



Assignment Cover Letter

(Individual Work)

Student Information:

Surname

Given Names

Student ID Number

1.

Edgari

Eric

2301902352

Course Code : COMP6056

Course Name : Introduction to Programming

Class : L1AC

Name of Lecturer(s) : Ida Bagus Kerthyayana Manuaba,
S.T., Ph.D.

Major : CS

Title of Assignment : Inventory System with GUI
(if any)

Type of Assignment : Final Project

Submission Pattern

Due Date : 17-01-20

Submission Date : 17-01-20

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. Brenda Spears

“Inventory System with GUI”

Name : Eric Edgari

ID : 2301902352

I. Project Specification**The reason I make this program:**

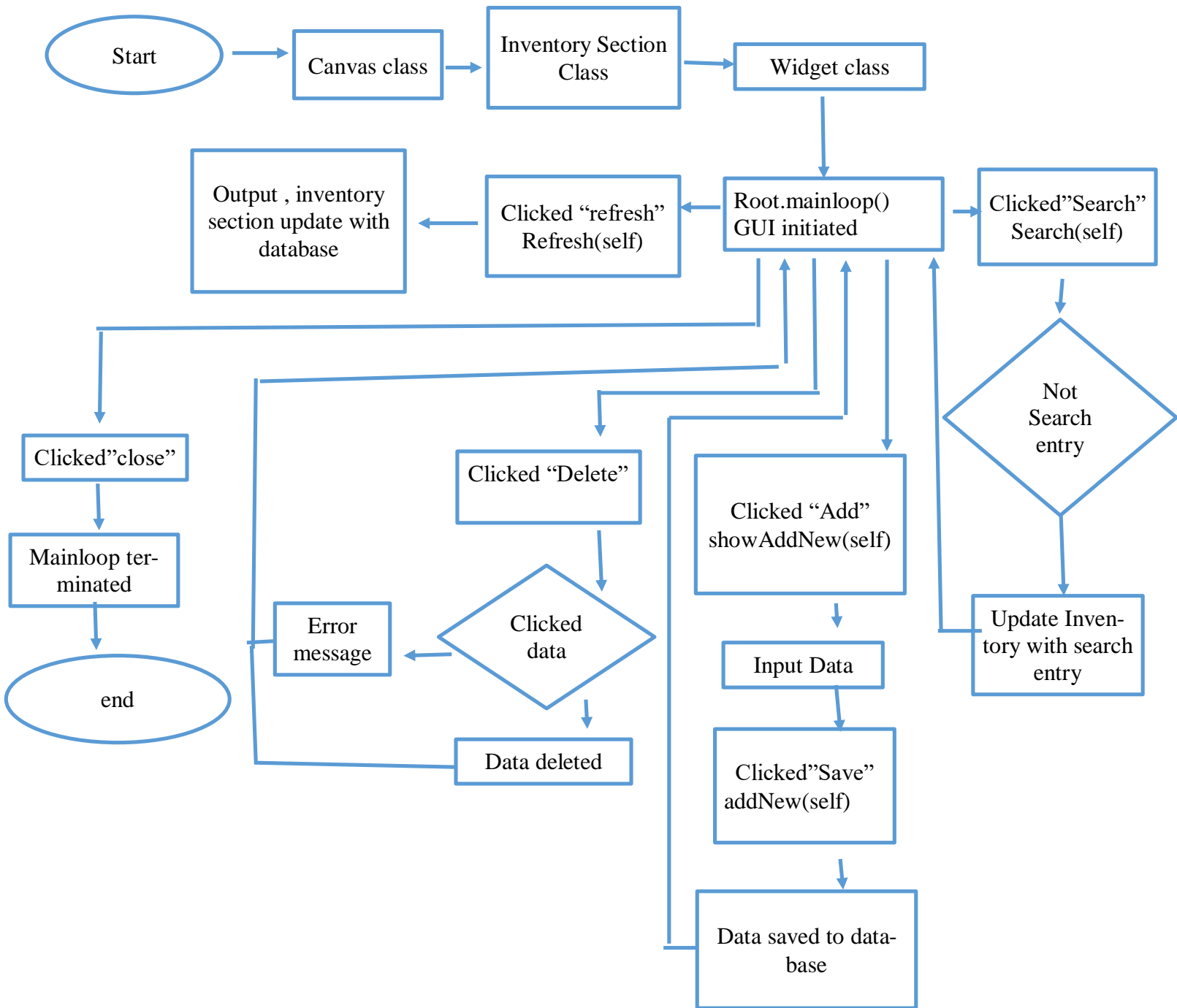
My family come from a merchant background. If I have a free time , I always help my family in the shop. While I was working there , I found that the inventory system still traditional , they still use paper and take notes. With that in mind , I set a goal for myself to make a simple inventory system that aim for the older person . With that in mind , I need to make a simple and easy to use program that is easy to use

