PROJECT REPORT

Abstract

The problem statement is -

A small-scale vegetable vendor wants to develop a system to manage his door-delivery service. The system should maintain the list of available vegetables for the day. The system should accept a list of vegetables with the individual quantities required by the customer and assign order numbers for tracking. The system should be able to generate the bill for the order and ensure that orders should be placed for more than Rs. 100/-. The system should confirm the payment through notifications to the customer and the vendor. Appending the order should be disabled after payment. The availability of the vegetables should be updated to the vendor and the customer (during ordering). When the order is ready, a notification should be sent to the customer mentioning the tentative time of delivery.

Objective

The main objective of the project is to design a vegetable management system that can be used by the vendors.

The customers must be able to use this to place their orders. This software must also help the customer to view their order once it has been placed. This must be done through a notification system.

The notification system sends a bill of the order along with the order details to the mail Id of the customer.

The mail ids of the customers are stored as their usernames for their login.

Another important aspect of the software is the management of vegetable details by the vendor. The vendor must be able to make changes to the details of the products

Introduction

Need for this software

Vegetable vendors have a lot of vegetables to handle and they are usually required to manually maintain the details of their vegetables. The sales of the vegetables also become a tedious task for them. Maintaining the stock after each purchase for each product must be done manually.

The delivery of the orders along with duration of the delivery is done by keeping everything in the human memory.

The customers also face trouble while purchasing the products due to lack of knowledge about the list of available vegetables along with their price and about the availability of the stock.

The bill generated by the vendor is a hard copy and keeping track of all the bills is an impossible job for the customers.

This software helps to solve all the problems. It takes care of maintaining the list of vegetables. It also allows the vendor to update the details of the vegetables. This comes as an advantage to the vendor as they need not keep track of the vegetables manually.

The customers also have the liberty of seeing the available list of vegetables along with their price. This makes the ordering easier where the customers only enter the required quantity for the necessary items.

This software also sends a copy of the bill to the customer’s mail Id so that they can keep track of their previous orders.

Requirements engineering

Client details

Vijay V

SSN vegetable vendor

Ph-6379435481

Sprints

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint #** | **Epic** | **User Story #** | **Requirement / User Story** | **Essential or Desirable** | **Description of the Requirement** | **Remarks** |
| 1 | LOGIN | RS1 | As a customer, I want to login to the system to buy products. | Essential | The customer must be able to login to the system to place orders. | Files are used to maintain the details. |
| RS2 | As a vendor, I want to login to the system to access the system. | Essential | The vendor must be able to login to the system to modify the details of the vegetables. | Giving rights to vendor. |
|  |  | RS3 | As a new customer, I want to register to access the system | Essential | The new customer should be able to set his username and password | First time customer accessing the system. |
| 2 | Ordering | RS1 | As a customer, I would like to view all the products along with their details | Essential | The customer must be in a position to view all products so that he can place his order | Displaying a file |
| RS2 | As a customer, I would like to search for the presence of a product | Desirable | In order to place an order for a particular item, the customer must know about its availability. | Searching from a file |
|  |  | RS3 | As a customer, I would like to place an order | Essential | The customer must be able to input the products along with the quantity. | The input is given as a list. |
|  | Billing and notification | RS1 | As a vendor, I must be able to generate a bill | Essential | The vendor must be able to get the customers input and calculate the bill and update the stock | The products must be updated. |
|  |  | RS2 | As a customer, I would like to receive an email of my bill | Essential | The customer must be able to receive a mail after placing the order. The bill must be sent in the mail. | The tentative time of schedule must also be mentioned. |
| 3 | Product details | RS1 | As a vendor, I would like to update all the details of the product | Essential | The vendor must be able to access the product file and update the details of the product. | All the details must be considered. |

Implementation and risk management

Vasundhara B 3122 21 5002 119

Role – Developer, Compilation of Code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint #** | **Epic** | **User Story #** | **Requirement / User Story** | **Remarks on implementation** |
| 3 | Product details | RS1 | As a vendor, I would like to update all the details of the product | The detail to be implemented must be given by the vendor and every detail of the product must be updated on the bases of vendor input. The vendor must also be able to view the product file. |
| 2 | Ordering | RS2 | As a customer, I would like to search for the presence of a product | The customer must be able to input the product name and the file must be traversed to find that product or else an appropriate message must be generated. |

Major contribution is the work with the product file. The vendor is provided with certain options and based on the option the detail of the product file is changed. It is then rewritten into the file. A separate class is defined for displaying the product detail in the format of a table. The vendor inputs are taken and then it is checked with the existing details before processing.

The existing details of the file are stored in a list and the file is traversed through checking for the presence of the product then the necessary details are changed.

The same method is used for displaying the details of a specific product. The list of products is linearly searched and then a table is displayed using the table class defined.

The compilation of the different functions and the GUI part was also a major contribution.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk #** | **Risk Description** | **Probability** | **Impact** | **Mitigation Plan** |
| 1 | While appending products, the vendor must enter details of correct data types | High | Low | Asking the vendor to input values of the correct data types |
| 2 | Deleting a product not in the product list | High | Low | Asking the vendor to enter an existing product |
| 3 | Entering invalid product name or code during modification | High | Low | Displaying an error message and asking the vendor to enter correct details |
| 4 | Entering invalid product name or code | Low | Low | Displaying an error message |

**Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC id** | **RS #** | **Test case description/ condition** | **Test case input** | **Expected Output** | **Result (PASS/ FAIL)** |
| 1 | R1 | Successfully appending the product | The vendor must enter the details of the product | Product successfully added to the list | PASS |
| 2 | R1 | Successfully deleting a product | The vendor must enter either the name or the code of the product | Product successfully deleted from the list | PASS |
| 3 | R1 | Successfully modifying a product | The vendor must enter a correct code or product name, in order for the item to be found and change its stock or price value | Product successfully modified in the list | PASS |
| 4 | R1 | Unsuccessful in deleting the product | The vendor entering a product name not existing in the list | “Product not found” message should be thrown and asking the vendor to enter correct product name | PASS |

Vemula Muni Karthik 3122 21 5002 120

Role – Developer, Client

Main contribution is generating the bill and notifying the customer. The dictionary of products is obtained after the customer orders and the product details which are stored in a list is traversed, checking for the availability of the product. The items are then traversed through the list and the total amount is generated. The product file is updated after reducing the stock.

