GROCERY CALCULATOR

A COURSE PROJECT REPORT

By

B YOGESH (RA2011027010038)
M LALITH KIRAN (RA2011027010052)
S.K SRICHARAN(RA2011027010053)

Under the guidance of

Dr. Anand M

(Assistant Professor, Department of Data Science and Business Systems)

In partial fulfilment for the Course

of

18CSC302J - COMPUTER NETWORKS
In Data Science and Business Systems

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**Grocery Calculator**" is the bonafide work of B Yogesh (RA2011027010038), M Lalith Kiran (RA2011027010052) and S.K Sricharan (RA2011027010053) who carried out the project work under my supervision.

SIGNATURE

Dr. Anand M
Assistant Professor
Department of Data Science and Business Systems
SRM Institute of Science and Technology

ABSTRACT

The grocery store project deals with the automation of supermarket. The project Grocery Calculator is developed with the objective of making the system reliable, easier, fast and more informative. There is a lot of reason for the introduction of this project. In the manual System, there are number of inefficiencies that a salesperson faces. Large record books have to be maintained where relevant and irrelevant information has to be stored which is very untidy and clumsy process. But our System reduces paper works. On the other hand, there are many inherent problems that exist in any manual system. Usually, they lack efficiency. Less efficiency has a great impact on the productivity of any human being keeping the data up-to-date.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.
We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement
We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project
We wish to express my sincere thanks to **Course Audit Professor Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty Dr. Anand M**, Assistant Professor, Department of Data Science and Business Systems** for his assistance, timely suggestion and guidance throughout the duration of this course project.
We extend my gratitude to our **HoD, Dr.Vadivu, Professor, Department of Data Science and Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1. INTRODUCTION

This project can compute the grocery price that customer have been brought.
The project is a simple GUI application that use tkinter module in order to generate a readable design view. The project can be accessed without entering a login information. The customer will choose an item in the list while the user checked the customer selected item and enter the quantity. The project will auto calculate the total amount of the successful purchase of groceries when user click the calculate button. This project contains all the basic python functionalities in order for you to learn and enhance your knowledge about python.

2. LITERATURE SURVEY

ARTICLE: Journal of Inventions in Computer Science and Communication Technology (JICSCT).ISSN(O): 2455-5738. Volume-3 – Issue 4, Jul-Aug, 2017

AUTHOR: Pooja Parashar, Sarvesh Singh

This report presentation a detail overview in developing socket programming. The Network programming is similar to socket programming or Client-Server programming where Socketprogramming is important to understand how internet based or access interposes communication work. In this we describe about different types of sockets used in interposes communicate on. Network programming basically uses the Client Server model. In Client- Server programming there are two different programs or process, one which in itiates communication called Client process and other who is waiting for communication to start called Server process. the application is built using java and issuing TCP, IP, UDP datagram.the primary objective of this report is to present the principles behind socket programming and the libraries available for socket programming application in java.

ARTICLE: (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5(3), 2014, 4802-4807

AUTHOR: Limi Kalita

The aim of the paper is to introduce sockets, its deployment pertaining to network programming. Sockets play a vital role in client server applications. The client and server can communicate with each other by writing to or reading from these sockets. They were invented in Berkeley as part of the BSD flavor of UNIX operating systems. And they spread like wildfire with the Internet. This paper introduces elements of network programming and concepts involved in creating network applications using sockets. One of the most basic network programming tasks likely to be faced as a java programmer is performing the socket functions/methods because java has been preferred mostly for establishing client server communications using sockets.

Design and Implementation of Client-Server based Application Using SocketProgramming in a Distributed Computing Environment

ARTICLE: Conference: 2017 IEEE International Conference on Computational Intelligence and Computing Research

AUTHOR: Rolou Lyn R. Maata, Ronald Cordova, Balaji Sudramurthy, Alrence Halibas

This research study discusses the detail overview in developing a client server-based application using socket programming in a distributed computing environment. The researchers developed a client-server-based application called OpTel Billing System (OBS) using Java NetBeans and TCP datagram to demonstrate the concepts of socket programming and its communication in a distributed computing. Interface design, socket programming style,java classes, and exceptions are also considered in the development stage. The communications between client server application processes using socket mechanism were mainly analyzed. The main objective of this research study is to demonstrate the principles and concepts behindsocket programming as well as the libraries available in Java. In conclusion, socket programming is one of the best methods in distributed computing that can improve system's performance.