The function of this program:

The purpose of this program is to make use of the class Inventory system that takes an input from a user and save it to a local database. But this time I’m going to take it to the next level by making GUI (Graphical User Interface) take makes it more user friendly. The GUI library that I’m going to use is from Tkinter and the database that it’s going to connect is from Sqlite3.

II.a. Solution Design

Project's Hierarchy Chart



II.b. Explanation of Each Function Inside and Outside classes

inventory.py

- **Def database() :**
 - Make the conn and cursor variable global so that it can be used in all the class
 - Make the connection to the sqlite3 and create local file with the name inventory_data-base.db
 - Make the connection between the con and the cursor
 - By using the command cursor.execute , it'll pass the argument product and create a product if its not yet been made , the product table contain , (product name, product qty ,product price.)
- **Class Canvas**
- **init (self,master) :**
 - making a variable called 'canvas' that will store tkinter method called Canvas
 - put the canvas into the GUI by using 'pack' function from tkinter
- **Class Title**
- **init (self,master) :**
 - making a variable called frame_title that will store tkinter method called Frame
 - put the canvas into the GUI by using 'place' function from tkinter
 - make a label by using self.title.label and store tkinter function for label
 - put the label inside the GUI using the 'place' function
- **Class Widgets**
- **Init(self,master)**
 - Making a variable to contain the search result

- Making other widgets frame and frame inventory list using the tkinter 'frame' module
- Making a scroll bar for the inventory section
- Making a listbox for the inventory section
- Connecting the inventory section to the sqlite database
- Calling the displaydata function from the same class
- Making the search bar by using the tkinter module 'entry' and place it using tkinter 'place' module
- Making the search button , add button , refresh button , and delete button using the tkinter module 'button'

- **Displaydata(self) :**

- Accessing the database function
- Using the execute function from sqlite to access the 'product' table
- Making a variable fetch to fetch all the data
- Make a loop to insert the data to the inventory box
- And close the cursor and conn

- **showAddnew()**

- making a variable that can contain the input of product name,price,and quantity
- making a new window using the tk.toplevel and store it to addnewform
- making the product name , price , and quantity label and entry box using the tkinter.label and tkinter.entry
- making a save button that runs the function addNew , to save the input to the database

- **addNew(self)**

- calling the database function to access it

- and insert the product.name , product.qty, product.price that are already been input to the sqlite database

- **Refresh(self)**

- Delete the inventorybox display
- Call the displaydata function to print the new one

