الجُمهُوريَّة العَربيّة السّورِيّة
وِزَارُة التّعلِيم العَاليْ - جَامِعَةُ تِشْرِين
كلّية الهَندَسَة المِيكَانيكيَّة وَالكَهربَائيَّة
قِسم هَندَسةُ الاتّصالاتِ وَالالكترونيَّات
السَّنة الدِّراسيّة الخامسة **2023-2024**

# Network Programming

# Homework No2

**إعْدَادْ الطَّالِب:**
محمّد محسن التزه
الرقم الجامعي **2482**

**إِشرَاف الدّكتُور : مهند عيسى**

## 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.

## Solution / Code:

server.py

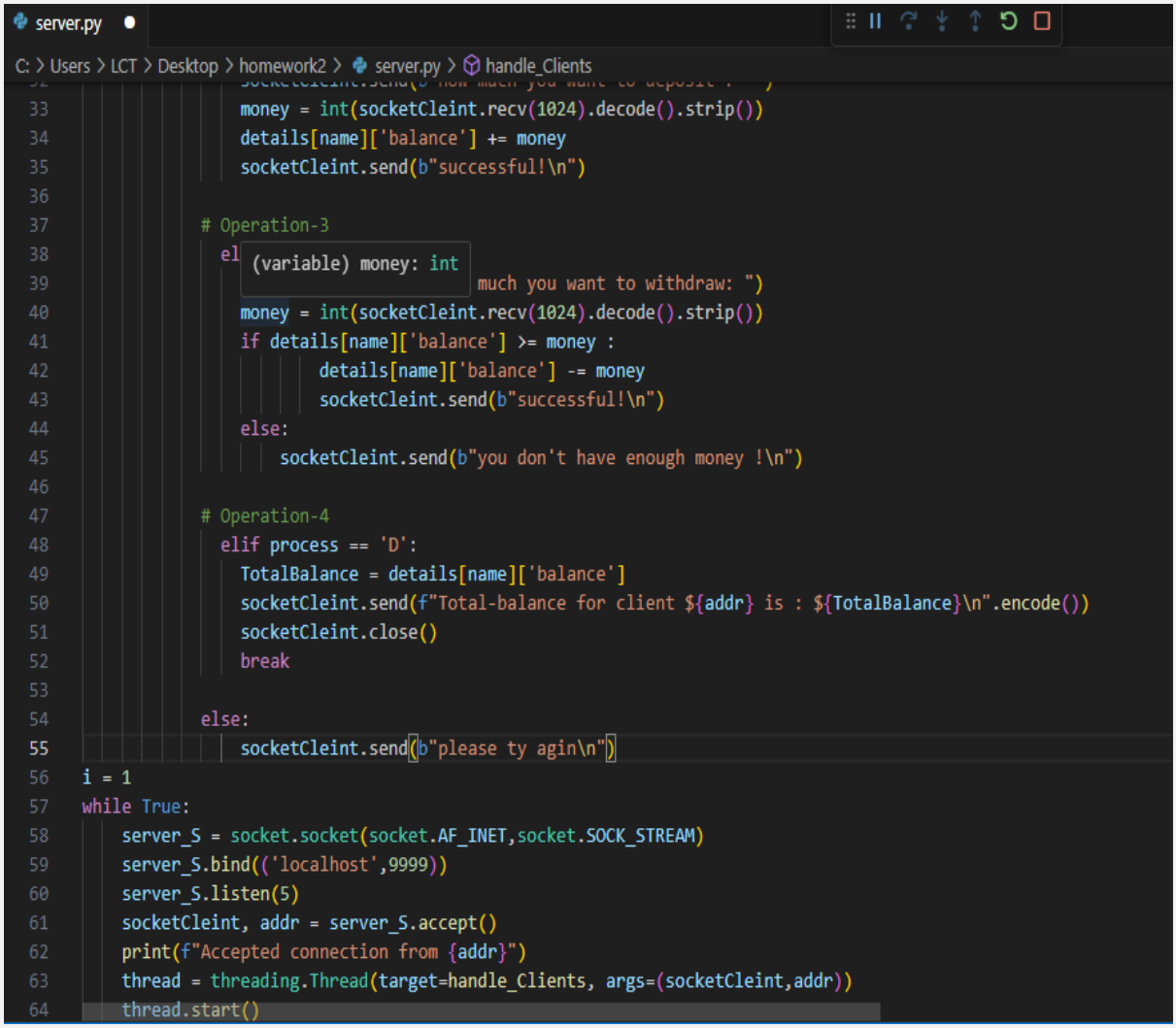
C: > Users > LCT > Desktop > homework2 > server.py > ...

```python
import socket
import threading
# bank accounts
details = {'Mohammed': {'password': '555', 'balance': 2000},'Ali': {'password': '444', 'balance': 1000}, 'Ahmad'
}
answer = ["A" , "B" , "C" , "D"]

def handle_Clients(socketCleint, addr):
        # accsess client
        socketCleint.send(b"Enter your name : ")
        name = socketCleint.recv(1024).decode().strip()
        socketCleint.send(b"Enter your Password : ")
        password = socketCleint.recv(1024).decode().strip()
        if name in details and details[name]['password'] == password:
            socketCleint.send(b"accsess successful!\n")
        else:
            socketCleint.send(b"wrong details please try later !\n")
            socketCleint.close()
            return

        while True:
            socketCleint.send(b"what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit\n")
            process = socketCleint.recv(1024).decode().strip()
            if process in answer :
            # Operation-1
              if process == 'A':
                socketCleint.send(f"Your balance is: ${details[name]['balance']}\n".encode())

            # Operation-2
              elif process == 'B':
                socketCleint.send(b"how much you want to deposit ?  ")
```




