PES UNIVERSITY

A Project based learning report

TOPIC: IMPLEMENTATION OF POINT OF SALE

Submitted in Partial fulfilment of the Requirements for VI Semester
Bachelor of Technology in Electronics and Communication

Jan - May 2019

Under the guidance of

Mrs. Prajeesha Emmanuel
Asst. prof, Dept of ECE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PES UNIVERSITY, BANGALORE-85

TEAM MEMBERS

CHIRANJEEVI N – 01FB16EEC083

CHINMAY D KUCHINAD – 01FB16EEC081

D G SUHAAS KIRAN – 01FB16EEC085

JOSEPH SUNIL – 01FB16EEC113

POS - Point of sale

INTRODUCTION

Any business selling in person will have a ‘point of sale’, commonly abbreviated ‘POS’.

POS is the setup you have in place for processing face-to-face payments from customers.

POS is a constellation of things that together enable you to process customer-facing transactions efficiently and streamline business processes connected with your sales.

The setup will vary in look and functionality depending

1. On your choice of technology.
2. What payment methods you accept (whether you print paper receipts).

3. How you record sales and organize end-of-day bookkeeping, and the inventory systems you have in place for your products.

BACKGROUND

Earlier, a point of sale system was just a cash register. The person operating the till would manually enter the prices of purchased items, often with the help of price tickets. Further POS became more computerized, storing a product database on a computer server. A barcode reader was used to avoid manual price entry and store transaction details electronically. Now advanced cloud-based POS systems are being used where data is stored online.

Basic Overview

Softwares and programming languages used-

- wxPython(For graphical user interface)
- SQL (Database creation at the server and the client)
- Python 3.0(For UDP connection between server and the clients)

System Hierarchy

- Graphical user interface for interactive communication between user(customer) and the client program.
- The data fed to the GUI by the user is used to query the database and provide information to the user on the GUI.A copy is appended to the list and is sent to the server for further operations via UDP connection.
- SQL database at the server to maintain records of recent transactions, keep track of store commodities, display items' availability.

Client Side

GRAPHIC USER INTERFACE – (Tool used-wxPython)

- The **graphical user interface** (**GUI**)is a form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.
- Graphic user interface in our project is designed using wxPython. The cashier is expected to feed the system the "Item ID" and get the item description and price in return.
- A set of valid item ids are predefined and stored in the "items database". If the item ID entered is valid,the sql database is queried with that ID and the information is displayed.

UDP

- **UDP** or User Datagram protocol is a transport layer source independent protocol.
- Once the database is queried, the relevant information that is in our case, item ID and item price is sent via UDP to the server where the respective databases are updated accordingly.

SQL Database

The client side has a copy of the items database. Once a valid item id is entered in the GUI, that value is used to traverse through the items database and get the required information which in our case is the item description and its price.

This information is appended into a list and is sent to the server.

Functions used

send_data

Used to send the data retrieved from the database to the server via a UDP connection.

OnCellChange

This function first retrieves the item id from the GUI,checks if it is valid , and if yes , retrieves the corresponding data from the dataset and appends it into a list.

OnClicked

It defines the mouse click event and assigns the consequence of this event to a particular action. IN our case the submit button is linked to this and upon clicking it, the action is terminated, the data is written to the list and it ultimately is transmitted through UDP.

MyForm

Used to create the basic table structures in the GUI and define the headers. Buttons are also defined which can be assigned functionalities such as terminating a particular action.

<u>Server Side</u>

receive_data

The data send from the client data is received and using sql operations, the respective databases are updated.

create_socket

Creation of sockets and the initiation of UDP process.

bind_socket

The socket created is bounded to a particular address on the host network.

socket_accept

Acknowledging an incoming request from a client program**.**

<u>SQL Databases</u>

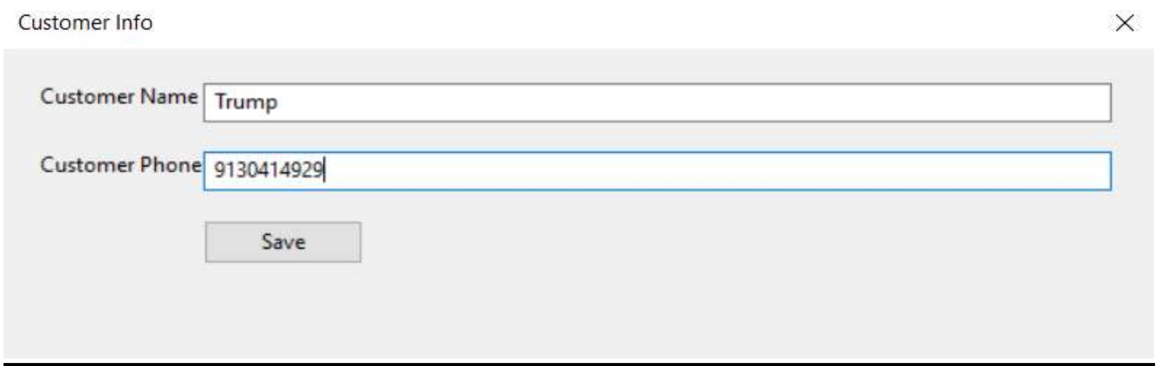
- **Items Database**
- **Transaction Database –** The data sent by the client is used to query the copy of the items' database present at the server side. This database contains purchase date, product description, customer details and is password secured.

<u>Describing the working in real time</u>

- In the GUI at the client side, the cashier enters the customer name and customer phone number. Then the cashier enters the Item IDs of the items purchased by that customer into the program. After checking for its validity, this id is used to traverse through the items database and information is obtained.
- This obtained information which in our case are item descriptions and item prices are displayed on the GUI and a copy is sent to the server (which has a centralized database) via UDP connection.
- After sending the items list to server, a bill of transaction is generated for the customer on the client side.
- The sever receives this information and updates its existing databases accordingly.
- The stored and secured information of the transactions can thus be accessed at the server whenever needed by the admin.

