال Server كما نرى أنه داخل حلقة While يتم في كل مرة تخزين العلامة المستقبلية من ال Server **في حال كان هناك رد من قبله** حيث تم وضع شرط وهو في حال استقبال العلامة من ال Server أي أن المتغير Result يحوي إجابة من Server يتم طباعتها أي طباعة العلامة ثم يتم إغلاق ال Socket الخاصة بالمستخدم أي قفل الجلسة ومن ثم يتم كسر الحلقة While ثم يخرج من الحلقة الأساسية بشكل نهائي ولا يتابع العمل ومن ثم يتم طباعة Done...! **أما إذا كان جواب الشرط N** فيطبع رسالة "You have to try later...!" ثم يغلق ال Socket الخاصة بالمستخدم و ثم يكسر الحلقة الأساسية (حلقة for) ومن ثم يطبع رسالة Done...! .

فيما يلي سوف نستعرض الكود الخاص بال User && Server :

Server_Code :

```

File Edit Selection View Go Run Terminal Help
socket [Administrator]

Server.py x User_1.py User_2.py User_3.py User_4.py
Server.py > handle_Users
1 import socket , threading
2 q = open("quiz.txt","r")
3 qe = []
4 for i in range(20):
5     qe.append(q.readline()[0:-3])
6
7 Check_Answers = ["9", "6", "15", "30", "150", "8", "24", "1000", "60", "80",
8 "1200", "90", "100", "2000", "99", "88", "625", "2500", "36", "180"]
9
10 So = socket.socket()
11 So.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
12 So.bind(('127.0.0.1',7777))
13 So.listen(5)
14
15 def handle_Users (i ,UserSocket,UserAdd):
16     result = 0
17     print(f'+++ Accepted User_{i} and his Address is : {UserAdd}')
18     UserSocket.send(f'{qe}'.encode())
19     Ans = ''
20     while True:
21         Ans=UserSocket.recv(4096).decode()
22         if Ans :
23             break
24
25     User_Answers = Ans[1:len(Ans)-1].split(',')
26
27     for j in range(20) :
28         if User_Answers[j] == Check_Answers[j]:
29             result +=0.5
30
31     UserSocket.send(f'Your Result is : {result} // Thank You'.encode())
32     UserSocket.close()
33     print(f'Done from User {i} and His result is {result}')
34
35
36 i = 1
37 while True :
38     print('The Server is Waiting for new Users .....')
39     us , uadd = So.accept()
40     user = threading.Thread(target=handle_Users,args=(i, us, uadd))
41     user.start()
42     i +=1
43
44

```

User_Code :

الآن سوف نستعرض عملية تشغيل ال Server وتشغيل ال Users والمعلومات التي يعرضها ال Server عندما يتم الاتصال به :

هنا عندما يكون ال Server في حال تنصت وانتظار أي مستخدم ليتصل معه

الآن سوف نشغل مستخدمين وسوف نرى النتائج من جهة ال Server ومن جهة المستخدمين :

المستخدم الأول .. نلاحظ أنه استلم نموذج الأسئلة من ال Server ونرى أنه تم عرض السؤال الأول للإجابة عليه ومن ثم سوف يظهر السؤال الذي يليه

المستخدم الثاني .. كما نلاحظ حصلنا على نفس النتيجة مثل المستخدم الأول وتم استقبال الأسئلة من ال Server

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HADI\Desktop\socket> & 'C:\Python38\python.exe' 'c:\Users\HADI\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\
\lib\python\debugpy\adapter\..\..\debugpy\launcher' '63736' '--' 'c:\Users\HADI\Desktop\socket\Server.py'
The Server is Waiting for new Users .....
The Server is Waiting for new Users .....
+++ Accepted User_1 and his Address is : ('127.0.0.1', 63753)
The Server is Waiting for new Users .....+++ Accepted User_2 and his Address is : ('127.0.0.1', 63758)
```

Server... كما نلاحظ أنه ظهر على جانب ال Server أنه هناك مستخدمين متصلين مع ال Server

الآن سوف نجيب عن بعض الأسئلة من قبل المستخدمين ونرى كيف يتم إرسال النتيجة لكل منهما :