## Explanation:

هذا الكود يقوم بإنشاء خادم TCP يعمل على منفذ 9999 ويستطيع معالجة اتصالات متعددة من العملاء (أو الزبائن) بشكل متزامن باستخدام التعدد الخيطي (Multi-threading). هدف التطبيق هو تطبيق نظام أساسي لصراف آلي بنكي يُمكِن العملاء من القيام بعمليات مصرفية مثل التحقق من الرصيد، الإيداع، والسحب.

الكود يتكون من الأجزاء الرئيسية التالية:

1. تفاصيل الحسابات البنكية: هي قاموس في Python يحتوي على حسابات المستخدمين مع كلمات المرور والأرصدة المرتبطة بهم.

2. دالة معالجة العملاء (handle Clients): وهي تتولى التفاعل مع العميل:

- يتم طلب اسم المستخدم وكلمة المرور.
- يتحقق من صحتهما وإذا كانت تفاصيل الدخول خطأ، يغلق الاتصال.
- إذا كانت صحيحة، تعرض على العميل قائمة بالعمليات المتاحة وتنفذها استجابةً لإدخالات العميل:
  - التحقق من الرصيد: يعرض الرصيد الحالي للعميل.
  - الإيداع: يستفسر عن مبلغ الإيداع ويضيفه إلى رصيد العميل.
  - السحب: يستفسر عن مبلغ السحب ويخصمه من رصيد العميل إذا توفر رصيد كاف.
  - الخروج: يعرض إجمالي الرصيد ويغلق الاتصال.

3. الكود الرئيسي: الذي ينشئ الخادم ويبدأ بالاستماع للاتصالات الواردة. عندما يتصل عميل، يُطبع عنوانه ويتم إنشاء خيط جديد لمعالجة الاتصال بحيث لا يؤثر على القدرة على قبول اتصالات جديدة.

## Code Agent_1:

server.py ● Agent-1.py ×

C: > Users > LCT > Desktop > homework2 > Agent-1.py > ...

```python
import socket
Agent = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
Agent.connect(('127.0.0.1', 8888))
incoming = ''
while True:
        incoming = Agent.recv(1024).decode()
        if incoming == '':
            break
        print(incoming, end="")
        Input = input()
        Agent.send(Input.encode())
        if Input == "D" :
            print(incoming, end="")

```

## Code Agent_2:

server.py ● Agent-1.py Agent-2.py ×

C: > Users > LCT > Desktop > homework2 > Agent-2.py > ...

```python
import socket
Agent = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
Agent.connect(('127.0.0.1', 8888))
incoming = ''
while True:
        incoming = Agent.recv(1024).decode()
        if incoming == '':
            break
        print(incoming, end="")
        Input = input()
        Agent.send(Input.encode())
        if Input == "D" :
            print(incoming, end="")

```

**Explanation:**

هذا الكود يمثل جهة العميل في تطبيق شبكة يستخدم بروتوكول TCP. إليك شرح مختصر لكل جزء:

1. استيراد موديول socket: يتم استيراد الموديول الذي يسمح بإجراء العمليات على الشبكة.

