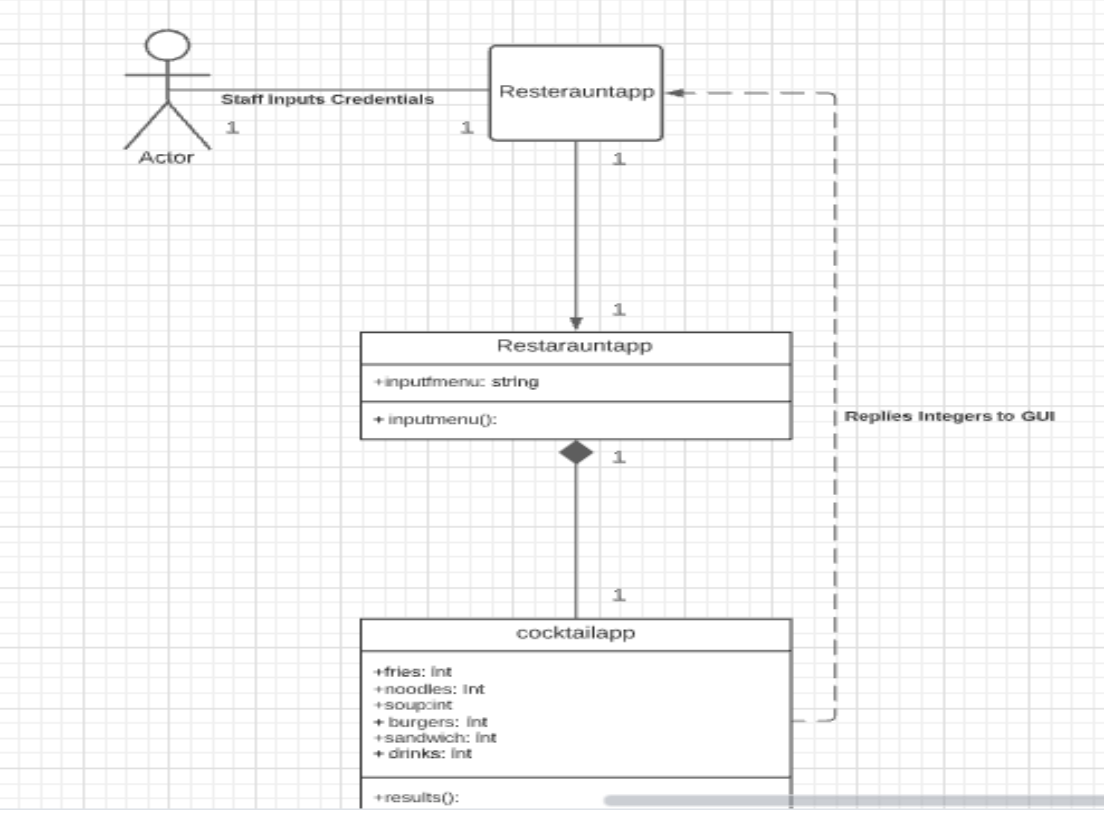
Sarah Gaines

CYBR7100

Final Report

Assignment 2



2. The first secure aspect of this application is the confidentiality aspect. This application is for the staff of the restaurant, so hiding the information a prominent factor. The username and password are a requirement because only the manager will have the credentials. This will allow for the staff to not give away any credentials to malicious users. The manager will receive the login from Headquarters than they will be the only people who have access to the system.

The second security principle in the application is a fail-safe default. If a user types in anything unrecognizable or unknown the application will display an "Username is not correct" message. The failsafe default does not allow users explicit access to the application. If the user does not know the username and password, then the system will not give the user access to the application. This is essential when hackers try to gain unauthorized access to the system because it will keep them away from implementing viruses, worms, trojan horses, etc.

The last security principle used is least privileges. Least privileges are defined as properly distributing privileges to users based on their position. The user can only input the data and then

obtain the total. They do not get to change the code or explore among the application. The privilege the users have are minor. However, the developer will be the only one who has access to the code of the application. This allows the developer to rewrite the code for the applicait6on and even add on different properties and options. The goal for this security principle is to keep everything at an order. No application developer will ever allow their users to rewrite their codes because at that point what is the purpose of the application. These three security principles will allow for the application to run fast and stay secure while the staff has a quick restaurant management system.