```
PROBLEMS  DEBUG CONSOLE  TERMINAL  JUPYTER

'q_1:3x3='9
'q_2:2x3='6
'q_3:3x5='15
'q_4:10x3='30
'q_5:50x3='
'q_6:4x2='
'q_7:3x8='
'q_8:10x100='
'q_9:20x3='
'q_10:2x40='
'q_11:2x600='
'q_12:3x30='
'q_13:4x25='
'q_14:20x100='
'q_15:3x33='
'q_16:8x11='
'q_17:25x25='
'q_18:50x50='
'q_19:6x6='
'q_20:60x3='
Are you Finish ? Y/N Y
Your Result is : 2.0 // Thank You
Done....!
PS C:\Users\HADI\Desktop\socket>
```

نلاحظ أنه عند انتهاء المستخدم الأول استقبل نتيجته ورسالة شكر ثم تم طباعة Done....!

```
PROBLEMS  DEBUG CONSOLE  TERMINAL  JUPYTER

\\lib\python\debugpy\adapter\..\..\debugpy\launcher' '63756' '--' 'c:\Users\HADI\Desktop\socket\User_2.py'
'q_1:3x3='9
'q_2:2x3='6
'q_3:3x5='15
'q_4:10x3='30
'q_5:50x3='150
'q_6:4x2='
'q_7:3x8='
'q_8:10x100='
'q_9:20x3='
'q_10:2x40='
'q_11:2x600='
'q_12:3x30='
'q_13:4x25='
'q_14:20x100='
'q_15:3x33='
'q_16:8x11='
'q_17:25x25='
'q_18:50x50='
'q_19:6x6='
'q_20:60x3='
Are you Finish ? Y/N Y
Your Result is : 2.5 // Thank You
Done....!
PS C:\Users\HADI\Desktop\socket>
```

نلاحظ أن المستخدم الثاني استقبل نتيجته أيضا وتم الطباعة مثل المستخدم الأول ..
أجبنا بشكل مختلف لنظهر أن آلية معالجة العلامة تظهر وتتم بشكل صحيح

```
PROBLEMS  DEBUG CONSOLE  TERMINAL  JUPYTER

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HADI\Desktop\socket> & 'C:\Python38\python.exe' 'c:\Users\HADI\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\
\lib\python\debugpy\adapter\..\..\debugpy\launcher' '63736' '--' 'c:\Users\HADI\Desktop\socket\Server.py'
The Server is Waiting for new Users .....
The Server is Waiting for new Users .....
+++ Accepted User_1 and his Address is : ('127.0.0.1', 63753)
The Server is Waiting for new Users .....+++ Accepted User_2 and his Address is : ('127.0.0.1', 63758)