2. إنشاء السوكيت (Socket): يتم إنشاء سوكيت جديد باستخدام IPv4 (AFINET) وتحديد أن طبيعة الاتصال ستكون بروتوكول TCP (SOCKSTREAM).

3. إنشاء الاتصال: يقوم الكود بمحاولة الاتصال بالخادم الذي يعمل على الجهاز المحلي (127.0.0.1) والمنفذ رقم 9999.

4. حلقة لاستقبال البيانات:

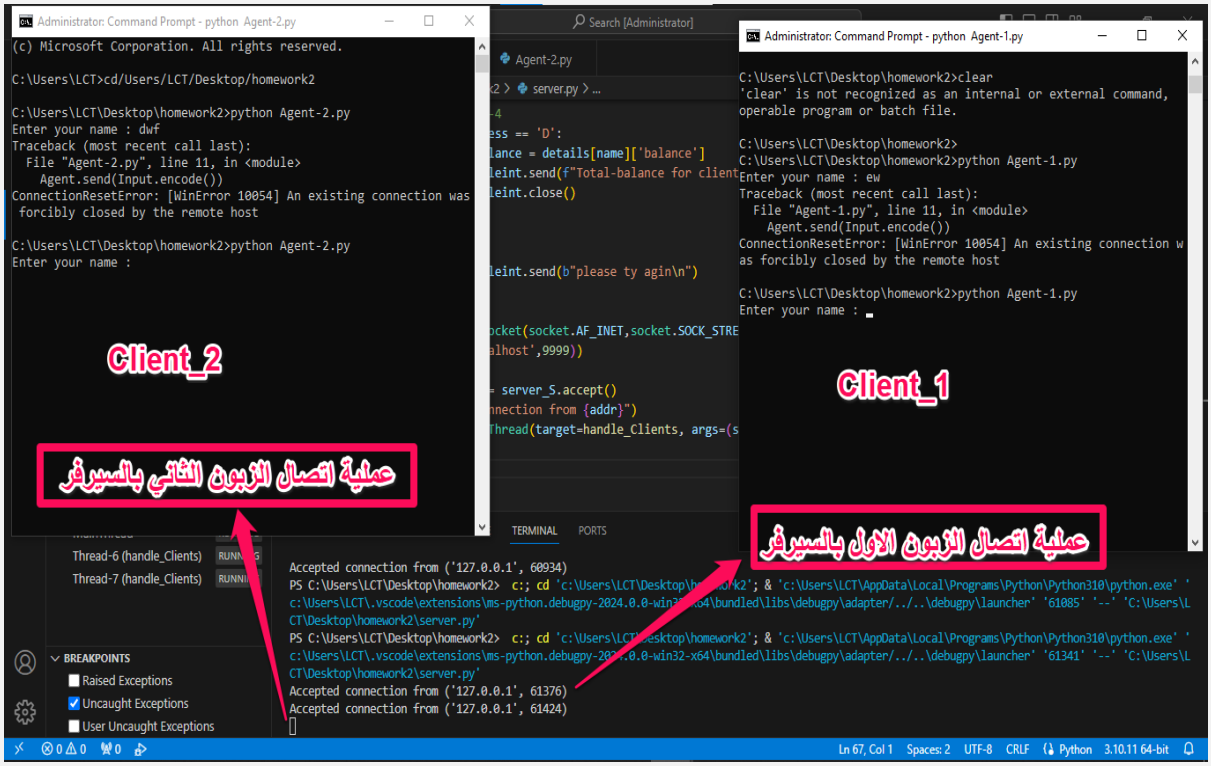
- يستمر العميل في تلقي البيانات من الخادم شريطة ألا يكون السلسلة الواردة فارغة. إذا تلقى سلسلة فارغة، يفهم العميل أن الاتصال قد انتهى ويقوم بالخروج من الحلقة.

