

**Second Network Programming Homework**

**إعداد الطالبتين:**

**ضحى خالد زنجرلي 2431 دعاء محمد جمعه زينب 2499**

**إشراف الدكتور المهندس:**

**مهند عيسى**

**الجمهورية العربية السورية**

**وزارة التعليم العالي والبحث العملي**

**جامعة تشرين – اللاذقية**

**كلية الهمك**

**قسم هندسة الاتصالات والالكترونيات**

**السنة الخامسة**

**Question 1:** **Bank ATM Application with TCP Server/Client and Multi-threading:**

import socket

import threading

HOST = '0.0.0.0'

PORT = 9999

accounts = {

    "Doha": 1050,

    "Doaa": 5050

}

def handle\_client(conn, addr):

    print(f"Connected by {addr}")

    while True:

        data = conn.recv(1024).decode()

        if not data:

            break

        account\_number, operation, amount = data.split()

        if account\_number not in accounts:

            conn.sendall("Invalid account number".encode())

            continue

        try:

            amount = float(amount)

        except ValueError:

            conn.sendall("Invalid amount".encode())

            continue

        if operation == "check\_balance":

            balance = accounts[account\_number]

            conn.sendall(f"Your balance is: {balance}".encode())

        elif operation == "deposit":

            accounts[account\_number] += amount

            conn.sendall(f"Deposit successful. New balance: {accounts[account\_number]}".encode())

        elif operation == "withdraw":

            if accounts[account\_number] < amount:

                conn.sendall("Insufficient funds".encode())

            else:

                accounts[account\_number] -= amount

                conn.sendall(f"Withdrawal successful. New balance: {accounts[account\_number]}".encode())

        else:

            conn.sendall("Invalid operation".encode())

    conn.close()

    print(f"Client {addr} disconnected")

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

    s.bind((HOST, PORT))

    s.listen()

    print(f"Server listening on {HOST}:{PORT}")

    while True:

        conn, addr = s.accept()

        thread = threading.Thread(target=handle\_client, args=(conn, addr))

        thread.start()

كود السيرفر: حتى نجعل السيرفر يخدم عدد كبير من المستخدمين بنفس الوقت يجب الاستفادة من المودل threading، تم تعيين IP السيرفر على 0.0.0.0 من أجل نخديم أي عنوان بالشبكة، ورقم المنفذ على 9999. خزنت الحسابات في dictionary له الاسم accounts، اعتمدت على رقم الحساب في التخزين بحيث جعلت رقم الحساب هو المفتاح والمبلغ المالي هو القيمة المقابلة.

بتعرف التابع handle\_client(conn, addr) أتعامل مع اتصالات العملاء بحيث مررت له سوكيت العميل وهو البارمتر conn وعنوان العميل addr. استقبل معلومات العميل باستخدام:

        data = conn.recv(1024).decode()

وثم عن طريق تعريف المتحولات رقم الحساب ونوع العملية المرادة وإجمالي القيمة المضافة أو المسحوبة أستطيع فصل هذه البيانات باستخدام data.split().

من ثم حسب العملية التي يرد القيام بها العميل أستطيع تنفيذ ما يريد كما يلي:

try:

            amount = float(amount)

        except ValueError:

            conn.sendall("Invalid amount".encode())

            continue

        if operation == "check\_balance":

            balance = accounts[account\_number]

            conn.sendall(f"Your balance is: {balance}".encode())

        elif operation == "deposit":

            accounts[account\_number] += amount

            conn.sendall(f"Deposit successful. New balance: {accounts[account\_number]}".encode())

        elif operation == "withdraw":

            if accounts[account\_number] < amount:

                conn.sendall("Insufficient funds".encode())

            else:

                accounts[account\_number] -= amount

                conn.sendall(f"Withdrawal successful. New balance: {accounts[account\_number]}".encode())

        else:

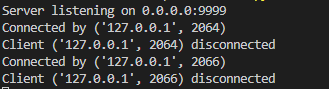
            conn.sendall("Invalid operation".encode())

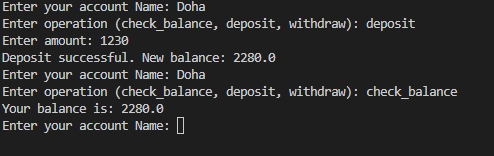
    conn.close()

    print(f"Client {addr} disconnected")

بإنشاء الأوبجيكت (الغرض) thread أستطيع جعل السيرفر يتعامل مع المستخدمين بنفس الوقت بإمرار التابع target=handle\_client ومن ثم args=(conn, addr).