Access Control

3. All application users must use access control because it has a control. The access control of this application starts from the beginning with the Restaurant diagram above. Once the user gains access to the system the discretionary access control process begins due to the results being based on the identity of the user. This means if the user gains access using the credentials then they can use the application. If the user does not have access, then the user will not be able to gain access in the management application., In contrast to the developer who has access to everything. There are six concepts that are necessary to model access controls in this application: subject, principle, resource, privilege, safeguard, and policy. The subject is the staff member who needs to use the application to input a customer's order. The principle of this application is the total cost. The staff member needs to get the total cost of the order, to tell the customers what they owe the restaurant. If the staff member tries to input any other variables into the quality of a menu then the application will not compute a total cost. The resources are the menu items. Permission is granted to all users by the management when the managers sign in to the system for the staff. Permission is the same term as privilege. The safeguard in this application is the fail-safe device. The application policy is a notion explaining all the above concepts.

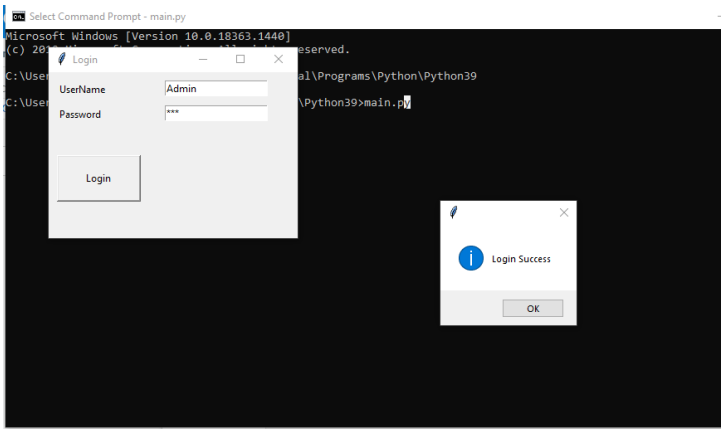
4. The trust model for this application consists of a reputation-based system. The reputation-based system consists of gaining reputation as an expectation about an individual's behavior based on that user's reaction. This application will have a review and ratings session at the end of the users search. The review will give the users a chance to give suggests and write about their experience using the application. The ratings will be available to rate the application on a five-star grading scale. If the user does not want to do the review, the applicant will be able to skip this option and return to the beginning search screen. If application were published the user's feedback would be available for all users to see from the Apple App store or Google Play store. The reviews being open for the public would allow for other customers to buy the application or perhaps steer them away from the application. The reviews will be beneficial for the developer because they give insight on how the application users truly feel toward the application. The developer will be able to see if there should be any changes or updates to the application that can increase popularity. Even though these reviews will be public, if the application is running well and easily handled than this should be a five-star application.

Assignment 3

1. The first part of the app shows Authentication and Password Management, this will help secure the program by steering away from anyone being able to log in. The login section should only be for employees so after the login window pops up the restaurant manager screen comes in to play. This is to keep hackers out and only give the right people authority. I felt as though this was the best secure principle to have for this application because managers would want their computer systems to be secure rather than be available for everyone. The username and password will verify that the application uses a single authentication mechanism that is known to be secure. If this were a real application, I would only give a username and password to the managers which is why the fail-safe is there for any other users that want to gain unauthorized access. The servers would be logged in for the whole day and if they accidently got logged out, they would just go get a manager to sign them back in. This is also an indication of access control due to the managers only having access to the authentication words and numbers. The password is "123" right now, but it could easily be changed with simple coding. If this application were active for a restaurant, the password would be strong with a minimum of 12 characters. The more difficult and long the password is, the less risk of a brute force attack occurs.

