Syrian Arab Republic

Lattakia - Tishreen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No2

الجمهورية العربية السورية

اللاذقية -جامعـــة تشريـــــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة2 برمجة شبكات

الوظيفة الثانية

محمد أحمد علي 2039

ناصر أكثم حسن 2474

السؤال الأول

كود السيرفر

```
import socket
import threading

# Predefined bank accounts (account number -> (PIN, balance))
accounts = {
    '12345': ('1111', 1000),
    '67890': ('2222', 2000),
}

# Lock for synchronizing access to account balances
lock = threading.Lock()


def handle_client(client_socket):
    try:
        authenticated = False
        account_number = None

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

            if not authenticated:
                account_number, pin = request.split(',')
                if account_number in accounts and accounts[account_number][0] == pin:
                    authenticated = True
                    client_socket.send(b'Authenticated\n')
                else:
                    client_socket.send(b'Authentication Failed\n')
                    break
            else:
                command, *args = request.split(',')

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

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

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

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

```python
def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('0.0.0.0', 9999))
    server.listen(5)
    print('Server listening on port 9999')

    while True:
        client_socket, addr = server.accept()
        print(f'Accepted connection from {addr}')
        client_handler = threading.Thread(target=handle_client, args=(client_socket,))
        client_handler.start()


if __name__ == '__main__':
    start_server()
```

هذا الكود يُنشئ خادم بنكي بسيط باستخدام مكتبتي socket وthreading في البداية تعريف قاموس accounts الذي يُمثل الحسابات البنكية المُسبقة التعريف. في هذا القاموس يُمثل المفتاح رقم الحساب والقيمة هي مزيج من رقم التعريف الشخصي (PIN) والرصيد المتوفر في الحساب.

لضمان التعديلات على رصيد الحسابات، يتم استخدام قفل threading.Lock() هذا القفل سيتم استخدامه عند تنفيذ العمليات البنكية كالاستعلام عن الرصيد، والإيداع، والسحب لقفل المتغير accounts من اجل ضمان وصول اكثر من ثريد اليه في نفس اللحظة.

التابع handle_client() هو المسؤول عن معالجة طلبات العملاء. يبدأ هذا التابع بالتحقق من صحة رقم الحساب وكلمة المرور المُرسلة من قبل العميل. إذا نجح التحقق، يتم تحديد الحساب على أنه مُصادق عليه، وبعد ذلك يُمكن للعميل إجراء العمليات البنكية المختلفة.

العمليات البنكية المُتاحة هي: الاستعلام عن الرصيد، والإيداع، والسحب، والخروج. عند تنفيذ هذه العمليات، يتم استخدام القفل lock لضمان التعديلات على رصيد الحساب.

التابع start_server() هو المسؤول عن إنشاء مقبس الخادم وجعله في وضع الاستماع على المنفذ 9999. عند استقبال اتصال جديد، يتم إنشاء ثريد جديد لمعالجة طلبات العميل باستخدام التابع handle_client

كود الكلاينت

```python
import socket
def start_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 9999))

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

    response = client.recv(1024).decode('utf-8').strip()
    if response == 'Authenticated':
        print('Login successful!')
    else:
        print('Authentication failed!')
        client.close()
        return

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

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

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

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

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

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

    client.close()


if __name__ == '__main__':
    start_client()
```

أولاً، يتم تعريف التابع start_client() الذي هو نقطة الإدخال الرئيسية للعميل. في هذا التابع يتم إنشاء مقبس عميل جديد باستخدام socket.socket() ويتم الاتصال بالخادم الموجود على عنوان IP 127.0.0.1 والمنفذ 9999

بعد ذلك يطلب البرنامج من المُستخدم إدخال رقم الحساب والرقم التعريفي الشخصي(PIN) ، ويرسلهما إلى الخادم.

إذا نجح التحقق من صحة البيانات المُرسلة، يُعرض رسالة "Login successful!" على الخرج. وإذا فشل التحقق يُعرض رسالة "Authentication failed!" ويتم إغلاق اتصال العميل.

إذا تم المصادقة بنجاح، يدخل العميل في حلقة while التي توفر القائمة التالية للعمليات البنكية:

1. Check Balance يرسل العميل رسالة "BALANCE" إلى الخادم ويعرض الرصيد الحالي للحساب.
2. Deposit Money يطلب العميل المبلغ المُراد إيداعه، ويرسله إلى الخادم مع رسالة "DEPOSIT"
3. Withdraw Money يطلب العميل المبلغ المُراد سحبه، ويرسله إلى الخادم مع رسالة WITHDRAW
4. Logout يرسل العميل رسالة "LOGOUT" إلى الخادم ويتم إغلاق الاتصال.

إذا تم إدخال خيار غير صالح، يعرض رسالة "Invalid choice, please try again."

في النهاية، يتم إغلاق اتصال العميل باستخدام client.close

الخرج:

السيرفر واتصال عميل

