جامعة تشرين
كلية الهندسة الميكانيكية والكهربائية
هندسة الاتصالات والالكترونيات

تطبيق مخدم / زبون لحل المعادلات الرياضية

اعداد الطلاب :

هبه سهيل عبدالله

ايفا محمد شكوحي

بإشراف :

الدكتور مهند عيسى

ملخص البحث:

سوف نقوم بهذا البحث بإنشاء مخدم يقوم بحل معادلات من الدرجات الأولى والثانية والثالثة يدخلها الزبون عبر اتصال مخدم / زبون.

في البداية كتبنا مقدمة عن بنية نظام مخدم / زبون ثم تطرقنا لأهمية استخدام هذه البنية . ثم تم شرح مكونات الشبكة : المخدم والزبون وأدوات الاتصال.

ثم التطرق بعدها لخصائص بنية مخدم / زبون التي تؤمن تشاركية وتكاملية المعطيات وعملها على عدة منصات وشفافية الموقع وتغليف الخدمة.

تم بعدها شرح مساوئ هذه البنية مقارنة ببنية الند للند .

وأخيرا تم شرح بنى هذه الشبكة وهي وحيدة الطبقات وثنائية الطبقات وثلاثية الطبقات ومتعددة الطبقات .

وأخيرا تم شرح مراحل بناء الأكواد والمخدم والزبون .

مقدمة:

إن بنية مخدم / زبون هي شبكة حاسوبية يقوم من خلالها الزبائن (أجهزة بعيدة) بطلب خدمة واستقبالها من مخدم مركزي.

يؤمن الزبون واجهة للسماح للمستخدم بطلب خدمة من المخدم وعرض النتائج التي حصل عليها وأعادها المخدم له. بينما ينتظر المخدم الطلبات الواصلة من الزبائن ومن ثم يجيب عليهم.

بشكل عام يؤمن المخدم واجهة شفافة للزبائن وبالتالي ليس بالضرورة للزبون أن يحوي مكونات برمجية أو صلبة محددة للحصول على الخدمة، وحيث تكون هذه الخدمة عامة وقابلة للعمل على مختلف أنظمة التشغيل والأجهزة.

يتوضع الزبون عادة في حاسوب محلي بينما يتوضع المخدم ضمن آلات ذات مقدرات أكبر. وتكون بنية مخدم / زبون فعالة أكثر عندما يحوي المخدم والزبون مهام مختلفة ويحتاجون لتكرارها بشكل دائم.

– على سبيل المثال: فإنه في نظام مشفى ما يمكن لحاسوب الزبون أن يشغل برنامج تطبيق لإدخال معلومات المريض بينما يقوم المخدم بتشغيل برنامج آخر يدير قواعد البيانات التي تخزن البيانات بشكل دائم

يمكن لعدة زبائن مختلفين أن يصلوا لبيانات قاعدة بيانات المخدم بشكل متزامن، وبنفس الوقت يمكن لحاسوب الزبون أن يقوم بعدة مهام أخرى مثل إرسال الإيميلات وغيرها.

بما أن كل حواسيب المخدم والزبون هي أجهزة ذكية، يمكن اعتبار بنية مخدم / زبون بنية مختلفة كليا عن النظام القديم المسمى "main frame" الذي يحوي حاسوب مركزي واحد يقوم بجميع المهام المتعلقة بالطرفيات المرتبطة به .

الغاية من بنية مخدم / زبون :

نحن اليوم في عصر تلعب فيه تكنولوجيا المعلومات دورا حيويا في التطبيقات التجارية بحيث تسعى دوما الكيانات التجارية لاستثمار هذه التقنية للتوسع أكثر في الأسواق العالمية بالإضافة لأن هذه المنظمات يجب أن تخضع لآلية لجلب ومعالجة البيانات اللازمة لجعل الإجراءات التجارية أكثر فعالية للوصول أو الاستمرار في الأسواق العالمية.
يؤمن نموذج مخدم / زبون طريقة منطقية logical ذات عمليات معالجة موزعة حيث يقوم بها المخدم باستقبال الطلبات من جميع الزبائن ويعالجها ويرد عليها.