The same bill is sent to the customers gmail id. The bill is also displayed on the screen for customers verification of the order.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint #** | **Epic** | **User Story #** | **Requirement / User Story** | **Remarks on implementation** |
| 2 | Billing and notification | RS1 | As a vendor, I must be able to generate a bill. | The dictionary of products with the product name as their keys and the quantity as their value is processed to generate a bill |
| RS2 | As a customer, I would like to receive an email of my bill | Sending the bill to their email id, which is the same email id that they use to login to the software. |

Decided the requirements of the software and the various restrictions and functionalities of the software

Risk Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk #** | **Risk Description** | **Probability** | **Impact** | **Mitigation Plan** |
| 1 | Invalid Gmail ID | High | High | Asking user to input valid email Id. |
| 2 | Getting negative bill value | Low | High | Displaying appropriate message to the user. |

**Test Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC id** | **RS #** | **Test case description/ condition** | **Test case input** | **Expected Output** | **Result (PASS/ FAIL)** |
| 1 | RS1 | Entered quantities should be in positive values | Given product quantities in negative values | To display a message to enter in positive values | pass |
| 2 | RS1 | Entered quantities should be in positive values | Given product quantities in positive values | Displaying the bill without any error | pass |
| 3 | RS2 | Customer should receive a notification after successful | Giving invalid G-mail | Email should be sent | fail |
| 4 | RS2 | Customer should receive a notification after successful | Giving invalid G-mail | An error message should be displayed | pass |

Vigneshwaran R 3122 21 5002 122

Role – Scrum master, Developer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint #** | **Epic** | **User Story #** | **Requirement / User Story** | **Remarks on implementation** |
| 1 | Login | RS1 | As a customer, I want to login to the system to buy products. | The username details are stored in a file and accessed in the form of a dictionary. The dictionary is traversed checking for the user input. |
| 1 | Login | RS2 | As a vendor, I want to login to the system to access the system. | The details of the vendor stored in a file are accessed and checked for login |
| 1 | Login | RS3 | As a new customer, I want to register to access the system | The inputs of the new customer are checked based on specific condition and either the new details are stored in the login files or an appropriate message is declared. |

Main contribution is as scrum master. Assigned roles to group members and monitored the progress of the project.

Contributed in designing the login part of the software. The user input is taken and checked with the existing details. The existing details are accessed from the files in the form as a dictionary. If the details are present then the customer or vendor are able to do their tasks.

New user login is taken from the customer and stored in the file for future login. The email id stored as username is also accessed for sending the bill to the customer.

Risk Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk #** | **Risk Description** | **Probability** | **Impact** | **Mitigation Plan** |
| 1 | Invalid username or password | High | High | Displaying appropriate message and asking the user to input the correct details |
| 2 | Using an improper email Id while registration | High | High | Asking the user to input proper mail Id |
| 3 | Without typing the username and password, trying to login | Low | Low | Showing invalid username or password to the user and asking him/her to enter the valid username and password or to register if he/she is a new customer. |

Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case No.** | **Test case condition** | **Test input** | **Expected Outcome** | **Actual Outcome** | **The requirement in RS that is being tested** | **Status** |
| 1 | Successful User Verification | The login to the system should be tried with the login assigned by the admin and the correct password | Login should be successful and the user should enter in to the system | As expected, the login was successful | RS1 | Passed |
| 2 | Unsuccessful User Verification due to wrong password or wrong login id | Login to the system with a wrong detail | Login should fail with an error ‘Invalid Username or Password’ | As expected, the login was unsuccessful and raised Invalid Username or Password | RS1 | Passed |
| 3 | Successful vendor verification | The login to the system should be tried with the login assigned by the admin and the correct password. | Login should be successful and the vendor should enter into the system | As expected, the login was successful | RS2 | Passed |
| 4 | Unsuccessful User Verification due to wrong password or wrong login id | Login to the system with a wrong detail | Login should fail with an error ‘Invalid Username or Password’ | As expected, the login was unsuccessful and raised Invalid Username or Password | RS2 | Passed |
| 5 | Successful new user registration | Register into the system with a unique username and a strong password | Registration should be successful with the unique username and a strong password | As expected, the registration was successful | RS3 | Passed |

Vijay V 3122 21 5002 123

Role – Developer, Client

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint #** | **Epic** | **User Story #** | **Requirement / User Story** | **Remarks on implementation** |
| 2 | Ordering | RS1 | As a customer, I would like to view all the products along with their detail | Class table is used and the details of the product file are accessed and shown to the customer as a table |
| RS3 | As a customer, I would like to place an order | The products along with their price and stock will be displayed alphabetically, and the customer will have to place their order by increasing or decreasing the quantity of the desired products |

Contribution involves taking orders from customers and producing a favorable data structure for the generation of bills. The quantity of the desired products is entered while placing the order. This is then converted into a dictionary. The keys of the dictionary contain the product name whereas the values are its quantities.

This is then used for generation of bill.

The displaying of all the product details is also done. It is done with the help of table class.

Also contributed as the client. Tested the working of the software with various inputs.

