Syrian Arab Republic

Lattakia - Tishreen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No2

الجمهورية العربية السورية

اللاذقية -جامعــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة2 برمجة شبكات

Second Network Programming Homework

علي ناظم القوزي 2993 فرح احمد ميهوب 2693

السؤال الأول:

نقوم بإنشاء سيرفر TCP لاستقبال العديد من الاتصالات والتعامل معها في نفس الوقت.

1- استيراد المكتبات وتعريف الحسابات البنكية :

يتم استيراد المكتبات socket من أجل انشاء السوكيت واستخدام التوابع الخاصة بالتعامل مع السيرفر لأننا سنقوم بإنشاء سوكت TCP وإرسال واستقبال البيانات عبر الشبكة و threading لإنشاء ثريد خاصة بكل كلاينت ينجح بالاتصال. المتغير data نوع dict يحوي الحسابات على شكل رقم حساب كمفتاح ومتغير tuple يحوي رمز الدخول والرصيد الابتدائي.

```
import socket
import threading
data = {
    '2993': ('1111', 1000),
    '2693': ('2222', 2000),
}
```

2- تعريف التابع handle_client:

يستقبل معلومات دخول المستخدم ويتحقق من صحتها فاذا كانت صحيحة يقوم بالعمليات المصرفية المطلوبة (معرفة الرصيد، شحن الحساب، السحب من الحساب، تسجيل الخروج) والا يغلق الاتصال.

```
def handle_client(client_socket,addr):
    try:
        AUTH = False

        while True:
            request = client_socket.recv(1024).decode('utf-8').strip()
            if not request:
                break

            if not AUTH:
```

```
                account_number, pin = request.split(',')
                if account_number in data and data[account_number][0] == pin:
                    AUTH = True
                    client_socket.send(b'TRUE\n')
                else:
                    client_socket.send(b'False\n')
                    break
            if request:
                command, *args = request.split(',')
                print(command)

            if command == 'BALANCE':
                balance = data[account_number][1]
                client_socket.send(f'Balance: {balance}\n'.encode('utf-8'))

            elif command == 'DEPOSIT':
                amount = float(args[0])
                data[account_number] = (data[account_number][0],
data[account_number][1] + amount)
                client_socket.send(b'Deposit Successful\n')

            elif command == 'WITHDRAW':
                amount = float(args[0])
                if data[account_number][1] >= amount:
                    data[account_number] = (
                        data[account_number][0], data[account_number][1] -
amount)
                    client_socket.send(b'Withdrawal Successful\n')
                else:
                    client_socket.send(b'Insufficient Funds\n')

            elif command == 'LOGOUT':
                balance = data[account_number][1]
                client_socket.send(f'Final Balance: {balance}\n'.encode('utf-
8'))
                break
    finally:
        client_socket.close()
        print(f' {addr} Closed!')
```

3- نقوم بإنشاء كائن socket وتعيين عنوان IP ورقم المنفذ له والاستماع الى الاتصالات الواردة عن طريق استخدام التابع listen. يأخذ السيرفر جميع عنوانين IP المتاحة ورقم المنفذ 1234 ويقوم

```
4- ssock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

   ssock.bind(('0.0.0.0', 1234))
   ssock.listen(5)
```

5- بمجرد أن يتم اتصال المستخدم يتم إنشاء ثريد جديد وارسال يقوم بتشغيل التابع handle_client ونمرر له البارمترات التي هي عبارة عن السوكت الخاصة بالكلاينت والعنوان.

```
    csock, addr = ssock.accept()
    print(f'New Client connection At {addr}')
    client_handler = threading.Thread(target=handle_client, args=(csock,addr))
    client_handler.start()
```

