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

بشرى عقل سلامه 1519

السؤال 1:

1. استيراد المكتبتين socketو threadingاللتان ستُستخدَمان في تهيئة الخادم وإدارة الاتصالات مع العملاء.
2. تعريف قاموس bank_accountsالذي يحتوي على الحسابات البنكية المسبقة الفرض مع كلمات المرور والأرصدة.
3. تعريف تابع handle_clientالذي سيتعامل مع كل عميل على حدة. هذا التابع يتضمن الخطوات التالية:
   - طباعة عنوان العميل المتصل.
   - إجراء عملية المصادقة: يطلب من العميل إدخال اسم الحساب وكلمة المرور، ويتحقق من صحة المعلومات.
   - بعد المصادقة الناجحة، يدخل العميل في حلقة تشغيل تتيح له التحقق من الرصيد، والإيداع، والسحب، والخروج.
4. إنشاء الخادم وتهيئته:
   - إنشاء مقبس خادم باستخدام .(socket.socket
   - تعيين خيار إعادة استخدام العنوان.
   - ربط المقبس بعنوان محدد (في هذه الحالة '0.0.0.0' والمنفذ 5555).
   - بدء الاستماع على المقبس للاتصالات الواردة.
   - طباعة رسالة لإشعار بدء تشغيل الخادم.
5. بدء قبول الاتصالات الواردة في حلقة لانهائية:
   - قبول الاتصال الواردة باستخدام .(server_socket.accept
   - إنشاء ثريد جديد لمعالجة كل عميل باستخدام (threading.Threadمع تمرير handle_clientكتابع لمعالجة العميل.
   - بدء تشغيل الثريد الجديد.

```
import socket
import threading

# Pre-defined bank accounts with balances
bank_accounts = {
    "John": {"password": "pass123", "balance": 500.0},
    "Alice": {"password": "alice456", "balance": 1000.0},
    "Bob": {"password": "bob789", "balance": 750.0}
}
```

```
# Function to handle each client
def handle_client(client_socket, client_address):
    print(f"New client connected from {client_address}")

    authenticated = False
    account_name = ""

    # Authentication loop
    while not authenticated:
        client_socket.send("Enter your account name and password (space-separated): ".encode())
        credentials = client_socket.recv(1024).decode().strip().split()

        if len(credentials) == 2:
            account_name, password = credentials
            if account_name in bank_accounts and bank_accounts[account_name]["password"] ==
password:
                authenticated = True
                client_socket.send("Authentication successful.\n".encode())
            else:
                client_socket.send("Invalid account name or password. Try again.\n".encode())
        else:
            client_socket.send("Invalid input. Try again.\n".encode())

    # Operations loop
    while True:
        client_socket.send("Choose an option (1: Check Balance, 2: Deposit, 3: Withdraw, 4:
Quit): ".encode())
        option = client_socket.recv(1024).decode().strip()

        if option == "1":
            balance = bank_accounts[account_name]["balance"]
            client_socket.send(f"Your balance is: {balance:.2f}\n".encode())
        elif option == "2":
            client_socket.send("Enter amount to deposit: ".encode())
            amount = float(client_socket.recv(1024).decode().strip())
            bank_accounts[account_name]["balance"] += amount
            client_socket.send(
                f"Deposit successful. New balance is:
{bank_accounts[account_name]['balance']:.2f}\n".encode())
        elif option == "3":
            client_socket.send("Enter amount to withdraw: ".encode())
            amount = float(client_socket.recv(1024).decode().strip())
            if amount > bank_accounts[account_name]["balance"]:
                client_socket.send("Insufficient funds.\n".encode())
            else:
                bank_accounts[account_name]["balance"] -= amount
                client_socket.send(
                    f"Withdrawal successful. New balance is:
{bank_accounts[account_name]['balance']:.2f}\n".encode())
        elif option == "4":
            client_socket.send(f"Final balance is:
{bank_accounts[account_name]['balance']:.2f}\nGoodbye!\n".encode())
            break
        else:
            client_socket.send("Invalid option. Try again.\n".encode())

    client_socket.close()
    print(f"Client from {client_address} disconnected.")


# Create and start the server
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind(('0.0.0.0', 5555))
server_socket.listen()

print("Server started. Waiting for client connections...")

while True:
    client_socket, client_address = server_socket.accept()
```

```
    client_thread = threading.Thread(target=handle_client, args=(client_socket, client_address))
    client_thread.start()
```