ما هو المخدم والزبون ؟

الزبون: هو عبارة عن مستخدم يؤمن خدمات التقديم presentation service وخدمات قواعد البيانات data base services ويؤمن الاتصالات عبر واجهة معينة للمستخدم للتفاعل وطلب الخدمات.

المخدم : هو عبارة عن معالج واحد وأكثر مع مقدرات أعلى وأكبر وذواكر مشتركة تؤمن الاتصالية وخدمات قواعد البيانات بالإضافة للواجهات المتعلقة بها لتأمين الخدمة المطلوبة من الزبون.

يقوم بروتوكول محدد بإدارة عمليات إرسال الطلبات ما قبل الزبون وعمليات إرسال الإجابات من قبل المخدم وتكون بنية مخدم زبون مستقلة عن المنصة Platform – Independent أي أنها غير مرتبطة بالعمل مع أجهزة أو أنظمة تشغيل محددة بل هي متوفرة للعمل مع مختلف أنواع الزبائن.

خصائص المخدم والزبون :

إن المخدم والزبون هما الكيانات المنطقية التي تعمل سوية عبر شبكة ما لتنفيذ مهمة ما وسنعرف بعض المفاهيم المتعلقة بهذه البنية:

١ – الخدمة Service :

إن خدمة مخدم / زبون هي علاقة بين عمليتين مختلفتين تعملان عبر جهازين مختلفين ، عملية المخدم تعتبر مؤمن الخدمات بينما تعتبر عملية الزبون مستهلك للخدمة باختصار ، هذه الطريقة تؤمن فصل بالوظائف بين الخدمات المقدمة.

٢ – مشاركة المصادر Resources Sharing :

يكون المخدم قادر على مشاركة مصادره (المعالجة ، الذواكر) بين عدة زبائن مختلفين حيث يستطيع استقبال أكثر من اتصال من عدة زبائن مختلفين بنفس الوقت.

٣– بروتوكولات غير متناظرة:

تعتبر خدمة مخدم / زبون علاقة من النوع عدة – إلى – واحدة (Many – to – one) والتي تبدأ من الزبائن بإرسال طلبات باتجاه المخدم لطلب الخدمة بينما يقوم المخدم بانتظارها

والرد عليها. في بعض الأحيان يحرر الزبون كانت إعادة اتصال يسمى **callback object** عندما يطلب الخدمة من الزبون (لما يجعل المخدم زبون بحد ذاته) .

٤ – شفافية الموقع Transparency Of Location :

يمكن أن تتواجد عملية المخدم ضمن جهاز الزبون نفسه (مخدم محلي) أو ضمن أي جهاز آخر على الشبكة في هذه الحالة تكون برمجية الزبون مسؤولة عن تحديد موقع المخدم عبر توجيه طلبات الحصول على الخدمة إلى المخدم عبر رقم المنفذ وبالتالي يمكن الجهاز أن يكون زبونا أو مخدما أو الاثنين معا.

٥ – التواصل عبر الرسائل :

التفاعل بين الزبائن والمخدمات يتم عبر تمرير الرسائل بشكل أساسي لتوصيل طلبات وإجابات الخدمة .

٦ – تغليف الخدمة Encap Sulation Of Services :

يكون المخدم مسؤولا عن تحقيق طلبات الزبائن بشكل ملائم بحيث يمكن ترقية المخدم دون تأثر البيئة الخارجية المحيطة به (الزبائن ، المصادر المشتركة، ...) بالإضافة إلى الحفاظ على واجهة الرد دون تغيير.

٧ – التوسعيةScalability :

يمكن توسيع أنظمة توسيع مخدم / زبون بشكل أفقي أو عمودي عبر إضافة أو حذف وحدات زبائن (توسيع شاقولي) أو وحدات مخدم / توسيع أفقي ، ويتم ذلك دون أي تأثيرات جوهرية على عمل أو أداء النظام.

٨ – التكاملية Integrity :

طالما أن أكواد المخدم والزبون تدار بشكل مركزي فإن كلفة الصيانة تكون أقل وتنتج حالة من عدم اعتماد البيانات على الزبائن.