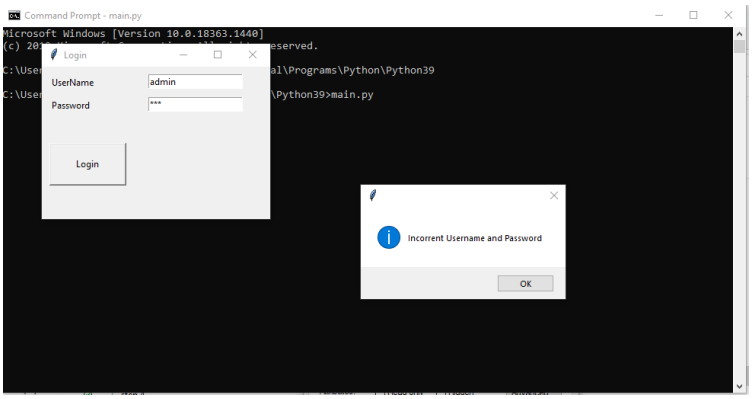
   The code for the Log in:

```
from tkinter import *
from tkinter import messagebox


def Ok():
    uname = e1.get()
    password = e2.get()

    if(uname == "" and password == "") :
        messagebox.showinfo("", "Blank Not allowed")


    elif(uname == "Admin" and password == "123"):

        messagebox.showinfo("","Login Success")
        root.destroy()

    else :
        messagebox.showinfo("","Incorrent Username and Password")


root = Tk()
root.title("Login")
root.geometry("300x200")
global e1
global e2

Label(root, text="UserName").place(x=10, y=10)
Label(root, text="Password").place(x=10, y=40)

e1 = Entry(root)
e1.place(x=140, y=10)

e2 = Entry(root)
e2.place(x=140, y=40)
e2.config(show="*")


Button(root, text="Login", command=Ok ,height = 3, width = 13).place(x=10, y=100
```



3.Input Validation is shown here in the beginning of the application code. The program inputs are an essential factor of the problems. If external data is not validated in this application, it will lead to a Syntax error. This input validation is to ensure that it contains the right type of information, the right amount of information, and the right structure of information. This security principle helps prevents improperly formed data from entering an information system. Input validation is one of the most important things you have in an application to make sure code security is available because input is always the one thing about your application you cannot directly control. If you can not control the application, then you can not trust it. The detection of the malicious user is difficult to discover because there are so many back door and hidden ways that people can infiltrate systems. This is where input validation come in because the applications will check and validate all input entered a system should occur when data is received from an external party, especially if the data is from untrusted sources. Incorrect input validation can lead to viruses, worms, man-in-the-middle, etc.

```
def Ok():
    uname = e1.get()
    password = e2.get()

    if(uname == "" and password == "") :
        messagebox.showinfo("", "Blank Not allowed")


    elif(uname == "Admin" and password == "123"):

        messagebox.showinfo("","Login Success")
        root.destroy()

    else :
        messagebox.showinfo("","Incorrent Username and Password")
```



Configuration Architectural Requirements shown through the button widget because it verifies the input and output requirements clearly by defining how to handle and process data based on fries, drinks, order numbers, and entrees. The API is reliable because the right information comes up and does not take the user somewhere else when pressing the buttons. All the options on this application help to Verify that coding is configured to produce the proper application. There is no misleading or misguiding, so the application is very secure and simple. Input and output are important because what is an application that do no output information? This requirement is needed in all applications so that they can be successful. The coding of the buttons took some time because everyone uses different programs and for me, I used Windows basis with python and tkinter and it is way different then just completing it on python.

