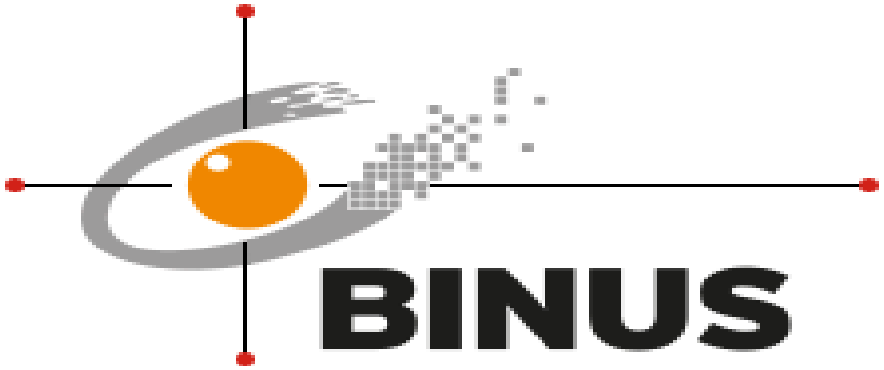
Odd Semester (2023)



BINUS
UNIVERSITY
INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	**Lucianto**	**William**	**2301890390**

Course Code	**: COMP6502**	**Course Name**	**: Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	**:Ida Bagus Kerthyayana**
Major	**: CS**		
Title of Assignment (if any)	: Inventory management		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 14-01-20**	**Submission Date**	**: 14-01-20**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. William Lucianto Santoso

"inventory management"