محاسن شبكات مخدم / زبون:

إن تطبيق بنية شبكات مخدم / زبون في العديد من التطبيقات يسمح بتحسين أدائها وتسهيل إدارتها وذلك من خلال:

١ – تحسين آلية تشارك المعطيات:

يتم جلب ومعالجة المعطيات المتعلقة بالخدمة في المخدم الذي يتم الوصول إليه من قبل عدة زبائن باستخدام آلية موثوقة.

إن استخدام لغة SQL (Stryctured Query Language) يدعم الوصول من قبل جميع الزبائن إلى الخدمة كما يدعم شفافية الخدمة أي السماح بالحصول على الخدمة بغض النظر عن الزبون الذي يطلب هذه الخدمة .

٢ – تكاملية الخدمات :

يسمح لأي زبون بالوصول إلى البيانات المشتركة عبر استخدام واجهة حاسوبية مباشرة دون الحاجة للدخول بوضع الطرفية أو استخدام جهاز(معالج آخر) حيث يمكن استخدام منصات مثل بوربوينت Power point , Spread sheet .... للتعامل مع معطيات متشاركة بالاستعانة بقاعدة بيانات ومخدم تطبيق متواجدة ضمن الشبكة لإنتاج معلومات مفيدة.

٣ – مشاركة المصادر بين عدة منصات:

إن التطبيقات التي يتم استخدامها في نموذج مخدم / زبون يتم بناؤها بغض النظر عن الكيان الصلب للزبونHardware أو حتى الكيان البرمجي Software (نظام التشغيل) مما يسمح بتأمين بيئة مفتوحة للحوسبة ويسمح بالحصول على الخدمة عبر مشاركتها بين مختلف انوع الزبائن

٤– إمكانية معالجة المعطيات بغض النظر عن الموقع:

نحن الآن في عصر يخضع للتجول من الأنظمة المعتمدة على الآلة إلى الأنظمة المعتمدة على الآلة إلى الأنظمة المعتمدة على المستخدم . حيث أن الأنظمة المعتمدة على الآلة مثل Main Frame لها منصات وآليات عمل خاصة بها بالإضافة لآليات أمنية وتشغيلية خاصة. أما في نظام مخدم / زبون فيمكن للمستخدمين الدخول مباشرة إلى النظام بغض النظر عن موقعهم أو نوع معالجاتهم أو التقنية التي يستخدمونها .

٥ – سهولة الصيانة:

إن بنية مخدم / زبون هي بنية موزعة تقوم بتوزيع المهام على عدة حواسيب مستقلة متلائمة عبر الشبكة وهذا شيء جيد من أجل الصيانة لأنه يجعل عملية وضع صيانة، ترقية أو إعادة توضيع مخدم ما عملية بسيطة دون أن تأثر على الزبون. هذه العملية التي تجعل الخدمة غير متعلقة بالتغيرات المحتملة تسمى بالتغليف Encapsulation .

٦ – الأمن:

لدى المخدمات تحكم بالوصول والمصادر أفضل من الزبائن مما يؤكد أن الكيانات الموثوقة فقط من الزبائن هي المسموح لها بالوصول وطلب المعطيات من المخدم ، مما يؤكد إدارة أمنية فعالة للشبكة من قبل المخدم .

مساوئ نظام مخدم / زبون مقارنة بنظام الند / للند p2p(Peer – to – Peer) :

يعتمد نظام الند للند على شبكة غير مركزية تكون فيها جميع العقد متساوية المهام والوظائف والقدرات ويمكن لأي منها أن تعمل كزبون أو مخدم و تؤمن هذه الأنظمة لكلفة أقل (لاتحوي مخدمات)

ويمكن تفصيل مساوئ نظام مخدم / زبون مقارنة بنظام الند للند P2P كما يلي :

١ – ازدحام المخدمات :

عند وصول طلبات زبائن كبيرة ومتزامنة بنفس الوقت إلى المخدم فإن هذه المخدمات سوف تزدحم، بينما في أنظمة P2P تقوم الشبكة بإضافة عقد أخرى مما يزيد من عرضة حزمة النظام وسعته التي تساوي مجموع عرض الحزمة واسعة في كل عقدة من عقد الشبكة.

