Name: Sara Najjar,2974, Number:, Submitted To GitHub:_ _____

Name: Shaza ramadan, Number:2973, Submitted To GitHub:_ _____

Second Network Programming Homework

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.

Notes:

• Write a brief report describing the design choices you made and any challenges faced during implementation.
• You can choose to create a TCP Server/Client Bank ATM application or any other appropriate application that fulfills all requirements.

Server.py > ...

```python
import socket
import threading
# Bank account details
accounts = {
    '123456': {'balance': 1000, 'password': 'pass123'},
    '789012': {'balance': 500, 'password': 'pass456'}
}
# Function to handle client requests
def handle_client(client_socket):
    while True:
        # Receive client request
        request = client_socket.recv(1024).decode()

        # Parse request
        req_parts = request.split()
        command = req_parts[0]

        # Authenticate client
        if command == 'LOGIN':
            account_number = req_parts[1]
            password = req_parts[2]
            if account_number in accounts and accounts[account_number]['password'] == password:
                client_socket.send("Authenticated".encode())
            else:
                client_socket.send("Invalid credentials".encode())
        # Check balance
        elif command == 'BALANCE':
            account_number = req_parts[1]
            balance = accounts[account_number]['balance']
            client_socket.send(f"Balance: {balance}".encode())
        # Deposit
        elif command == 'DEPOSIT':
            account_number = req_parts[1]
            amount = int(req_parts[2])
            accounts[account_number]['balance'] += amount
            client_socket.send("Deposit successful".encode())
        # Withdraw
        elif command == 'WITHDRAW':
            account_number = req_parts[1]
            amount = int(req_parts[2])
            if accounts[account_number]['balance'] >= amount:
                accounts[account_number]['balance'] -= amount
                client_socket.send("Withdrawal successful".encode())
            else:
                client_socket.send("Insufficient funds".encode())
        # Close connection
        elif command == 'EXIT':
            break
    # Send final balance and close connection
    client_socket.send(f"Final balance: {accounts[account_number]['balance']}".encode())
    client_socket.close()
# Main function to start the server
```

```python
def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('0.0.0.0', 9999))
    server.listen(5)
    print("Server started")
    while True:
        client_socket, addr = server.accept()
        print(f"Connection from {addr}")
        # Start a new thread to handle the client
        client_handler = threading.Thread(target=handle_client, args=(client_socket,))
        client_handler.start()
if __name__ == "__main__":
    main()
```

client.py > ...

```python
import socket
def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('localhost', 9999))  # Replace 'localhost' with the server's IP if running remotely
    # Login
    account_number = input("Enter your account number: ")
    password = input("Enter your password: ")
    login_request = f"LOGIN {account_number} {password}"
    client.send(login_request.encode())
    response = client.recv(1024).decode()
    print(response)
    # If authenticated, allow banking operations
    if response == "Authenticated":
        while True:
            print("\nChoose operation:")
            print("1. Check balance")
            print("2. Deposit")
            print("3. Withdraw")
            print("4. Exit")
            choice = input("Enter choice: ")
            if choice == '1':
                client.send(f"BALANCE {account_number}".encode())
                balance_response = client.recv(1024).decode()
                print(balance_response)
            elif choice == '2':
                amount = input("Enter amount to deposit: ")
                client.send(f"DEPOSIT {account_number} {amount}".encode())
                deposit_response = client.recv(1024).decode()
                print(deposit_response)
```

```
            elif choice == '3':
                amount = input("Enter amount to withdraw: ")
                client.send(f"WITHDRAW {account_number} {amount}".encode())
                withdraw_response = client.recv(1024).decode()
                print(withdraw_response)
            elif choice == '4':
                client.send("EXIT".encode())
                final_balance_response = client.recv(1024).decode()
                print(final_balance_response)
                break
            else:
                print("Invalid choice")
    else:
        print("Authentication failed")
    client.close()
if __name__ == "__main__":
    main()
```

```
Enter your account number: 123456
Enter your password: pass123
Authenticated

Choose operation:
1. Check balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 3
Enter amount to withdraw: 500
Withdrawal successful

Choose operation:
1. Check balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Balance: 500

Choose operation:
1. Check balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 4
Final balance: 500
```

```
Enter your account number: 789012
Enter your password: pass456
Authenticated

Choose operation:
1. Check balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 2
Enter amount to deposit: 500
Deposit successful

Choose operation:
1. Check balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Balance: 1000

Choose operation:
1. Check balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 4
Final balance: 1000
```

```
Server started
Connection from ('127.0.0.1', 64732)
Connection from ('127.0.0.1', 64755)
```