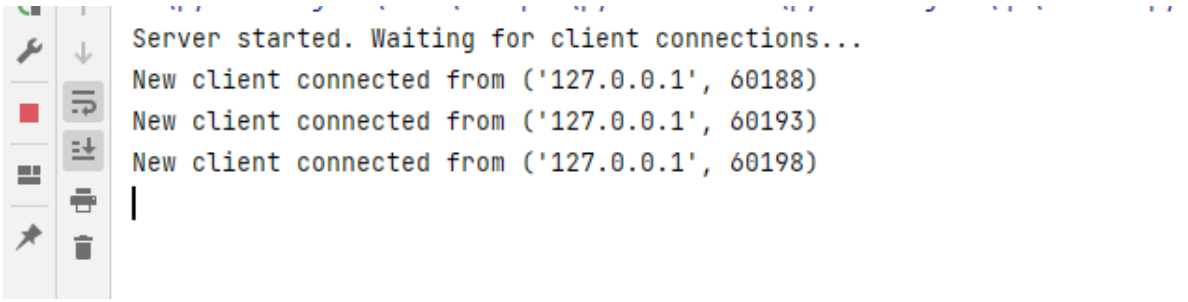
العميل:

1. برنامج عميل الذي يتواصل مع الخادم على منفذ 5555 على الجهاز المحلي.(localhost)
2. العميل يدخل في حلقة while True التي تستمر إلى أن يتوقف الخادم عن إرسال الرسائل.
3. عندما يستقبل العميل رسالة من الخادم، يتم طباعتها على الشاشة.
4. يتم فحص الرسالة المستقبلة للكشف عن أي طلبات إدخال من المستخدم مثل اسم الحساب وكلمة المرور أو المبلغ أو اختيار خيار من قائمة. في هذه الحالات، يطلب من المستخدم إدخال البيانات المطلوبة، والتي يتم إرسالها إلى الخادم.
5. في النهاية، يتم إغلاق المقبس.

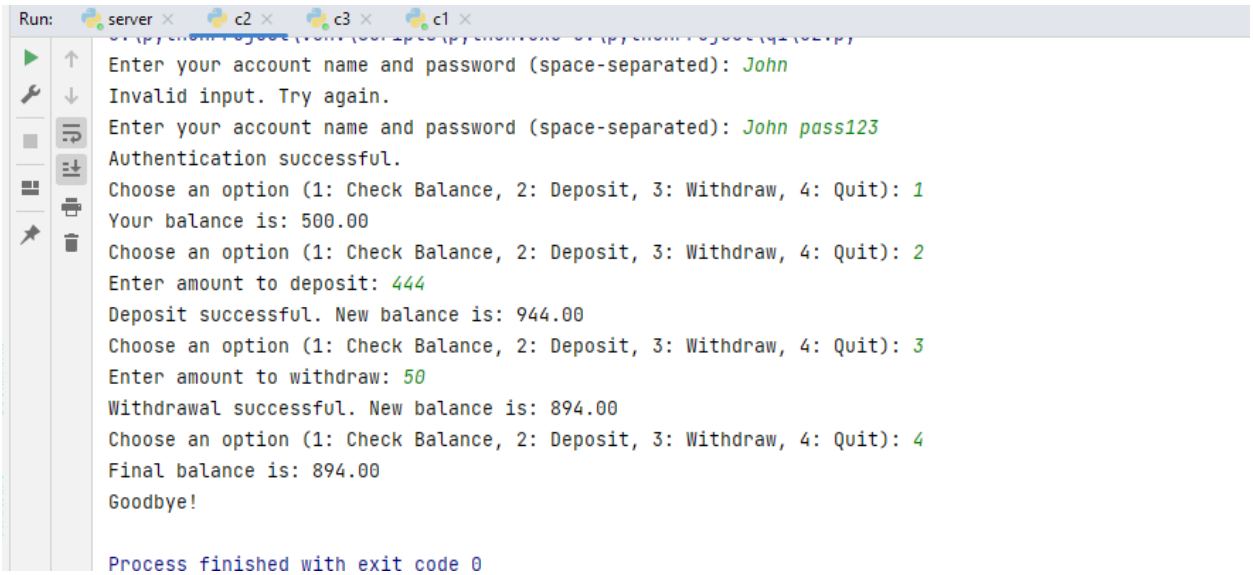
```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 5555))

try:
    while True:
        message = client_socket.recv(1024).decode()
        if not message:
            break
        print(message, end="")

        if "Enter your account name and password" in message or "Enter amount" in message or
"Choose an option" in message:
            user_input = input()
            client_socket.send(user_input.encode())

finally:
    client_socket.close()
```