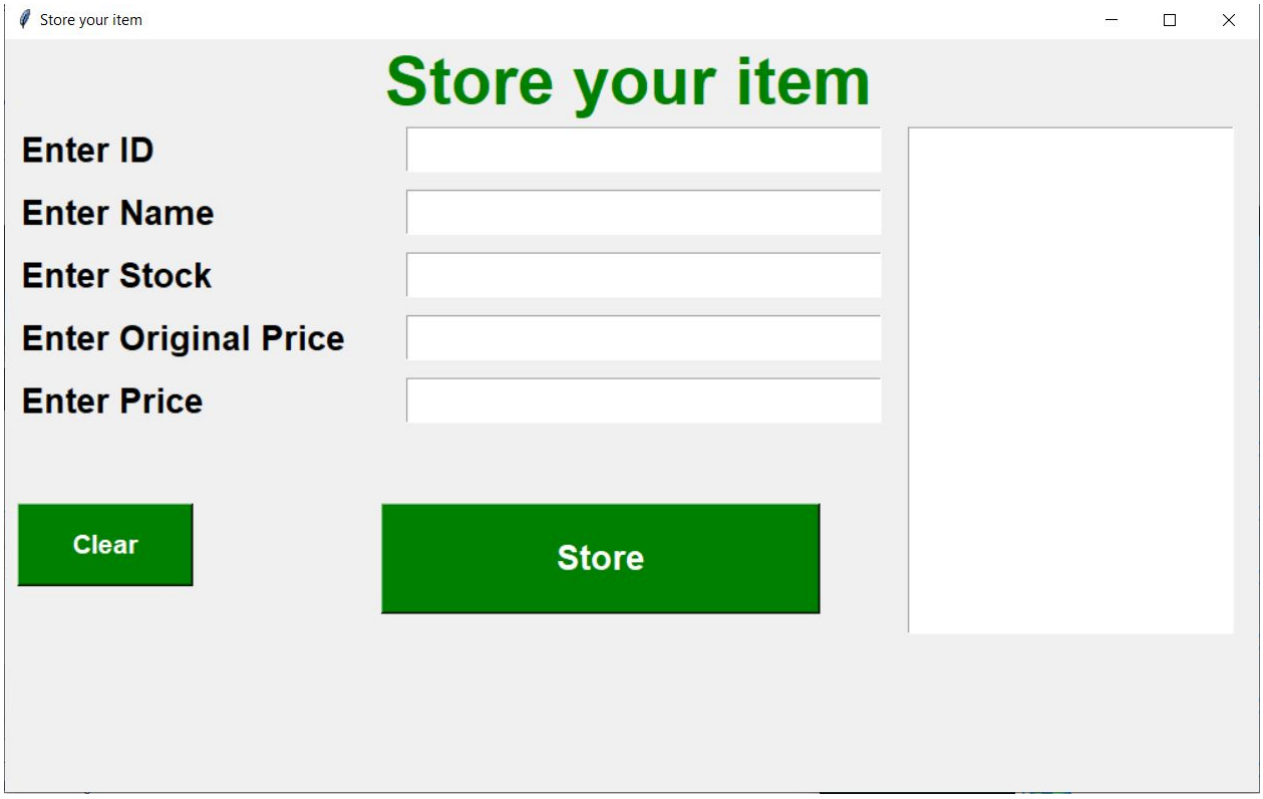
Name :William Lucianto Santoso

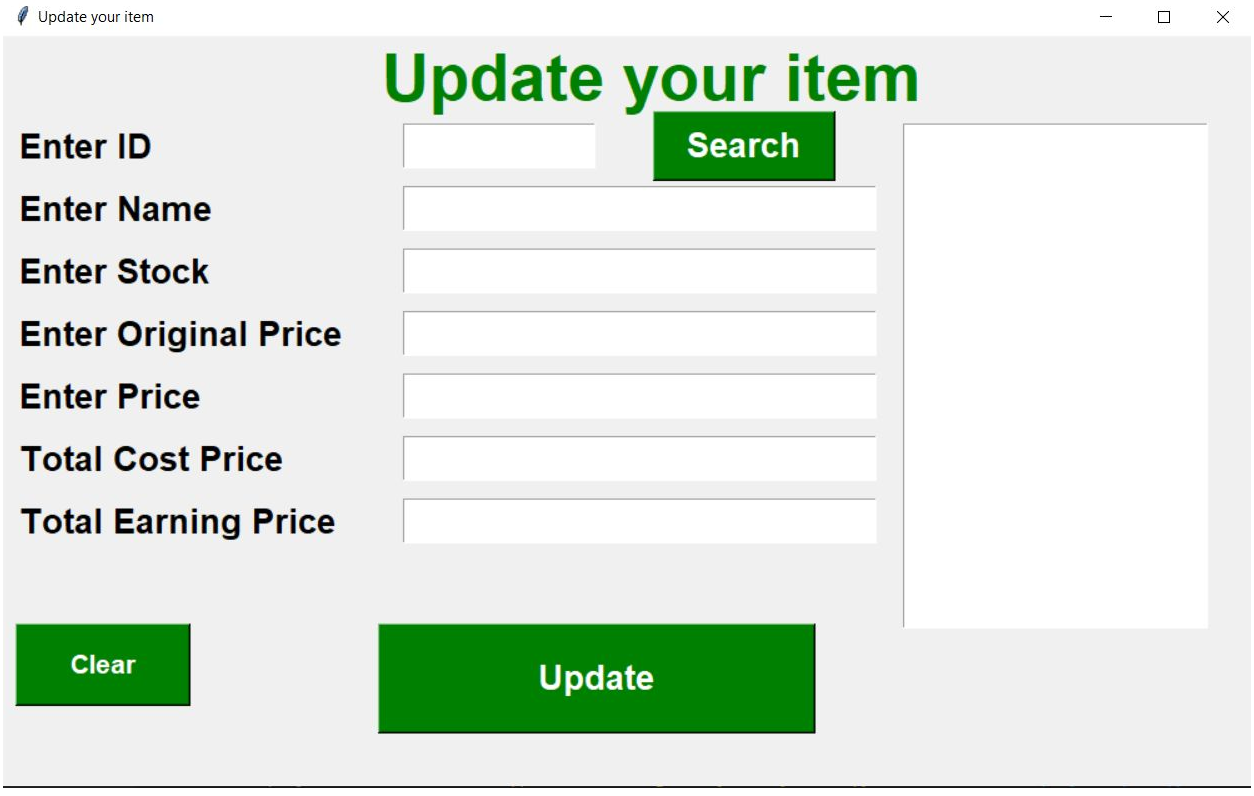
ID : 2301890390

I. Description

The function of this program:

The purpose of this program is to store and update item into database using sqlite3 so the user can easily track their items and they can search the item and modify it with this program. This program use Graphic User Interface by tkinter for a better UI & UX