السؤال الأول:

السيرفر:

إنشاء خادم بنكي أولاً، تم تعريف حسابات بنكية بمعلوماتها، حيث يحتوي كل حساب على رقم الحساب، الرصيد وكلمة المرور. يتم تخزين هذه الحسابات في قاموس.

ثم تم تعريف دالة لمعالجة طلبات العملاء. تستمر هذه الدالة في تلقي طلبات العملاء عبر مقبس الشبكة، وتحلل الطلب المطلوب تنفيذه. تبدأ الأوامر بأمر تسجيل الدخول الذي يطلب من العميل تقديم رقم الحساب وكلمة المرور للتحقق من صحة بياناته. في حالة نجاح عملية التحقق، يتم إعلام العميل بأنه تم المصادقة عليه، وإلا يتم إخباره بأن البيانات غير صحيحة.

إذا طلب العميل معرفة الرصيد، يتم إرسال الرصيد الحالي للحساب. أما في حالة الإيداع، فيتم زيادة رصيد الحساب بالمبلغ المودع، ويتم إعلام العميل بنجاح العملية. في حالة السحب، يتم التحقق أولاً من توفر الرصيد الكافي، وإذا كان الرصيد كافياً، يتم خصم المبلغ من الحساب وإبلاغ العميل بنجاح العملية، وإلا يتم إخباره بعدم توفر رصيد كافٍ. يمكن للعميل أيضًا إنهاء الاتصال بإرسال أمر "EXIT"، حيث يتم إرسال الرصيد النهائي للعميل قبل إغلاق الاتصال.

أخيراً، تحتوي الدالة الرئيسية على إعدادات الخادم، حيث يتم إنشاء مقبس الشبكة وربطه بعنوان IP ورقم منفذ. يبدأ الخادم بالاستماع لطلبات الاتصال. عند تلقي طلب اتصال من عميل، يتم قبول الاتصال وإنشاء اتصال جديد للتعامل مع هذا العميل، مما يسمح للخادم بمعالجة عدة عملاء في نفس الوقت.

العميل:

يتم إنشاء مقبس شبكة للعميل ثم يتصل بالخادم عبر عنوان IP المحلي والمنفذ 9999.

عند الاتصال بالخادم، يطلب البرنامج من المستخدم إدخال رقم الحساب وكلمة المرور. ثم يتم إرسال طلب تسجيل الدخول إلى الخادم بصيغة نصية تتضمن الأمر LOGIN، رقم الحساب، وكلمة المرور. ينتظر البرنامج رد الخادم، فإذا كان الرد "Authenticated"، فهذا يعني أن المصادقة قد نجحت.

في حالة نجاح المصادقة، يسمح البرنامج للمستخدم بإجراء عمليات بنكية مثل التحقق من الرصيد، الإيداع، السحب، أو إنهاء الاتصال. يعرض البرنامج قائمة بالخيارات المتاحة للمستخدم، ويطلب منه إدخال الرقم المقابل للعملية المطلوبة.