٢ – الأثر السلبي لمركزية الشبكة:

بما أن نظام مخدم / زبون نظام مركزي يعتمد بكل عمله على المخدم ، فإن فشل هذا المخدم سوف لن يسمح باستقبال طلبات الزبائن وبالتالي انهيار الشبكة، وبالمقابل فإن فشل عقدة ما في شبكات P2P لن يؤثر على عمل الشبكة كونه يبحث عن أفضل بديل لهذه العقدة ويقوم بتشغيله.

أنواع بنى شبكات مخدم / زبون :

يتم تقسيم بنية مخدم / زبون إلى إصناف رئيسية هي:

١ – وحيدة الطبقة one – tier .

٢ – ثنائي الطبقة two – tier .

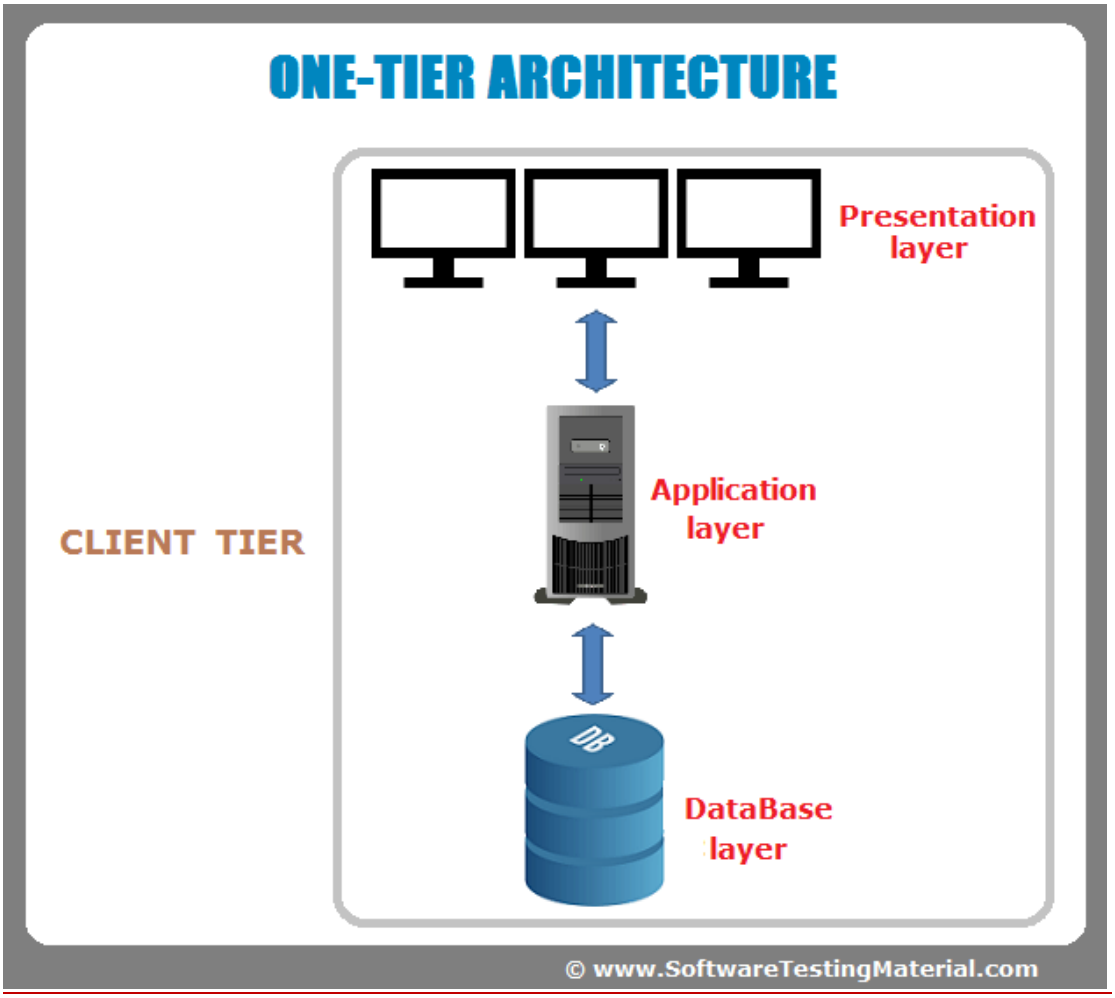
٣ – ثلاثي الطبقة three – tier .