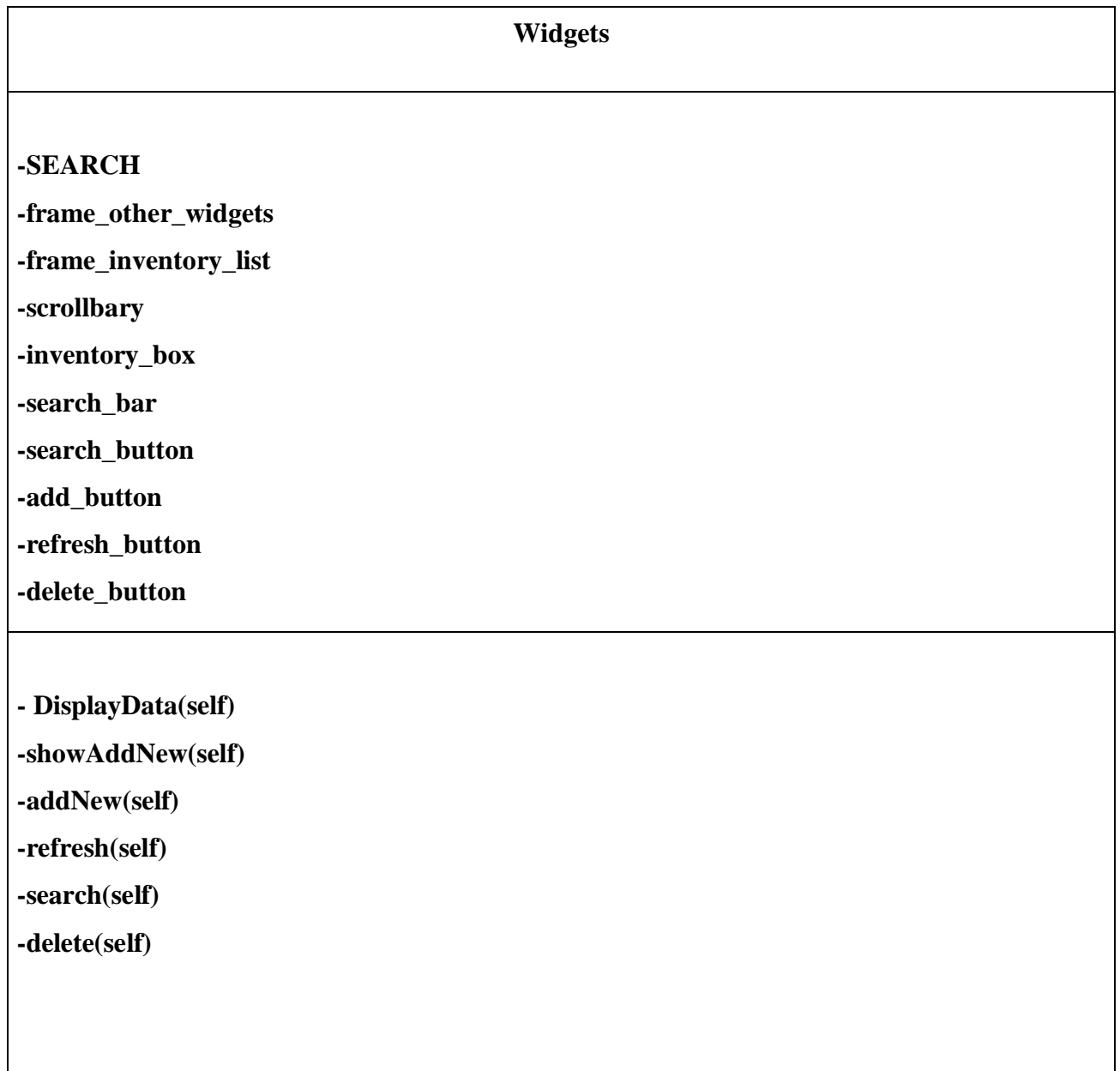
- **Search(self)**

- If the search bar contain result , it will delete the inventorybox display
- Accessing the database() .
- Get the string that is put in the search bar , and select that specific product
- Put the specific product that is matched to the search string, to the inventory box by using a loop

- **Delete(self)**

- if the self.inventory_box not selected , It will conjure a message box that's called error
- else , it will pop a messagebox that said do you want to delete the record
- and if the messagebox got the 'yes' answer
- it will make a variable to focus to the selected part
- accessing the database function and the delete the specific data

Class Diagram



Canvas
-canvas
- init(self,master)