Code for the above application:

```
from tkinter import *

from tkinter import messagebox



def Ok():

    uname = e1.get()

    password = e2.get()


    if(uname == "" and password == "") :

        messagebox.showinfo("", "Blank Not allowed")



    elif(uname == "Admin" and password == "123"):


        messagebox.showinfo("","Login Success")

        root.destroy()


    else :

        messagebox.showinfo("","Incorrent Username and Password")
```

```
root = Tk()
root.title("Login")
root.geometry("300x200")
global e1
global e2

Label(root, text="UserName").place(x=10, y=10)
Label(root, text="Password").place(x=10, y=40)

e1 = Entry(root)
e1.place(x=140, y=10)

e2 = Entry(root)
e2.place(x=140, y=40)
e2.config(show="*")



Button(root, text="Login", command=Ok ,height = 3, width = 13).place(x=10, y=100)

root.mainloop()

from tkinter import*
import random
import time
import datetime

root=Tk()
root.geometry("1600x8000")
root.title("Python Restaurant Management Software")

Tops=Frame(root, width=1600,relief=SUNKEN)
Tops.pack(side=TOP)

f1=Frame(root,width=800,height=700,relief=SUNKEN)
```

```
f1.pack(side=LEFT)

#=============================================================================
#                TIME
#=============================================================================
localtime=time.asctime(time.localtime(time.time()))

lblInfo=Label(Tops,font=('helvetica',40,'bold'),text="Python Restaurant Management Software ",fg="red",bd=10,anchor='w')
lblInfo.grid(row=0,column=0)

lblInfo=Label(Tops,font=('arial',20,'bold'),text=localtime,fg="blue",bd=10,anchor='w')
lblInfo.grid(row=1,column=0)
lblInfo=Label(Tops,font=('arial',10,'bold'),text="By The Codezine[https://thecodezine.com] ",fg="blue",bd=10,anchor='w')
lblInfo.grid(row=3,column=0)



def Ref():
    x=random.randint(10908,500876)
    randomRef=str(x)
    rand.set(randomRef)

    if (Fries.get()==""):
        CoFries=0
    else:
        CoFries=float(Fries.get())




    if (Noodles.get()==""):
        CoNoodles=0
    else:
        CoNoodles=float(Noodles.get())



    if (Soup.get()==""):
```

```
    CoSoup=0
else:
    CoSoup=float(Soup.get())



if (Burger.get()==""):
    CoBurger=0
else:
    CoBurger=float(Burger.get())


if (Sandwich.get()==""):
    CoSandwich=0
else:
    CoSandwich=float(Sandwich.get())


if (Drinks.get()==""):
    CoD=0
else:
    CoD=float(Drinks.get())


CostofFries =CoFries * 140
CostofDrinks=CoD * 65
CostofNoodles = CoNoodles* 90
CostofSoup = CoSoup * 140
CostBurger = CoBurger* 260
CostSandwich=CoSandwich * 300

CostofMeal= "Rs", str('%.2f' % (CostofFries+CostofDrinks+CostofNoodles+CostofSoup+CostBurger+CostSandwich))

PayTax=((CostofFries+CostofDrinks+CostofNoodles+CostofSoup+CostBurger+CostSandwich) * 0.2)

TotalCost=(CostofFries+CostofDrinks+CostofNoodles+CostofSoup+CostBurger+CostSandwich)
```

```
    Ser_Charge= ((CostofFries+CostofDrinks+CostofNoodles+CostofSoup+CostBurger+CostSandwich)/99)

    Service = "Rs", str ('%.2f' % Ser_Charge)

    OverAllCost ="Rs", str ('%.2f' % (PayTax+TotalCost+Ser_Charge))

    PaidTax= "Rs", str ('%.2f' % PayTax)

    Service_Charge.set(Service)
    Cost.set(CostofMeal)
    Tax.set(PaidTax)
    SubTotal.set(CostofMeal)
    Total.set(OverAllCost)

def qExit():
    root.destroy()

def Reset():
    rand.set("")
    Fries.set("")
    Noodles.set("")
    Soup.set("")
    SubTotal.set("")
    Total.set("")
    Service_Charge.set("")
    Drinks.set("")
    Tax.set("")
    Cost.set("")
    Burger.set("")
    Sandwich.set("")

#===================================Restaraunt Info
1=========================================================

rand = StringVar()
Fries=StringVar()
Noodles=StringVar()
```