وبالتالي يكون شكل الكود النهائي:

```
import socket
import threading

# Predefined bank data (account number :(PIN, balance))
data = {
    '2993': ('1111', 1000),
    '2693': ('2222', 2000),
}

def handle_client(client_socket,addr):
    try:
        AUTH = False

        while True:
            request = client_socket.recv(1024).decode('utf-8').strip()
            if not request:
                break

            if not AUTH:
                account_number, pin = request.split(',')
                if account_number in data and data[account_number][0] == pin:
                    AUTH = True
                    client_socket.send(b'TRUE\n')
                else:
                    client_socket.send(b'False\n')
                    break
            if request:
                command, *args = request.split(',')
                print(command)

            if command == 'BALANCE':
                balance = data[account_number][1]
                client_socket.send(f'Balance: {balance}\n'.encode('utf-8'))

            elif command == 'DEPOSIT':
                amount = float(args[0])
                data[account_number] = (data[account_number][0],
data[account_number][1] + amount)
                client_socket.send(b'Deposit Successful\n')

            elif command == 'WITHDRAW':
                amount = float(args[0])
                if data[account_number][1] >= amount:
                    data[account_number] = (
                        data[account_number][0], data[account_number][1] -
amount)
                    client_socket.send(b'Withdrawal Successful\n')
```

```
                else:
                    client_socket.send(b'Insufficient Funds\n')

            elif command == 'LOGOUT':
                balance = data[account_number][1]
                client_socket.send(f'Final Balance: {balance}\n'.encode('utf-
8'))
                break
    finally:
        client_socket.close()
        print(f' {addr} Closed!')



ssock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ssock.bind(('0.0.0.0', 1234))
ssock.listen(5)
print('Server listening on port : 1234')

while True:
    csock, addr = ssock.accept()
    print(f'New Client connection At {addr}')
    client_handler = threading.Thread(target=handle_client,
args=(csock,addr))
    client_handler.start()
```

بعدها نقوم بانشاء كلاينت TCP الذي يتصل بالسيرفر الموجود في الكود السابق

1- الاتصال بالسيرفر:

نقوم بإنشاء سوكيت خاصة بالكلاينت من أجل الاتصال من خلاله بالسيرفر باستخدام التابع connect.

```
csock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
csock.connect(('127.0.0.1', 1234))
```

2- ارسال معلومات الدخول:

```
   account_number = input('Enter Your account number: ')
   pin = input('Enter Your PIN: ')
   csock.send(f'{account_number},{pin}\n'.encode('utf-8'))
```

3- استقبال نتيجة التحقق:

```
msg = csock.recv(1024).decode('utf-8').strip()
if msg == 'TRUE':
    print('Login successful!')
else:
    print('Authentication failed!')
    csock.close()
```

حلقة while من اجل القيام بالعمليات المصرفية بالتزامن مع السيرفر

```
while True:
    print("\nOptions:\n1. Check\n2. Deposit\n3. Withdraw \n4. Logout")
    choice = input("Enter operation: ")

    if choice == '1':
        csock.send(b'BALANCE\n')
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)

    elif choice == '2':
        amount = input('Enter amount to deposit: ')
        csock.send(f'DEPOSIT,{amount}\n'.encode('utf-8'))
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)

    elif choice == '3':
        amount = input('Enter amount to withdraw: ')
        csock.send(f'WITHDRAW,{amount}\n'.encode('utf-8'))
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)

    elif choice == '4':
        csock.send(b'LOGOUT\n')
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)
        break

    else:
        print('Invalid choice, please try again.')
```

4- إغلاق الاتصال:

يتم إغلاق الاتصال بعد تسجيل الخروج باستخدام التابع close

```
csock.close()
```

وبالتالي يكون شكل الكود النهائي:

```python
import socket

csock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
csock.connect(('127.0.0.1', 1234))

account_number = input('Enter Your account number: ')
pin = input('Enter Your PIN: ')
csock.send(f'{account_number},{pin}\n'.encode('utf-8'))

msg = csock.recv(1024).decode('utf-8').strip()
if msg == 'TRUE':
    print('Login successful!')
else:
    print('Authentication failed!')
    csock.close()


while True:
    print("\nOptions:\n1. Check\n2. Deposit\n3. Withdraw \n4. Logout")
    choice = input("Enter operation: ")

    if choice == '1':
        csock.send(b'BALANCE\n')
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)

    elif choice == '2':
        amount = input('Enter amount to deposit: ')
        csock.send(f'DEPOSIT,{amount}\n'.encode('utf-8'))
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)

    elif choice == '3':
        amount = input('Enter amount to withdraw: ')
        csock.send(f'WITHDRAW,{amount}\n'.encode('utf-8'))
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)

    elif choice == '4':
        csock.send(b'LOGOUT\n')
        msg = csock.recv(1024).decode('utf-8').strip()
        print(msg)
        break

    else:
        print('Invalid choice, please try again.')

csock.close()
```