Done from User 1 and His result is 2.0
Done from User 2 and His result is 2.5
```

نلاحظ أنه من جهة ال Server تم طباعة عبارة الانتهاء من المستخدم الأول والثاني وطباعة علامة كل منهما

التحديات والصعوبات التي صادفت الخوارزمية :

- كان من الصعب إيجاد طريقة لتخزين الأسئلة الموجودة على المستند النصي في ال Server
- كما أنه كان من الصعب جدا إيجاد آلية لإرسال نموذج الأسئلة للمستخدمين المتصلين مع ال Server
- التحدي الأصعب باعتقادي هو إيجاد آلية لمعالجة نموذج الأسئلة المستقبلية من جهة ال Server حيث تستقبل كسلسلة نصية وكان من الصعب فصل الأسئلة عن بعضها وتحويلها إلى مصفوفة كل عنصر بها عبارة عن سؤال منفصل, والذي باعتقادي أنه محور الخوارزمية هو آلية معالجة الرسائل المستقبلية من جهة المستخدم ومن جهة ال Server
- من ناحية التفكير المنطقي كان هناك ثغرات من ناحية أنه يجب على الخوارزمية بشكل عام أن تعمل بشكل متوازي على كل من ال Server و ال Users بعبارة أخرى , أنه يجب إضافة فترات انتظار للسيرفر حتى الانتهاء من حل الأسئلة من قبل المستقبل , ويجب إضافة فترة انتظار من قبل المستخدم ريثما يقوم السيرفر من التحقق من الأجوبة وإرسال النتيجة إلى المستخدم أي بشكل عام يجب التفكير في هذه الآلية التي لم تؤخذ بالحسبان في بداية الأمر والتي سببت خلل في أداء الخوارزمية ومن ثم تم تدارك هذه الثغرة المنطقية .. مما جعل الخوارزمية تعمل بشكل صحيح .

Question 2: Simple Website with Python Flask Framework

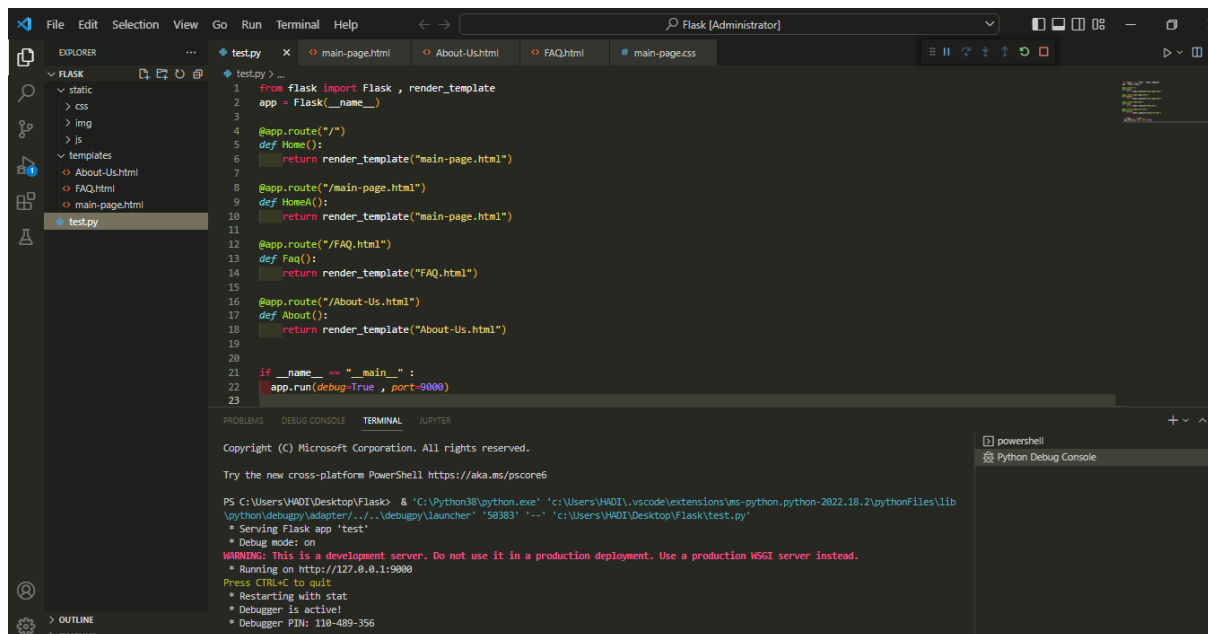
Create a simple website with multiple pages using Flask, HTML, CSS, and Bootstrap. The website should demonstrate your understanding of web design principles.

Requirements:

- Set up a local web server using XAMPP, IIS, or Python's built-in server (using Flask).
- Apply CSS and Bootstrap to style the website and make it visually appealing.
- Ensure that the website is responsive and displays correctly on different screen sizes.
- Implement basic server-side functionality using Flask to handle website features.

Solution :

فيما يلي سوف نستعرض كود ال Server باستخدام Flask ونرى النتائج عند تشغيل ال Server :