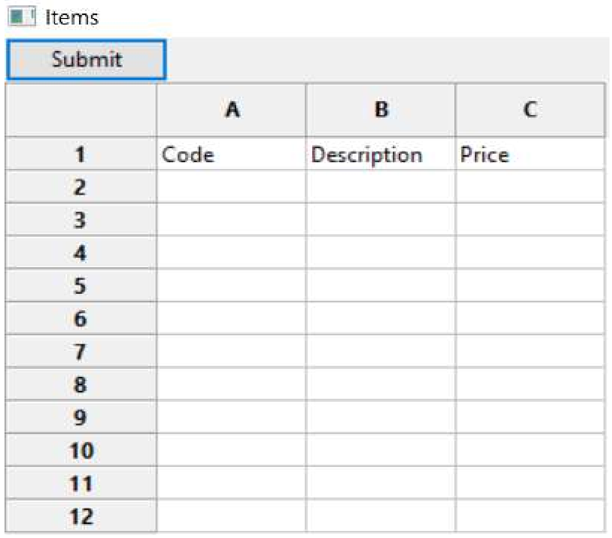
Code Execution

Graphical user interface



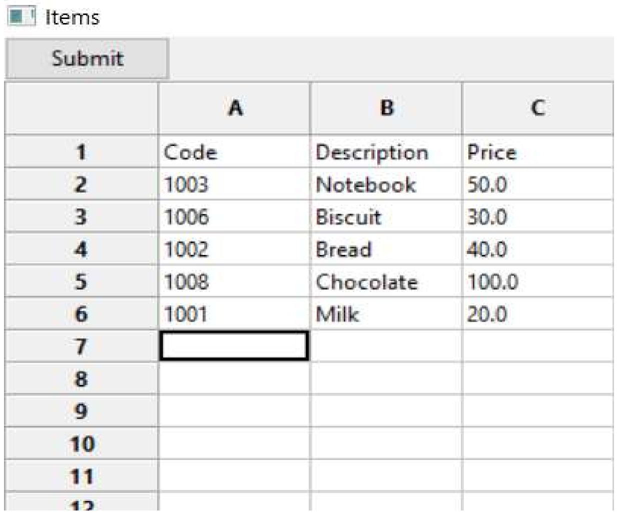
Customer Name	Trump
Customer Phone	9130414929

Before entering



	A	B	C
1	Code	Description	Price
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

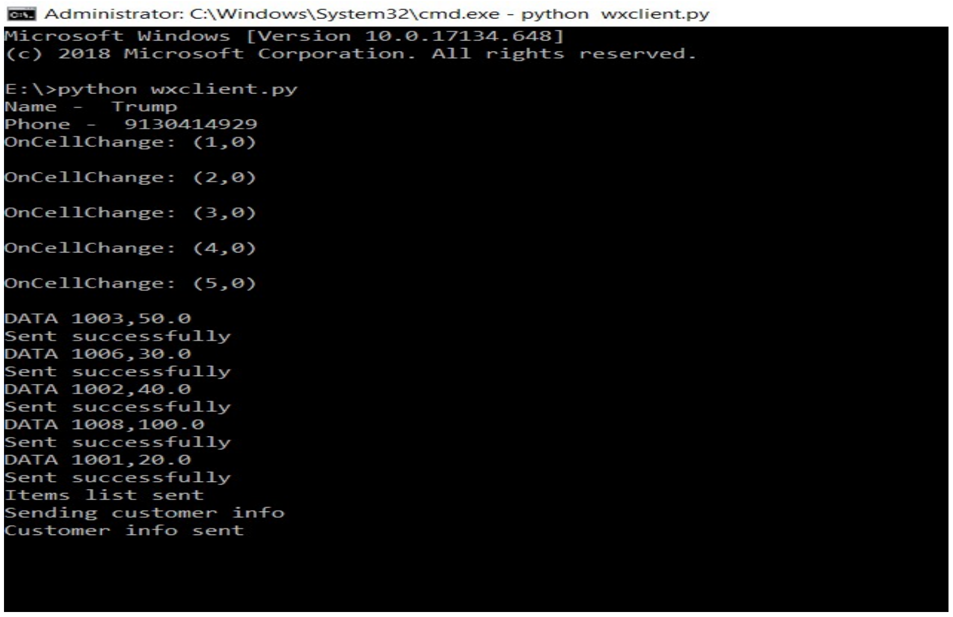
After entering



	A	B	C
1	Code	Description	Price
2	1003	Notebook	50.0
3	1006	Biscuit	30.0
4	1002	Bread	40.0
5	1008	Chocolate	100.0
6	1001	Milk	20.0
7			
8			
9			
10			
11			