إذا اختار المستخدم التحقق من الرصيد، يتم إرسال طلب بصيغة BALANCE متبوعاً برقم الحساب، ويستقبل العميل الرد من الخادم ويعرض الرصيد الحالي. في حالة الإيداع، يطلب البرنامج من المستخدم إدخال المبلغ المراد إيداعه، ثم يرسل طلب بصيغة DEPOSIT متبوعاً برقم الحساب والمبلغ، وينتظر الرد من الخادم ليعلم المستخدم بنجاح العملية. وبالمثل، في حالة السحب، يتم طلب المبلغ المراد سحبه، وإرسال طلب بصيغة WITHDRAW، واستقبال الرد المناسب من الخادم.

إذا اختار المستخدم إنهاء الجلسة، يتم إرسال طلب بصيغة EXIT إلى الخادم، ويتم استقبال الرد النهائي الذي يتضمن الرصيد النهائي للحساب، ثم يتم إنهاء الاتصال بالخادم وإغلاق مقبس الشبكة.

إذا فشلت المصادقة، يتم إبلاغ المستخدم بفشل عملية تسجيل الدخول، وينتهي البرنامج.

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 .

web > templates > index.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Comprehensive Networking and Software Communications Learning Platform</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a class="navbar-brand" href="{{ url_for('index') }}">Networking and Software Communications</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="nav
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('index') }}">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('about') }}">About</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('contact') }}">Contact</a>
                    </li>
                </ul>
            </div>
        </nav>
    </header>
```

```
32
33      <div class="container mt-5">
34          <section>
35              <h2>Lessons on Networking Protocols and Security</h2>
36              <ul>
37                  <li>Explore various networking protocols and security measures to ensure robust communication.</li>
38              </ul>
39          </section>
40
41          <section>
42              <h2>Interactive Exercises</h2>
43              <p>Engage in interactive exercises to apply wired and wireless networking concepts.</p>
44          </section>
45
46          <section>
47              <h2>Articles and Resources</h2>
48              <p>Access the latest articles and resources covering cutting-edge technologies and tools in the field of commu
49          </section>
50
51          <section>
52              <h2>Interface for Open-Source Software and Tools</h2>
53              <p>Utilize an interface for open-source software and tools to develop customized solutions.</p>
54          </section>
55      </div>
56
57  </body>
58  </html>
```

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>About - Networking and Software Communications</title>
7       <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
8       <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
9   </head>
10  <body>
11      <header>
12          <nav class="navbar navbar-expand-lg navbar-light bg-light">
13              <a class="navbar-brand" href="{{ url_for('index') }}">Networking and Software Communications</a>
14              <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="
15                  <span class="navbar-toggler-icon"></span>
16              </button>
17              <div class="collapse navbar-collapse" id="navbarNav">
18                  <ul class="navbar-nav">
19                      <li class="nav-item">
20                          <a class="nav-link" href="{{ url_for('index') }}">Home</a>
21                      </li>
22                      <li class="nav-item">
23                          <a class="nav-link" href="{{ url_for('about') }}">About</a>
24                      </li>
25                      <li class="nav-item">
26                          <a class="nav-link" href="{{ url_for('contact') }}">Contact</a>
27                      </li>
28                  </ul>
29              </div>
30          </nav>
31      </header>
```

```
32
33      <div class="container mt-5">
34          <section>
35              <h2>About Us</h2>
36              <p>Welcome to the Networking and Software Communications Learning Platform, your ultimate resource for masterin
37              <p>Our team comprises seasoned professionals with extensive experience in networking and software communicatior
38          </section>
39
40          <section>
41              <h2>Our Mission</h2>
42              <p>Our mission at Networking and Software Communications Learning Platform is to empower learners with the know
43          </section>
44
45          <section>
46              <h2>Why Choose Us?</h2>
47              <ul>
48                  <li>Expertly curated curriculum covering a wide range of networking protocols and software communication to
49                  <li>Hands-on learning experiences with simulations and practical exercises</li>
50                  <li>Engaging instructional materials developed by industry professionals</li>
51                  <li>Interactive community forums for collaboration and knowledge sharing</li>
52                  <li>Flexible learning options to accommodate diverse learning styles</li>
53                  <li>Opportunities for professional development and career advancement</li>
54              </ul>
55          </section>
56      </div>
57  </body>
58  </html>
59
```