٤ – متعدد الطبقات N – tier .

هذه الطبقات تحتوي على واجهة المستخدم User Interface ومنطق العمل Business والمعلومات المتشاركة shared data .

١ – البنية وحيدة الطبقات one – tier :

في هذه البنية يتوضع كل واجهة المستخدم ومنطق العمل ضمن كيان واحد (نفس الجهاز) ولا حاجة هناك لاتصال شبكة لربط المخدم مع الزبون



الشكل (٢): بنية 1–tier

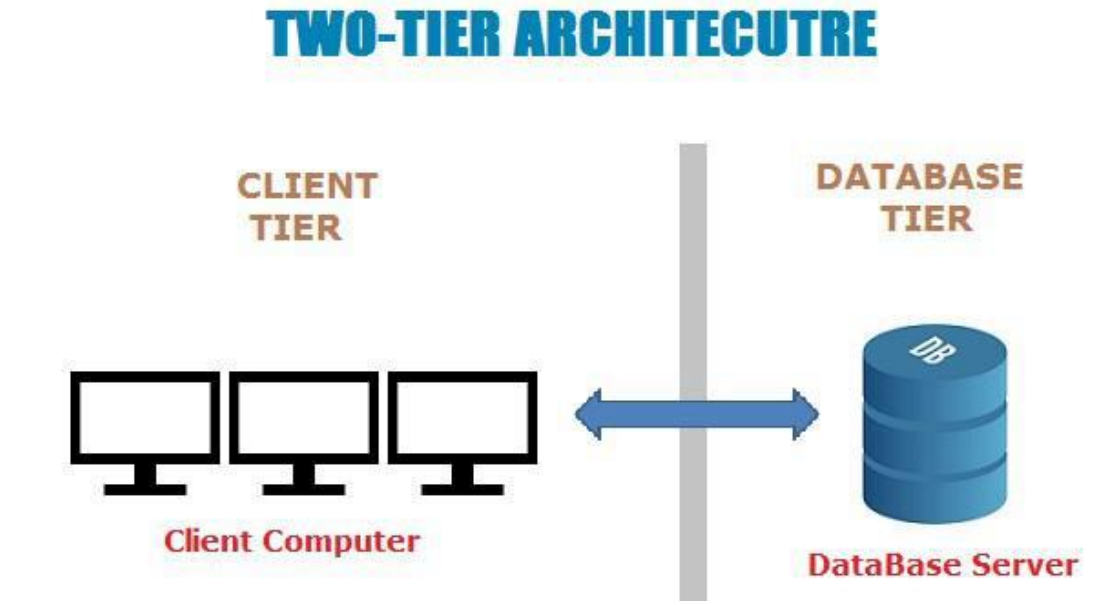
٢- البنية ثنائية الطبقات Two – tier :

أوجدت في العام ١٩٨٠ بالاعتماد على مخدم الملفات الذي تم إيجاده أساسا لتحسين الاستخدام عبر توفير واجهة مستخدم سهلة الاستخدام كما أنها توفر توسعية كبيرة عبر دعم حتى ١٠٠ مستخدم مقارنة بعشرات المستخدمين فقط في المخدمات السابقة. تحتاج هذه البنية لأقل تدخل ممكن من قبل الزبون.

تتألف هذه الطبقات الاثنين من ٣ مكونات مقسمة إلى طبقتين طبقة الزبون (الذي يطلب الخدمة) والمخدم (مزود الخدمة) وهذه المكونات هي :

- **واجهة المستخدم (إدخال نصوص ، أجهزة عرض .....)**
- **إدارة المعالجة (عمليات الم ا رقبة ، خدمات مصادر المعالجة .....)**
- **إدارة قواعد البيانات (خدمات المعطيات والملفات).**

في هذه البنية يتم تشغيل واجهة المستخدم كاملة في طرف الزبون بينما تتوضع إدارة قواعد البيانات ضمن المستخدم بينما يتم توزيع وتقاسم عمليات إدارة المعالجة بين الزبون والمخدم . يتصل الزبون مباشرة مع المخدم ويطلب الخدمة.