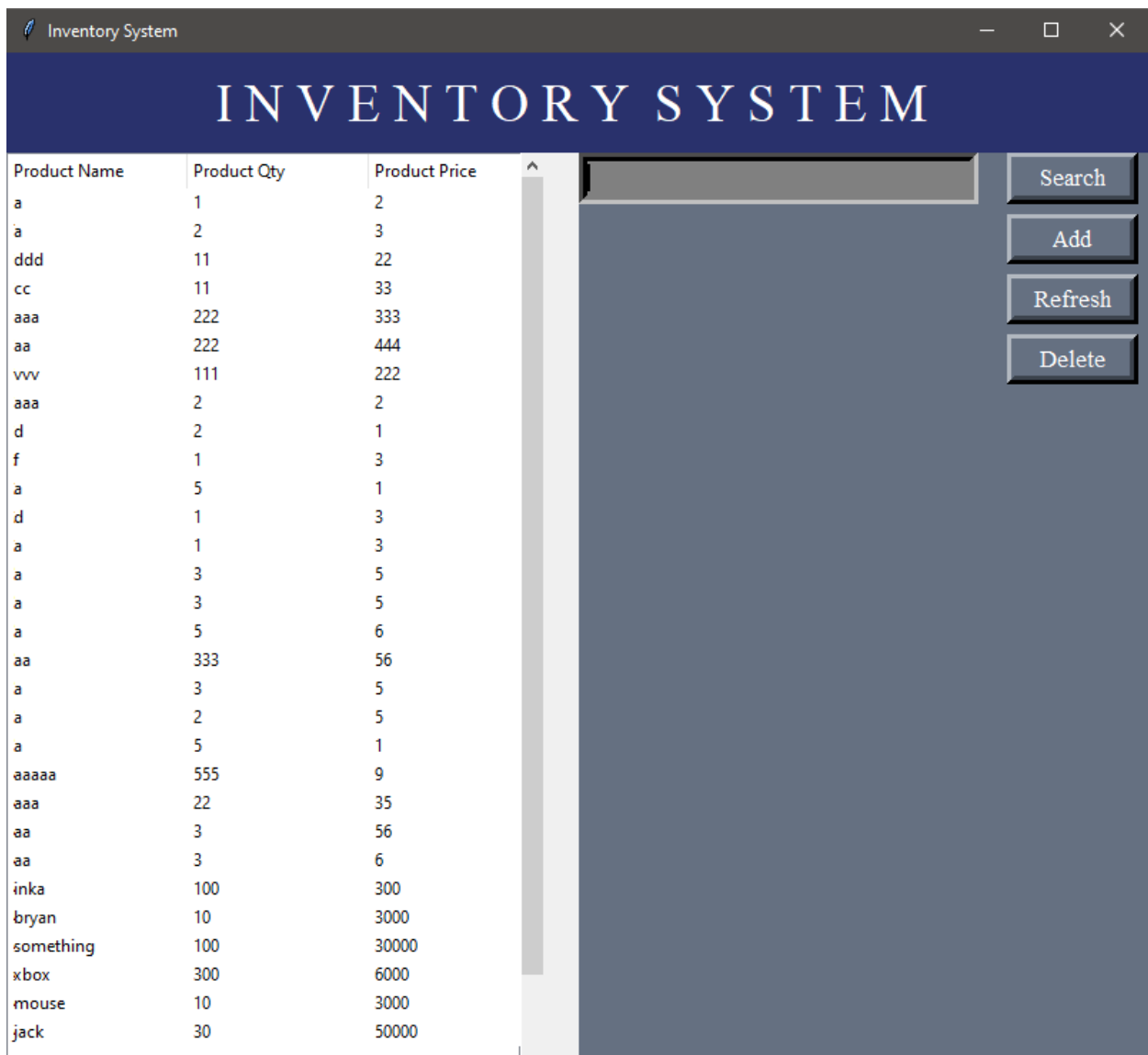
Title
-frame_title -title_label
- init(self,master)

III. Discussiong of what was implemented and how it works and evidence

The simple term how this program works is. First, you open the program, when the program run , it will display Graphical User Interface (GUI). In the left section there will be a list box that will display all the store data , and in the right section , there will be 1 entry box which used as search bar and there will be 4 button . each button named (search , add, refresh, delete).

Tkinter library implemented and how it works:

Each of the button , label , list box , and entry field are made by using tkinter module , After I made it , I need to carefully place it inside the screen that are already been made by the tkinter.