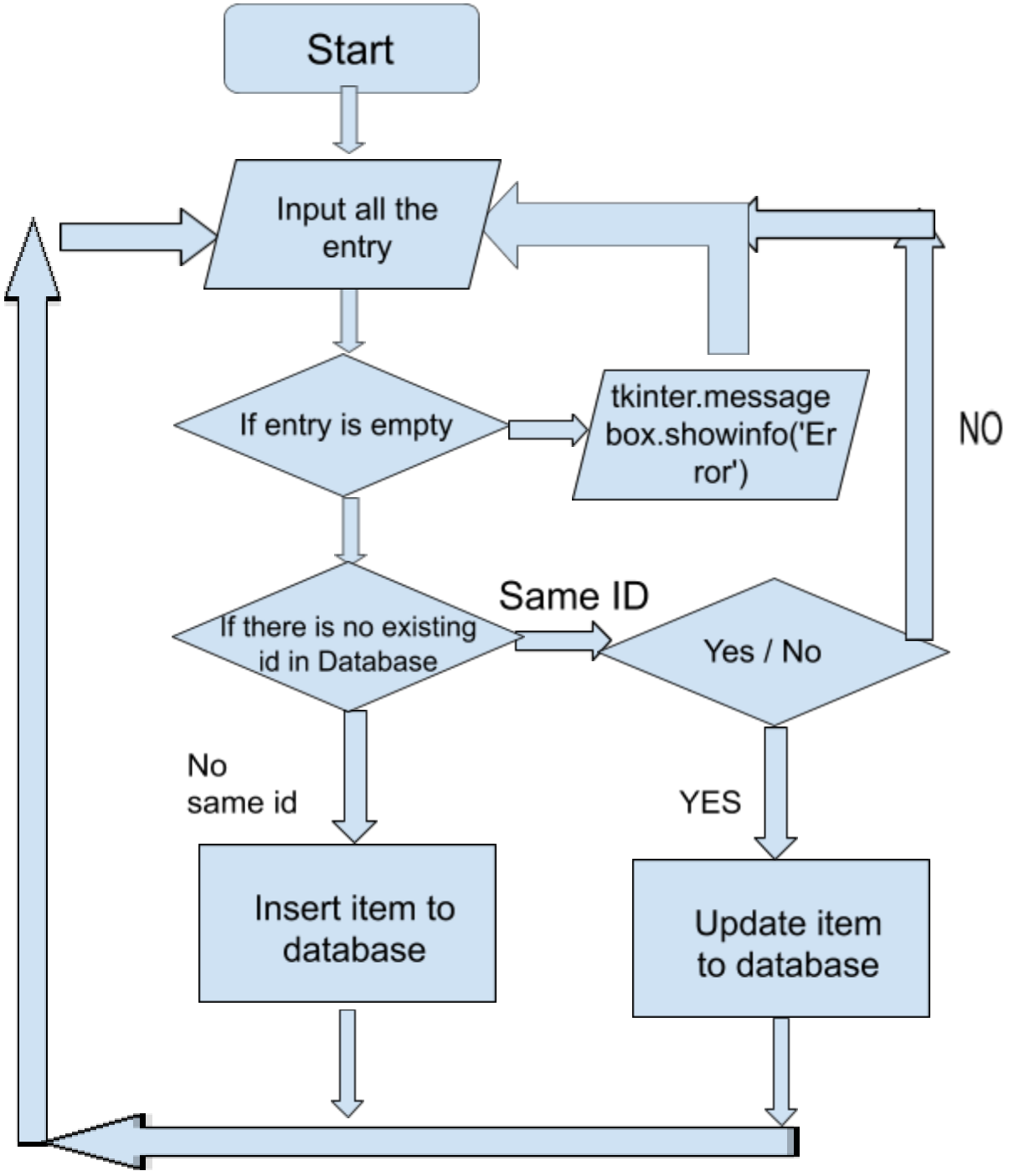

Store your item
Store your item
Enter ID
Enter Name
Enter Stock
Enter Original Price
Enter Price
Clear
Store


Update your item
Update your item
Enter ID
Search
Enter Name
Enter Stock
Enter Original Price
Enter Price
Total Cost Price
Total Earning Price
Clear
Update

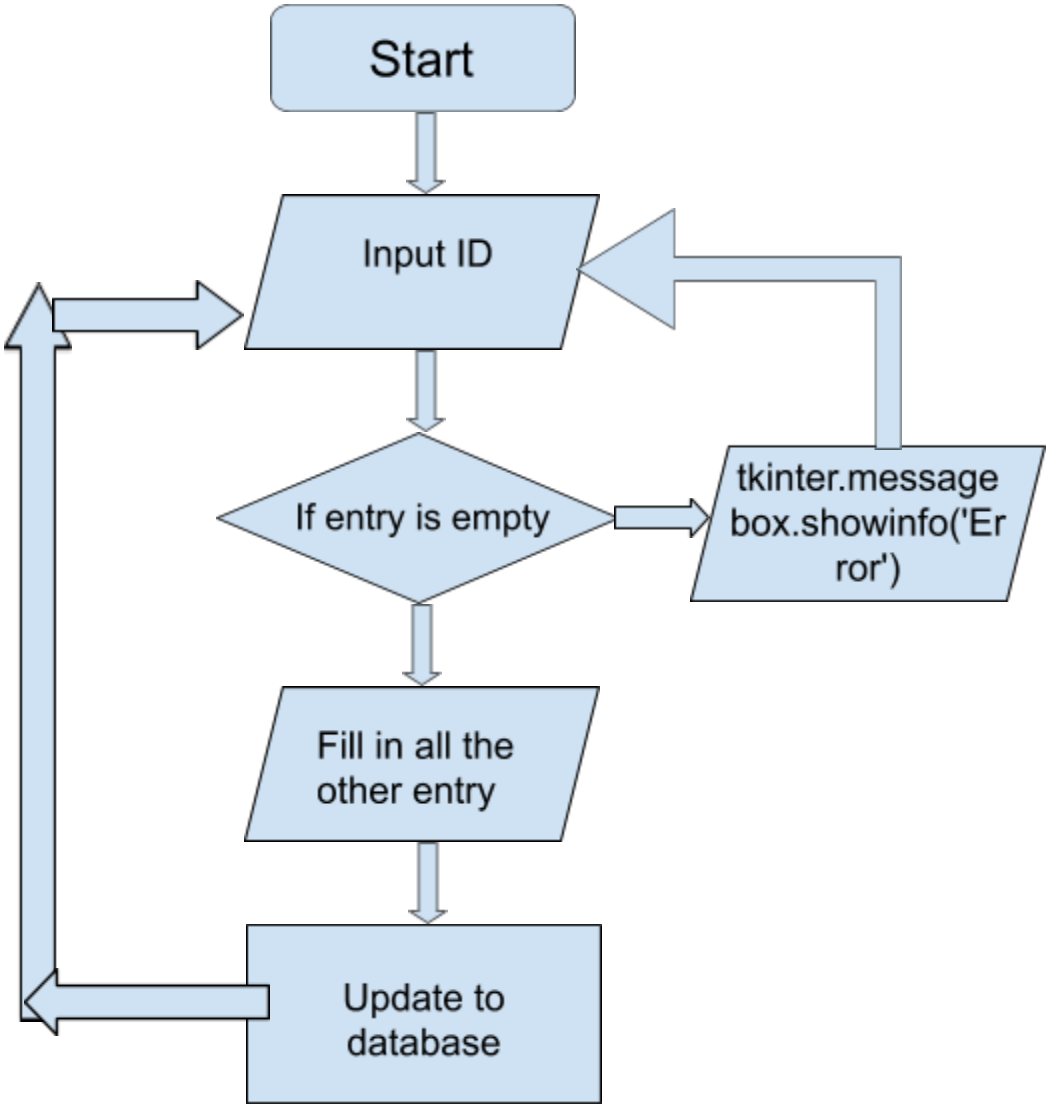
II.a. Design/Plan

Project's Flow Chart

#Database.py



#Update.py


Start
Input ID
If entry is empty
tkinter.message box.showinfo('Error')
Fill in all the other entry
Update to database

#Cashier.py

- Take input ID and Search to show the item name and price
- input Quantity and discount and then press the add to cart button
- after you put all the item you want press the total button
- the app will calculate the total price and show it to the user
- input the customer total money
- the app will calculate the change money
- use update the database to automatically reduce the stock in the database and calculate the totalCost, totalEarning, assumedProfit and update it into the database