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Contact - Networking and Software Communications</title>
7       <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
8       <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
9   </head>
10  <body>
11      <header>
12          <nav class="navbar navbar-expand-lg navbar-light bg-light">
13              <a class="navbar-brand" href="{{ url_for('index') }}">Networking and Software Communications</a>
14              <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="na
15                  <span class="navbar-toggler-icon"></span>
16              </button>
17              <div class="collapse navbar-collapse" id="navbarNav">
18                  <ul class="navbar-nav">
19                      <li class="nav-item">
20                          <a class="nav-link" href="{{ url_for('index') }}">Home</a>
21                      </li>
22                      <li class="nav-item">
23                          <a class="nav-link" href="{{ url_for('about') }}">About</a>
24                      </li>
25                      <li class="nav-item">
26                          <a class="nav-link" href="{{ url_for('contact') }}">Contact</a>
27                      </li>
28                  </ul>
29              </div>
30          </nav>
31      </header>
```

```
<div class="container mt-5">
    <section>
        <h2>Contact Us</h2>
        <p>Have a question or need assistance? Feel free to reach out to our team. We are here to help!</p>

        <form>
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" id="name" name="name" class="form-control" required>
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" class="form-control" required>
            </div>
            <div class="form-group">
                <label for="message">Message:</label>
                <textarea id="message" name="message" class="form-control" rows="5" required></textarea>
            </div>
            <button type="submit" class="btn btn-primary">Send</button>
        </form>
    </section>

    <section class="mt-5">
        <h2>Our Location</h2>
        <p>Our headquarters are located at:</p>
        <address>
            123 Network Street<br>
            City, State, ZIP<br>
            Country
        </address>
    </section>
```