```
1 from flask import Flask, render_template
2 app = Flask(__name__)
3
4 @app.route("/")
5 def Home():
6     return render_template("main-page.html")
7
8 @app.route("/main-page.html")
9 def HomeA():
10    return render_template("main-page.html")
11
12 @app.route("/FAQ.html")
13 def Faq():
14    return render_template("FAQ.html")
15
16 @app.route("/About-Us.html")
17 def About():
18    return render_template("About-Us.html")
19
20
21 if __name__ == "__main__":
22    app.run(debug=True, port=9000)
23
```

Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\VHADI\Desktop\Flask> & 'C:\Python38\python.exe' 'c:\Users\VHADI\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '50383' '-t' 'c:\Users\VHADI\Desktop\Flask\test.py'

* Serving Flask app 'test'

* Debug modes on

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

* Running on http://127.0.0.1:9000

Press CTRL+C to quit

* Restarting with stat

* Debugger is active!

* Debugger PIN: 110-489-356

Server_Code :

```
from flask import Flask , render_template
app = Flask(__name__)

@app.route("/")
def Home():
    return render_template("main-page.html")

@app.route("/main-page.html")
def HomeA():
    return render_template("main-page.html")

@app.route("/FAQ.html")
def Faq():
    return render_template("FAQ.html")

@app.route("/About-Us.html")
def About():
    return render_template("About-Us.html")