تشغيل السيرفر:





import socket

HOST = '127.0.0.1'

PORT = 9999

while True:

    account\_number = input("Enter your account Name: ")

    operation = input("Enter operation (check\_balance, deposit, withdraw): ")

    if operation in ("deposit", "withdraw"):

        amount = float(input("Enter amount: "))

    data = f"{account\_number} {operation} {amount}".encode()

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

        s.connect((HOST, PORT))

        s.sendall(data)

        response = s.recv(1024).decode()

        print(response)

**Question 2: Simple Calendar App using Kivy:**

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.label import Label

from kivy.uix.gridlayout import GridLayout

from datetime import datetime

import calendar

class DayOfWeekApp(App):

    def build(self):

        root\_layout = BoxLayout(orientation='vertical')

        day = datetime.today().strftime('%A')

        year = datetime.today().year

        calendar\_month = datetime.today().strftime('%B %Y')

        day\_label = Label(text=f"Today is {day}", font\_size=30)

        year\_label = Label(text=f"Current Year: {year}", font\_size=22)

        calendar\_label = Label(text=f"Calendar: {calendar\_month}", font\_size=22)

        root\_layout.add\_widget(day\_label)

        root\_layout.add\_widget(year\_label)

        root\_layout.add\_widget(calendar\_label)

        calendar\_table = self.create\_calendar\_table()

        root\_layout.add\_widget(calendar\_table)

        return root\_layout

    def create\_calendar\_table(self):

        calendar\_table = GridLayout(cols=7, spacing=5, padding=10, size\_hint\_y=None, height=400)

        calendar\_table.bind(minimum\_height=calendar\_table.setter('height'))

        day\_names = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']

        for day in day\_names:

            day\_label = Label(text=day, size\_hint\_y=None, height=40)

            calendar\_table.add\_widget(day\_label)

        today = datetime.today()

        calendar\_matrix = calendar.monthcalendar(today.year, today.month)

        for week in calendar\_matrix:

            for day in week:

                if day == 0:

                    day\_label = Label(text='', size\_hint\_y=None, height=40)

                else:

                    day\_label = Label(text=str(day), size\_hint\_y=None, height=40)

                calendar\_table.add\_widget(day\_label)

        return calendar\_table

if \_\_name\_\_ == '\_\_main\_\_':

    DayOfWeekApp().run()

هذا الكود هو تطبيق بسيط باستخدام مكتبة Kivy لعرض يوم الأسبوع وتقويم شهري بتنسيق جميل:

1. from kivy.app import App: يستورد الكائن الرئيسي لتطبيق Kivy.

2. from kivy.uix.boxlayout import BoxLayout: يستورد تخطيط مربعي لترتيب العناصر في صفوف وأعمدة.

3. from kivy.uix.label import Label: يستورد ويستخدم العنصر Label لعرض النص.

4. from kivy.uix.gridlayout import GridLayout: يستورد تخطيط الجدول لترتيب العناصر في شبكة.

5. from datetime import datetime: يستورد وقت التشغيل لاستخدام التواريخ والأوقات.

6. import calendar: يستورد مكتبة التقويم لإنشاء التقويمات.

الكلاس DayOfWeekApp :

7. class DayOfWeekApp(App):: يعرف الكلاس الرئيسي للتطبيق.

8. def build(self):: يعرف الوظيفة التي يتم استدعاؤها لإنشاء وتكوين واجهة المستخدم.

9. root\_layout = BoxLayout(orientation='vertical'): ينشئ تخطيطًا مربعيًا لترتيب العناصر بشكل رأسي.

10. تحديد معلومات اليوم والسنة والشهر الحالي.

11. إنشاء عناصر Label لعرض معلومات اليوم والسنة والشهر.

12. إضافة العناصر إلى التخطيط الرئيسي.

13. calendar\_table = self.create\_calendar\_table(): ينشئ جدول التقويم باستخدام الميثود create\_calendar\_table.

14. إضافة جدول التقويم إلى التخطيط الرئيسي.

15. return root\_layout: يعيد التخطيط الرئيسي كجزء من واجهة المستخدم.

الآن لنشرح الوظيفة create\_calendar\_table:

16. def create\_calendar\_table(self):: تعريف وظيفة لإنشاء جدول التقويم.

17. calendar\_table = GridLayout(cols=7, spacing=5, padding=10, size\_hint\_y=None, height=400): ينشئ جدولًا مكونًا من 7 أعمدة ويحدد البعد والتباعد.

18. calendar\_table.bind(minimum\_height=calendar\_table.setter('height')): يربط ارتفاع الجدول بحد أدنى ممكن للحفاظ على الحجم.

19. إنشاء تسميات الأيام (Sun, Mon, Tueإلخ وإضافتها إلى الجدول.

20. إنشاء مصفوفة تقويم للشهر الحالي وملء الجدول بأيام الشهر الحالي.

21. return calendar\_table: يعيد الجدول كجزء من واجهة المستخدم.

22 if \_\_name\_\_ == '\_\_main\_\_':

يتحقق مما إذا كان البرنامج يشغل مباشرة من الملف أو مستوردًا كمكتبة.

23 DayOfWeekApp().run(): تشغيل التطبيق.