```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 , port=25545)

```

يتم إنشاء تطبيق Flask باستخدام الكود app = Flask(__name__).

يتم تمرير __name__ كمعامل لتحديد اسم التطبيق وتحديد موقع ملفات html.

يتم تعريف المسارات (routes) باستخدام المزخرف @app.route

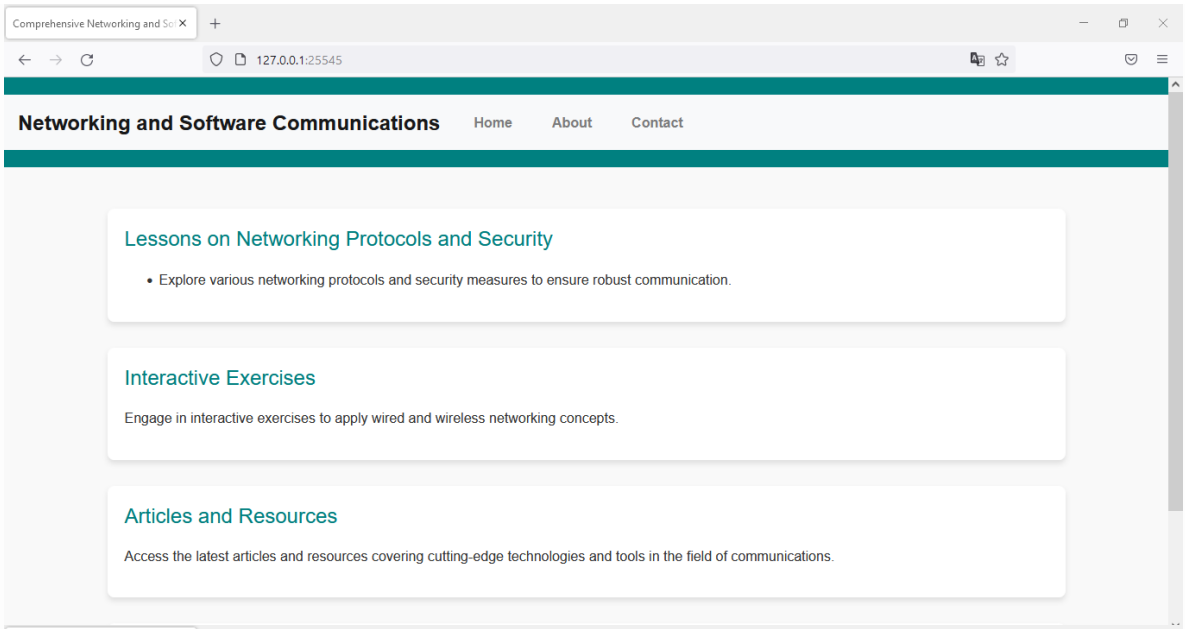
المسار /template يعود إلى الصفحة الرئيسية ويتم تعيينه لدالة home().

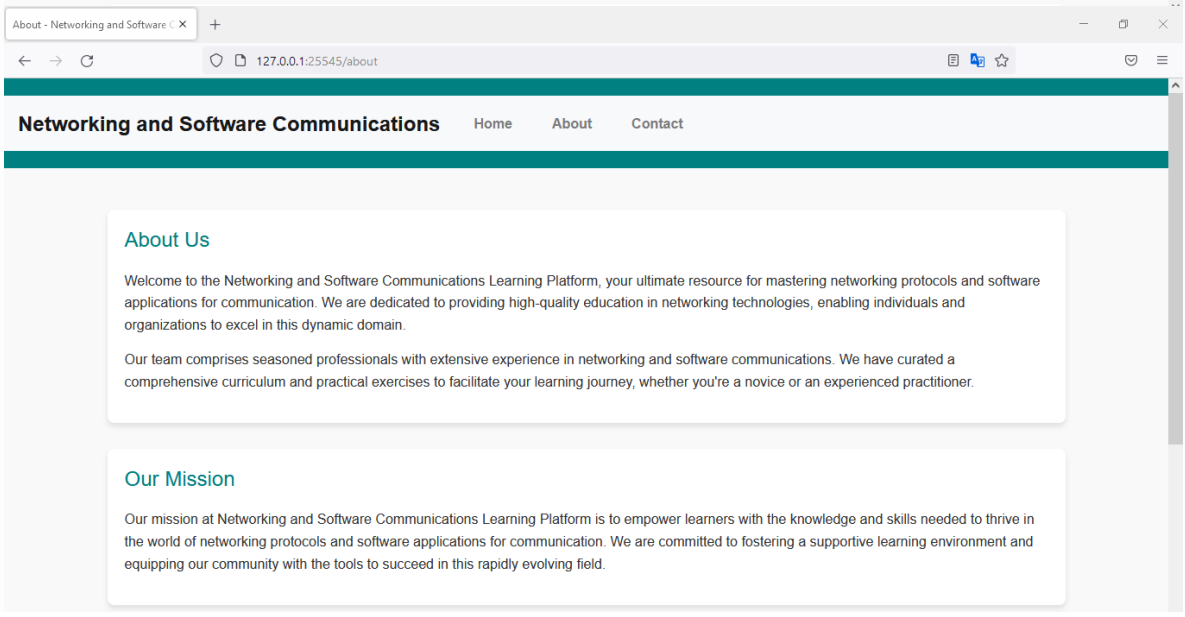
المسار /template/about يعود إلى صفحة "about" ويتم تعيينه لدالة about().

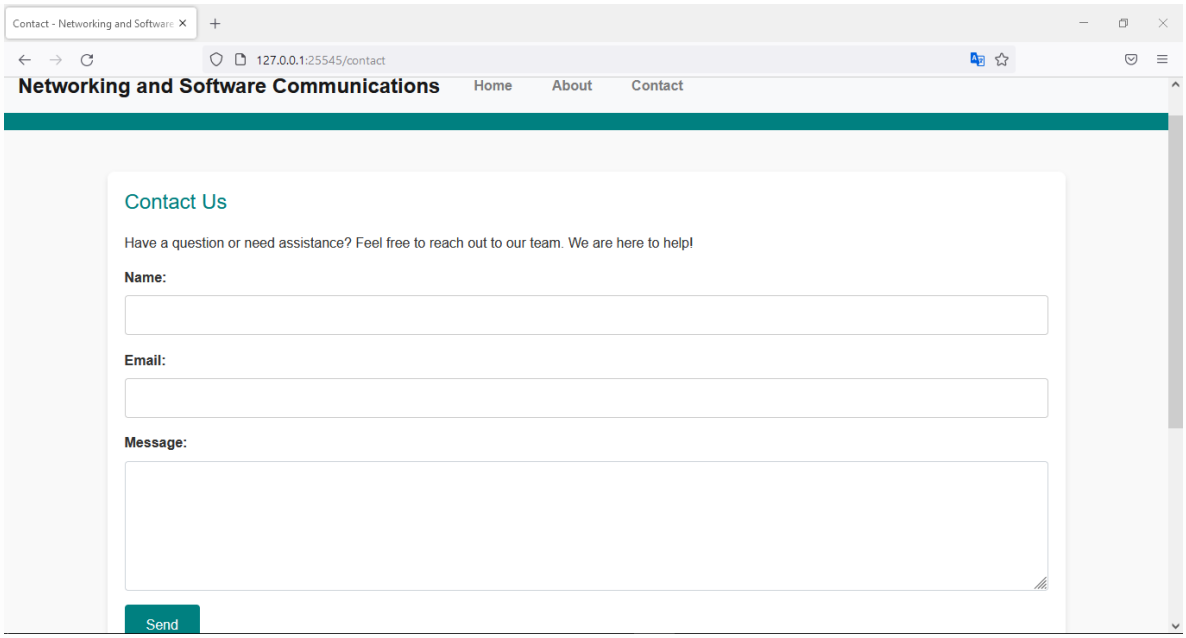
المسار /template/contact يعود إلى صفحة "contact " ويتم تعيينه لدالة contact().

إذا كان البرنامج يتم تشغيله مباشرة عن طريق تشغيل البرنامج الرئيسي ، فإنه يشغل التطبيق بتفعيل وضع التصحيح (debug mode) بواسطة الأمر app.run(debug=True).

يتم التشغيل على port 25545


Comprehensive Networking and Sol
127.0.0.1:25545
Networking and Software Communications
Home
About
Contact
Lessons on Networking Protocols and Security
Explore various networking protocols and security measures to ensure robust communication.
Interactive Exercises
Engage in interactive exercises to apply wired and wireless networking concepts.
Articles and Resources
Access the latest articles and resources covering cutting-edge technologies and tools in the field of communications.


About - Networking and Software C
127.0.0.1:25545/about
Networking and Software Communications
Home
About
Contact
About Us
Welcome to the Networking and Software Communications Learning Platform, your ultimate resource for mastering networking protocols and software applications for communication. We are dedicated to providing high-quality education in networking technologies, enabling individuals and organizations to excel in this dynamic domain.
Our team comprises seasoned professionals with extensive experience in networking and software communications. We have curated a comprehensive curriculum and practical exercises to facilitate your learning journey, whether you're a novice or an experienced practitioner.
Our Mission
Our mission at Networking and Software Communications Learning Platform is to empower learners with the knowledge and skills needed to thrive in the world of networking protocols and software applications for communication. We are committed to fostering a supportive learning environment and equipping our community with the tools to succeed in this rapidly evolving field.


Contact - Networking and Software
127.0.0.1:25545/contact
Networking and Software Communications
Home
About
Contact
Contact Us
Have a question or need assistance? Feel free to reach out to our team. We are here to help!
Name:
Email:
Message:
Send