**Risk Management**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk #** | **Risk Description** | **Probability** | **Impact** | **Mitigation Plan** |
| 1 | Not selecting any products | low | High | Goes back to the customer page where it displays the other options available to the customer. |

**Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC id | RS # | Test case description/ condition | Test case input | Expected Output | Result (PASS/ FAIL) |
| 1 | Rs1 | Quantities are entered properly | Given product is entered correctly only once | Displaying the products | pass |
| 2 | RS2 | Customer enters the correct entry | Giving correct input | Cart should be added | pass |

Vemuri Somesh 3122 21 5002 121

Role – Developer, Working with Jira tool and GitHub

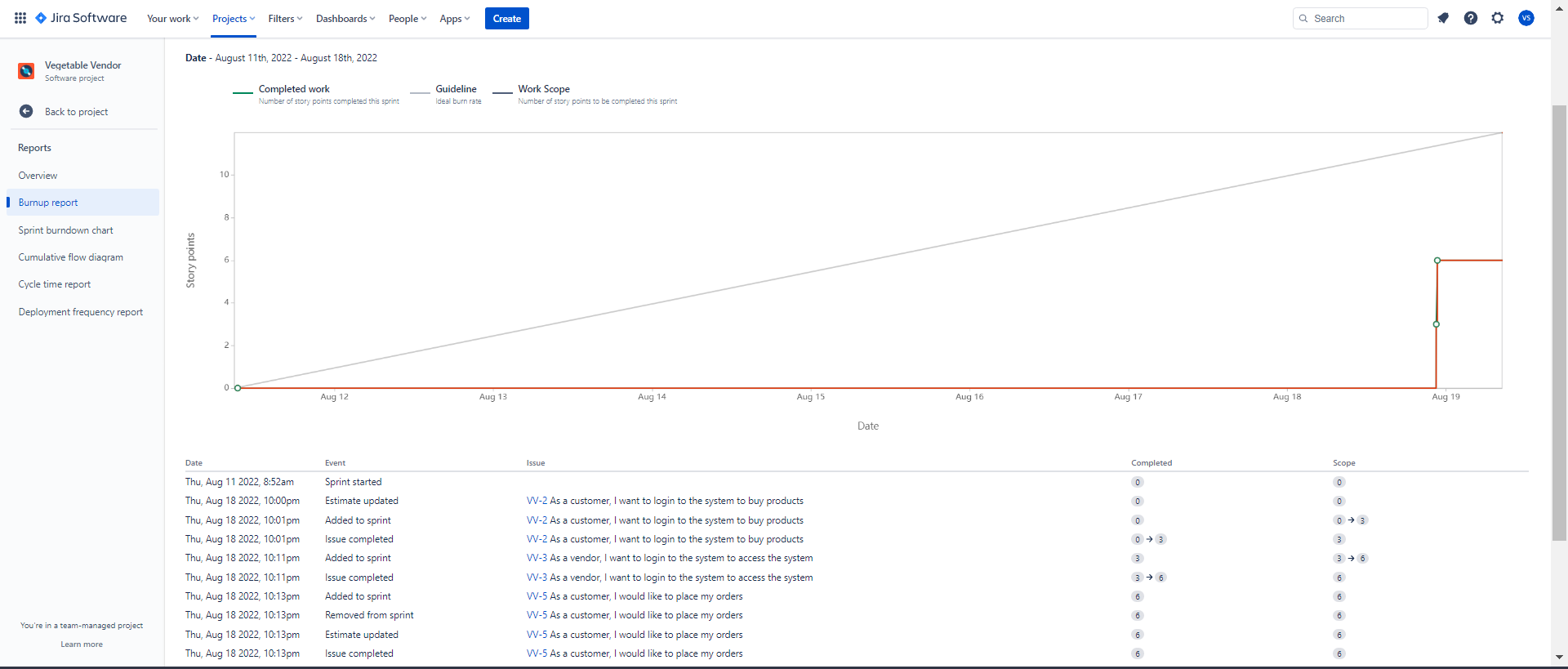
Assisted Karthik and Vijay in their tasks. Shared the same user stories as that of other developers.

Helped Karthik with his email segment and Vijay in designing the GUI.

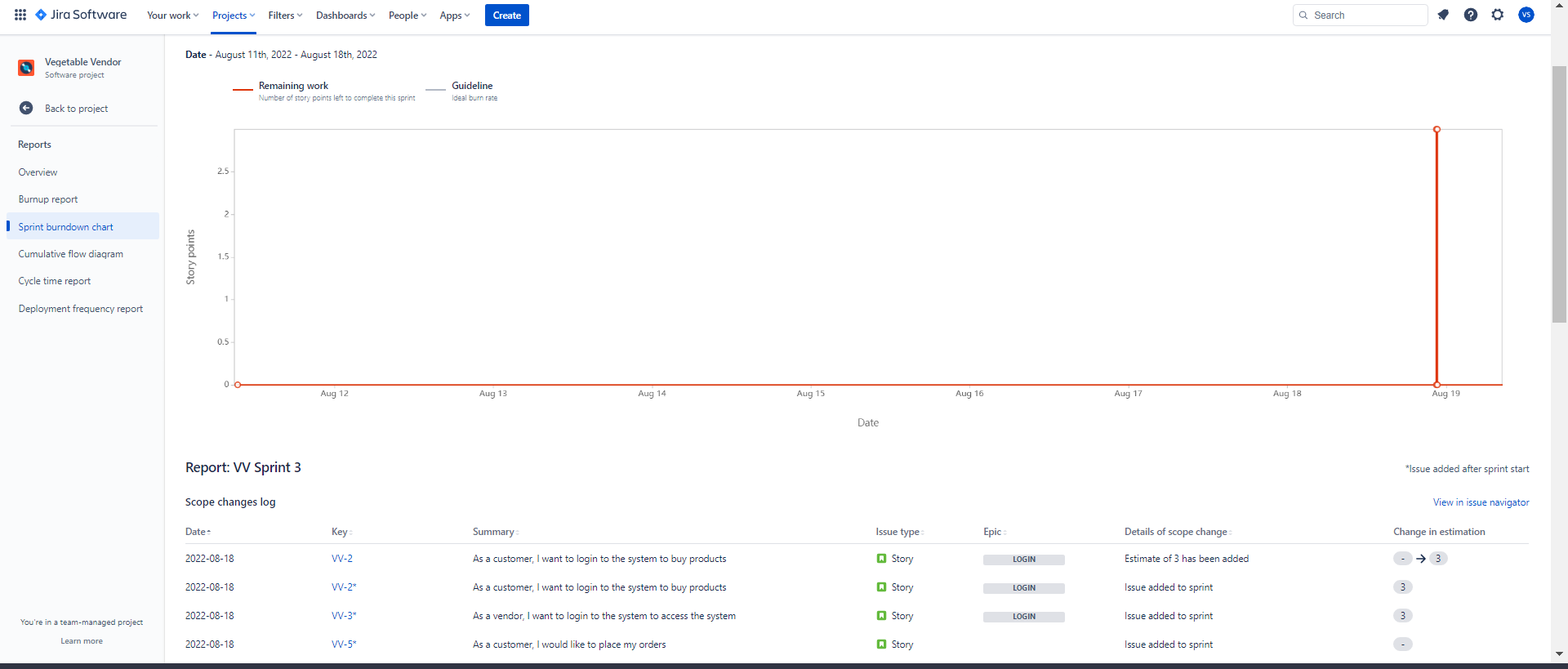
Maintained the Jira Tool for the project and GitHub.