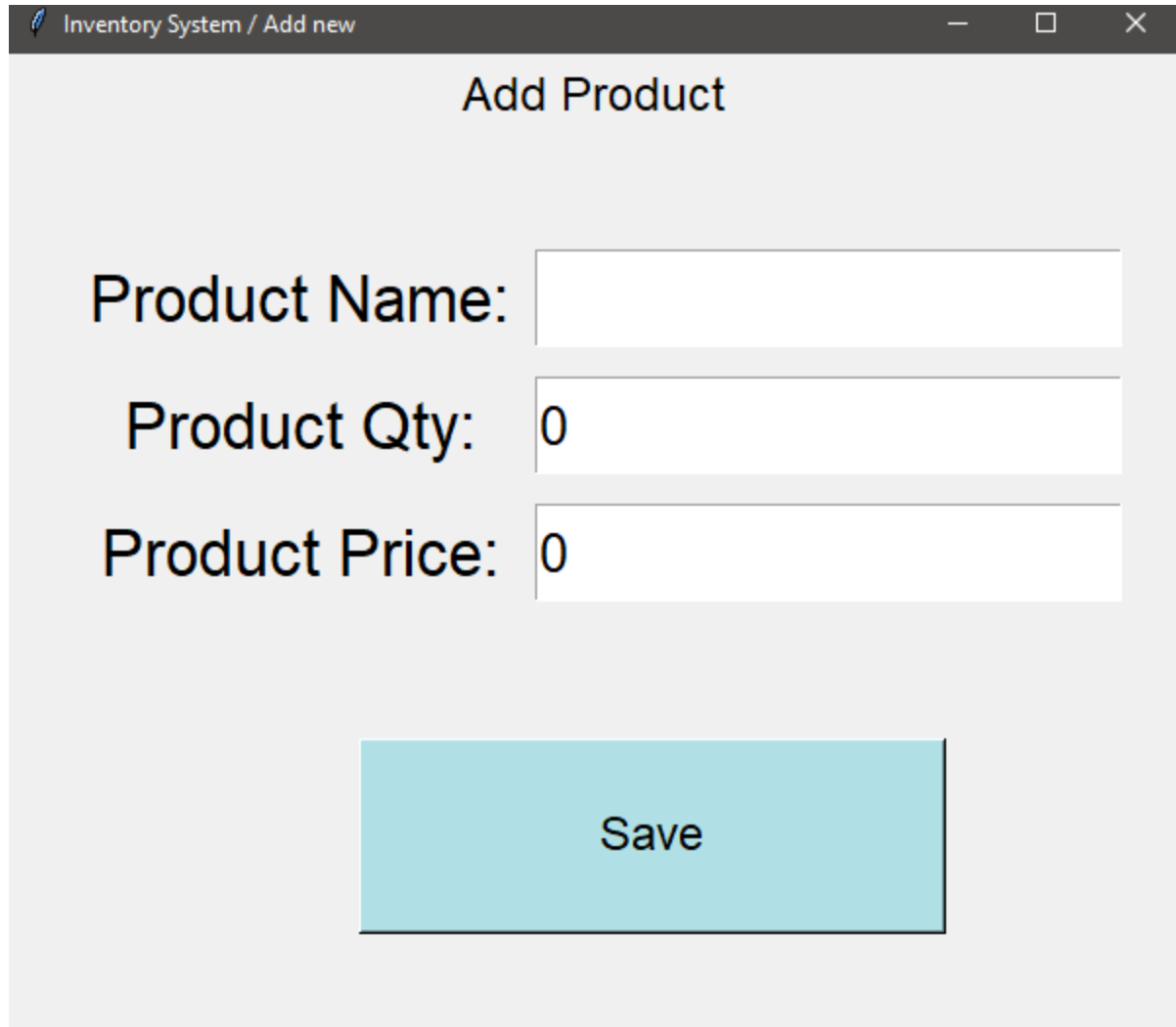
Sqlite3 library implemented and how it works:

This library is used to save the data or you can say as a database. It works by saving the data in a single stored disk file . It works like a csv or any file like you can write , save , modify , and delete . But they have their own function

Add button

In this specific button when pressed, it will connect to a function that will make a new window. Inside that window will conjure an Add form . In that add form that user will input a product name , product quantity , and product price inside an entry bar for each respective entry bar. At the bottom of the form , there will be a save button , that button is used to save and connected a function that save all the input inside a database from sqlite 3

The Add Form



The screenshot shows a window titled 'Inventory System / Add new'. Inside the window, the title 'Add Product' is centered at the top. Below the title, there are three input fields arranged vertically. The first field is labeled 'Product Name:' and is empty. The second field is labeled 'Product Qty:' and contains the value '0'. The third field is labeled 'Product Price:' and also contains the value '0'. At the bottom of the form, there is a large, light blue rectangular button with the word 'Save' centered on it.