if __name__ == "__main__":
    app.run(debug=True , port=9000)
```

explanation:

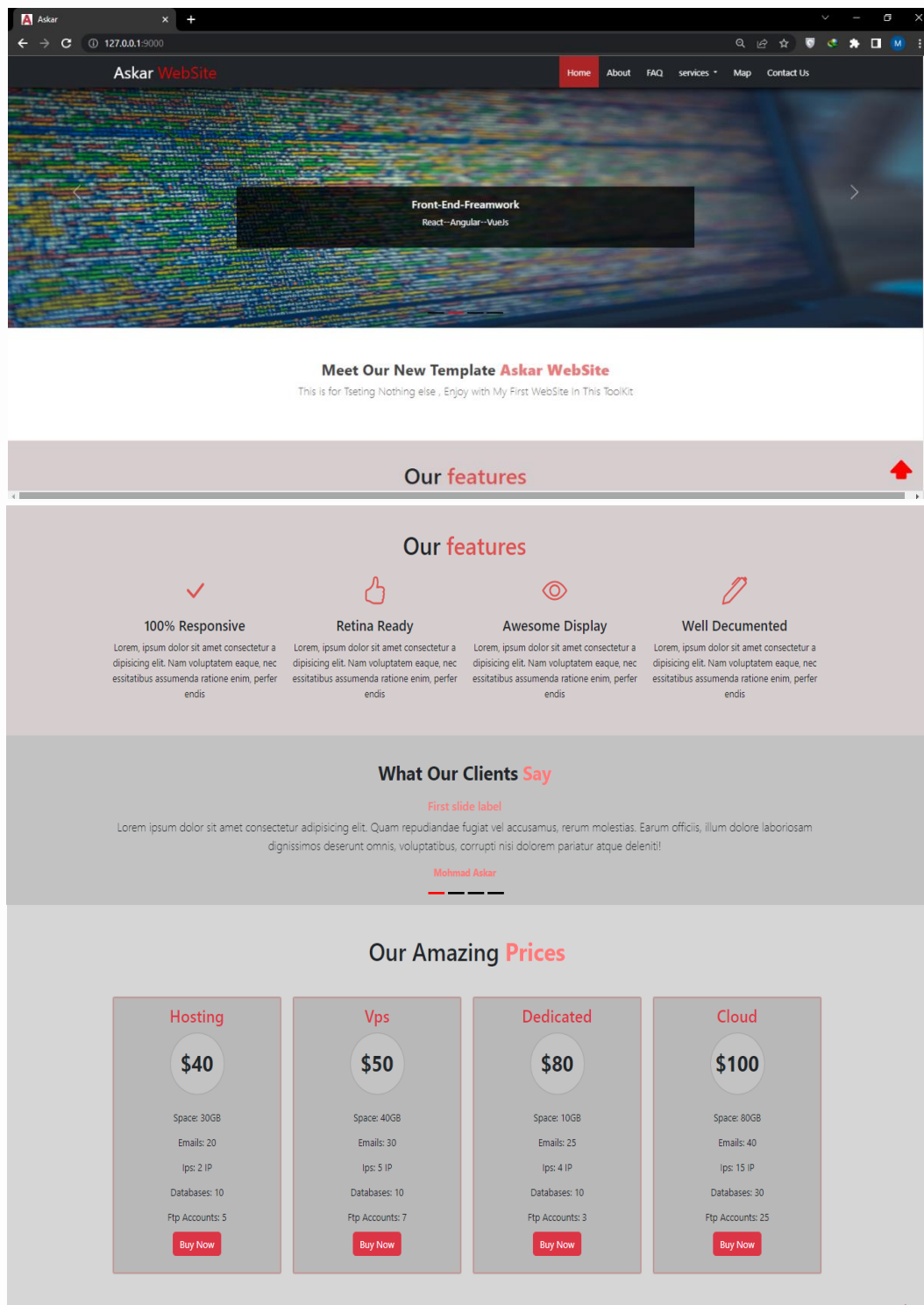
- كما نلاحظ من الصورة السابقة أن كود ال Server باستخدام Flask يتألف مما يلي :
- 1- عملية تضمين ل Flask و render_template وهي مسؤولة عن إظهار صفحات Html عند الاستدعاء .
 - 2- نلاحظ أنه عملية توزيع لمسارات الموقع وقد تم اختيار أن الجذر الرئيسي للمسارات أي الصفحة الأولى التي سوف تظهر قد تم اختيار صفحة (main-page.html) لتكون الجذر الرئيسي
 - 3- في هذا الموقع قمنا بعمل ثلاث صفحات html وبالتالي لدينا ثلاث مسارات فرعية .. وهم :
FAQ.html و About-Us.html و main-page.html ... قمنا بإعادة كتابة الجذر بهذه الطريقة من أجل آلية الانتقال بين الصفحات لان هذه الصفحات سوف تكون مبروطة فيما بينها.
 - 4- نلاحظ أنه بكل مسار يعيد لنا صفحة ال Html المطلوبة حيث أن التابع يعيد ال Method اسمها render_template والذي بدوره يمرر له صفحة ال Html فيعيد لها .
 - 5- نلاحظ أن دور الشرط هنا أن هذا التطبيق لن يعمل إلا اذا تحقق الشرط وهو شرط التشغيل .. وعند التحقق سوف يقوم بعملية Run للتطبيق ونلاحظ أنه غيرنا port الخاص بالتطبيق و قمنا بتفعيل وضع debug .

النتائج عند تشغيل ال Server :

نلاحظ من الصورة السابقة أنه عند تشغيل ال Server تظهر النتائج التالية :

- 1- الملف الذي اسمه test.py قد تم تشغيله وهو ال Server
 - 2- ونلاحظ أنه تم تفعيل وضع debug mode
 - 3- ونلاحظ أنه أعاد لنا رابط الموقع (جذر الموقع) وهو عبارة عن localhost للحاسب المحلي ويظهر معه port الذي اخترناه .
- كما ذكرنا سابقا موقعنا يتألف من ثلاث صفحات html سوف نستعرضها ومن ثم نرى النتائج من جهة ال Server لنرى الطلبات التي تم عملها .

main-page.html :



Our Team



Chris Coyler

This is Chris Coyler The Founder Of The Amazing Web-Site Csstricks



Steve Jobs

This is Steve Jobs And He Is Web-Developer And he is Very Clever



Leah Culver

This is Leah Culver And She Is Amazing In Hacker Stuff



Sam Anderson

This is Sam Anderson And He Is Team Leader In This Project