- يتم طباعة البيانات الواردة ليتم رؤيتها من قبل المستخدم دون نهاية السطر (end="") لتتمكن من إدخال البيانات في نفس السطر.

- يقوم المستخدم بإدخال الرد ويتم إرسال هذه البيانات مرة أخرى إلى الخادم مُشفرة بتنسيق UTF-8.

5. شرط لإغلاق الاتصال: إذا كان الإدخال الذي يقوم به المستخدم هو الحرف "D"، فيفترض الكود أن هذا هو إشارة لإنهاء الجلسة فور طباعة الردود السابقة، ومع ذلك، يبدو أن هناك خطأ ما في الكود لأن المتغير incoming لن يتم تحديثه قبل الطباعة في هذا الشرط، ولا يُغلق السوكيت بشكل واضح.

## RESULT:

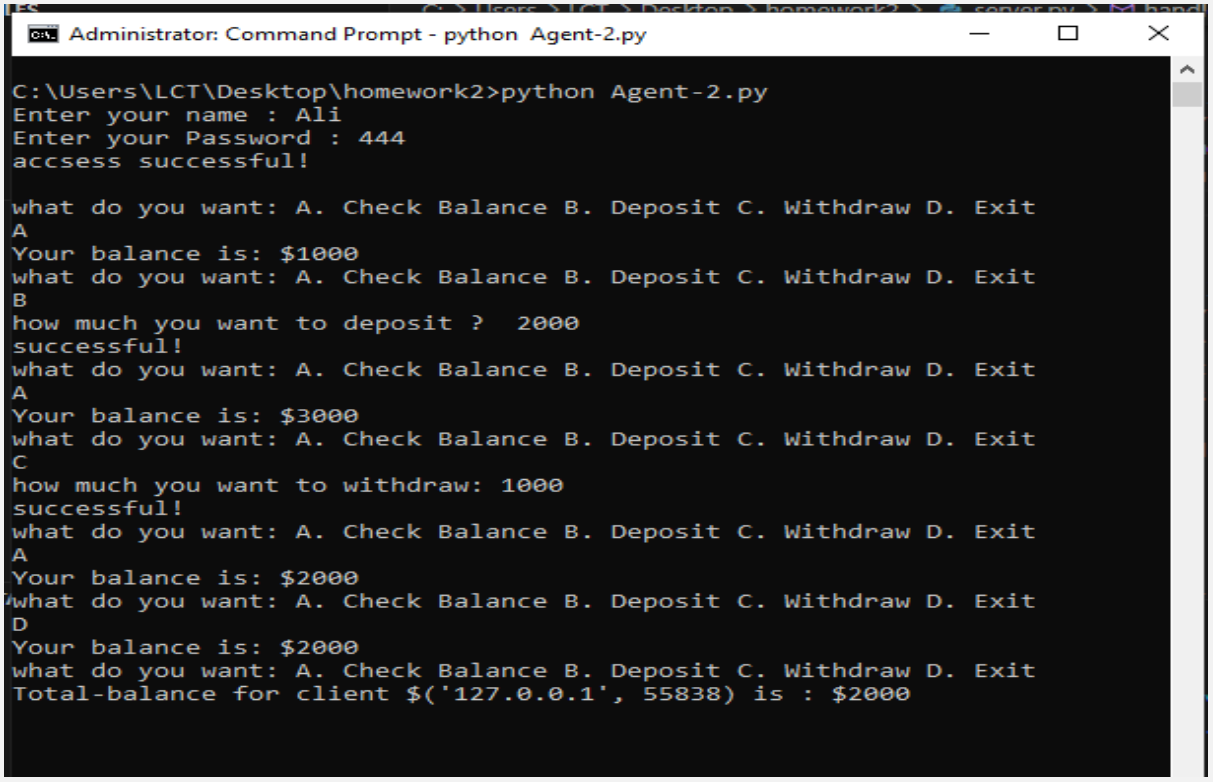



## Agent-1:

```
Administrator: Command Prompt - python  Agent-1.py

C:\Users\LCT\Desktop\homework2>python Agent-1.py
Enter your name : Mohammed
Enter your Password : 555
accsess successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $2000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
B
how much you want to deposit ?  2000
successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $4000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
C
how much you want to withdraw: 1000
successful!
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
A
Your balance is: $3000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
D
Your balance is: $3000
what do you want: A. Check Balance B. Deposit C. Withdraw D. Exit
Total-balance for client $('127.0.0.1', 53872) is : $3000
```