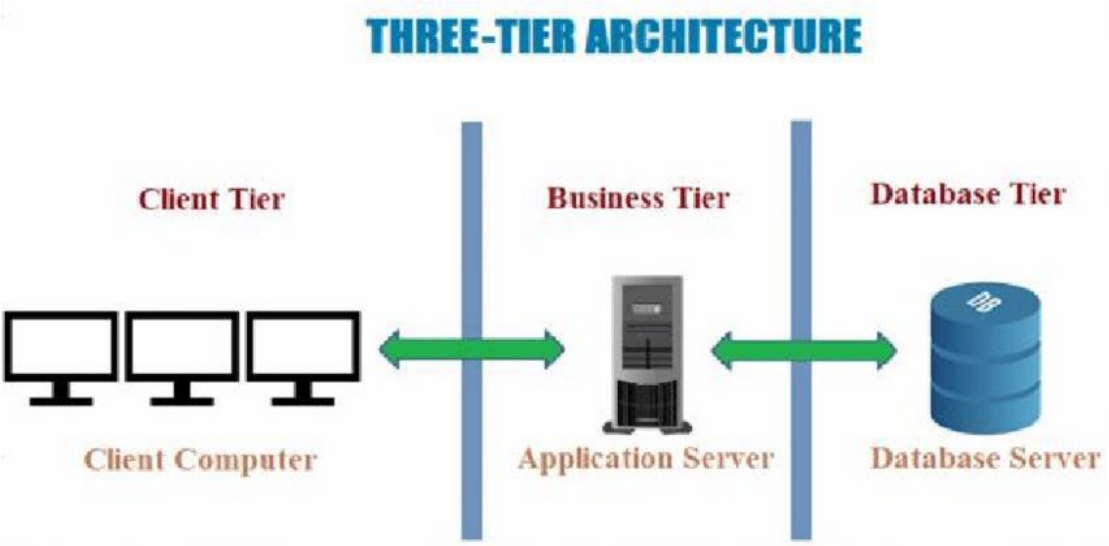


الشكل:(3) بنيةtier–2

٣- البنية ثلاثية الطبقات Three – tier :

تصنيف هذه البنية طبقة جديدة بالإضافة لطبقات البنية ثنائية الطبقات ، في هذه البنية لا يتصل الزبون مباشرة مع المخدم بل يقدم بتوجيه طلباته إلى السيرفر المحلي عبر مخدم وسيط

.على سبيل المثال إذا كنت تحاول الوصول لموقع ما ب(٣ طبقات) سوف يقوم حاسوبك بالاتصال مع الموقع عبر سيرفر التطبيق الذي يقوم بدوره بالنيابة عنك بإرسال طلب المخدم المركزي. سوف يقوم المخدم المركزي باستقبال الطلب ومعالجته والرد عليه عبر إرسال الرد إلى المخدم الوسيط (مخدم التطبيقات) الذي بدوره يوصل المعلومات إليك.



الشكل(٤): بنية 3-tier

٤- البنية رباعية الطبقات Four – tier :

تطابق عمل البنية ثلاثية الطبقات مع فرق أن عدد المخدمات الوسيطية هو أكثر من مخدم واحد.

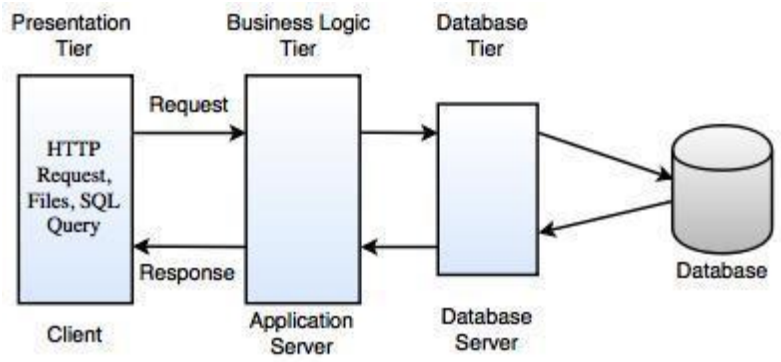


Fig. Multi-Tier Architecture

قسم التطبيق العملي :