PROJECT MANAGEMENT:

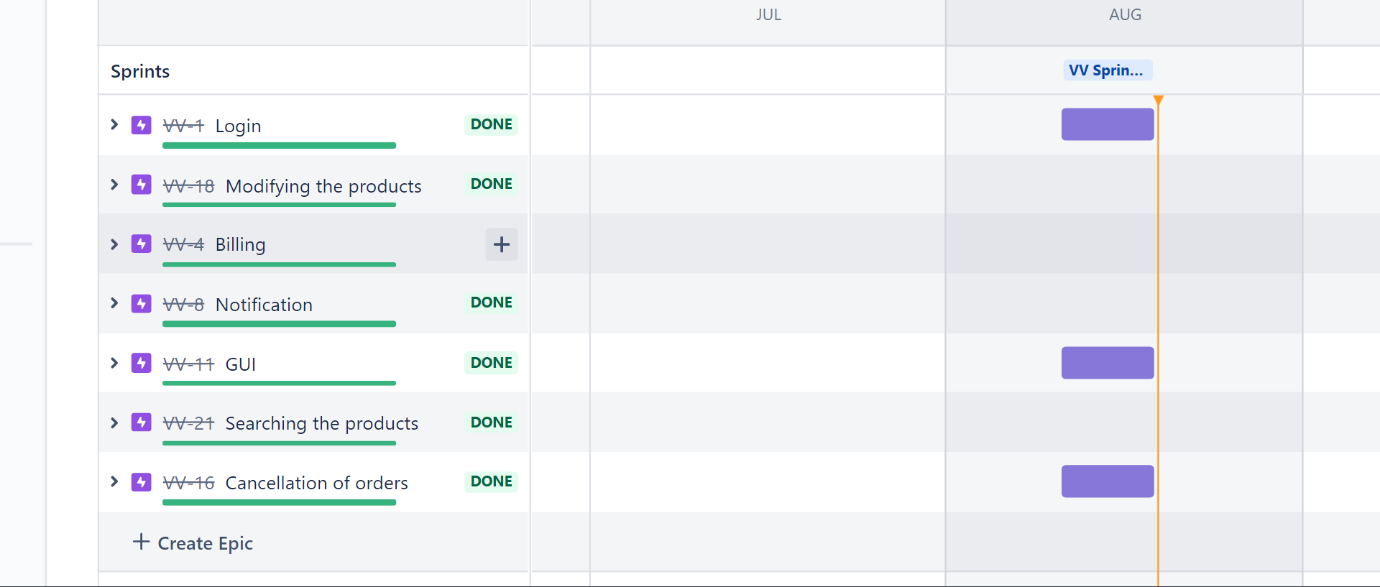
1. BURN-UP CHART



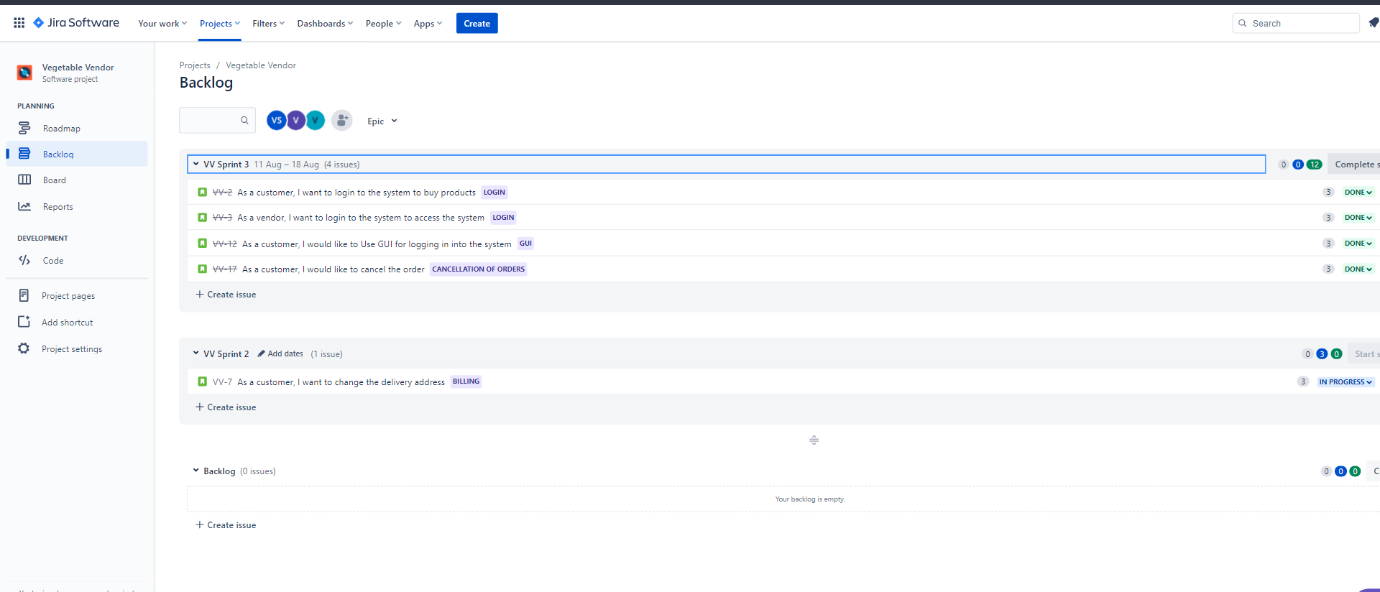
1. BURN – DOWN CHART



1. ROAD-MAP



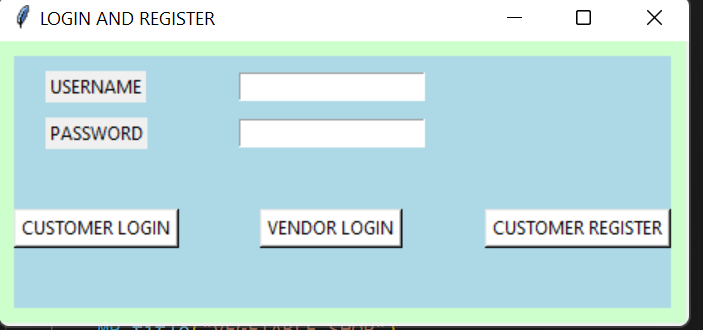
1. BACKLOG



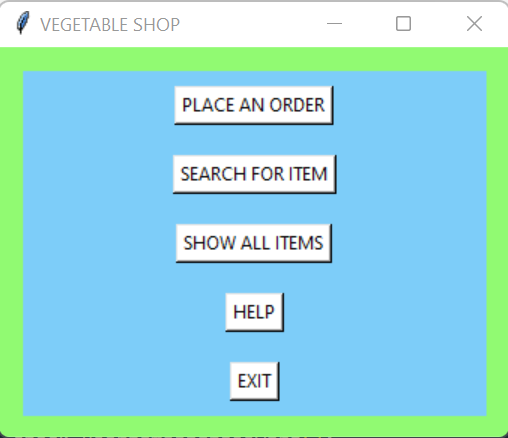
**PROJECT OUTCOME:**

SCREENSHOTS:

1. LOGIN WINDOW



1. OPTIONS PROVIDED TO THE CUSTOMER

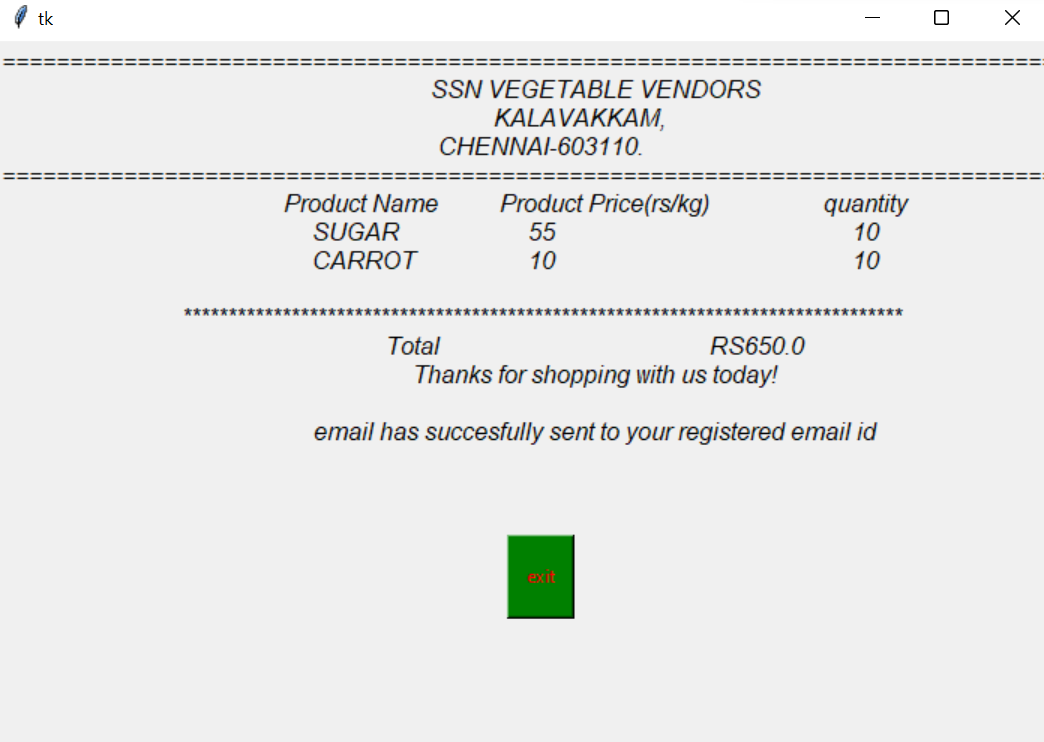


1. PLACING AN ORDER

Graphical user interface

Description automatically generated

1. BILL IS GENERATED



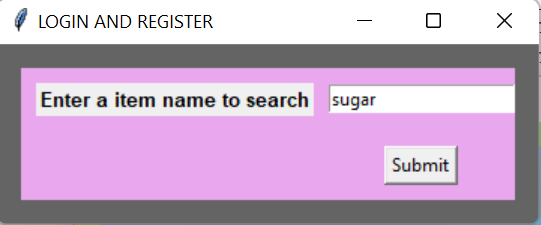
1. IF THE BILL GENERATED IS LESS THAN Rs.100

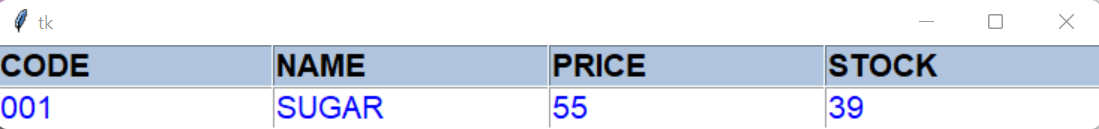
Graphical user interface

Description automatically generated

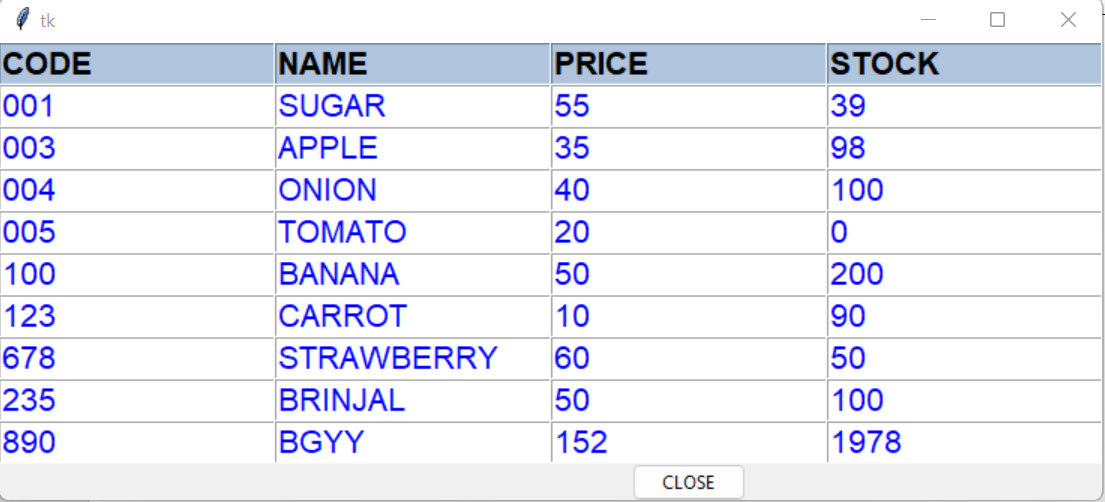


1. SEARCHING FOR AN ITEM

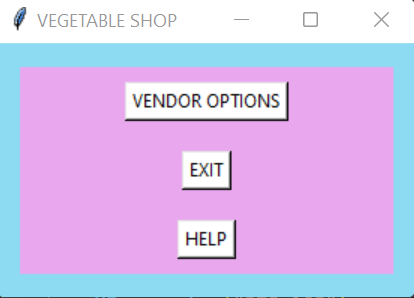


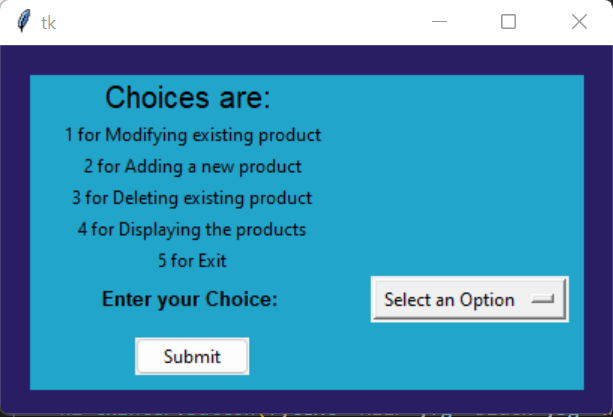


1. DISPLAYING ALL ITEMS

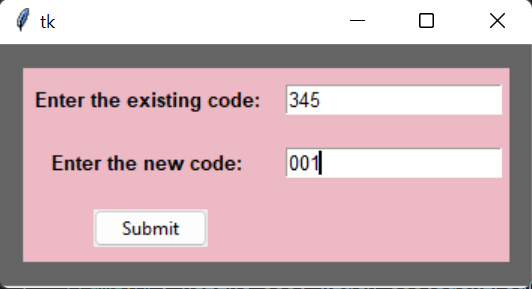


1. VENDOR OPTIONS