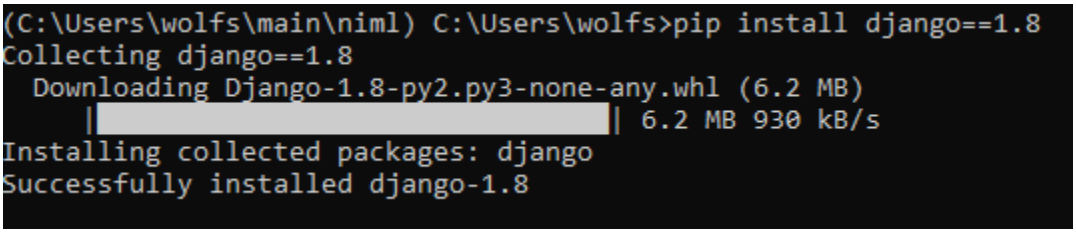
```
Soup=StringVar()
SubTotal=StringVar()
Total=StringVar()
Service_Charge=StringVar()
Drinks=StringVar()
Tax=StringVar()
Cost=StringVar()
Burger=StringVar()
Sandwich=StringVar()

lblReference= Label(f1, font=('arial', 16, 'bold'),text="Order No",bd=16,anchor="w")
lblReference.grid(row=0, column=0)
txtReference=Entry(f1, font=('arial',16,'bold'),textvariable=rand,bd=10,insertwidth=4,bg="red",justify='right')
txtReference.grid(row=0,column=1)

lblFries= Label(f1, font=('arial', 16, 'bold'),text="Fries",bd=16,anchor="w")
lblFries.grid(row=1, column=0)
txtFries=Entry(f1, font=('arial',16,'bold'),textvariable=Fries,bd=10,insertwidth=4,bg="red",justify='right')
txtFries.grid(row=1,column=1)


lblNoodles= Label(f1, font=('arial', 16, 'bold'),text="Noodles",bd=16,anchor="w")
lblNoodles.grid(row=2, column=0)
txtNoodles=Entry(f1, font=('arial',16,'bold'),textvariable=Noodles,bd=10,insertwidth=4,bg="red",justify='right')
txtNoodles.grid(row=2,column=1)


lblSoup= Label(f1, font=('arial', 16, 'bold'),text="Soup",bd=16,anchor="w")
lblSoup.grid(row=3, column=0)
txtSoup=Entry(f1, font=('arial',16,'bold'),textvariable=Soup,bd=10,insertwidth=4,bg="red",justify='right')
txtSoup.grid(row=3,column=1)

lblBurger= Label(f1, font=('arial', 16, 'bold'),text="Burger",bd=16,anchor="w")
lblBurger.grid(row=4, column=0)
txtBurger=Entry(f1, font=('arial',16,'bold'),textvariable=Burger,bd=10,insertwidth=4,bg="red",justify='right')
txtBurger.grid(row=4,column=1)
```

```
lblSandwich= Label(f1, font=('arial', 16, 'bold'),text="Sandwich",bd=16,anchor="w")

lblSandwich.grid(row=5, column=0)

txtSandwich=Entry(f1, font=('arial',16,'bold'),textvariable=Sandwich,bd=10,insertwidth=4,bg="red",justify='right')

txtSandwich.grid(row=5,column=1)


#=============================================================================================================#

#                         RESTAURANT OTHER INFO                                    #

#=============================================================================================================#


lblDrinks= Label(f1, font=('arial', 16, 'bold'),text="Drinks",bd=16,anchor="w")

lblDrinks.grid(row=0, column=2)

txtDrinks=Entry(f1, font=('arial',16,'bold'),textvariable=Drinks,bd=10,insertwidth=4,bg="red",justify='right')

txtDrinks.grid(row=0,column=3)


lblCost= Label(f1, font=('arial', 16, 'bold'),text="Cost of Meal",bd=16,anchor="w")

lblCost.grid(row=1, column=2)

txtCost=Entry(f1, font=('arial',16,'bold'),textvariable=Cost,bd=10,insertwidth=4,bg="red",justify='right')

txtCost.grid(row=1,column=3)



lblService= Label(f1, font=('arial', 16, 'bold'),text="Service Charge",bd=16,anchor="w")

lblService.grid(row=2, column=2)

txtService=Entry(f1, font=('arial',16,'bold'),textvariable=Service_Charge,bd=10,insertwidth=4,bg="red",justify='right')

txtService.grid(row=2,column=3)



lblStateTax= Label(f1, font=('arial', 16, 'bold'),text="Tax",bd=16,anchor="w")

lblStateTax.grid(row=3, column=2)

txtStateTax=Entry(f1, font=('arial',16,'bold'),textvariable=Tax,bd=10,insertwidth=4,bg="red",justify='right')

txtStateTax.grid(row=3,column=3)


lblSubTotal= Label(f1, font=('arial', 16, 'bold'),text="Sub Total",bd=16,anchor="w")

lblSubTotal.grid(row=4, column=2)

txtSubTotal=Entry(f1, font=('arial',16,'bold'),textvariable=SubTotal,bd=10,insertwidth=4,bg="red",justify='right')
```

```
txtSubTotal.grid(row=4,column=3)


lblTotalCost= Label(f1, font=('arial', 16, 'bold'),text="Total Cost",bd=16,anchor="w")

lblTotalCost.grid(row=5, column=2)