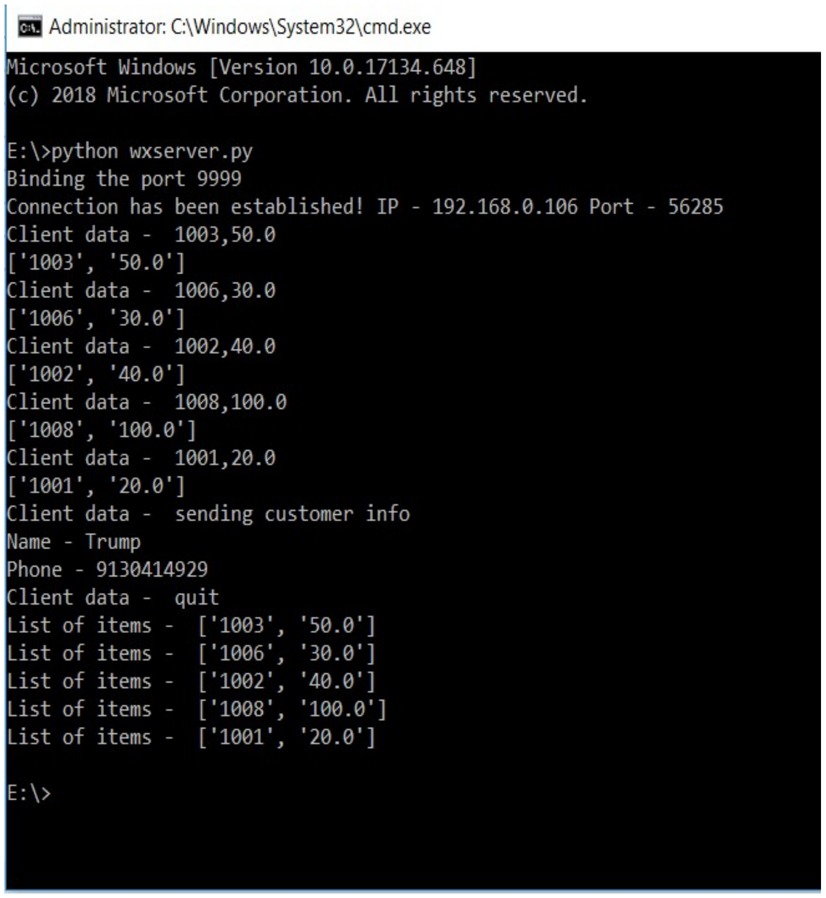
>>Click Submit

Client program execution


Administrator: C:\Windows\System32\cmd.exe - python wxclient.py

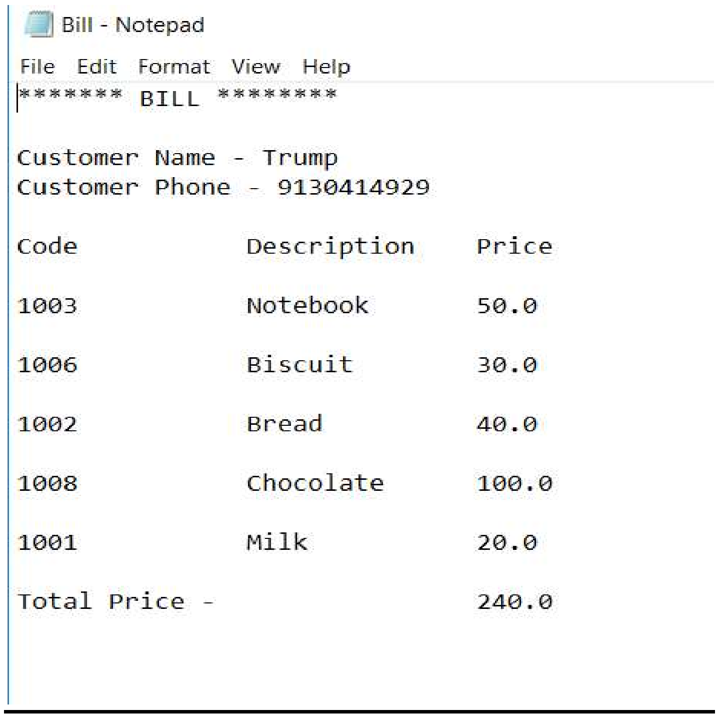
```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

E:\>python wxclient.py
Name -  Trump
Phone -  9130414929
OnCellChange: (1,0)

OnCellChange: (2,0)

OnCellChange: (3,0)

OnCellChange: (4,0)

OnCellChange: (5,0)

DATA 1003,50.0
Sent successfully
DATA 1006,30.0
Sent successfully
DATA 1002,40.0
Sent successfully
DATA 1008,100.0
Sent successfully
DATA 1001,20.0
Sent successfully
Items list sent
Sending customer info
Customer info sent
```

Output seen at server upon receiving client data



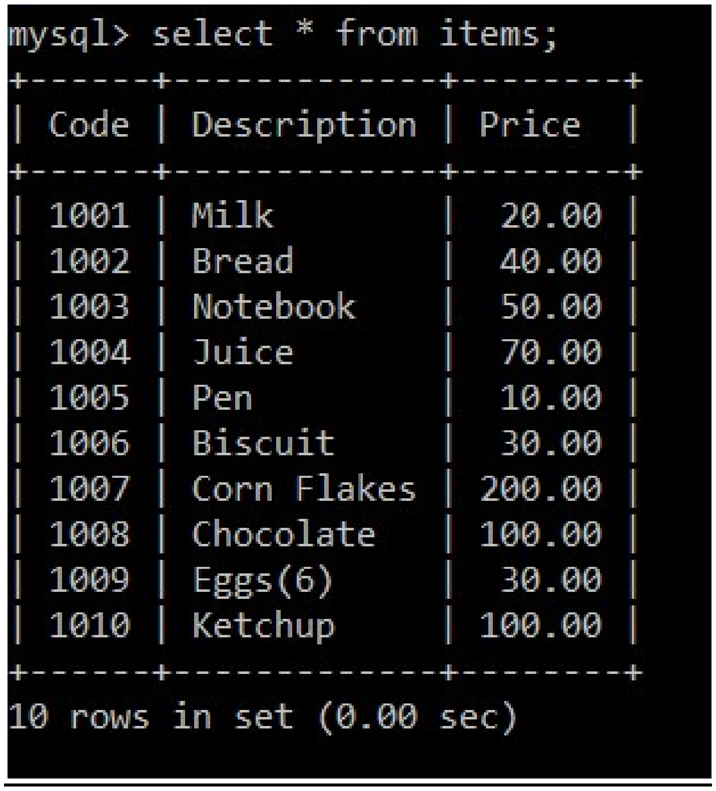
```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

E:\>python wxserver.py
Binding the port 9999
Connection has been established! IP - 192.168.0.106 Port - 56285
Client data -  1003,50.0
['1003', '50.0']
Client data -  1006,30.0
['1006', '30.0']
Client data -  1002,40.0
['1002', '40.0']
Client data -  1008,100.0
['1008', '100.0']
Client data -  1001,20.0
['1001', '20.0']
Client data -  sending customer info
Name - Trump
Phone - 9130414929
Client data -  quit
List of items -  ['1003', '50.0']
List of items -  ['1006', '30.0']
List of items -  ['1002', '40.0']
List of items -  ['1008', '100.0']
List of items -  ['1001', '20.0']

E:\>
```

Bill generated at the client for the user



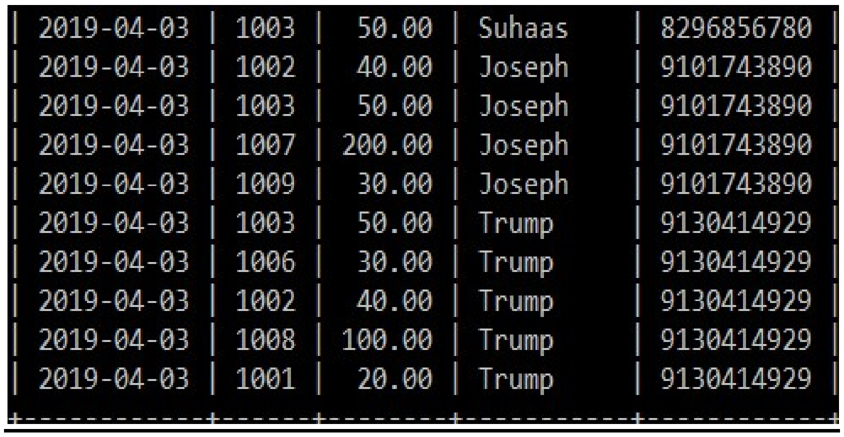
```
******* BILL ********

Customer Name - Trump
Customer Phone - 9130414929

Code            Description     Price

1003            Notebook        50.0

1006            Biscuit         30.0

1002            Bread           40.0

1008            Chocolate       100.0

1001            Milk            20.0

Total Price -                   240.0
```