Refresh button

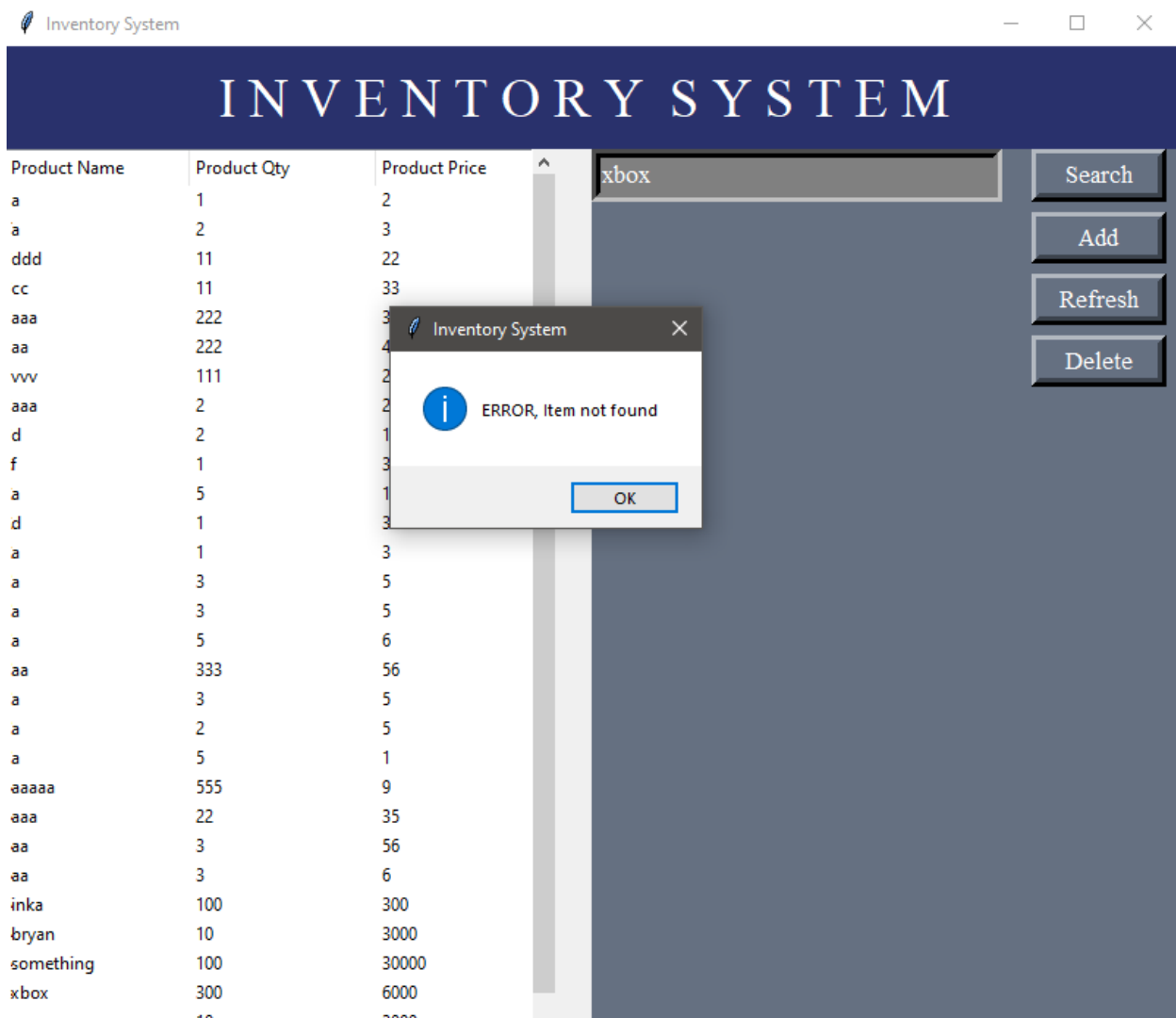
In this specific button when pressed , it will connect to a function that going to delete the list box for the inventory box and fetch all the data from the sqlite3 database , and display it in the list box. Basiclly this works to update the listbox to the latest database

Delete Button

In this specific button when pressed , it will connect to a function that take a parameter , that parameter is ,you need to click the item in the inventory box , if you don't select an item , it will conjure an error box . if you clicked the item and pressed the delete button , it will conjure an

message box that said , ‘do you really want to delete the item” , if you clicked ‘yes’ , it will delete the specific item from the database and in the listbox.