الخرج:

اتصل عميلين في نفس اللحظة

```
Server listening on port : 1234
New Client connection At ('127.0.0.1', 55572)
2993
New Client connection At ('127.0.0.1', 55729)
2693
```

العميل الاول

```
Enter Your account number: 2993
Enter Your PIN: 1111
Login successful!

Options:
1. Check
2. Deposit
3. Withdraw
4. Logout
Enter operation: 1
Balance: 1000

Options:
1. Check
2. Deposit
3. Withdraw
4. Logout
Enter operation: 3
Enter amount to withdraw: 222
Withdrawal Successful

Options:
1. Check
2. Deposit
3. Withdraw
4. Logout
Enter operation: 4
Final Balance: 778.0

Process finished with exit code 0
```

العميل الثاني

```
Enter Your account number: 2693
Enter Your PIN: 2222
Login successful!

Options:
1. Check
2. Deposit
3. Withdraw
4. Logout
Enter operation: 2
Enter amount to deposit: 8000
Deposit Successful

Options:
1. Check
2. Deposit
3. Withdraw
4. Logout
Enter operation: 4
Final Balance: 10000.0

Process finished with exit code 0
```

السيرفر

```
Server listening on port : 1234
New Client connection At ('127.0.0.1', 55572)
2993
New Client connection At ('127.0.0.1', 55729)
2693
BALANCE
WITHDRAW
LOGOUT
 ('127.0.0.1', 55572) Closed!
DEPOSIT
LOGOUT
 ('127.0.0.1', 55729) Closed!
```

السؤال الثاني:

```python
from flask import Flask, render_template

app = Flask(__name__)

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

@app.route('/page2')
def page2():
    return render_template('2.html')

@app.route('/page3')
def page3():
    return render_template('3.html')

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

يمثل هذا الكود تطبيق ويب بسيط باستخدام إطار عمل Flask في لغة Python. يتألف التطبيق من ثلاث صفحات، حيث يتم عرض كل صفحة عن طريق نموذج HTML مختلف باستخدام التابع render_template.

يتم في البداية تعريف متغير app باستخدام الكلاس Flask وتمرير name إليه كمعرف للتطبيق. ثم يتم تعريف ثلاث توابع مختلفة باستخدام توجيهات route لتحديد الصفحات المختلفة في التطبيق. يتم استدعاء تابع render_template في كل تابع لتحميل ملف HTML المرتبط بكل صفحة.

يتحقق الشرط :'__main__' == __name__ if من أن التطبيق يعمل فقط عندما يتم تشغيله مباشرة من نفس الملف ويتم تشغيل التطبيق باستخدام تابع run الموجود في Flask.
السيرفر يعمل على العنوان http://127.0.0.1:5000

الصفحة الأولى:
هي ملف HTML يتم استخدامه في تطبيق Flask ويتضمن بعض العناصر المختلفة لتنسيق الصفحة ويحتوي الملف على العناصر التالية:

<DOCTYPE html!>: هذا يحدد نوع المستند كـ HTML5.

<html> و </html>: هذا يحدد بداية ونهاية الصفحة HTML.

<head> و </head>: هذا يحدد بداية ونهاية ترويسة الصفحة وتحتوي على عناصر التعريف مثل عنوان الصفحة ونمط CSS.

<title> و </title>: يتم وضع عنوان الصفحة بين هذه العناصر.

<"link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}>: يتم استخدام هذا العنصر لتضمين ملف نمط CSS الذي يحتوي على تنسيقات المظهر للموقع. يتم تضمين الملف بواسطة استخدام url_for لتوليد عنوان URL المناسب للملف.

link rel="stylesheet" >

<"href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css: يتم استخدام هذا العنصر لتضمين ملف نمط CSS الذي يحتوي على تنسيقات Bootstrap للموقع. يتم تضمين الملف من Bootstrap CDN.

<body> و </body>: هذا يحدد بداية ونهاية جسم الصفحة ويحتوي على عناصر HTML الفعلية التي يتم عرضها.

 و : هذا يحدد بداية ونهاية عنصر div الذي يحتوي على العناصر الأخرى في الصفحة.

" و : هذا يحدد بداية ونهاية عنصر div آخر يستخدم لإنشاء صفوف في الصفحة.

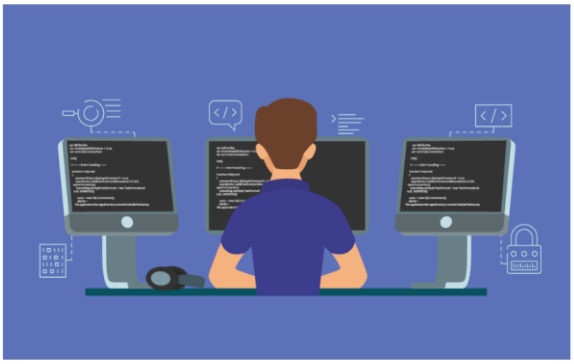
" و : هذا يحدد بداية ونهاية عنصر div يتم استخدامه لتحديد عمود في الصفحة. تم استخدام هذا المصفوفة في صفحة الرئيسية لعرض صورة ونص.

<img src="{{ url_for('static', filename='image.png') }}" alt="Image">: يتم استخدام هذا العنصر لعرض صورة على الصفحة. يتم تضمين الملف بواسطة استخدام url_for لتوليد عنوان URL المناسب للملف.

<h1> و </h1>: يتم استخدام هذا العنصر لإنشاء عنوان رئيسي.

 و : يتم استخدام هذا العنصر لإنشاء فقرات.