Keep In Touch

Sign Up For Free Newsletter Dont Worry About Spam We Hate It Too .

Our Main Statistics



9,912

Satisfied Users



2,923

Posted Comments



25,94

Projects Done



44,32

Tickets Achieved

Our Skills

Html / Csst3

JavaScript / JQuery

PHP / MySQL

Wordpress

About The Team

We are programming Team What Ever You want It will Be a Digital Dream With Us We are The Best In Web Design , Your Dream Will Be Real

Tell Us What you Fell

Fell Free To Contact Us Anytime

Tiny
white



Smaller

BeMosca



Sitemap

Home

About

Company

Code

Design

Host

Solutions

Sitemap

Contact



Latest Articles



Programming

Lorem ipsum dolor sit amet, consectetur adipisicing elit, from Askar for web



Coding

Lorem ipsum dolor sit amet, consectetur adipisicing elit, from Askar for web



Web Design

Lorem ipsum dolor sit amet, consectetur adipisicing elit, from Askar for web

Our Awsome Work



About-Us.html :

Askar WebSite

HomeAboutFAQservicesMapContact Us

About Us

Think About Us When You Want To Do It , Just Contact With Us And Will Be Here AnyTime You Want !



We Love Code



Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta suscipit fuga debitis sequi. Sunt praesentium nemo veritatis illo nesciunt iste.

We Are Social



Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta suscipit fuga debitis sequi. Sunt praesentium nemo veritatis illo nesciunt iste.

We Are Happy



Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta suscipit fuga debitis sequi. Sunt praesentium nemo veritatis illo nesciunt iste.



We are Happy To Work And Help You