Item database



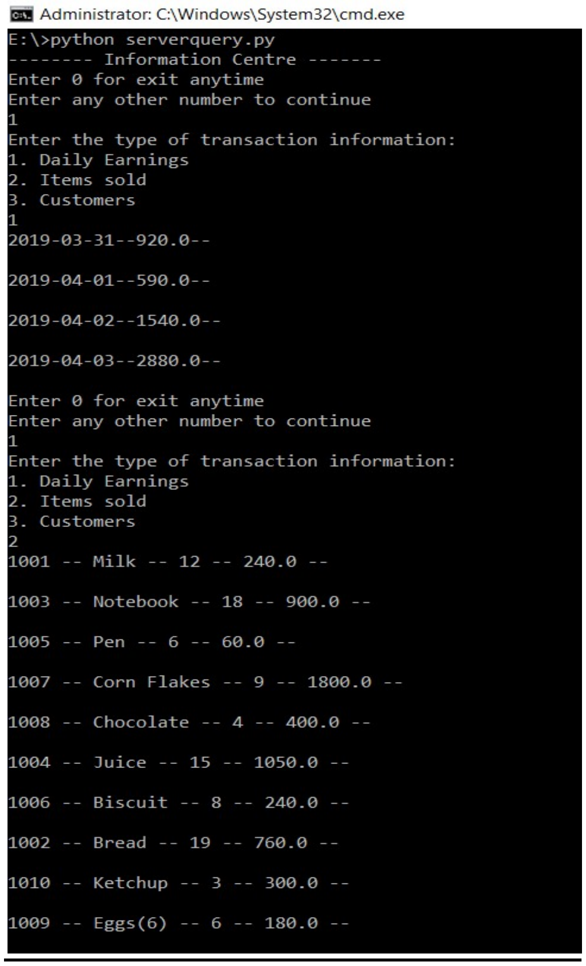
```
mysql> select * from items;
+------+-------------+--------+
| Code | Description | Price  |
+------+-------------+--------+
| 1001 | Milk        |  20.00 |
| 1002 | Bread       |  40.00 |
| 1003 | Notebook    |  50.00 |
| 1004 | Juice       |  70.00 |
| 1005 | Pen         |  10.00 |
| 1006 | Biscuit     |  30.00 |
| 1007 | Corn Flakes | 200.00 |
| 1008 | Chocolate   | 100.00 |
| 1009 | Eggs(6)     |  30.00 |
| 1010 | Ketchup     | 100.00 |
+------+-------------+--------+
10 rows in set (0.00 sec)
```

Transaction database



```
| 2019-04-03 | 1003 |  50.00 | Suhaas    | 8296856780 |
| 2019-04-03 | 1002 |  40.00 | Joseph    | 9101743890 |
| 2019-04-03 | 1003 |  50.00 | Joseph    | 9101743890 |
| 2019-04-03 | 1007 | 200.00 | Joseph    | 9101743890 |
| 2019-04-03 | 1009 |  30.00 | Joseph    | 9101743890 |
| 2019-04-03 | 1003 |  50.00 | Trump     | 9130414929 |
| 2019-04-03 | 1006 |  30.00 | Trump     | 9130414929 |
| 2019-04-03 | 1002 |  40.00 | Trump     | 9130414929 |
| 2019-04-03 | 1008 | 100.00 | Trump     | 9130414929 |
| 2019-04-03 | 1001 |  20.00 | Trump     | 9130414929 |
+------------+------+--------+-----------+------------+
```

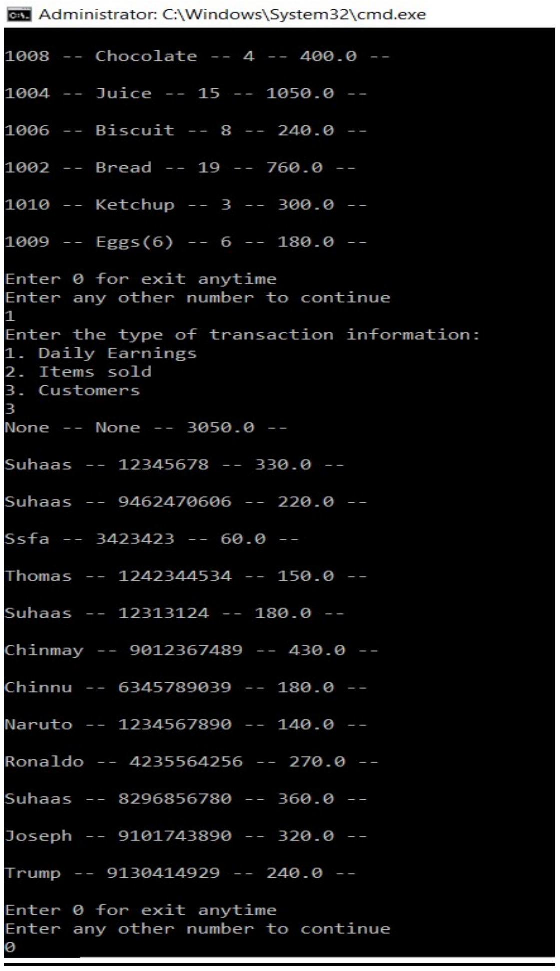
Querying item database



Administrator: C:\Windows\System32\cmd.exe

```
E:\>python serverquery.py
-------- Information Centre -------
Enter 0 for exit anytime
Enter any other number to continue
1
Enter the type of transaction information:
1. Daily Earnings
2. Items sold
3. Customers
1
2019-03-31--920.0--

2019-04-01--590.0--

2019-04-02--1540.0--

2019-04-03--2880.0--

Enter 0 for exit anytime
Enter any other number to continue
1
Enter the type of transaction information:
1. Daily Earnings
2. Items sold
3. Customers
2
1001 -- Milk -- 12 -- 240.0 --

1003 -- Notebook -- 18 -- 900.0 --

1005 -- Pen -- 6 -- 60.0 --

1007 -- Corn Flakes -- 9 -- 1800.0 --

1008 -- Chocolate -- 4 -- 400.0 --

1004 -- Juice -- 15 -- 1050.0 --

1006 -- Biscuit -- 8 -- 240.0 --

1002 -- Bread -- 19 -- 760.0 --

1010 -- Ketchup -- 3 -- 300.0 --

1009 -- Eggs(6) -- 6 -- 180.0 --
```