I.b. Explanation of Each Function Inside the Class

- **delete_entry (self, *args, **kwargs) :**
    - delete all Entry to make all the Entries empty
- **store_items (self, *args, **kwargs) :**
    - use get() to take the value from the “clothe” table
    - if there is an empty entry, show error messages by using if and else
    - use “SELECT id FROM clothe” and loop it to see if there are existing id in the Database
    - if there are same id in the database, show message with yes or no answer if user want to rewrite the data or not
    - if yes = update the same id in the database to the one we enter in the entry
    - if no = delete the id entry
    - if there are no data with the same id in the database, use sql command “INSERT INTO clothe” to input all the data into the table
    - use c.execute() to perform the sql command
    - commit change by using conn.commit()
    - add message box to show that the storing process is completed
    - input a “Added {} into your database” ({} = item you want to store) into your entry log
- **search (self, *args, **kwargs) :**
    - use try and except to make sure there are no error by putting error message box in the except
    - use sql command “SELECT * FROM clothe WHERE id=?

- use c.execute to use the sql command
- since there are 7 key in the table, we make a 7 new variable with loop where each variable have the values for each keys
- commit change by using conn.commit()
- use delete to remove all the entries if there are any, then insert the new variable to get the values in the entry

- **update (self, *args, **kwargs):**
    - use of try and except to make sure there are no errors
    - make a new variable to keep the updated values
    - put the updated values to the new variable by using get()
    - use sql command = "UPDATE clothe SET name=?, id=?, and so on
    - c.execute to use the sql command
    - conn.commit() to commit the change in table
    - insert message in the entry log that the update is successful
    - insert message showinfo to make a pop up to show success
- add_to_cart(self, *args, **kwargs):
    - get the quantity and discount entry value
    - get total price by multiply price and quantity
    - check if the quantity is not more than the stock
    - append the product name, quantity, price, id to the list of each one
    - use loop to create the label in the right frame
- total(self, *args, **kwargs):
    - get the sum of product prices
    - create the label for total price, entry for the given money from customer and the button to calculate the change
- cash_change(self, *args, **kwargs):
    - create the button to update the database
    - get the total money by customer from the entry

- calculate the change
- check if the given money is not lower than the total price
- show the change using label

- reduce_stock(self, *args, **kwargs):
  - use “SELECT * FROM clothe WHERE id=?” with the product_id list using loop
  - get new stock by stock - product_quantity list
  - update it to the database with the new stock
  - destroy all the entry and label with only remaining of the top one
  - clear all the list

III.a. Lessons that Have Been Learned

```
from tkinter import *
import sqlite3
import tkinter.messagebox
from fungsi import *
```

- I learned how to import all by using from …. import *

```
root = Tk()
b = Database(root)

root.geometry('1000x600')
root.title("Update your item")
root.mainloop()
```

- I have learned the basics of tkinter where you create root window, create an instance for it, create the size & title before you use mainloop() to loop & spawn you the window.

```
conn = sqlite3.connect("D:\Final project\Database\stock.db")
c = conn.cursor()
```

- learned the basic of sqlite where you connect the file into the database file

```
class Database:
        def __init__(self, master, *args, **kwargs):

                self.master = master
                self.heading = Label(master, text="Store your item",
font=('arial 40 bold'), fg='green')
                self.heading.place(x=300, y=0)
```

- the basic of tkinter where you can make label, entry, checkbox, button, and many more by making the variable and then use the (x,y) to put it wherever you want

```
                        sql = 'INSERT INTO clothe (id, name, stock,
originalPrice, price, totalCost, totalEarning, assumedProfit)
VALUES(?,?,?,?,?,?,?,?)'
                        c.execute(sql, (self.id, self.name, self.stock,
self.originalPrice, self.price, self.totalCost, self.totalEarning,
self.assumedProfit))
                        conn.commit()
```

- the use of sql command such as “INSERT, UPDATE, INTO, SELECT, FROM, WHERE, and many more)
- also the use of c.execute to use the sql command
- and the use of conn.commit() to commit the change of the database

```
tkinter.messagebox.showinfo("SUCCESS", 'Your item have been stored')
```

- how to make a pop up message box

```
                        test = 'SELECT id FROM clothe'

                        test2 = c.execute(test)

                        counter = 0

                        for r in test2:

                                self.ids = list(r)

                                if int(self.id) in self.ids:

                                        counter += 1

                        if counter != 0:

                                MsgBox = tkinter.messagebox.askquestion
('Error','There is existing id in database, do you want to rewrite
it?',icon = 'warning')

                                if MsgBox == 'yes':

                                        self.totalCost =
float(self.originalPrice) * float(self.stock)

                                        self.totalEarning =
float(self.price) * float(self.stock)

                                        self.assumedProfit =
float(self.totalEarning - self.totalCost)
```

```
                                        updet = "UPDATE clothe SET name=?,
stock=?, originalPrice=?, price=?, totalCost=?, totalEarning=?,
assumedProfit=? WHERE id=?"

                                        c.execute(updet,
(self.name,self.stock, self.originalPrice, self.price,self.totalCost,
self.totalEarning, self.assumedProfit, self.id))

                                        conn.commit()


tkinter.messagebox.showinfo("Success", "Your Database has been updated")
```