## Agent-2:




- عند عملية التنفيذ لدينا اربع احرف لها وظائف محددة عند ادخالها في الكود وهي :
  - A فحص الرصيد
  - B الايداع
  - C السحب
  - D اغلاق

- لقد قمت في الكود السابق بادخال عمليات فحص الرصيد في حسابي ومن ثم الايداع ومن ثم اعادة فحص الرصيد ومن ثم السحب ومن ثم فحص الرصيد ومن ثم الاغلاق .

## 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).

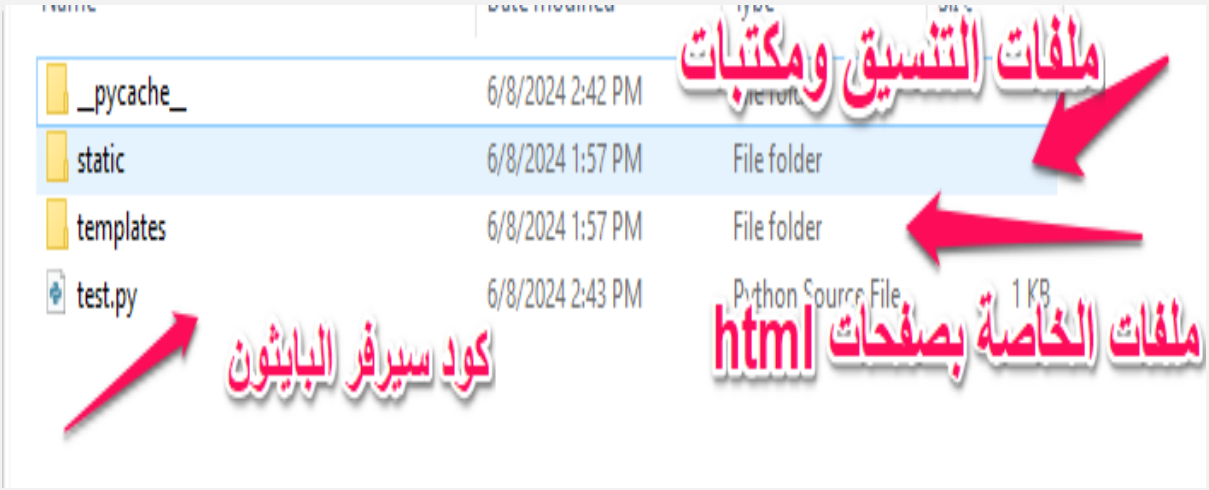
H. Apply CSS and Bootstrap to style the website and make it visually appealing.

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.