Querying transaction database

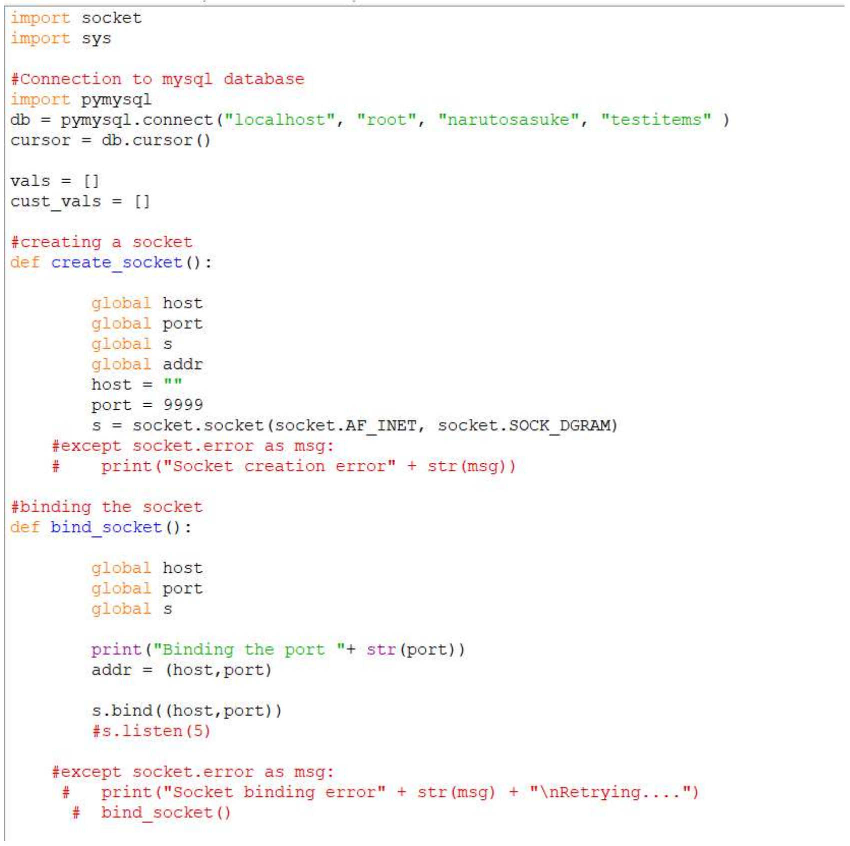


Administrator: C:\Windows\System32\cmd.exe

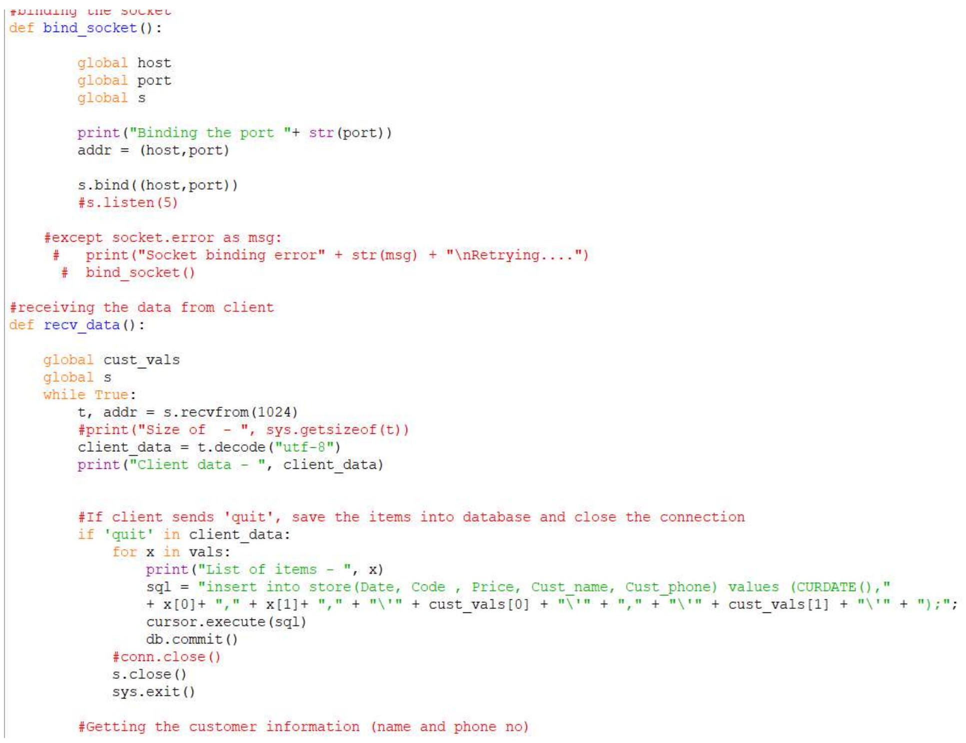
```
1008 -- Chocolate -- 4 -- 400.0 --

1004 -- Juice -- 15 -- 1050.0 --

1006 -- Biscuit -- 8 -- 240.0 --

1002 -- Bread -- 19 -- 760.0 --

1010 -- Ketchup -- 3 -- 300.0 --

1009 -- Eggs(6) -- 6 -- 180.0 --

Enter 0 for exit anytime
Enter any other number to continue
1
Enter the type of transaction information:
1. Daily Earnings
2. Items sold
3. Customers
3
None -- None -- 3050.0 --

Suhaas -- 12345678 -- 330.0 --

Suhaas -- 9462470606 -- 220.0 --

Ssfa -- 3423423 -- 60.0 --

Thomas -- 1242344534 -- 150.0 --

Suhaas -- 12313124 -- 180.0 --

Chinmay -- 9012367489 -- 430.0 --

Chinnu -- 6345789039 -- 180.0 --

Naruto -- 1234567890 -- 140.0 --

Ronaldo -- 4235564256 -- 270.0 --

Suhaas -- 8296856780 -- 360.0 --

Joseph -- 9101743890 -- 320.0 --

Trump -- 9130414929 -- 240.0 --

Enter 0 for exit anytime
Enter any other number to continue
0
```

The Python codes used in this project are as follows

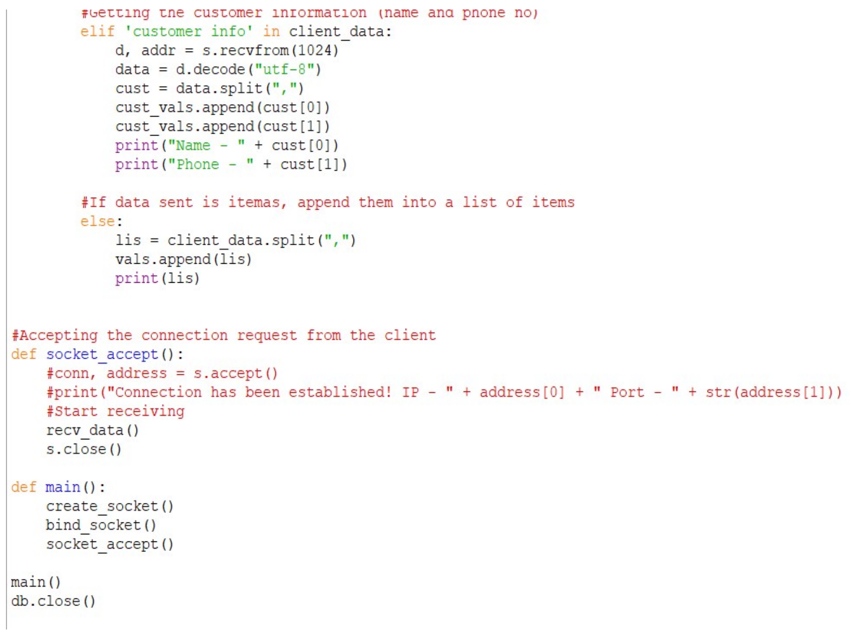
wxserver.py code



```
import socket
import sys

#Connection to mysql database
import pymysql
db = pymysql.connect("localhost", "root", "narutosasuke", "testitems" )
cursor = db.cursor()

vals = []
cust_vals = []

#creating a socket
def create_socket():

        global host
        global port
        global s
        global addr
        host = ""
        port = 9999
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    #except socket.error as msg:
    #    print("Socket creation error" + str(msg))

#binding the socket
def bind_socket():

        global host
        global port
        global s

        print("Binding the port "+ str(port))
        addr = (host,port)

        s.bind((host,port))
        #s.listen(5)

    #except socket.error as msg:
     #    print("Socket binding error" + str(msg) + "\nRetrying....")
      #  bind_socket()
```