يقوم مشروعا بتخديم عدد من الزبائن بشكل متواز عبر استخدام مكتبة Threading حيث يتخلص عمل المخدم بحل معادلات من مختلف الدرجات : درجة أولى ، ثانية، ثالثة، يطلب المستخدم أولا من الزبون إدخال درجة المعادلة ومن ثم يطلب إدخال ثوابت للمعادلة هذه ، المعادلة وإعادة إرسالها للزبون.

برنامج المخدم :

١ – استدعاء المكتبة اللازمة للعمل : مكتبة threading لتخديم عدة زبائن بشكل متزامن ، مكتبة Socket لإنشاء سوكيت اتصال بين المخدم والزبون ومكتبة Cmath ومكتبة Sympy لحل المعادلات من الدرجة الثالثة.

```
import socket , threading , cmath
import symtable as sp      # import package sympy
```

٢– إنشاء السوكيت الخاصة بالمخدم باستخدام بروتوكول Tcp ورقم منفذ ٨٨٨٨ وعنوان IP هو العنوان المحلي.

```
server_socket =socket.socket()
server_socket.bind(('127.0.0.1', 8888))
server_socket.listen(10)
print('server is waiting client')
```

٣– تعريف التابع First الذي يقوم بحل المعادلات من الدرجة الأولى ويأخذ بارامتر واحد هو عبارة عن List تحوي قيم الثوابت: a , b , c .

```
def first (values):
    res = - value[1] / value[0]
    return str(res)
```

٤ – تعريف التابع Second الذي يقوم بحل المعادلات من الدرجة الأولى ويأخذ بارامتر واحد هو عبارة عنlist تحوي قيم الثوابت: a , b, c .

```
def second (values):
    a = values[0]
    b = values[1]
    c = values[2]
    print("Equation")
    delta = (b**2) - (4*a*c)
    if delta ==0:
        res = -b / (2*a)
        res = "the one root is :"+str(res)
        return res
    else:
```

```
        res1 = str((-b + cmath.sqrt(delta)) / (2*a))
        res2 = str ((-b - cmath.sqrt(delta)) / (2*a))
        return ("the roots are:"+res1+","+res2)
```

٥ – تعريف التابع Third الذي يحل المعادلات من الدرجة الثالثة ويأخذ بارامتر وهو عبارة عن List تحوي قيم الثوابت a , b, c ,d .

٦ – يستدعي هذا التابع المكتبة Sympy وبعد تمرير الثوابت لتعيد حل هذه المعادلة والذي بدوره يعيدها للمستخدم.

```
def third (values):
    x = sp.symbol('x')
    f = values[0]*(x**3)+values[1]*(x**2)+values[2]*(x)+values[3]
    res = sp.solve(f)
    res = "the roots are :\n"+str(res[0])+"\n"+str(res[1])+"\n"+str(res[2])
    return res
```

٧ – تعريف التابع excute الذي يتم استدعاؤه عند كل قبول لزبون ما. يطلب أولا درجة المعادلة وعلى أساسها يستخدم تعليمات << if, elif >>