ARTICLE: 2018 JETIR May 2018, Volume 5, Issue 5

AUTHOR: Diksha Walia (M. Tech Scholar, Department of Computer Science and Engineering, MVN University, Palwal), Divyanshu Sinha (Assistant Professor, Department of Computer Science and Engineering, MVN University, Palwal)

This paper aims to summarize the client server protocol implementation. Client-server is a system that performs both the functions of client and server so as to promote the sharing of information between them. It allows many users to have access to the same database at the same time, and the database will store much information. The internet protocol TCP/IP uses computers called gateways, which provide all interconnections among physical network. The purpose of this system is to communicate between clients that are connected to different servers. This paper will provide information about client server mobile computing in terms ofits paradigms, architecture, and framework.

ARTICLE: IEEE Distributed Systems Online (Volume:6, Issue: 10, 2005)

AUTHOR: M. Gordon (Technische Universität Berlin, Germany)

This article is a review of the book, 'Foundations of Python Network Programming' by John Goerzen, published by Apress, 2004. The book covers the standard Python distribution's clientprotocol implementations-such as HTTP, DNS (Domain Name System), SMTP (Simple Mail- Transfer Protocol), FTP, POP (Post Office Protocol), IMAP (Internet Message Access Protocol), and XML-RPQ and server frameworks (for "out of the box" HTTP and XML-RPC servers). The Database Clients chapter admirably introduces DB-API 2.0, the Python equivalent of Java database connectivity or open database connectivity. Other chapters explainPython interfaces to the Secure Sockets Layer, HTML/XML parsing, and the mod/spl I.bar/python Apache module, as well as more advanced network programming topics such as IPv6 support, multithreaded servers, and nonblocking 1/0. The book also incorporates third- party networking libraries such as Twisted and PyDNS. Each chapter's text thoroughly explains the code samples, which are far more comprehensive than in the standard documentation. The reviewer feels that the book complements other introductory texts.

USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS

AUTHOR: D. B. Beniz, A. M. Espindola, Brazilian Synchrotron Light Laboratory, Campinas, Brazil

Tkinter, or “Tk interface”, is a module of python that provides an interface to Tk GUI toolkit, developed in TCL (Tool Command Language) and multiplatform, with support for Linux, MAC OS and MS Windows. Tk is natively present in Linux and MAC OS, and can be easily installed on MS Windows, it is not part of Python. Tkinter is part of Python, being called “Tkinter” in versions prior to 3, and “tkinter” on version. Widgets, geometry management and event handling are the three main concepts of Tk, which also apply for Tkinter.

Widgets

Often referred to as controls, or window elements, widgets are all visible components on a graphical inter-face. Some examples are frames, labels, buttons, text entries, checkboxes, tree views, scrollbars, and text areas.

https://www.researchgate.net/publication/320827237

Published in: IEEE Distributed Systems Online (Volume: 6, Issue: 10, 2005)
DOI: 10.1109/MDSO.2005.50
Publisher: IEEE

Foundations of Python Network Programming covers the standard Python distribution's client protocol implementations-such as HTTP, DNS (Domain Name System), SMTP (Simple Mail-Transfer Protocol), FTP, POP (Post Office Protocol), IMAP (Internet Message Access Protocol), and XML-RPC) and server frameworks (for "out of the box" HTTP and XML-RPC servers). The Database Clients chapter admirably introduces DB-API 2.0, the Python equivalent of Java database connectivity or open database connectivity. Other chapters explain Python interfaces to the Secure Sockets Layer, HTML/XML parsing, and the mod python Apache module, as well as more advanced network programming topics such as IPv6 support, multithreaded servers, and nonblocking *I/O*. The book also incorporates third-party networking libraries such as Twisted and PyDNS. Each chapter's text thoroughly explains the code samples, which are far more comprehensive than in the standard documentation (and a primary selling point of Goerzen's book).