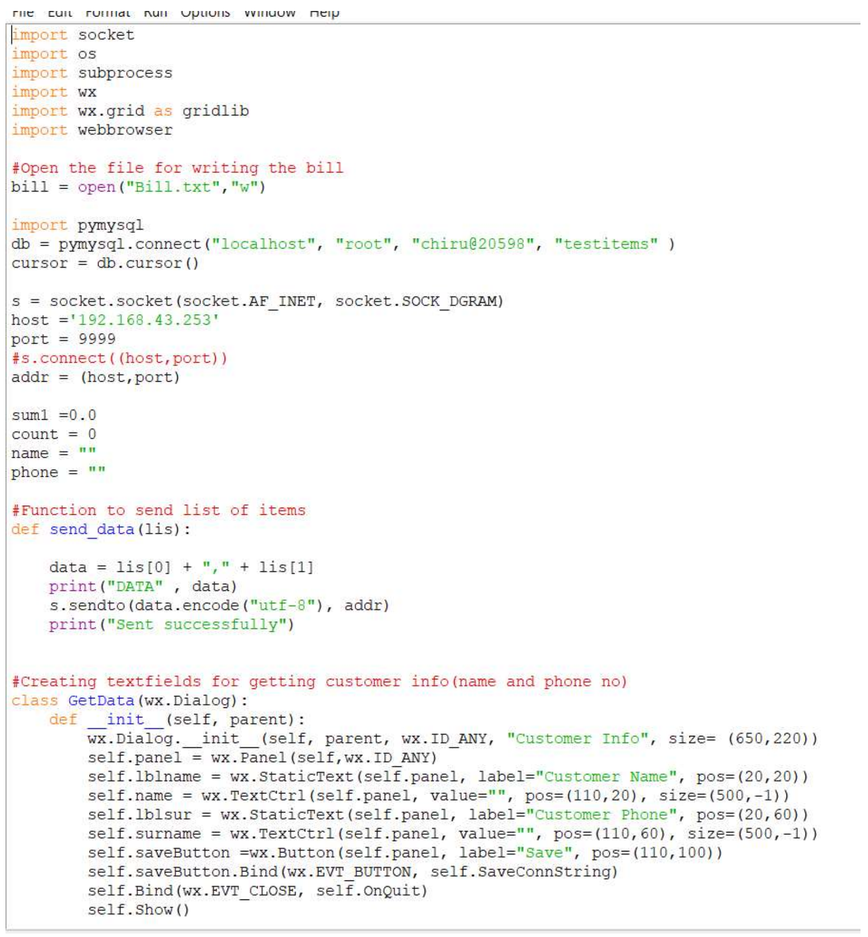


```
#binding the socket
def bind_socket():

        global host
        global port
        global s

        print("Binding the port "+ str(port))
        addr = (host,port)

        s.bind((host,port))
        #s.listen(5)

    #except socket.error as msg:
     #    print("Socket binding error" + str(msg) + "\nRetrying....")
      #  bind_socket()

#receiving the data from client
def recv_data():

    global cust_vals
    global s
    while True:
        t, addr = s.recvfrom(1024)
        #print("Size of  - ", sys.getsizeof(t))
        client_data = t.decode("utf-8")
        print("Client data - ", client_data)


        #If client sends 'quit', save the items into database and close the connection
        if 'quit' in client_data:
            for x in vals:
                print("List of items - ", x)
                sql = "insert into store(Date, Code , Price, Cust_name, Cust_phone) values (CURDATE(),"
                + x[0]+ "," + x[1]+ "," + "\'" + cust_vals[0] + "\'" + "," + "\'" + cust_vals[1] + "\'" + ");";
                cursor.execute(sql)
                db.commit()
            #conn.close()
            s.close()
            sys.exit()

        #Getting the customer information (name and phone no)
```

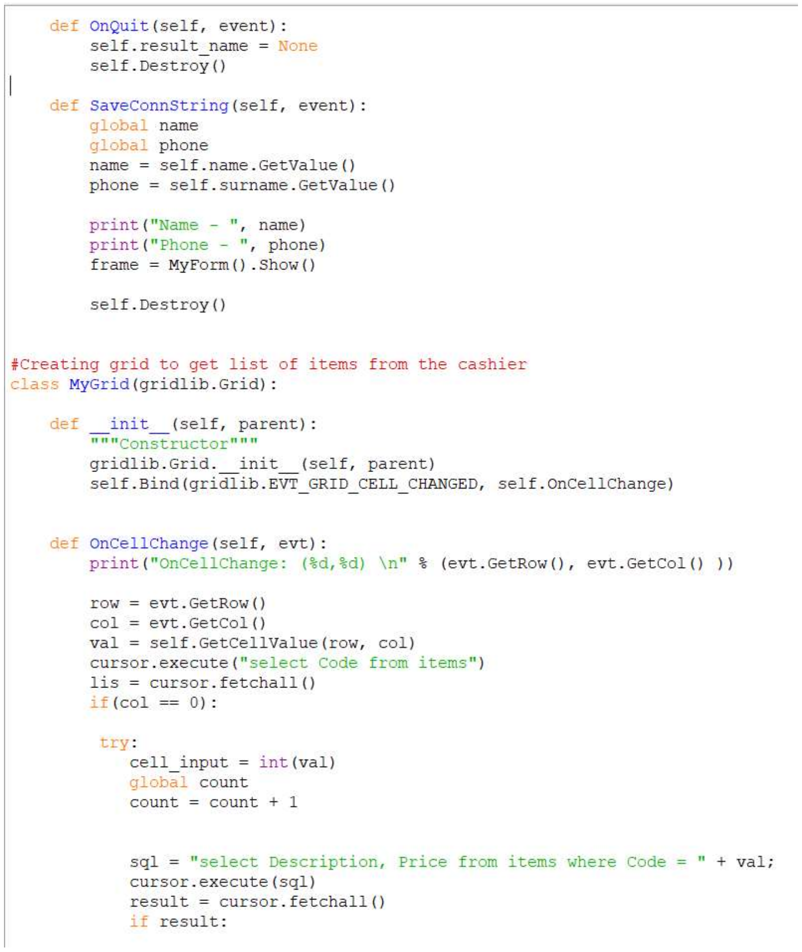


```
        #Getting the customer information (name and phone no)
        elif 'customer info' in client_data:
            d, addr = s.recvfrom(1024)
            data = d.decode("utf-8")
            cust = data.split(",")
            cust_vals.append(cust[0])
            cust_vals.append(cust[1])
            print("Name - " + cust[0])
            print("Phone - " + cust[1])

        #If data sent is itemas, append them into a list of items
        else:
            lis = client_data.split(",")
            vals.append(lis)
            print(lis)


#Accepting the connection request from the client
def socket_accept():
    #conn, address = s.accept()
    #print("Connection has been established! IP - " + address[0] + " Port - " + str(address[1]))
    #Start receiving
    recv_data()
    s.close()

def main():
    create_socket()
    bind_socket()
    socket_accept()

main()
db.close()
```