txtTotalCost=Entry(f1, font=('arial',16,'bold'),textvariable=Total,bd=10,insertwidth=4,bg="red",justify='right')

txtTotalCost.grid(row=5,column=3)


#=========================================Buttons=========================================================
==================================

btnTotal=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=10,text="Total",bg="red",command=Ref).grid(row=7,colu
mn=1)


btnReset=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=10,text="Reset",bg="red",command=Reset).grid(row=7,col
umn=2)


btnExit=Button(f1,padx=16,pady=8,bd=16,fg="black",font=('arial',16,'bold'),width=10,text="Exit",bg="red",command=qExit).grid(row=7,colum
n=3)


def price():

    roo = Tk()

    roo.geometry("600x220+0+0")

    roo.title("Item Price List")

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="ITEM", fg="black", bd=5)

    lblinfo.grid(row=0, column=0)

    lblinfo = Label(roo, font=('aria', 15,'bold'), text="_____", fg="white", anchor=W)

    lblinfo.grid(row=0, column=2)

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="PRICE", fg="black", anchor=W)

    lblinfo.grid(row=0, column=3)

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Fries", fg="red", anchor=W)

    lblinfo.grid(row=1, column=0)

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="140", fg="red", anchor=W)

    lblinfo.grid(row=1, column=3)

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Noodles", fg="red", anchor=W)

    lblinfo.grid(row=2, column=0)

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="90", fg="red", anchor=W)

    lblinfo.grid(row=2, column=3)

    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Soup", fg="red", anchor=W)

    lblinfo.grid(row=3, column=0)
```

```
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="140", fg="red", anchor=W)
    lblinfo.grid(row=3, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Burger", fg="red", anchor=W)
    lblinfo.grid(row=4, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="260", fg="red", anchor=W)
    lblinfo.grid(row=4, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Sandwich", fg="red", anchor=W)
    lblinfo.grid(row=5, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="300", fg="red", anchor=W)
    lblinfo.grid(row=5, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Drinks", fg="red", anchor=W)
    lblinfo.grid(row=6, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="65", fg="red", anchor=W)
    lblinfo.grid(row=6, column=3)


    roo.mainloop()


btnprice=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="PRICE", bg="red",command=price)
btnprice.grid(row=7, column=0)


root.mainloop()
```

3.For the secure application part: I decided to use Safety because I felt it was mor simple for my type of application. I had to break my explanation up in to pieces because I wanted you to see the process of how it all went. Safety is vulnerability database updated once a month, and the vulnerability database is only licensed for non-commercial use. This means that if you want more timely updates or to use it in a commercial project you'll have to pay because the one I used is the free version. I will mention below or on top of each screenshot the process of running this safety check.



First step I had to install Django, and I used Django before, but not on the Anaconda database, so I had to consistently use “pip” before giving the machine any command to do.

Then I had to install safety. When I installed safety, it took about two minutes to fully go through because of all the packages, but those will go away once it is updated,