3. REQUIREMENTS

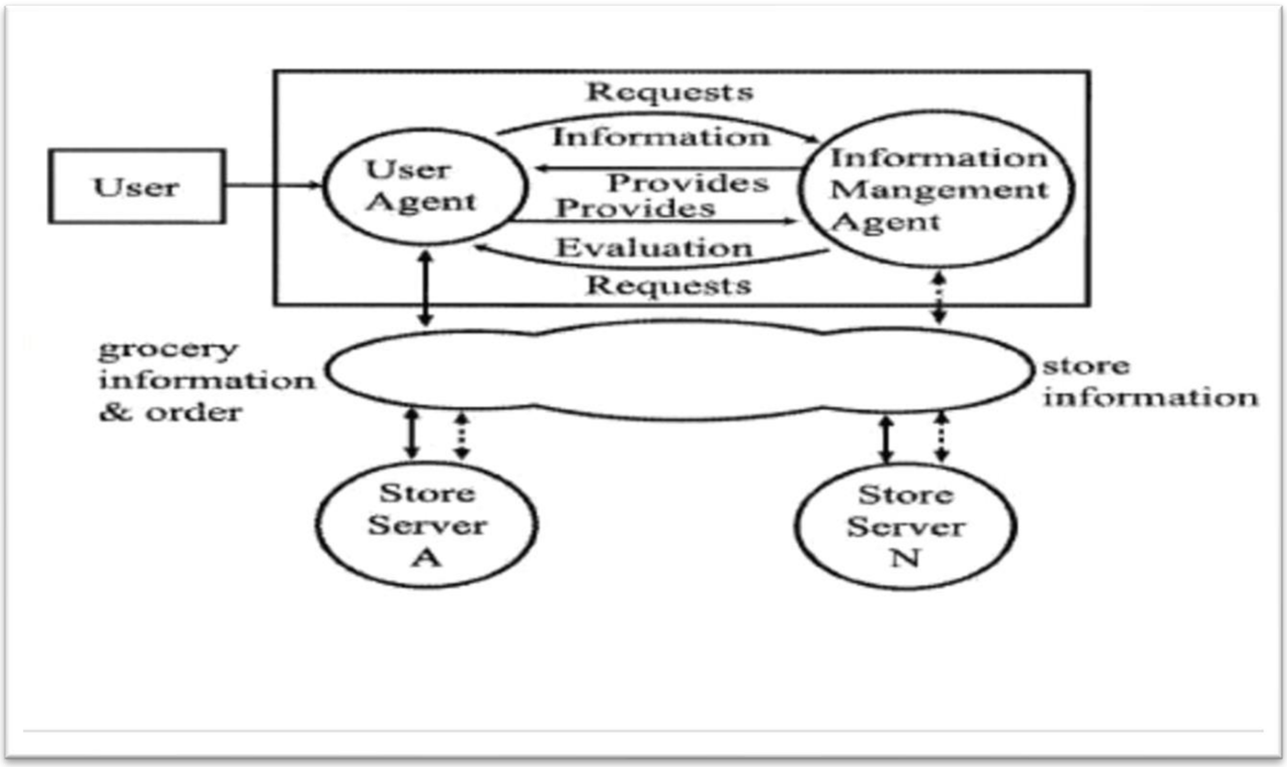
Software Requirements

Operating system	:	Windows XP/8/8.1 or linux
User interface	:	Tkinter
Programming language	:	Python