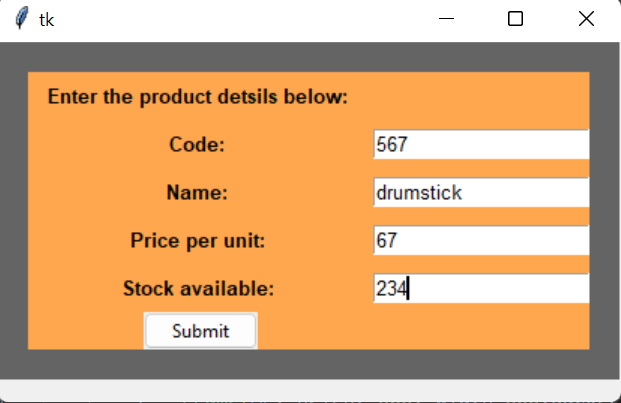




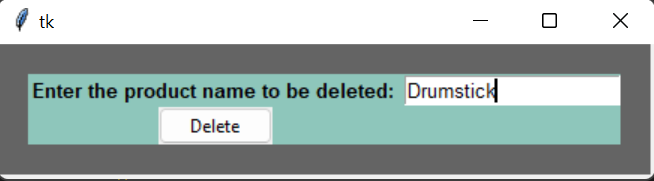
1. MODIFYING THE PRODUCT

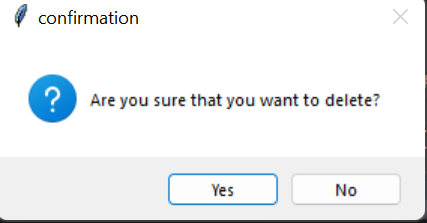


1. ADDING A NEW PRODUCT

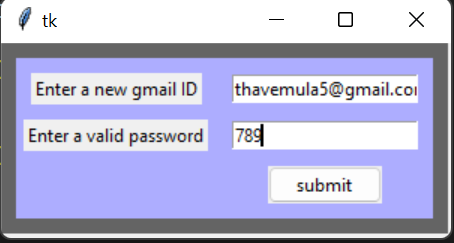


1. DELETING A PRODUCT





1. CUSTOMER REGISTER



**FEEDBACK FROM THE CLIENT:**

§The software seems to work fine, and it has reduced my work significantly. The options for vendor helps me to maintain the stocks.

§The customers are also assisted with this software as ordering of vegetables become easier and the bill is also generated in a transparent way.

§While ordering, the customer must give the products as a list only which is a setback.