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Nam explicabo suscipit est mollitia non et nesciunt hic numquam quae minus officia, incididunt rem debitis cum excepturi itaque? Reiciendis, vel laboriosam. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Nam explicabo suscipit est mollitia non et nesciunt hic numquam quae minus officia, incididunt rem debitis cum excepturi itaque? Reiciendis, vel laboriosam. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Nam explicabo suscipit est mollitia non et nesciunt hic numquam quae minus officia, incididunt rem debitis cum excepturi itaque? Reiciendis, vel laboriosam.

Contact With Us

Sitemap

HomeAboutCompany

CodeDesignHost

SolutionsSitemapContact



Latest Articles



Programming

Lorem ipsum dolor sit amet, consectetur adipisicing elit, from Askar for web.



Coding

Lorem ipsum dolor sit amet, consectetur adipisicing elit, from Askar for web.



Web Design

Lorem ipsum dolor sit amet, consectetur adipisicing elit, from Askar for web.

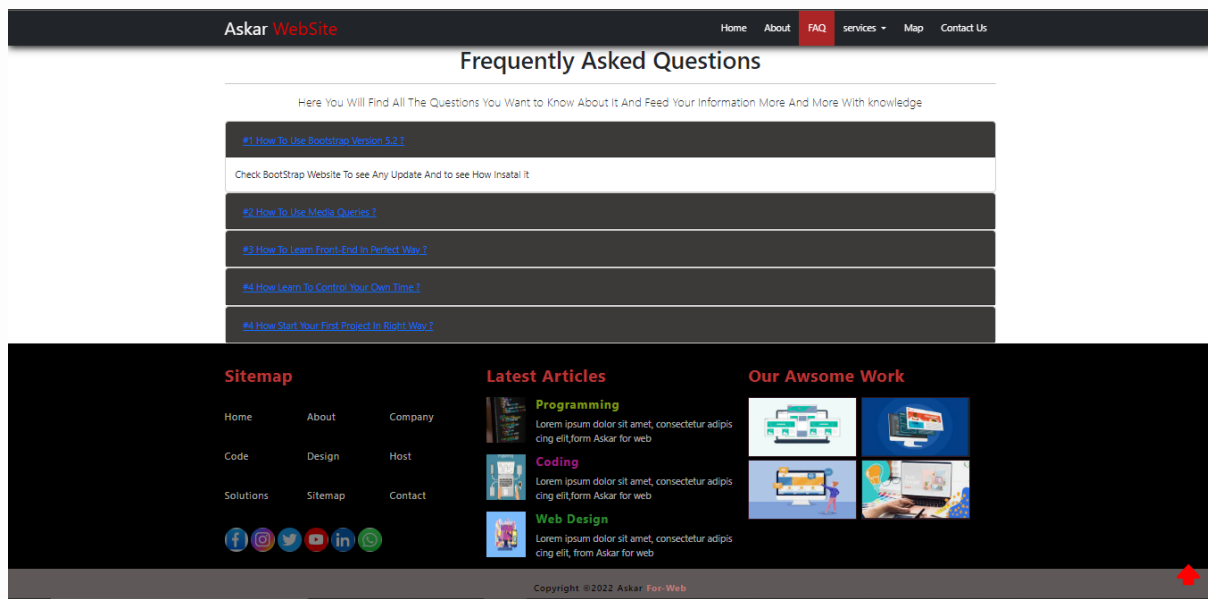
Our Awsome Work



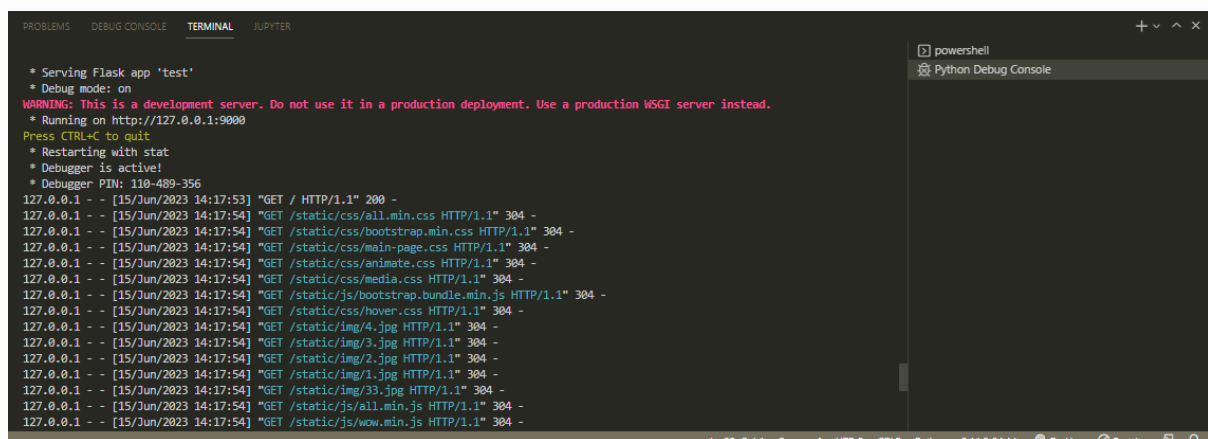


Copyright ©2022 Askar For Web

FAQ.html :



الصورة التالية توضح الطلبات التي يعالجها ال Server ويبين Status Code لكل طلب كما هو مبين الشكل عند الدخول على جذر الموقع ("/") أي الصفحة الرئيسية :



نلاحظ أنه عثر على صفحة main-page.html وقد أرجع بيانات هذه الصفحة والعناصر التي تحويها .. نلاحظ هناك العديد من الطلبات لإرجاع كامل الصفحة .

- فيما يلي سوف نستعرض أكواد الموقع الأجزاء المهمة منها وهي طريقة ربط الصفحات والربط مع ملفات CSS&&JS ومعمارية الملفات وكيف يجب تخزينها عند التعامل مع Flask-FreamWork :

main-page.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="icon" type="img/png" href="{{ url_for('static', filename='img/logo/we.png') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/all.min.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/main-page.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/hover.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/animate.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/media.css') }}">
</head>
<title>Askar</title>
```