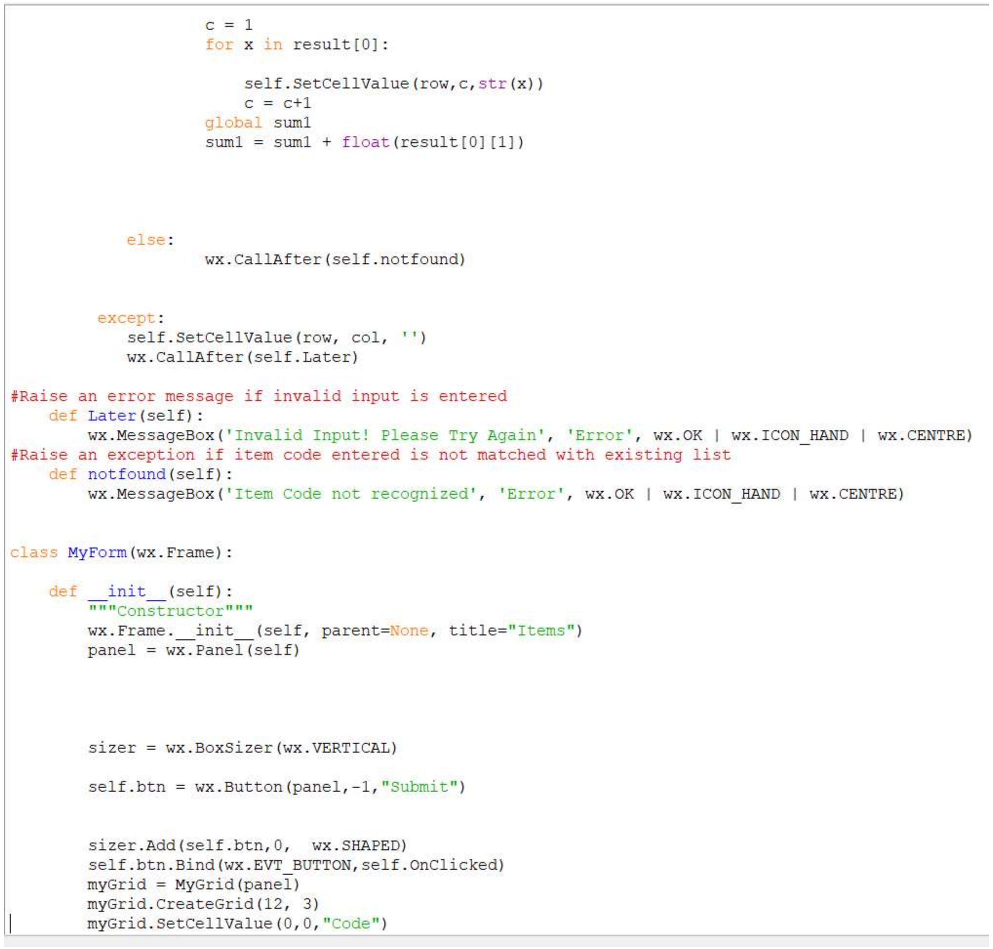
wxclient.py program



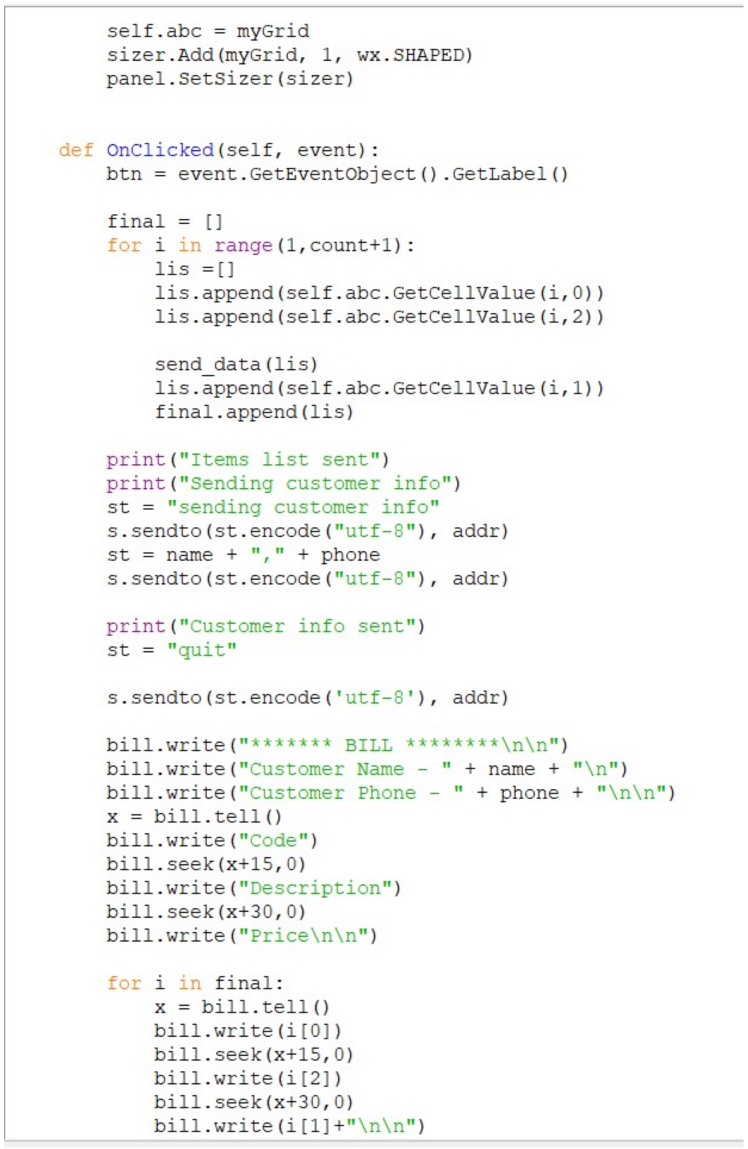
```
import socket
import os
import subprocess
import wx
import wx.grid as gridlib
import webbrowser

#Open the file for writing the bill
bill = open("Bill.txt","w")

import pymysql
db = pymysql.connect("localhost", "root", "chiru@20598", "testitems" )
cursor = db.cursor()

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
host ='192.168.43.253'
port = 9999
#s.connect((host,port))
addr = (host,port)

sum1 =0.0
count = 0
name = ""
phone = ""

#Function to send list of items
def send_data(lis):

    data = lis[0] + "," + lis[1]
    print("DATA" , data)
    s.sendto(data.encode("utf-8"), addr)
    print("Sent successfully")


#Creating textfields for getting customer info(name and phone no)
class GetData(wx.Dialog):
    def __init__(self, parent):
        wx.Dialog.__init__(self, parent, wx.ID_ANY, "Customer Info", size= (650,220))
        self.panel = wx.Panel(self,wx.ID_ANY)
        self.lblname = wx.StaticText(self.panel, label="Customer Name", pos=(20,20))
        self.name = wx.TextCtrl(self.panel, value="", pos=(110,20), size=(500,-1))
        self.lblsur = wx.StaticText(self.panel, label="Customer Phone", pos=(20,60))
        self.surname = wx.TextCtrl(self.panel, value="", pos=(110,60), size=(500,-1))
        self.saveButton =wx.Button(self.panel, label="Save", pos=(110,100))
        self.saveButton.Bind(wx.EVT_BUTTON, self.SaveConnString)
        self.Bind(wx.EVT_CLOSE, self.OnQuit)
        self.Show()
```