CODE SNIPPETS:

1. Searching for a product

This option is provided to the customer where the customer has to enter the name of the product to be searched and the details of the product will be displayed. All the products are stored in a separate file, from which the name is searched.

def search(s):

    detail=[]

    print(s.get())

    with open("product.txt","r+") as fh:

            n=len(fh.readlines())

    with open("product.txt","r+") as fh:

        details={}

        for i in range(n):

            prod=(fh.readline()).split()

            key=prod[1]

            prod.remove(prod[1])

            details[key]=prod

    for i in details:

        print(i)

        if i.upper()==s.get().upper():

            sp=[]

            detail.insert(0,["CODE","NAME","PRICE","STOCK"])

            sp.append(details[i][0])

            sp.append(i)

            sp.append(details[i][1])

            sp.append(details[i][2])

            detail.append(sp)

    print(detail)

    total\_rows = len(detail)

    total\_columns = len(detail[0])

    root = Tk()

    t = Table(total\_rows,total\_columns,detail,root)

    root.mainloop()

1. New registration

This option is provided to the customer, if the customer doesn’t have an existing account. The customer is requested to fill in the details (Such as username, password, mobile no. and address. If the entered input is not in the correct format, then the respected error message pops up. If all the details entered are correct, then it takes the customer to the customer page.

def register(gm,npw,ph,ad):

    with open("USERNAME.txt","r+") as f1:

        n=len(f1.readlines())

    with open("USERNAME.txt","r+") as f1:

        user=[]

        for i in range(n-1):

            user.append(f1.readline().split("\n")[0])

        user.append(f1.readline())

    f1=open("USERNAME.txt","a+")

    f2=open("PASSWORD.txt","a+")

    x=gm.get()

    y=npw.get()

    a=ph.get()

    b=ad.get()

    flag=False

    flag1=False

    flag2=False

    flag3=False

    x1=x[-len(y):-8]

    # Checking if the input entered by the user is proper ...

    if not(y.isalnum()):

        flag1=True

    else:

        flag1=False

    if x[-10:]=="@gmail.com":

        flag=True

    if len(a)==10 and a.isdigit():

        flag2=True

    if b and a and x and y:

        flag3=True

    if flag==False:

        impg=Tk()

        impg.geometry("300x130")

        impg.config(bg='#f7ef38')

        window2=tkinter.Label(impg,text="ENTER A PROPER GMAIL ID",font=(30)).grid(row=2,column=2,pady=40,padx=13)

        impg.after(3000,lambda:impg.destroy())

        impg.mainloop()

    elif flag1==True:

        impp=Tk()

        impp.geometry("600x130")

        impp.config(bg='#f7ef38')

        window2=tkinter.Label(impp,text="USE ONLY ALPHABETS OR NUMBERS FOR PASSWORD",font=(30)).grid(row=2,column=2,pady=40,padx=13)

        impp.after(3000,lambda:impp.destroy())

        impp.mainloop()

    elif flag2==False:

        impr=Tk()

        impr.geometry("350x130")

        impr.config(bg='#f7ef38')

        window2=tkinter.Label(impr,text="ENTER A VALID PHONE NO.",font=(30)).grid(row=2,column=2,pady=40,padx=13)

        impr.after(3000,lambda:impr.destroy())

        impr.mainloop()

    elif flag3==False:

        impf=Tk()

        impf.geometry("350x130")

        impf.config(bg='#f7ef38')

        window2=tkinter.Label(impf,text="ENTER A VALID INPUT",font=(30)).grid(row=2,column=2,pady=40,padx=13)

        impf.after(3000,lambda:impf.destroy())

        impf.mainloop()

    if x in user:

        win1=Tk()

        win1.geometry("350x130")

        win1.config(bg='#f7ef38')

        window2=tkinter.Label(win1,text="ACCOUNT ALREADY EXISTS",font=(30)).grid(row=2,column=2,pady=40,padx=13)

        win1.after(3000,lambda:win1.destroy())

        win1.mainloop()

    elif flag==True and flag1==False and flag2==True and flag3==True:

        f1.write("\n")

        f1.writelines(str(x)+" "+str(a)+" "+str(b))

        print("hi")

        f2.write("\n")

        f2.writelines([y])

        f1.close()

        f2.close()

        MP=tkinter.Tk()

        MP.geometry("340x260")

        MP.state("zoomed")

        MP.title("VEGETABLE SHOP")

        f = Frame(MP, height=600, width=600,background='#7dcdf9', highlightbackground='#91fa72',highlightthickness=40)

        f.grid(row=0,column=0)

        sc=tkinter.Button(f,text="SEARCH FOR ITEM",fg="Black",bg="White",command=searchitem,font=('Arial',15)).grid(row=4,column=2,padx=500,pady=43)

        si=tkinter.Button(f,text="SHOW ALL ITEMS",fg="Black",bg="White",command=cpd,font=('Arial',15)).grid(row=6,column=2,pady=35)

        op=tkinter.Button(f1,text="PLACE AN ORDER",fg="Black",bg="White",command=lambda:[PO()],font=('Arial',15)).grid(row=2,column=2,pady=35)

        ext=tkinter.Button(f,text="EXIT",fg="Black",bg="White",command=MP.destroy,font=('Arial',15)).grid(row=10,column=2,pady=35)

        hl=tkinter.Button(f,text="HELP",fg="Black",bg="White",command=help\_,font=('Arial',15)).grid(row=8,column=2,pady=35)

        MP.mainloop()

1. Sending a notification to the vendor

The customer’s order will be placed only if the condition satisfies i.e., the total price amount must be above Rs.100.

Once the order is placed, the bill is sent to the customer’s Gmail for their reference.

def bill(gmid,order):

 print(gmid)

 print(order)

 d=[i for i in order.keys()]

 e=[i for i in order.values()]

 f=open("Product.txt","r")

 g=len(f.readlines())

 f.close()

 f=open("Product.txt","r")

 l=[]

 for i in range(g):

   h=(f.readline()).split()

   l.append(h)

 t=0

 k=[]

 for i in d:

    for j in l:

        if i==j[1]:

         k.append(j[2])

         a=d.index(i)

         t=t+(float(e[a])\*float(j[2]))

 print(t)

 u=[]

 for i in range(len(d)) :