- to check in Database.py if there is the same id in Database, if there is the same id we can choose to rewrite or no

```
            x_loop=0

            y_loop=80

            counter = 0


            for self.p in product_list:

                self.temp_name = Label(self.right,
text=str(product_list[counter]), font=('arial 20 bold'),
bg='SpringGreen3', fg='white')

                self.temp_name.place(x=x_loop, y=y_loop)

                list_of_label.append(self.temp_name)


                x_loop += 225


                self.temp_price = Label(self.right,
text=int(product_quantities[counter]), font=('arial 20 bold'),
bg='SpringGreen3', fg='white')

                self.temp_price.place(x=x_loop, y=y_loop)

                list_of_label.append(self.temp_price)
```

```
                x_loop += 225


                self.temp_quantities = Label(self.right,
text=float(product_prices[counter]), font=('arial 20 bold'),
bg='SpringGreen3', fg='white')
                self.temp_quantities.place(x=x_loop, y=y_loop)
                list_of_label.append(self.temp_quantities)
                x_loop = 0
                y_loop += 40
                counter += 1
```

- how to create label using loop

```
        for lbl in list_of_label:
            lbl.destroy()
```

- how to delete the looped label

```
        self.x = 0




        for i in product_list:
            initial = "SELECT * FROM clothe WHERE id=?"
            result = c.execute(initial, (product_id[self.x], ))


            for r in result:
                self.old_stock = r[2]
                self.og_price = r[3]
                self.price = r[4]
            self.new_stock = int(self.old_stock) -
int(product_quantities[self.x])
            self.total_cost = float(self.og_price) * float(self.new_stock)
```

```
            self.total_earn = float(self.price) * float(self.new_stock)

            self.assumed_profit = float(self.total_earn) -
float(self.total_cost)

            # updating the stock

            sql = "UPDATE clothe SET stock=?, totalCost=?, totalEarning=?,
assumedProfit=? WHERE id=?"

            c.execute(sql, (self.new_stock,self.total_cost,
self.total_earn, self.assumed_profit, product_id[self.x]))

            conn.commit()
```

- how to get new stock after customer buys it
- update the new stock with loop

III.b. Problem that Have Been Overcome

Creating this is program is easy yet difficult since I only started to learn the basic of tkinter and sqlite3. I manage to make the store function and update without any difficulty, and after learning more about the sql command, I manage to make it so you can't store the same ID in database after many times of trials and errors because it keeps store the item and not update the item. The cashier is probably the hardest one for this project because it was very different than the other two. I manage to learn how to create a label for the loop but I stuck for so long to find out how to delete all the label because it keeps deleting only the last label in the cart section, I also have a lot of difficulties in making the decrease stock in the Database where i need to add list for each of the type and eventually manage to make it.

Resources :

-https://www.youtube.com/watch?v=Wgjja1-K_kQ&list=PLeyK9Dw9ShReNENDOoG5r133npUKF5IhD

-https://stackoverflow.com/

-https://www.w3schools.com/sql/sql_syntax.asp

-

V. Source Code

database.py

```
from tkinter import *
import sqlite3
import tkinter.messagebox



conn = sqlite3.connect("D:\Final project\Database\stock.db")
c = conn.cursor()

class Database:
    def __init__(self, master, *args, **kwargs):

        self.master = master
        self.heading = Label(master, text="Store your item", font=('arial 40 bold'),
fg='green')
        self.heading.place(x=300, y=0)

#Labels

        self.id1 = Label(master, text="Enter ID", font=('arial 20 bold'))
        self.id1.place(x=10, y=70)

        self.name1 = Label(master, text="Enter Name", font=('arial 20 bold'))
        self.name1.place(x=10, y=120)

        self.stock1 = Label(master, text="Enter Stock", font=('arial 20 bold'))
        self.stock1.place(x=10, y=170)

        self.originalPrice1 = Label(master, text="Enter Original Price", font=('arial 20
bold'))
        self.originalPrice1.place(x=10, y=220)

        self.price1 = Label(master, text="Enter Price", font=('arial 20 bold'))
        self.price1.place(x=10, y=270)


# ENTRY

        self.id_e = Entry(master, width=25,font=('arial 20 bold'))
        self.id_e.place(x=320,y=70)
```

```
        self.name_e = Entry(master, width=25,font=('arial 20 bold'))
        self.name_e.place(x=320,y=120)

        self.stock_e = Entry(master, width=25,font=('arial 20 bold'))
        self.stock_e.place(x=320,y=170)

        self.originalPrice_e = Entry(master, width=25,font=('arial 20 bold'))
        self.originalPrice_e.place(x=320,y=220)

        self.price_e = Entry(master, width=25,font=('arial 20 bold'))
        self.price_e.place(x=320,y=270)


#BUTTON

        self.btn_store = Button(master, text="Store",font=('arial 20 bold'), width=20,
height=2, bg='green', fg='white', command=self.store_items)
        self.btn_store.place(x=300,y=370)
        self.btn_clear = Button(master, text="Clear", font=('arial 16 bold'), width=10,
height=2, bg='green', fg='white', command=self.delete_entry)
        self.btn_clear.place(x=10,y=370)

#ENTRY LOG

        self.log = Text(master, width=32,height=25)
        self.log.place(x=720, y=70)
#FUNCTION
    def delete_entry(self, *args, **kwargs):
        self.id_e.delete(0, END)
        self.name_e.delete(0, END)
        self.stock_e.delete(0, END)
        self.originalPrice_e.delete(0, END)
        self.price_e.delete(0, END)

    def store_items(self, *args, **kwargs):

        self.id = self.id_e.get()
        self.name = self.name_e.get()
        self.stock = self.stock_e.get()
        self.originalPrice = self.originalPrice_e.get()
        self.price = self.price_e.get()
        if self.id == '' or self.name == '' or self.stock == '' or self.originalPrice == '' or
self.price == '':
            tkinter.messagebox.showinfo('Error', 'Please fill all the entries')
        else:
            test = 'SELECT id FROM clothe'
```

```
                test2 = c.execute(test)
                counter = 0
                for r in test2:
                    self.ids = list(r)
                    if int(self.id) in self.ids:
                        counter += 1
                if counter != 0:
                    MsgBox = tkinter.messagebox.askquestion ('Error','There is existing id in
database, do you want to rewrite it?',icon = 'warning')
                    if MsgBox == 'yes':
                        self.totalCost = float(self.originalPrice) * float(self.stock)
                        self.totalEarning = float(self.price) * float(self.stock)
                        self.assumedProfit = float(self.totalEarning - self.totalCost)
                        updet = "UPDATE clothe SET name=?, stock=?, originalPrice=?,
price=?, totalCost=?, totalEarning=?, assumedProfit=? WHERE id=?"
                        c.execute(updet, (self.name,self.stock, self.originalPrice,
self.price,self.totalCost, self.totalEarning, self.assumedProfit, self.id))
                        conn.commit()
                        tkinter.messagebox.showinfo("Success", "Your Database has been
updated")
                    else:
                        self.id_e.delete(0, END )
                else:
                    self.totalCost = float(self.originalPrice) * float(self.stock)
                    self.totalEarning = float(self.price) * float(self.stock)
                    self.assumedProfit = float(self.totalEarning - self.totalCost)
                    sql = 'INSERT INTO clothe (id, name, stock, originalPrice, price, totalCost,
totalEarning, assumedProfit) VALUES(?,?,?,?,?,?,?,?)'
                    c.execute(sql, (self.id, self.name, self.stock, self.originalPrice, self.price,
self.totalCost, self.totalEarning, self.assumedProfit))
                    conn.commit()
                    self.log.insert(END, 'Added ' + str(self.name) + ' into your database\n')
                    tkinter.messagebox.showinfo("SUCCESS", 'Your item have been stored')

root = Tk()
b = Database(root)

root.geometry('1000x600')
root.title("Store your item")
root.mainloop()
```