آلية ربط ملفات Css و Bootstrap في Flask

```
</div>
<div class="carousel-inner">
  <div class="carousel-item active" data-bs-interval="5000">
    
    <div class="carousel-caption d-none d-md-block">
      <h5>Front-End-Roadmap-Basics</h5>
      <p>Html--Css--JavaScript--BootStrap</p>
    </div>
  </div>
  <div class="carousel-item" data-bs-interval="2000">
    
    <div class="carousel-caption d-none d-md-block">
      <h5>Front-End-Freamwork</h5>
      <p>React--Angular--VueJs</p>
    </div>
  </div>
  <div class="carousel-item" data-bs-interval="2000">
    
    <div class="carousel-caption d-none d-md-block">
      <h5>Back-End-Roadmap-Basics</h5>
      <p>PHP-SQL--DataBase 'Deeply'</p>
    </div>
  </div>
</div>
```

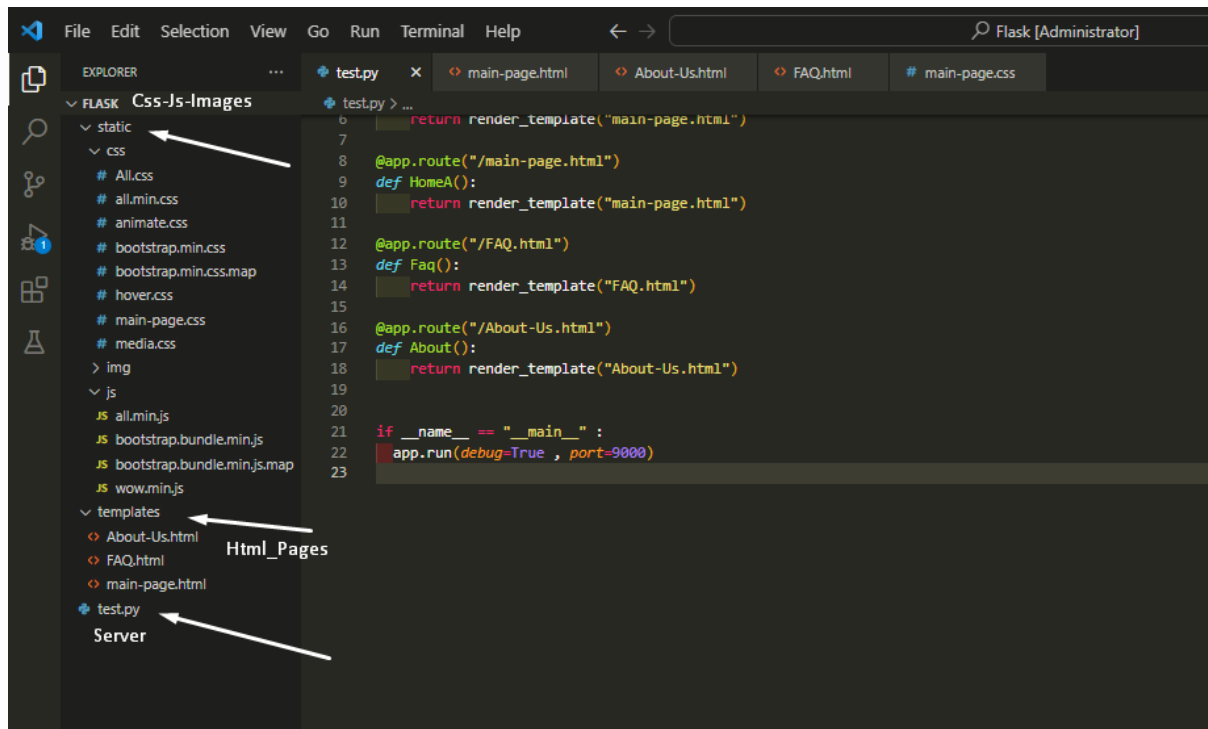
آلية إدراج مسار صورة في صفحة Html عند التعامل مع Flask

```
<script src="{{ url_for('static', filename='js/bootstrap.bundle.min.js') }}"></script>
<script src="{{ url_for('static', filename='js/all.min.js') }}"></script>
<script src="{{ url_for('static', filename='js/wow.min.js') }}"></script>
</script>
```

آلية إدراج مسار ملفات JS وربطها عند التعامل مع ال Flask

والأمر نفسه في بقية الصفحات الأخرى

- الآن سوف نستعرض معمارية الملفات عند التعامل مع Flask FreamWork :



- نلاحظ فيما يتعلق في صفحات html يجب تخزينها في مجلد اسمه template حيث أن Server يبحث عن الصفحات التي يعيدها في هذا المجلد
- كما نلاحظ أن ملف ال Server هو ملف منفصل عن بقية المجلدات والملفات
- فيما يخص ملفات Css && Js && Images يتم تخزينها في مجلد اسمه static حيث أن Server تلقائياً يبحث عن الملفات المرتبطة بصفحات ال Html في هذا المجلد من صور أو ملفات js أو css .