```
Run: s1 c1
C:\pythonProject\venv\Scripts\python.exe C:\pythonProject\s1.py
Server listening on port 9999
Accepted connection from ('127.0.0.1', 55545)
```

السيرفر بعد اتصال عميلين في نفس الوقت

```
Run: s1 c1 c12
C:\pythonProject\venv\Scripts\python.exe C:\pythonProject\s1.py
Server listening on port 9999
Accepted connection from ('127.0.0.1', 55545)
Accepted connection from ('127.0.0.1', 55609)
```

العميل الأول بعد نجاح المصادقة

```
Run: s1 c1 c12
C:\pythonProject\venv\Scripts\python.exe C:\pythonProject\c1.py
Enter account number: 12345
Enter PIN: 1111
Login successful!

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice:
```

العميل الأول بعد التحقق من الرصيد وشحن الرصيد وإعادة التحقق من الرصيد:

```
Run:  s1  c1  c12
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 1
Balance: 1000

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 2
Enter amount to deposit: 100
Deposit Successful

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 1
Balance: 1100.0

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Version Control  Run  Python Packages
```

العميل الثاني بعد تسجيل الدخول والقيام بعمللية سحب والعميل الأول متصل وبعدها يقوم بتسجيل الخروج ويتبعه العميل الاول

```
s1  c1  c12
C:\pythonProject\venv\Scripts\python.exe C:\pythonProject\c12.py
Enter account number: 67890
Enter PIN: 2222
Login successful!

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 3
Enter amount to withdraw: 40
Withdrawal Successful

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 4
Final Balance: 1960.0

Process finished with exit code 0
```

Run: s1 c1 c12

```
Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 2
Enter amount to deposit: 100
Deposit Successful

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 1
Balance: 1100.0

Options:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Logout
Enter choice: 4
Final Balance: 1100.0

Process finished with exit code 0
```

السؤال الثاني

يتم إنشاء التطبيق باستخدام Flask ويتم تعريف التطبيق باستخدام الكود التالي:

```
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)
```

حيث:

يتم اشتقاق غرض من الكلاس Flask باستخدام الأمر التالي:

app = Flask(__name__)

ويتم تعريف المسارات (routes) بواسطة الكود التالي:

```
@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')
```

يتم تعريف التابع render_template الذي يستخدم لعرض صفحات HTML حيث يتم استدعاء هذا التابع في التوابع المعرفة للمسارات على سبيل المثال في التابع index تم استدعاء التابع render_template لعرض صفحة HTML الخاصة بالصفحة الرئيسية حيث يتم تنفيذ التابع index عند ورود الرابط الأساسي من المتصفح.

يتم تشغيل التطبيق باستخدام التابع run حيث يتم تنفيذه اذا كان التشغيل مباشرة من هذا الكود:

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

الخطوة 2: إنشاء الصفحات HTML و CSS و Bootstrap

تم إنشاء ثلاث صفحات بسيطة: الصفحة الرئيسية (index.html) وصفحة "حول" (about.html) وصفحة "اتصل بنا" (contact.html).

نضع الصفحات في مجلد templates

تم انشاء ملف css بسيط نضعه في مجلد static في المجلد css.

ملف الcss:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f7f7f7;
    color: #333;
}

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

.btn {
    margin-right: 10px;
}
```

يتم تعريف خصائص الجسم (body) بواسطة الكود التالي:

```
body {
    font-family: Arial, sans-serif;
```

```
  background-color: #f7f7f7;
  color: #333;
}
```

يتم تعريف خصائص عنصر الحاوية (container) بواسطة الكود التالي:

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

يتم تعريف خصائص الزر (button) بواسطة الكود التالي:

```
.btn {
    margin-right: 10px;
}
```