update.py

```
from tkinter import *
import sqlite3
import tkinter.messagebox

conn = sqlite3.connect("D:\Final project\Database\stock.db")
c = conn.cursor()

class Database:
	def __init__(self, master, *args, **kwargs):

		self.master = master
		self.heading = Label(master, text="Update your item", font=('arial 40 bold'),
fg='green')
		self.heading.place(x=300, y=0)

#Label & Entry
		self.id1 = Label(master, text="Enter ID", font=('arial 20 bold'))
		self.id1.place(x=10, y=70)

		self.id_entry = Entry(master, font=('arial 20 bold'), width=10)
		self.id_entry.place(x=320,y=70)

		self.id_btn = Button(master, text='Search', font=('arial 20 bold'),width=8, height=1,
bg='green', fg='white', command=self.search)
		self.id_btn.place(x=520,y=60)
#Labels
		self.name1 = Label(master, text="Enter Name", font=('arial 20 bold'))
		self.name1.place(x=10, y=120)

		self.stock1 = Label(master, text="Enter Stock", font=('arial 20 bold'))
		self.stock1.place(x=10, y=170)

		self.originalPrice1 = Label(master, text="Enter Original Price", font=('arial 20 bold'))
		self.originalPrice1.place(x=10, y=220)

		self.price1 = Label(master, text="Enter Price", font=('arial 20 bold'))
		self.price1.place(x=10, y=270)

		self.totalCost1 = Label(master, text="Total Cost Price", font=('arial 20 bold'))
		self.totalCost1.place(x=10, y=320)

		self.totalEarning1 = Label(master, text="Total Earning Price", font=('arial 20 bold'))
		self.totalEarning1.place(x=10, y=370)
```

```
#ENTRY

        self.name_e = Entry(master, width=25,font=('arial 20 bold'))
        self.name_e.place(x=320,y=120)

        self.stock_e = Entry(master, width=25,font=('arial 20 bold'))
        self.stock_e.place(x=320,y=170)

        self.originalPrice_e = Entry(master, width=25,font=('arial 20 bold'))
        self.originalPrice_e.place(x=320,y=220)

        self.price_e = Entry(master, width=25,font=('arial 20 bold'))
        self.price_e.place(x=320,y=270)

        self.totalCost_e = Entry(master, width=25,font=('arial 20 bold'))
        self.totalCost_e.place(x=320,y=320)

        self.totalEarning_e = Entry(master, width=25,font=('arial 20 bold'))
        self.totalEarning_e.place(x=320,y=370)


#BUTTON
        self.btn_store = Button(master, text="Update",font=('arial 20 bold'), width=20,
height=2, bg='green', fg='white', command=self.update)
        self.btn_store.place(x=300,y=470)
        self.btn_clear = Button(master, text="Clear", font=('arial 16 bold'), width=10,
height=2, bg='green', fg='white', command=self.delete_entry)
        self.btn_clear.place(x=10,y=470)
#ENTRY LOG
        self.log = Text(master, width=30,height=25)
        self.log.place(x=720, y=70)
#FUNCTION
    def delete_entry(self, *args, **kwargs):
        self.id_entry.delete(0, END)
        self.name_e.delete(0, END)
        self.stock_e.delete(0, END)
        self.originalPrice_e.delete(0, END)
        self.price_e.delete(0, END)
        self.totalCost_e.delete(0, END)
        self.totalEarning_e.delete(0, END)

    def search(self, *args, **kwargs):
        try:
            sql = "SELECT * FROM clothe WHERE id=?"
            result = c.execute(sql, (self.id_entry.get(), ))
```

```
            for r in result:
                self.n1 = r[1]
                self.n2 = r[2]
                self.n3 = r[3]
                self.n4 = r[4]
                self.n5 = r[5]
                self.n6 = r[6]
                self.n7 = r[7]
            conn.commit()
            #insert the entries to update
            self.name_e.delete(0, END)
            self.name_e.insert(0, str(self.n1))

            self.stock_e.delete(0, END)
            self.stock_e.insert(0, str(self.n2))
            self.originalPrice_e.delete(0, END)
            self.originalPrice_e.insert(0, str(self.n3))

            self.price_e.delete(0, END)
            self.price_e.insert(0, str(self.n4))

            self.price_e.delete(0, END)
            self.price_e.insert(0, str(self.n4))

            self.price_e.delete(0, END)
            self.price_e.insert(0, str(self.n4))

            self.totalCost_e.delete(0, END)
            self.totalCost_e.insert(0, str(self.n5))

            self.totalEarning_e.delete(0, END)
            self.totalEarning_e.insert(0, str(self.n6))
        except:
            tkinter.messagebox.showinfo('Error', 'Please input correctly')

    def update(self, *args, **kwargs):
        try:
            #get all the updated value
            self.u_id = self.id_entry.get()
            self.u1 = self.name_e.get()
            self.u2 = self.stock_e.get()
            self.u3 = self.originalPrice_e.get()
            self.u4 = self.price_e.get()
            self.u5 = float(self.u3) * float(self.u2)
            self.u6 = float(self.u4) * float(self.u2)
            self.u7 = float(self.u6) - float(self.u5)
```

```
                query = "UPDATE clothe SET name=?, stock=?, originalPrice=?, price=?,
totalCost=?, totalEarning=?, assumedProfit=? WHERE id=?"
                c.execute(query, (self.u1, self.u2, self.u3, self.u4, self.u5, self.u6,self.u7,
self.u_id))
                conn.commit()
                self.log.insert(END, 'Your item have been updated\n')
                tkinter.messagebox.showinfo("Success", "Your item have been updated")
            except:
                tkinter.messagebox.showinfo('Error', 'Please input correctly')


root = Tk()
b = Database(root)

root.geometry('1000x600')
root.title("Update your item")
root.mainloop()
```