الخرج

السيرفر بعد اتصال 3 عملاء



نموذج عميل قام بجميع العمليات علما ان العميلين الاخرين متصلين



السؤال 2

نقوم بإنشاء ملف Flask ونسميه "app.py"
نقوم بإنشاء مجلد جديد ونسميه "templates"
نقوم بإنشاء ملفات HTML للصفحات الخمس داخل المجلد "templates".
نقوم بإنشاء مجلد جديد ونسميه "static"
نقوم بإنشاء ملف CSS وسمِّه "style.css" داخل المجلد "static".
نقوم بتحميل ملف ونضعه داخل المجلد "static"

تم استخدام لغة HTML لإنشاء صفحات الموقع وتحديد الهيكل العام للصفحات وفيما يلي شرح لبعض الأكواد والتنسيقات التي تم استخدامها في صفحات HTML:

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

<html>
هذا العنصر يحدد بداية مستند HTML وينهيه.

<head>
هذا العنصر يحتوي على المعلومات التي لا تظهر مباشرة للمستخدمين على الصفحة، مثل عنوان الصفحة، روابط CSS و JavaScript، وأوامر أخرى.

<title>
هذا العنصر يحدد عنوان الصفحة الذي يظهر في علامة التبويب للصفحة في المتصفح.

<body>
هذا العنصر يحدد الجزء الرئيسي من صفحة HTML التي تحتوي على جميع المحتويات الرئيسية مثل النصوص والصور والروابط والجداول.

<header>
هذا العنصر يحدد جزء الصفحة الذي يحتوي على عنوان الصفحة وغالباً ما يكون في الجزء العلوي من الصفحة.

<nav>
هذا العنصر يحدد الروابط التي تحتوي على قائمة التنقل في الموقع.

<h1> to <h6>
هذه العناوين تستخدم لتحديد مستوى العنوان في الصفحة، حيث أن <h1> هو الأعلى و <h6> هو الأدنى.

<p>
هذا العنصر يستخدم لإدخال النصوص في الصفحة.

<img>
هذا العنصر يستخدم لإدراج صورة في الصفحة.

<a>
هذا العنصر يستخدم لإنشاء رابط في الصفحة، ويمكن أن يحتوي على نص أو صورة أو أي عنصر آخر.

 و
هذه العناصر تستخدم لإنشاء قوائم غير مرتبة، حيث يتم استخدام <ul> لتحديد القائمة و <li> لتحديد كل عنصر في القائمة.

 و
هذه العناصر تستخدم لإنشاء قوائم مرتبة، حيث يتم استخدام <ol> لتحديد القائمة و <li> لتحديد كل عنصر في القائمة.

محتوى الصفحة الرئيسية index.html

```
<!DOCTYPE html>
<html lang="ar" dir="rtl">
<head>
```

```
    <meta charset="UTF-8">
    <title>الرئيسية</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
       <header>
          <img src="{{ url_for('static', filename='logo.png') }}" alt="Flask
App" class="img-fluid">
          <nav>
          <ul>
             <li><a href="#">الرئيسية</a></li>
             <li><a href="{{ url_for('about') }}">حول</a></li>
             <li><a href="{{ url_for('contact') }}">بنا اتصل</a></li>
          </ul>
       </nav>

       </header>
       </div>
    <div class="container">
          <h1>بشرى عقل سلامه 1519</h1>
          <h2>الثانية الوظيفة من الثاني بالسؤال الخاص موقعي</h2>
          <p>التي الويب تصميم مفاهيم فيها استخدمت صفحات 3 يحوي الموقع هذا
الشبكات برمجة مقرر من تعلمتها</p>

    </div>
</body>
</html>
```

محتوى صفحة about.html

```
<!DOCTYPE html>
<html lang="ar" dir="rtl">
```

```
<head>
    <meta charset="UTF-8">
    <title>حول</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <header>
            <img src="{{ url_for('static', filename='logo.png') }}" alt="Flask
App" class="img-fluid">
            <h1>حول</h1>
        </header>

        <nav>
            <ul>
                <li><a href="{{ url_for('index') }}">الرئيسية</a></li>
                <li><a href="#">حول</a></li>
                <li><a href="{{ url_for('contact') }}">بنا اتصل</a></li>
            </ul>
        </nav>

        <main>
            <h2>الموقع عن</h2>
            <p>موقع باستخدام الفلاسك أنشائته بشرى عقل سلامه </p>
        </main>
    </div>
</body>
</html>
```