```
<!DOCTYPE html>
<html lang="ar">
<head>
    <title>الرئيسية الصفحة</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.cs
s">
</head>
<body>
    <div class="container">
       <div class="row">
          <div class="col-md-6">
             <img src="{{ url_for('static', filename='image.png') }}">
          </div>
          <div class="col-md-6">
             <h1>البرمجة لتعلم موقع أفضل</h1>
             <p> ميهوب احمد ،فرح  القوزي ناظم علي تقديم </p>
          <p><a href="{{ url_for('page2') }}">الموقع عن</a></p>
             <p><a href="{{ url_for('page3') }}">الكورسات</a></p>
          </div>
       </div>
    </div>
</body>
</html>
```



أفضل موقع لتعلم البرمجة

تقديم علي ناظم القوزي ،فرح احمد ميهوب

عن الموقع

الكورسات

الصفحة الثانية:

```
<!DOCTYPE html>
<html lang="ar">
<head>
   <title>الموقع عن</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.cs
s">
</head>
<body>
   <div class="container">
      <div class="row">
         <div class="col-md-6">
            <img src="{{ url_for('static', filename='image.png') }}"
alt="Image">
         </div>
         <div class="col-md-6">
            <h1> الموقع عن</h1>
            <p>تهدف الإنترنت عبر تعليمية منصة في البرمجة تعلم في موقعنا يتمثل
يتميز .العالم أنحاء جميع في والمتوسطين للمبتدئين البرمجة تعلم عملية تسهيل إلى
في بما البرمجة، لتعلم المجانية التعليمية المواد من واسعة مجموعة بتقديم الموقع
العملية والمشاريع العملية والتمارين المصورة الدروس ذلك.</p>
            <p>
والتمارين التعليمية الفيديوهات مثل متعددة تعليمية أساليب المنصة تستخدم
بيئة توفير إلى المنصة وتهدف .التعلم عملية لتسهيل العملية والمشاريع العملية
البرمجة تعلم في أهدافهم تحقيق على المتعلمين تساعد ومحفزة ودية تعليمية.</p>
      </div>
   </div>
</body>
</html>
```

عن الموقع

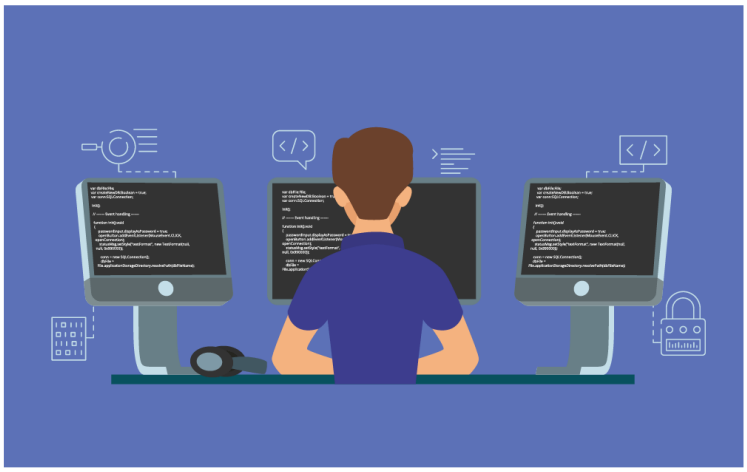
يتمثل موقعنا في تعلم البرمجة في منصة تعليمية عبر الإنترنت تهدف إلى تسهيل عملية تعلم البرمجة للمبتدئين والمتوسطين في جميع أنحاء العالم. يتميز الموقع بتقديم مجموعة واسعة من المواد التعليمية المجانية لتعلم البرمجة، بما في ذلك الدروس المصورة والتمارين العملية والمشاريع العملية.

تستخدم المنصة أساليب تعليمية متعددة مثل الفيديوهات التعليمية والتمارين العملية والمشاريع العملية لتسهيل عملية التعلم. وتهدف المنصة إلى توفير بيئة تعليمية ودية ومحفزة تساعد المتعلمين على تحقيق أهدافهم في تعلم البرمجة.

الصفحة الثالثة:

```
<!DOCTYPE html>
<html lang="ar">
<head>
   <title>الكورسات</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.cs
s">
</head>
<body>
   <div class="container">
      <div class="row">
         <div class="col-md.6">
            <img src="{{ url_for('static', filename='image.png') }}"
alt="Image">
         </div>
         <div class="col-md-6">
            <h1>الكورسات</h1>
            <ul>
  <li>دورة تعلم HTML و CSS</li>
  <li>دورة تعلم JavaScript</li>
```

```
  <li>دورة تعلم Python</li>
  <li>دورة تعلم PHP</li>
</ul>
          </div>
       </div>
    </div>
</body>
</html>
```



الكورسات

دوره تعلم HTML و CSS

دوره تعلم JavaScript

دوره تعلم Python

دوره تعلم PHP

ملف الcss

```
.container {
    margin-top: 50px;
    text-align: right;
}
```

- .container

- تحديد هامش علوي للعناصر التي تحمل الفئة container.بقيمة 50 بكسل.

- تحديد محاذاة النص داخل العناصر التي تحمل الفئة container.إلى اليمين.

```
img {
    max-width: 100%;
    height: auto;
}
```

- img
  - تحديد أقصى عرض ممكن للصور داخل الصفحة.
  - تحديد الارتفاع تلقائيا على أساس العرض لتفادي تشويه الصورة.

```
h1 {
    font-size: 36px;
    color: #0066cc;
}

p {
    font-size: 18px;
    line-height: 1.5;
    color: #333;
}

a {
    color: #0066cc;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

ul {
  list-style: none;
  margin: 0;
  padding: 0;
}

li {
  font-family: Arial, sans-serif;
  font-size: 18px;
  color: #333;
  margin-bottom: 10px;
  padding-left: 20px;
  background-repeat: no-repeat;
  background-position: left center;
}
```

- h1

- تحديد حجم العنوان الرئيسي ليكون 36 بكسل.
- تحديد لون النص ليكون #0066cc

- p
- تحديد حجم الفقرات ليكون 18 بكسل.
- تحديد ارتفاع الخط ليكون 1.5 مرة الحجم.
- تحديد لون النص ليكون #333.

- a
- تحديد لون الروابط ليكون #0066cc
- حذف أي تزيينات نصية من الروابط.

- a:hover
- تحديد تحتية النص عند تحويم المؤشر فوق الروابط.

- ul
- حذف علامات الترقيم الافتراضية من القائمة.
- تحديد الهامش الداخلي والخارجي للقائمة ليكون صفر.

- li
- تحديد حجم النص ليكون 18 بكسل.
- تحديد لون النص ليكون #333.
- تحديد هامش علوي واسفل لكل عنصر في القائمة بقيمة 10 بكسل.
- تحديد هامش إلى اليسار بقيمة 20 بكسل.
- تحديد صورة كخلفية لكل عنصر في القائمة باستخدام خاصية background-image.