Cashier.py

```
from tkinter import *
import sqlite3
import tkinter.messagebox
import datetime

conn = sqlite3.connect("D:\Final project\Database\stock.db")
c = conn.cursor()


#date
date = datetime.datetime.now().date()

#temp list
product_list = []
product_prices = []
product_quantities = []
product_id = []
list_of_label =[]

class Application:
    def __init__(self, master, *args, **kwargs):
        self.master = master
        #frames
        self.left = Frame(master, width=400, height=600, bg= 'white')
        self.left.pack(side=LEFT)
```

```
        self.right = Frame(master, width=600, height=600, bg='SpringGreen3')
        self.right.pack(side=RIGHT)

        #components
        self.heading = Label(self.left, text='Cashier', font=('arial 18 bold underline'),
bg='SpringGreen3', fg='white')
        self.heading.place(x=10,y=0)

        self.date_1 = Label(self.right, text="Today's Date: " + str(date), font=('arial 18 bold
underline'),bg='SpringGreen3', fg='white')
        self.date_1.place(x=0,y=0)

        #table invoice
        self.tproduct = Label(self.right, text='Products', font=('arial 18 bold underline'),
bg='SpringGreen3', fg='white')
        self.tproduct.place(x=0,y=40)

        self.tproduct = Label(self.right, text='Quantity', font=('arial 18 bold underline'),
bg='SpringGreen3', fg='white')
        self.tproduct.place(x=225,y=40)

        self.tproduct = Label(self.right, text='Total Price', font=('arial 18 bold underline'),
bg='SpringGreen3', fg='white')
        self.tproduct.place(x=450,y=40)

        #entry label
        self.itemlabel = Label(self.left, text='Enter product ID', font=('arial 18 bold'), bg='white')
        self.itemlabel.place(x=10,y=40)

        self.item_e = Entry(self.left, width=10,font=('arial 24 bold'),bg='grey')
        self.item_e.place(x=10,y=80)

        #button
        self.item_b = Button(self.left, text='Search', width=10, height=2, bg='SpringGreen3',
command=self.search)
        self.item_b.place(x=150,y=80)

    def search(self, *args, **kwargs):
        self.get_id = self.item_e.get()
        query = "SELECT * FROM clothe WHERE id=?"
        result = c.execute(query, (self.get_id, ))
        for self.r in result:
            self.get_name = self.r[1]
            self.get_stock = self.r[2]
            self.get_originalPrice = self.r[3]
            self.get_price = self.r[4]
```

```
            self.get_totalCost = self.r[5]
            self.get_totalEarning = self.r[6]
            self.get_assumedProfit = self.r[7]

        self.product_name = Label(self.left,text="Product's Name: " + str(self.get_name),
font=('arial 18 bold'), bg='white')
        self.product_name.place(x=10, y=140)

        self.product_price = Label(self.left,text='Price: ' + str(self.get_price), font=('arial 18 bold'),
bg='white')
        self.product_price.place(x=10, y=180)

        #quantity / discount
        self.quantity1 = Label(self.left, text='Enter Quantity: ', font=('arial 18 bold'), bg='white')
        self.quantity1.place(x=10, y=240)

        self.quantity_e = Entry(self.left,font=('arial 18 bold') ,width=10,bg='grey')
        self.quantity_e.place(x=200,y=240)
        self.quantity_e.focus()

        self.discount1 = Label(self.left,text='Enter Discount: ', font=('arial 18 bold'),bg='white')
        self.discount1.place(x=10, y=280)

        self.discount_e = Entry(self.left,font=('arial 18 bold') ,width=10,bg='grey')
        self.discount_e.place(x=200,y=280)
        self.discount_e.insert(END, 0)

        #add to cart button
        self.cart_b = Button(self.left, text='Add to cart', width=10, height=2, bg='SpringGreen3',
command=self.add_to_cart)
        self.cart_b.place(x=200,y=320)

        self.tot_b = Button(self.left, text='Total', width=10, height=2, bg='SpringGreen3',
command=self.total)
        self.tot_b.place(x=300,y=320)


    def add_to_cart (self, *args, **kwargs):
        self.u1 = self.quantity_e.get()
        self.u2 = self.discount_e.get()
        self.total_price = int(self.get_price) * int(self.u1)
        if int(self.u1) > int(self.get_stock):
            tkinter.messagebox.showinfo('Error', 'quantity exceed item stocks')
        else:
            if self.u2 == 0:
                product_prices.append(self.total_price)
            else:
```

```
            self.total_price -= int(self.total_price) * int(self.u2) / 100
            product_prices.append(self.total_price)

        product_list.append(self.get_name)
        product_quantities.append(self.u1)
        product_id.append(self.get_id)