يقوم بعدها بإرسال شكل المعادلة للزبون ويطلب بعدها إدخال الثوابت من قبل الزبون ولصقلها في ال List الخاصة بهم ومن ثم يستدعي التابع First أو Second أو Third الموافق وأخيرا يقوم بإرسال الحلول (الجذور) إلى الزبون .

```
def excute(cs , cadd):
    cs.send("enter the degree of equation:".encode())
    degree= cs.recv(2048).decode()
    if degree=='1':
        cs.send("equation (a*x+b)".encode())
        i = 0
        constants = ['a' , 'b']
        values = []
        while i <= int(degree):
            msg= 'enter' + constants[i] + ':'
            cs.send(msg.encode())
            var = cs.recv(2048).decode()
            values.append(int(var))
            i += 1
        cs.send('ok'.encode())
        res=second(values)
        cs.send(res.encode())
    elif degree == '2':
        cs.send('equation (a*x^2 + b*x + c)'.encode())
        constants = ['a' , 'b' , 'c' ]
        values = []
        i=0
        while i <= int(degree):
            msg = 'enter' + constants[i] + ":"
            cs.send(msg.encode())
            var = cs.recv(2048).decode()
            values.append(int(var))
            i+=1
        cs.send('ok'.encode())
        res=second(values)
        cs.send(res.encode())
    elif degree =='3':
```

```
        cs.send('equation (a*x^3 + b*x^2 + c*x + d)'.encode())
        constants = ['a', 'b', 'c' , 'd']
        values = []
        i = 0
        while i <= int(degree):
            msg = 'enter' + constants[i] + ":"
            cs.send(msg.encode())
            var = cs.recv(2048).decode()
            values.append(int(var))
            i += 1
        cs.send('ok'.encode())
        res = third(values)
        cs.send(res.encode())
    else:
        cs.send("not exit".encode())
        cs.send("ok".encode())
        cs.send("no result".encode())

    cs.close()
```

٨ – نعرف الحلقة اللانهائية التي تستقبل الطلبات من الزبائن بشكل دائم ثم تنشأ Thread خاصة بكل زبون تنفذ التابع excute السابق.

```
while True:
     cs,cadd = server_socket.accept()
     cs.send("welcome to equation solver server".encode())
     print("new user is connected" , cadd)
     th = threading.thread(target= excute ,arg = (cs,cadd))
     th.start()
```

برنامج الزبون:

١ – استدعاء مكتبة Socket .

٢ – إنشاء السوكيت الخاصة بالزبون وإنشاء اتصل مع المخدم.

```
import socket

client = socket.socket()
client.connect(('127.0.0.1',8888))
```

٣ – استقبال جملة الترحيب وإرسال درجة المعادلة .

```
res = client.recv(2048).decode()
print(res)
```

٤ – طباعة شكل المعادلة ومن ثم إدخال ثوابت هذه المعادلة .

٥ – طباعة حلول المعادلة المرسلة من قبل المخدم وإنهاء الاتصال.

```
res = client.recv(2048).decode()
inp = input(res)
client.send(inp.encode())

res = client.recv(2048).decode()
print (res)

while True:
    msg = client.recv(2048).decode()
    if msg == 'ok':
        break
    inp = input(msg)
    client.send(inp.encode())
res = client.recv(2048).decode()
print(res)


client.close()
```

تنفيذ التطبيق :

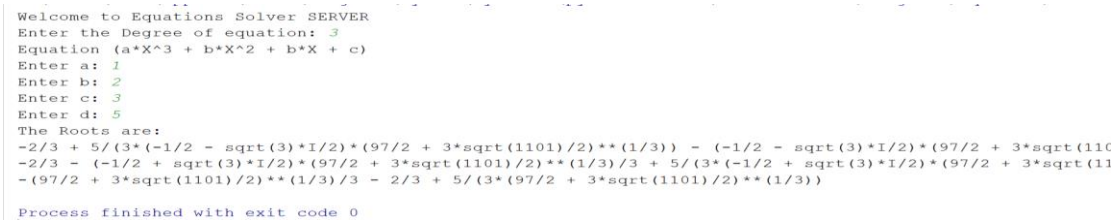
١ – السيناريو الأول : إدخال معادلة من الدرجة الأولى .

```
Welcome to Equations Solver SERVER
Enter the Degree of equation: 1
Equation (a*X + b)
Enter a: 5
Enter b: 3
The Result is: -0.6

Process finished with exit code 0
```

٢ – السيناريو الثاني: إدخال معادلة من الدرجة الثانية.

```
Welcome to Equations Solver SERVER
Enter the Degree of equation: 2
Equation (a*X^2+ b*X + c)
Enter a: 3
Enter b: 6
Enter c: 5
The Roots are: (-1+0.8164965809277259j), (-1-0.8164965809277259j)

Process finished with exit code 0
```

٣ – السيناريو الثالث: إدخال معادلة من الدرجة الثالثة.



المراجع:

١- Client Server Architecture, www.studymafia.org

٢- www.wikipedia.com