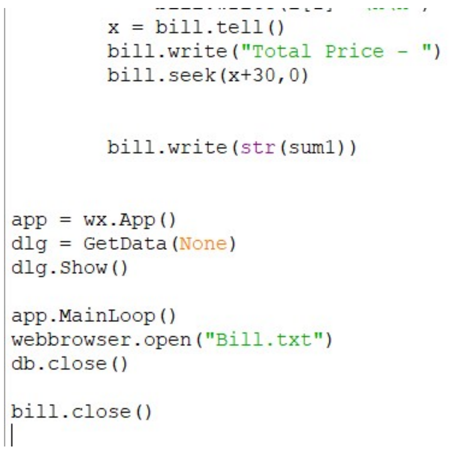
```
    def OnQuit(self, event):
        self.result_name = None
        self.Destroy()

    def SaveConnString(self, event):
        global name
        global phone
        name = self.name.GetValue()
        phone = self.surname.GetValue()

        print("Name - ", name)
        print("Phone - ", phone)
        frame = MyForm().Show()

        self.Destroy()


#Creating grid to get list of items from the cashier
class MyGrid(gridlib.Grid):

    def __init__(self, parent):
        """Constructor"""
        gridlib.Grid.__init__(self, parent)
        self.Bind(gridlib.EVT_GRID_CELL_CHANGED, self.OnCellChange)


    def OnCellChange(self, evt):
        print("OnCellChange: (%d,%d) \n" % (evt.GetRow(), evt.GetCol() ))

        row = evt.GetRow()
        col = evt.GetCol()
        val = self.GetCellValue(row, col)
        cursor.execute("select Code from items")
        lis = cursor.fetchall()
        if(col == 0):

         try:
            cell_input = int(val)
            global count
            count = count + 1


            sql = "select Description, Price from items where Code = " + val;
            cursor.execute(sql)
            result = cursor.fetchall()
            if result:
```



```
                    c = 1
                    for x in result[0]:

                        self.SetCellValue(row,c,str(x))
                        c = c+1
                    global sum1
                    sum1 = sum1 + float(result[0][1])




            else:
                    wx.CallAfter(self.notfound)


         except:
            self.SetCellValue(row, col, '')
            wx.CallAfter(self.Later)

#Raise an error message if invalid input is entered
    def Later(self):
        wx.MessageBox('Invalid Input! Please Try Again', 'Error', wx.OK | wx.ICON_HAND | wx.CENTRE)
#Raise an exception if item code entered is not matched with existing list
    def notfound(self):
        wx.MessageBox('Item Code not recognized', 'Error', wx.OK | wx.ICON_HAND | wx.CENTRE)


class MyForm(wx.Frame):

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, parent=None, title="Items")
        panel = wx.Panel(self)




        sizer = wx.BoxSizer(wx.VERTICAL)

        self.btn = wx.Button(panel,-1,"Submit")


        sizer.Add(self.btn,0,  wx.SHAPED)
        self.btn.Bind(wx.EVT_BUTTON,self.OnClicked)
        myGrid = MyGrid(panel)
        myGrid.CreateGrid(12, 3)
        myGrid.SetCellValue(0,0,"Code")
```



```
        self.abc = myGrid
        sizer.Add(myGrid, 1, wx.SHAPED)
        panel.SetSizer(sizer)


    def OnClicked(self, event):
        btn = event.GetEventObject().GetLabel()

        final = []
        for i in range(1,count+1):
            lis =[]
            lis.append(self.abc.GetCellValue(i,0))
            lis.append(self.abc.GetCellValue(i,2))

            send_data(lis)
            lis.append(self.abc.GetCellValue(i,1))
            final.append(lis)

        print("Items list sent")
        print("Sending customer info")
        st = "sending customer info"
        s.sendto(st.encode("utf-8"), addr)
        st = name + "," + phone
        s.sendto(st.encode("utf-8"), addr)

        print("Customer info sent")
        st = "quit"

        s.sendto(st.encode('utf-8'), addr)

        bill.write("******* BILL ********\n\n")
        bill.write("Customer Name - " + name + "\n")
        bill.write("Customer Phone - " + phone + "\n\n")
        x = bill.tell()
        bill.write("Code")
        bill.seek(x+15,0)
        bill.write("Description")
        bill.seek(x+30,0)
        bill.write("Price\n\n")

        for i in final:
            x = bill.tell()
            bill.write(i[0])
            bill.seek(x+15,0)
            bill.write(i[2])
            bill.seek(x+30,0)
            bill.write(i[1]+"\n\n")
```



```
        x = bill.tell()
        bill.write("Total Price - ")
        bill.seek(x+30,0)


        bill.write(str(sum1))


app = wx.App()
dlg = GetData(None)
dlg.Show()

app.MainLoop()
webbrowser.open("Bill.txt")
db.close()

bill.close()
```

Roles of team members

D. G. Suhaas Kiran – Creation of SQL databases for maintaining transaction details and also keep track of the commodities. These databases are modified dynamically with respect to user inputs.

Chiranjeevi N - Using wxPython to create a graphical user interface for interactive communication with the user at the client side.

Joseph Sunil - UDP client and server codes for full duplex communication and lining them to the GUI.

Chinmay D Kuchinad – Linking the graphic user interface at the client with the items database and relay the information to the server.

****