Hardware Requirements

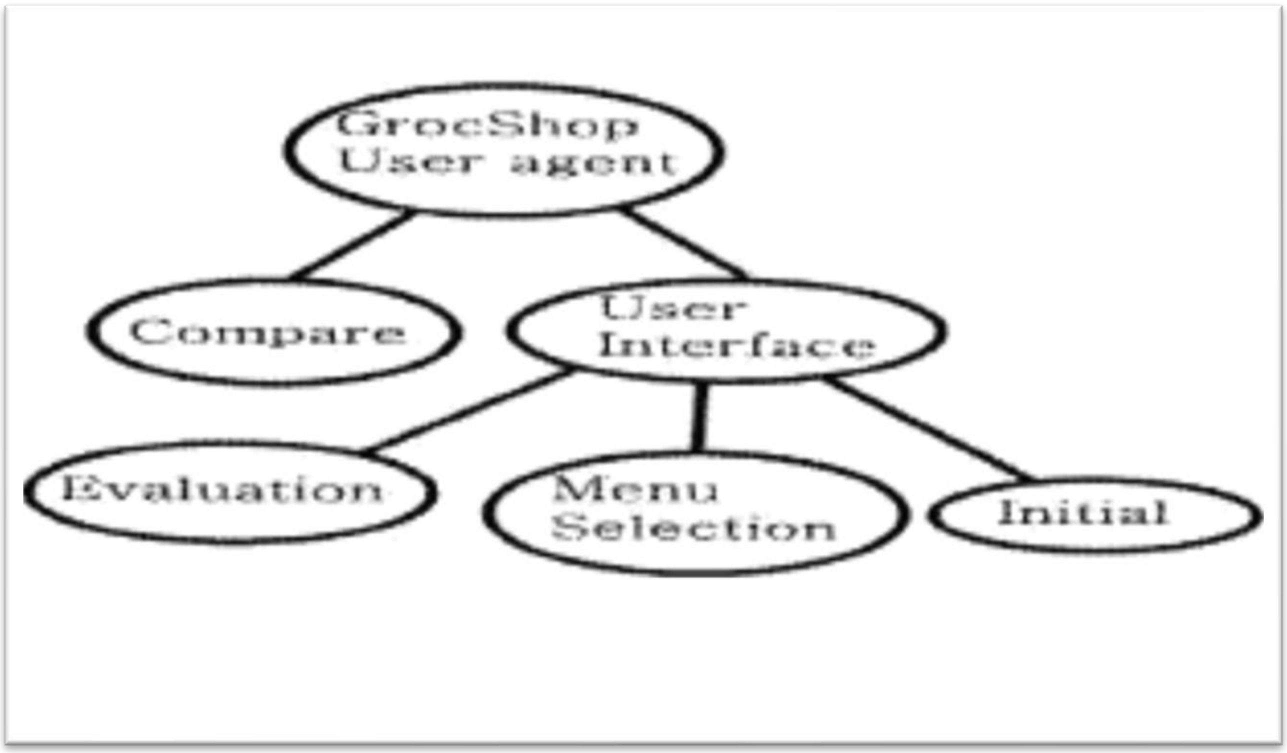
Memory	:	4 GB
Hard Disk Space	:	240 GB or less
CPU	:	Intel Core i3 and above
RAM	:	4GB
Pixel Shader	:	5.1

4. ARCHITECTURE AND DESIGN



Structure of grocery calculator

User’s activity of grocery shopping consists of many small tasks such as selection, information gathering, comparison, decision, and evaluation. A user conducts each task independently for other tasks, and chooses an alternative from multiple strategies for each task. We have developed an agent for a task, because it is natural and easy to develop an agent that imitates user’s activity. We call it a primitive agent. The primitive agent executes a specific small task/function. It is necessary to group primitive agents,because the role agents execute several activities to achieve the goal of the system. We have introduced an agent that groups primitive agents and call it an organization agent.The organization agent achieves user’s goal using domain knowledge.Natural logical sequence leads the programmer to build multiple primitive agents for each task.



Structure of user agent.

UA orders groceries from the best provider of each grocery item on behalf of a user. We have designed UA to cover the requirements. UA is a role agent that organizes agents to realize an easy-to-use human interface of the agent-based grocery shopping. UA is responsible for managing and controlling the whole process of automating grocery shopping.Compare agent is a primitive agent that gathers grocery information, compares it, and places orders with grocery stores. The compare agent utilizes knowledge of the user preference for both groceries and stores that are managed by IMA, to provide the user-oriented shopping services for the user. The user defines and

utilizes the user-specific compare agents with different comparison algorithms. User Interface agent is an organization agent to manage and control the following primitive agents which interact with the user by using knowledge of respective agents

5. IMPLEMENTATION

WORKING:

When the customer connects to the server using socket programming in python with the help of the socket module in python. Then the grocery items display on The screen along with their prices, after customer selects the items then total amount to be paid will display on the screen and customer pay bill for his groceries.

Server :

```
Import
socket
        import json
        import threading
        import config

        # handling the connection
        def threaded_answer(address, conn):

            while True:
                str_data = conn.recv(1024).decode()
                if not str_data:
                    break
                data = json.loads(str_data)

                print("got connection from address %s on port %s"%(address[0], address[1]))
                print("got request")
                print(data)
                # check the request
                request_type = data['request']
                resp = {}

                if request_type == 'cash':
                    resp['request'] = 'cash'
                    total = sum([config.PRICES[fruit]*data['data'][fruit] for fruit in
        data['data'].keys() ])
                    resp['data'] = total
                elif request_type == 'prices':
                    resp['request'] = 'prices'
                    resp['data'] = config.PRICES
                print("respones is :")
                print(resp)

                str_resp = json.dumps(resp).encode()
                conn.send(str_resp)
            conn.close()




        with socket.socket() as my_socket:
            # AF_INET refers to the address family ipv4. The SOCK_STREAM means connection
        oriented TCP protocol.
            my_socket.bind((config.IP_ADDRESS, config.PORT_NUMBER))

            my_socket.listen(5)
            counter = 0

            while True:
                conn, add = my_socket.accept()
                thread = threading.Thread(target=threaded_answer, args=(add, conn,))
                thread.start()
```