        x_loop=0
        y_loop=80
        counter = 0

        for self.p in product_list:
            self.temp_name = Label(self.right, text=str(product_list[counter]), font=('arial 20
bold'), bg='SpringGreen3', fg='white')
            self.temp_name.place(x=x_loop, y=y_loop)
            list_of_label.append(self.temp_name)

            x_loop += 225

            self.temp_price = Label(self.right, text=int(product_quantities[counter]), font=('arial
20 bold'), bg='SpringGreen3', fg='white')
            self.temp_price.place(x=x_loop, y=y_loop)
            list_of_label.append(self.temp_price)

            x_loop += 225

            self.temp_quantities = Label(self.right, text=float(product_prices[counter]), font=('arial
20 bold'), bg='SpringGreen3', fg='white')
            self.temp_quantities.place(x=x_loop, y=y_loop)
            list_of_label.append(self.temp_quantities)
            x_loop = 0
            y_loop += 40
            counter += 1
        self.item_e.delete(0, END)
        self.quantity_e.delete(0, END)
        self.product_name.destroy()
        self.product_price.destroy()



    def total(self, *args, **kwargs):
        total_price_bill = sum(product_prices)
        self.total_label = Label(self.left, text=('total: ' + str(total_price_bill)), font=("arial 20 bold"),
bg="white")
        self.total_label.place(x=10, y=360)

        #generate bill
        self.bill_name= Label(self.left, text='Given Amount', font=('arial 18 bold'), bg='white')
```

```
        self.bill_name.place(x=10, y=400)

        self.bill_name_e = Entry(self.left,font=('arial 18 bold') ,width=10,bg='grey')
        self.bill_name_e.place(x=200, y=400)

        #change button
        self.change_btn = Button(self.left, text='Calculate change', width=16, height=2,
bg='SpringGreen3',command=self.cash_change)
        self.change_btn.place(x=200,y=440)


    def cash_change(self, *args, **kwargs):
        #generate bill
        self.bill_btn = Button(self.left, text='Update the Database', width=50, height=2,
bg='SpringGreen3', command=self.reduce_stock)
        self.bill_btn.place(x=10,y=540)


        total_price_bill = sum(product_prices)
        self.total_customer_money = self.bill_name_e.get()
        if float(self.total_customer_money) < total_price_bill:
            tkinter.messagebox.showinfo('Error', 'need more money')
        else:
            labelll_change = float(self.total_customer_money) - total_price_bill
            self.label_change = Label(self.left, text=('change: ' + str(labelll_change)), font=('arial 20
bold'), bg='white')
            self.label_change.place(x=10, y=500)


    def reduce_stock (self, *args, **kwargs):
        # decrease the stock
        self.x = 0


        for i in product_list:
            initial = "SELECT * FROM clothe WHERE id=?"
            result = c.execute(initial, (product_id[self.x], ))

            for r in result:
                self.old_stock = r[2]
                self.og_price = r[3]
                self.price = r[4]
            self.new_stock = int(self.old_stock) - int(product_quantities[self.x])
            self.total_cost = float(self.og_price) * float(self.new_stock)
            self.total_earn = float(self.price) * float(self.new_stock)
            self.assumed_profit = float(self.total_earn) - float(self.total_cost)
            # updating the stock
```

```
            sql = "UPDATE clothe SET stock=?, totalCost=?, totalEarning=?, assumedProfit=?
WHERE id=?"
            c.execute(sql, (self.new_stock,self.total_cost, self.total_earn, self.assumed_profit,
product_id[self.x]))
            conn.commit()

        tkinter.messagebox.showinfo('Success','Database stock has been decreased')

        for lbl in list_of_label:
            lbl.destroy()

        self.product_name.destroy()
        self.product_price.destroy()
        self.quantity1.destroy()
        self.quantity_e.destroy()
        self.discount1.destroy()
        self.discount_e.destroy()
        self.cart_b.destroy()
        self.tot_b.destroy()
        self.change_btn.destroy()
        self.bill_btn.destroy()
        self.bill_name.destroy()
        self.bill_name_e.destroy()
        self.label_change.destroy()
        self.total_label.destroy()
        self.item_e.delete(0, END)

        product_list.clear()
        product_prices.clear()
        product_quantities.clear()
        product_id.clear()
        list_of_label.clear()


root = Tk()
b = Application(root)

root.geometry('1000x600')
root.title("Cashier")
root.mainloop()
```