# create a print statement for each item

         m=[d[i],k[i],e[i]]

         u.append(m)

 j=('=' \* 80+'\n\t{}'.format("SSN VEGETABLE VENDORS")+'\n\t{}\t'.format(" KALAVAKKAM,")+'\n{}'.format("CHENNAI-603110.\n")+('=' \* 80)+'\n\tProduct Name\tProduct Price(rs/kg)\t\tquantity\n')

 h=('\*' \* 60+'\n\t\t\t{}'.format("SSN VEGETABLE VENDORS")+'\n\t\t\t{}\t'.format("    KALAVAKKAM,")+'\n\t\t\t{}'.format("  CHENNAI-603110.\n")+('=' \* 45)+'\n\tProduct Name\tProduct Price(rs/kg)\tquantity\n')

 m=""

 c=""

 for i in u:

   m=m+'\t{}\t\t\t\t{}\t\t\t\t{}'.format(i[0],i[1],i[2])+"\n"

   c=c+'\t{}\t\t{}\t\t\t{}'.format(i[0],i[1],i[2])+"\n"

 d=j+c+"\n"+'\*' \* 80+'\n\tTotal\t\t\tRS{}'.format(t)+'\n\t{}\n\n\n'.format('Thanks for shopping with us today!'+"\n\n\t{}".format("email has succesfully sent to your registered email id"))

 l=h+m+"\n"+'\*' \* 60+'\n\tTotal\t\t\tRS{}'.format(t)+'\n\t{}\n'.format('Thanks for shopping with us today!')

# creates SMTP session

 s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security

 s.starttls()

#for i in range len(y)

 x=gmid[0:-10]

# Authentication

 s.login("vvvvvvicky2708@gmail.com", "ysavauxjzeojejlp")

 sub="CONFORAMTION OF YOUR VEGETABLE ORDER"

 body=("HELLO!!!,"+x+"\n\t{}".format("your order has sucessfully placed and it will be delivered soon\n")+"invoice for your order\n\n\n"+l+"\n\n\n\t{}".format("have a nice day :)"))

# message to be sent

 message = "Subject:{}\n\n{}".format(sub,body)

# sending the mail

 s.sendmail("vvvvvvicky2708@gmail.com",gmid, message)

# terminating the session

 s.quit()

 window=tkinter.Tk()

 window.geometry("700x1000")

 l=tkinter.Label(window,text=d,font=("arial",12,"italic")).grid(column=0,row=0)

 end=time.time()

 print("time taken to compile",end-start)

 bt=tkinter.Button(window,text="exit",command=window.destroy,fg="red",bg="green",height=3,width=5).grid(column=0,row=1)

 window.mainloop()

def cpd():

    with open("Product.txt","r+") as fh:

        n=len(fh.readlines())

    with open("Product.txt","r+") as fh:

        details=[]

        for i in range(n):

            prod=(fh.readline()).split()

            details.append(prod)

    details.insert(0,["CODE","NAME","PRICE","STOCK"])

    total\_rows = len(details)

    total\_columns = len(details[0])

    root = Tk()

    t = Table(total\_rows,total\_columns,details,root)

    but =Button(root,text = 'CLOSE', command = lambda:[root.destroy()]).grid(row=total\_rows+1,column=2)

    root.mainloop()

1. Placing an order

This option is provided to both the customer and vendor.

This option is provided to the vendor because in case a customer doesn’t have access to online facilities, they could approach the vendor for their order and the vendor will place the order on their behalf.

A dialogue box appears with a list of products alphabetically along with their price and availability in stock, the customer would just have to choose desired products and of how much quantity.

def PO():

    windown= Toplevel(window)

    windown.title("Vendor shop")

    with open("Product.txt","r+") as fh:

        lp=fh.readlines()

    n=0

    lon1=[]

    lop1=[]

    los1=[]

    d111={}

    for i in lp:

        p=i.split()

        lon1.append(p[1])

        lop1.append(p[2])

        los1.append(p[3])

        d111[p[1]]=[p[2],p[3]]

        n=n+1

    lop,lon,los=[],[],[]

    for i in range(0,len(lon1)):

        for j in range(0,len(lon1)-i-1):

            if lon1[j]>lon1[j+1]:

                lon1[j],lon1[j+1]=lon1[j+1],lon1[j]

    for i in lon1:

        lon.append(i)

        los.append(d111[i][1])

        lop.append(d111[i][0])

    sb=[]

    tln=Label(windown,text="PRODUCT").grid(row=0,column=0,padx=5,pady=5)

    tlp=Label(windown,text="PRICE PER KG").grid(row=0,column=1,padx=5,pady=5)

    tls=Label(windown,text="STOCK").grid(row=0,column=2,padx=5,pady=5)

    tlq=Label(windown,text="QUANTITY").grid(row=0,column=3,padx=5,pady=5)

    for i in range(1,1+n):

      if int(los[i-1])>0:

        dummy=IntVar()

        labeln=Label(windown,text=lon[i-1]).grid(row=i,column=0,padx=5,pady=5)

        labelp=Label(windown,text=lop[i-1]).grid(row=i,column=1,padx=5,pady=5)

        labels=Label(windown,text=los[i-1]).grid(row=i,column=2,padx=5,pady=5)

        sbx=Spinbox(windown,from\_=0,to\_=50,width=3,textvariable=dummy).grid(row=i,column=3,padx=5,pady=5)

        sb.append([lon[i-1],dummy])

    button=Button(windown,text="Order",command=lambda:[passtobill(sb)]).grid(row=i+2,column=0,padx=5,pady=5)

    butclose=Button(windown,text="Exit",command=lambda:[window.destroy()]).grid(row=i+2,column=1,padx=5,pady=5)

def passtobill(sb):

    global diclop

    username2=username.get()

    for i in range(len(sb)):

        if sb[i][1].get()>0:

            diclop[sb[i][0]]=sb[i][1].get()

    bill(username2,diclop)

CONCLUSION AND FUTURE SCOPE

The project is successful in helping the vendor with maintaining his products and also in placing an order by the customer. And vendor on behalf of the customer.

The basic idea of the software is a positive output. Successful handling of the customer is also a positive outlook.

The process of creating a front end and back end was a source outcome for each of the team members

There seem to be many future scopes for the project as it avails the prevalent public.

One of the further developments of the project seems to be in incorporating the cancellation of orders. We are currently working on it.

Google maps can be integrated into the project to facilitate the delivery of orders.

Customer interaction with the delivery boy can be incorporated into the system which will come in handy to both the customer as well as the delivery person.