Client:

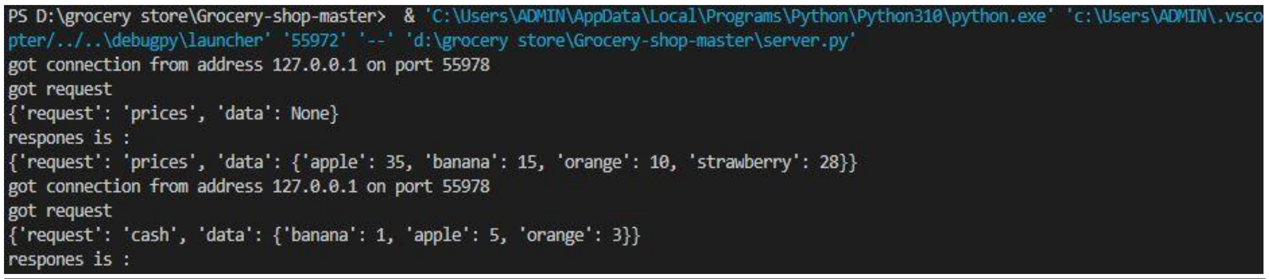
```
Import
json
         import socket

         import config


         class Client:
             def __init__(self):
                 try:
                     self.socket = socket.socket()
                     self.socket.connect((config.IP_ADDRESS, config.PORT_NUMBER))
                 except:
                     raise Exception("make sure the server is up")

             def get_prices(self):
                 # create a prices request
                 prices = {
```

```
                'request': 'prices',
                'data': None
            }
            prices_str = json.dumps(prices).encode()
            self.socket.send(prices_str)

            response_str = self.socket.recv(1024).decode()
            response = json.loads(response_str)
            return response['data']

        def get_cash(self, orders):
            request = {'request': 'cash'}
            request['data'] = orders
            orders_str = json.dumps(request).encode()
            self.socket.send(orders_str)

            response = self.socket.recv(1024).decode()
            response = json.loads(response)
            return response['data']


    if __name__ == '__main__':
        # for testing
        prices = {
        'request': 'prices',
        'data': None
        }

        cash = {
            'request': 'cash',
            'data': {
                'banana': 1,
                'apple': 5,
                'orange': 3
            }
        }

        my_socket = socket.socket()

        my_socket.connect((config.IP_ADDRESS, config.PORT_NUMBER))

        print("testing the prices request")
        prices_str = json.dumps(prices).encode()
        my_socket.send(prices_str)

        response_str = my_socket.recv(1024).decode()
        response = json.loads(response_str)
        print("got response:")
        print(response)

        print("testing the cash request")
        cash_str = json.dumps(cash).encode()
        my_socket.send(cash_str)

    response = my_socket.recv(1024).decode()
        print("got response:")
        print(response)
    my_socket.close()
```

Shopy:

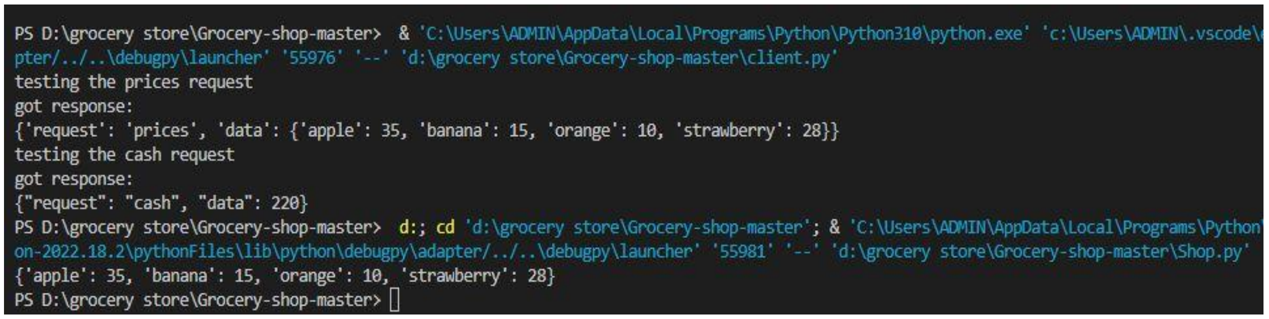
```
import
tkinter

import client


class Shop:
    def __init__(self, master):
        # build the gui

        self.client = client.Client()
        self.prices = self.client.get_prices()
        print(self.prices)

        # add the apple widgets
        self.apple_img = tkinter.PhotoImage(file='res/apple.png')
        self.apple_info = tkinter.StringVar()
        self.apple_info.set('Apples %.2f$' % (self.prices['apple']))
        self.apple_label = tkinter.Label(textvariable=self.apple_info,
image=self.apple_img,
                                        width=250, height=250,
compound=tkinter.TOP)
        self.apple_label.config(font=("console", 18, 'bold'))
        self.apple_label.grid(row=0, column=1)
        self.apple_label.configure(background='white')
        self.apple_spin = tkinter.Spinbox(from_=0, to=100, font=("console", 20,
'bold'), width=5, state='readonly')
        self.apple_spin.grid(row=1, column=1)

        # add the banana widgets
        self.banana_img = tkinter.PhotoImage(file='res/banana.png')
        self.banana_info = tkinter.StringVar()
        self.banana_info.set('Banana %.2f$' % (self.prices['banana']))
        self.banana_label = tkinter.Label(textvariable=self.banana_info,
image=self.banana_img,
                                  width=250, height=250,
compound=tkinter.TOP)
        self.banana_label.config(font=("console", 18, 'bold'))
        self.banana_label.grid(row=0, column=2)
        self.banana_label.configure(background='white')
        self.banana_spin = tkinter.Spinbox(from_=0, to=100, font=("console", 20,
'bold'), width=5, state='readonly')
        self.banana_spin.grid(row=1, column=2)

        # add the orange widgets
        self.orange_img = tkinter.PhotoImage(file='res/orange.png')
        self.orange_info = tkinter.StringVar()
        self.orange_info.set('Orange %.2f$' % (self.prices['orange']))
        self.orange_label = tkinter.Label(textvariable=self.orange_info,
image=self.orange_img,
                                  width=250, height=250,
compound=tkinter.TOP)
        self.orange_label.config(font=("console", 18, 'bold'))
        self.orange_label.grid(row=0, column=3)
        self.orange_label.configure(background='white')
        self.orange_spin = tkinter.Spinbox(from_=0, to=100, font=("console", 20,
'bold'), width=5, state='readonly')
        self.orange_spin.grid(row=1, column=3)

        # add the strawberry widgets
        self.strb_img = tkinter.PhotoImage(file='res/strawberry.png')
        self.strb_info = tkinter.StringVar()
        self.strb_info.set('Strawberry %.2f$' % (self.prices['strawberry']))
        self.strb_label = tkinter.Label(textvariable=self.strb_info,
image=self.strb_img, width=250,
                                 height=250, compound=tkinter.TOP)
        self.strb_label.config(font=("console", 18, 'bold'))
        self.strb_label.grid(row=0, column=4)
        self.strb_label.configure(background='white')
        self.strb_spin = tkinter.Spinbox(from_=0, to=100, font=("console", 20,
'bold'), width=5, state='readonly')
        self.strb_spin.grid(row=1, column=4)

        # add total cost label
        self.total_cost = tkinter.StringVar()
        self.total_cost.set("Total Cost is 0$")
        self.total_cost_widget = tkinter.Label(textvariable=self.total_cost,
font=("console", 20, 'bold'))
        self.total_cost_widget.configure(background='white')
        self.total_cost_widget.grid(row=3, column=3)
        # add the checkout button
        self.check_out = tkinter.Button(text="checkout", command=self.checkout, 
font=("console", 20, 'bold'))
        self.check_out.grid(row=3, columns=2)

    def checkout(self):
        # get the purchased list
        purchased = dict()
        purchased['apple'] = int(self.apple_spin.get())
        purchased['orange'] = int(self.orange_spin.get())
        purchased['banana'] = int(self.banana_spin.get())
        purchased['strawberry'] = int(self.strb_spin.get())
        # get total
        cost = self.client.get_cash(purchased)
        print("cost is")
        print(cost)
        self.total_cost.set("Total Cost is %.2f$" % cost)



root = tkinter.Tk()
root.minsize(320, 240)
root.configure(background='white')
root.grid_rowconfigure(2, minsize=100)
root.grid_rowconfigure(4, minsize=100)

obj = Shop(root)
root.mainloop()
```

Server:



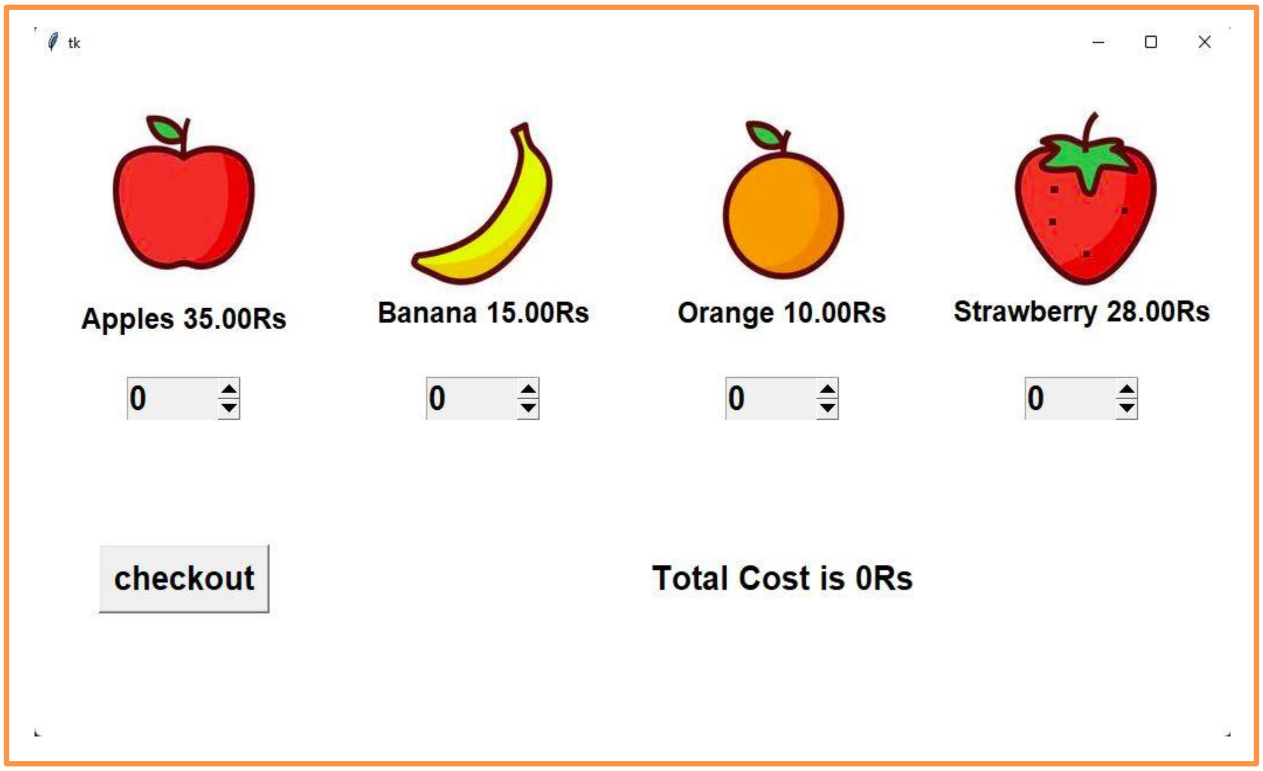
```
PS D:\grocery store\Grocery-shop-master>  & 'C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\ADMIN\.vsco
pter/../..\debugpy\launcher' '55972' '--' 'd:\grocery store\Grocery-shop-master\server.py'
got connection from address 127.0.0.1 on port 55978
got request
{'request': 'prices', 'data': None}
respones is :
{'request': 'prices', 'data': {'apple': 35, 'banana': 15, 'orange': 10, 'strawberry': 28}}
got connection from address 127.0.0.1 on port 55978
got request
{'request': 'cash', 'data': {'banana': 1, 'apple': 5, 'orange': 3}}
respones is :
```

Client:

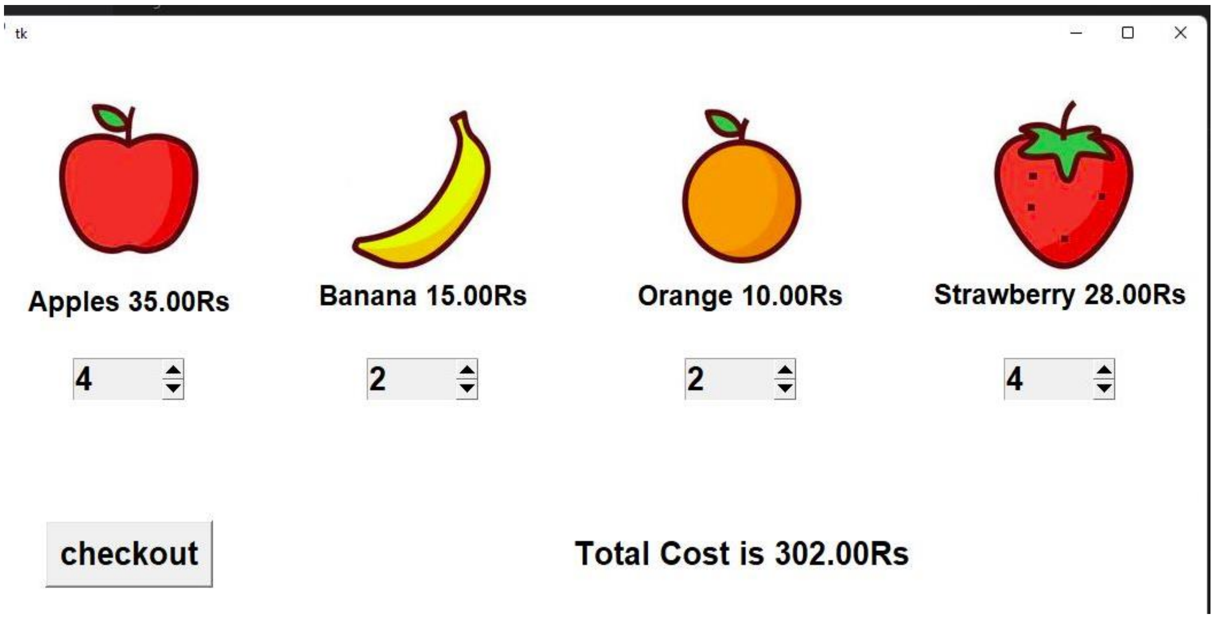


```
PS D:\grocery store\Grocery-shop-master>  & 'C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\ADMIN\.vscode\
pter/../..\debugpy\launcher' '55976' '--' 'd:\grocery store\Grocery-shop-master\client.py'
testing the prices request
got response:
{'request': 'prices', 'data': {'apple': 35, 'banana': 15, 'orange': 10, 'strawberry': 28}}
testing the cash request
got response:
{"request": "cash", "data": 220}
PS D:\grocery store\Grocery-shop-master>  d:; cd 'd:\grocery store\Grocery-shop-master'; & 'C:\Users\ADMIN\AppData\Local\Programs\Python
on-2022.18.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '55981' '--' 'd:\grocery store\Grocery-shop-master\Shop.py'
{'apple': 35, 'banana': 15, 'orange': 10, 'strawberry': 28}
PS D:\grocery store\Grocery-shop-master> 
```

6. RESULTS AND DISCUSSION



Total Cost is 0Rs



Total Cost is 302.00Rs

7. CONCLUSION AND FUTURE ENHANCEMENT

The project entitled ‘Grocery calculator ‘is very convenient for the Grocery Stores. This system is very convenient for customer or users to buy grocery items.

It can be observed that the information can be obtained easily and accurately. The grocery calculator is made more user friendly to the users, so that anyone can run this application.

The grocery calculator can be later be updated by adding more grocery items to the list for the user to purchase and the list is not limited.

Another enhancement that can be done is that we can also complete the payment process in the calculator itself.

8. REFERENCES

1. **ARTICLE:** Journal of Inventions in Computer Science and Communication Technology (JICSCT).ISSN(O): 2455-5738. Volume-3 – Issue 4, Jul-Aug, 2017

   AUTHOR: Pooja Parashar, Sarvesh Singh

2. **ARTICLE:** (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5(3), 2014, 4802-4807

   AUTHOR: Limi Kalita

3. **ARTICLE:** Conference: 2017 IEEE International Conference on Computational Intelligence and Computing Research

   AUTHOR: Rolou Lyn R. Maata, Ronald Cordova, Balaji Sudramurthy, Alrence Halibas

4. **ARTICLE:** 2018 JETIR May 2018, Volume 5, Issue 5

   AUTHOR: Diksha Walia (M. Tech Scholar, MVN University, Palwal), Divyanshu Sinha (Assistant Professor, MVN University, Palwal)

5. **ARTICLE:** Using Tkinter of python to create Graphical User Interface (GUI) for scripts in LNLS

   AUTHOR: D. B. Beniz, A. M. Espindola, Brazilian Synchrotron Light Laboratory, Campinas, Brazil