حول

الرئيسية حول اتصل بنا

عن الموقع

موقع باستخدام الفلاسك أنشائته بشرى عقل سلامه

محتوى صفحة contact.html

```
<!DOCTYPE html>
<html lang="ar" dir="rtl">
<head>
    <meta charset="UTF-8">
    <title>بـنـا  اتـصل</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
       <header>
          <img src="{{ url_for('static', filename='logo.png') }}" alt="Flask
App" class="img-fluid">
          <h1>Contact</h1>
       </header>

       <nav>
          <ul>
             <li><a href="{{ url_for('index') }}">الـرئـيسية</a></li>
             <li><a href="{{ url_for('about') }}">حول</a></li>
             <li><a href="#">بـنـا  اتـصل</a></li>
          </ul>
       </nav>

       <main>
          <h2>الـهـاتـف رقـم</h2>          <p>+963 991 233 693</p>
       </main>
    </div>
</body>
</html>
```

Contact

الرئيسية حول اتصل بنا

رقم الهاتف

693 233 991 963+

محتوى ملف style.css

header { text-align: center; margin-top: 20px; }
هذا الكود ينص على توسيط جميع النصوص في عنصر <header> وتحديد المسافة بين العنصر والأعلى بـ 20 بكسل.

nav { margin-top: 20px; }
هذا الكود ينص على تحديد المسافة بين العنصر <nav> والأعلى بـ 20 بكسل.

nav ul { list-style-type: none; display: flex; justify-content: center; }
هذا الكود ينص على إزالة نقاط التعليم الخاصة بالقائمة وتحديد نوع العرض بشكل قائمة وتوسيط العناصر في القائمة.

nav ul li { margin-right: 20px; }
هذا الكود ينص على تحديد المسافة بين عناصر القائمة بـ 20 بكسل في الجانب الأيمن.

main { margin-top: 20px; text-align: center; }
هذا الكود ينص على تحديد المسافة بين عنصر <main> والأعلى بـ 20 بكسل وتوسيط النصوص التي تحتويه.

img { max-width: 100%; height: auto; }
هذا الكود ينص على تحديد أقصى عرض للصورة بنسبة 100٪ من حجم العنصر الأب وتحديد الارتفاع بشكل تلقائي للحفاظ على نسبة العرض إلى الارتفاع الأصلية للصورة.

```
header {
   text-align: center;
   margin-top: 20px;
}

nav {
   margin-top: 20px;
}

nav ul {
   list-style-type: none;
   display: flex;
   justify-content: center;
}

nav ul li {
   margin-right: 20px;
}

main {
   margin-top: 20px;
   text-align: center;
}

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

محتوى ملف app.py

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    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)
```

from flask import Flask, render_template
يستورد فئتين من مكتبة Flask، وهي Flask و render_template، والتي تستخدم لإنشاء التطبيق وتقديم الصفحات على التوالي.

app = Flask(__name__)
ينشئ تطبيق Flask باستخدام اسم الملف الحالي كاسم التطبيق.

@app.route('/')
يستخدم لتحديد المسار الذي يتم الوصول إليه عندما يتم الوصول إلى الصفحة الرئيسية للموقع.

def index():
يعرف تابع باسم index والتي سيتم استدعاؤه عند الوصول إلى المسار الرئيسي / المحدد سابقاً.

return render_template('index.html')
هذا الكود يقوم بتقديم صفحة HTML المسماة index.html باستخدام تابع render_template الموجود في Flask.

@app.route('/about')
يستخدم لتحديد المسار الذي يتم الوصول إليه عندما يتم الوصول إلى صفحة حول الموقع.

def about():
هذا الكود يعرف تابع باسم about والتي سيتم استدعاؤه عند الوصول إلى المسار /about المحدد سابقا.

return render_template('about.html')
هذا الكود يقوم بتقديم صفحة HTML المسماة about.html باستخدام تابع render_template

@app.route('/contact')
هذا الكود يستخدم لتحديد المسار الذي يتم الوصول إليه عندما يتم الوصول إلى صفحة الاتصال بنا.

def contact():
هذا الكود يعرف تابع باسم contact والتي سيتم استدعاؤه عند الوصول إلى المسار /contact المحدد سابقاً.

return render_template('contact.html')
هذا الكود يقوم بتقديم صفحة HTML المسماة contact.html باستخدام تابع render_template

if __name__ == '__main__':
app.run(debug=True)
هذا الكود يتحقق من أن البرنامج يتم تشغيله مباشرة وليس استدعاؤه من ملف آخر، ويشغل التطبيق باستخدام تابع run() بإضافة خاصية debug=True لتمكين وضع التصحيح الآلي Debugging mode لتسهيل عملية تطوير التطبيق.