الصفحة الرئيسية (index.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>About | Flask Website</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1>About</h1>
        <p>This website was built using Flask.</p>
        <a href="{{ url_for('index') }}" class="btn btn-primary">Home</a>
        <a href="{{ url_for('contact') }}" class="btn btn-
primary">Contact</a>
    </div>
</body>
</html>
```

تم استدعاء ملف CSS الخاص بالصفحة بواسطة الكود التالي:

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
```

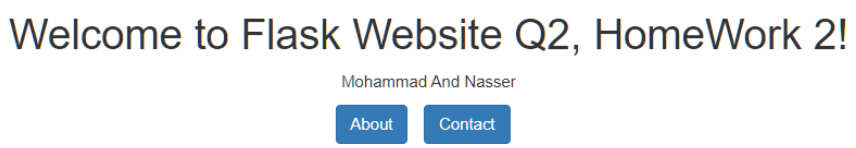
تم استدعاء Bootstrap بواسطة الكود التالي:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

تم إضافة روابط لصفحات حول و اتصل بنا باستخدام الكود التالي:

```
<a href="{{ url_for('about') }}" class="btn btn-primary">About</a>
    <a href="{{ url_for('contact') }}" class="btn btn-primary">Contact</a>
```

الشكل التالي هو شكل الصفحة في المتصفح

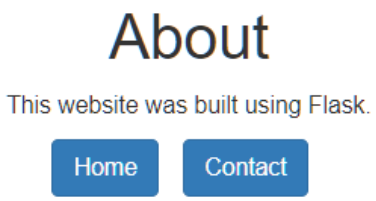


صفحة حول (about.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>About | Flask Website</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1>About</h1>
        <p>This website was built using Flask.</p>
        <a href="{{ url_for('index') }}" class="btn btn-primary">Home</a>
        <a href="{{ url_for('contact') }}" class="btn btn-
primary">Contact</a>
    </div>
</body>
</html>
```

تم إضافة روابط لصفحة الرئيسية وصفحة اتصل بنا

الشكل التالي هو شكل الصفحة في المتصفح

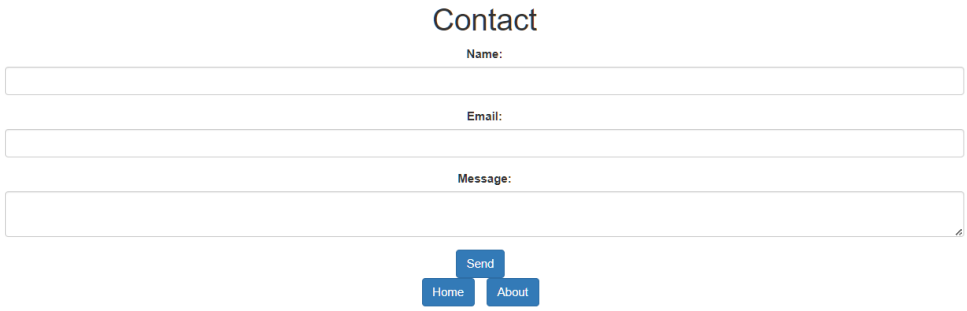


صفحة اتصل بنا (contact.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Contact | Flask Website</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css')
}}">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1>Contact</h1>
        <form action="" method="post">
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" name="name" id="name" class="form-control"
required>
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" name="email" id="email" class="form-control"
required>
            </div>
            <div class="form-group">
                <label for="message">Message:</label>
                <textarea name="message" id="message" class="form-control"
required></textarea>
            </div>
            <button type="submit" class="btn btn-primary">Send</button>
        </form>
        <a href="{{ url_for('index') }}" class="btn btn-primary">Home</a>
```

```
        <a href="{{ url_for('about') }}" class="btn btn-primary">About</a>
    </div>
</body>
</html>
```

تم إنشاء نموذج اتصال يتضمن حقول الاسم والبريد الإلكتروني والرسالة باستخدام الكود التالي:

```
<form action="" method="post">
        <div class="form-group">
          <label for="name">Name:</label>
          <input type="text" name="name" id="name" class="form-control" required>
        </div>
        <div class="form-group">
          <label for="email">Email:</label>
          <input type="email" name="email" id="email" class="form-control" required>
        </div>
        <div class="form-group">
          <label for="message">Message:</label>
          <textarea name="message" id="message" class="form-control" required></textarea>
        </div>
        <button type="submit" class="btn btn-primary">Send</button>
      </form>
```

الشكل التالي هو شكل الصفحة في المتصفح