# Solution / Code:

### Data structure:

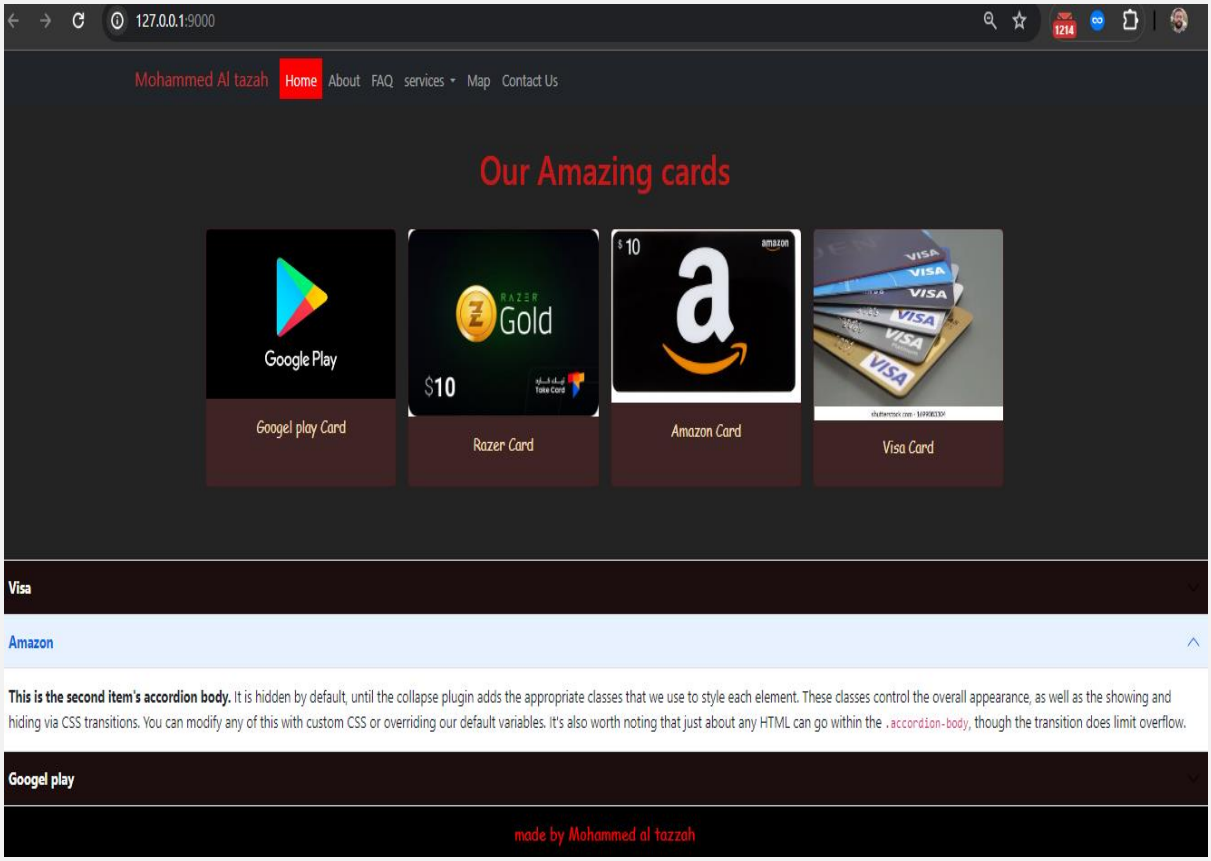


## Server code:

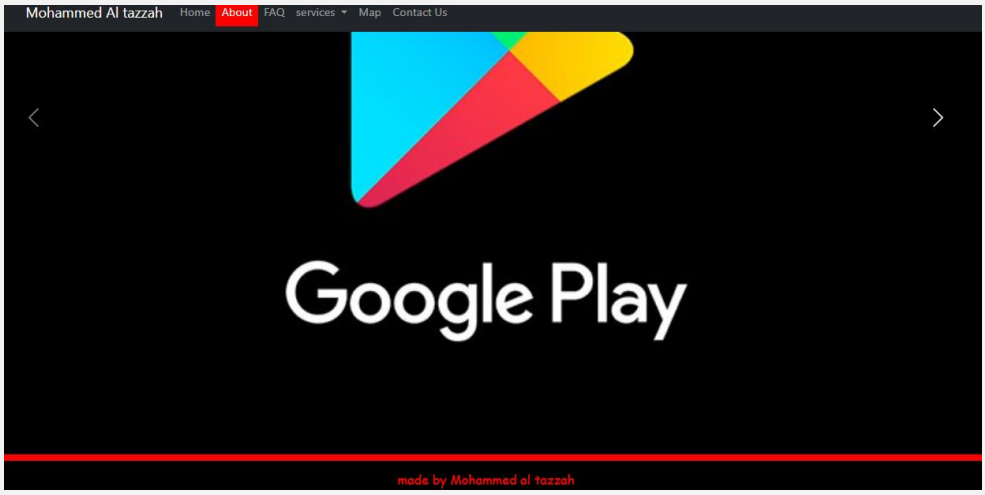

Lunch code server:

Web Site :

## Home-page




## About-Us:




***The End .....***