When no item was clicked if



When there's item clicked

INVENTORY SYSTEM

Product Name	Product Qty	Product Price
vvv	111	222
aaa	2	2
d	2	1
f	1	3
a	5	1
d	1	3
a	1	3
a	3	5
a	3	5
a	5	6
aa	333	56
a	3	5
a	2	5
a	5	1
aaaaa	555	9
aaa	22	35
aa	3	56
aa	3	6
inka	100	300
bryan	10	3000
something	100	30000
xbox	300	6000
mouse	10	3000
jack	30	50000
ufo	300	5000
laptop	300	60000
kartu	30	5000
paku	300	7000
handphone	30	600000

Inventory System

Are you sure you want to delete this record?

something	100	30000
xbox	300	6000
jack	30	50000
ufo	300	5000
laptop	300	60000

Item deleted

Search Button

In this specific button when pressed, it will connect to a function that take a parameter in the search entry box. It will get the string and find it in the database , after that it will display it in the list box .

The search button used

Inventory System

INVENTORY SYSTEM

Product Name	Product Qty	Product Price
xbox	300	6000

xbox

Search

Add

Refresh

Delete

III.a. Lessons that Have Been Learned

```
import sqlite3
import tkinter as tk
from tkinter import messagebox, ttk
```

1. The use of “sqlite3”:

I found some more easy way to store the data off the inventory rather than using the csv or a normal text file , using the sqlite3 It literally create you selfcontained database engine and stored it in a single disk file . it have a bunch of useful module like:

```
def DisplayData(self):
    database()
    cursor.execute("SELECT * FROM `product`")
    selecting a data
```

```
def addNew(self):
    database()
    cursor.execute("INSERT INTO `product` (product_name, product_qty, product_price)
VALUES(?, ?, ?)",
                (str(PRODUCT_NAME.get()), int(PRODUCT_QTY.get()), int(PROD-
UCT_PRICE.get()))
    and inserting a new data
```

2. The use of tkinter :

This module is really amazing , I always get bored when making a a program cause it only runs in a terminal , but this time , I can make a Graphical interface or should I say a normal program that everyone can use , I have the freedom of designing rather using the monotone of terminal . You can put button , label , entry , and much more .some of the uses

```
frame_other_widgets = tk.Frame(master, bg="#657081") #MAKING FRAME FOR THE RIGHT SIDE
OF PROGRAM
frame_other_widgets.place(relx=0.5, rely=0, relheight=1, relwidth=0.5)

frame_inventory_list = tk.Frame(master, bg="black") #MAKING FRAME FOR TGE LEFT SIDE
OF PROGRAM
frame_inventory_list.place(relx=0, rely=0, relheight=1, relwidth=0.45)
```

III.b. Problem that Have Been Overcome

The start of the projects its really hard , cause this time , I’m really doing it on my own , at first I tried to make website using the django but its really hard cause I know my limitation . so Im going to the alternative way to making a simple inventory system. This doesn’t go as easy as I though too , in this project I really learned a lot of things , I understand I’m still a beginner programmer , I really have a hard time finding the good documentation and how to use it . thank god , in the end if found the right documentation and tutorial that can teach me . I found out that not

all documentation and tutorial is easy to understand , you need to find the one you really get it , if not , in the end you just copy someone code without really understand what it really meant . As this was really my project this is gonna be my first experience on making a project , so I think a lot of rookie mistake has been made but I tried my best . I find that , in this project I can implement all the uses of the basic programming from the class , mostly on class parts and global variable . I found out that , there's not only one way to solve a problem , but it has a multiple way . And what best suit you is the best for you . This is one of the solution concept for the problem that I have when I tried to find the documentation and tutorial for this program . I think this is really a good experience for my first project as an programmer

<https://bastibe.de/2013-05-30-speeding-up-matplotlib.html>

Resources :

(<https://code-projects.org/simple-inventory-system-in-python-with-source-code/>)

(the source code as a reference for making this project possible for me)

https://www.tutorialspoint.com/python/tk_entry.htm (the documentation for the tkinter module that I really understand)

<https://www.youtube.com/watch?v=D8-snVfekto> (the tutorial that I use to getting hang of tkinter)

V. Source Code

```
import sqlite3
import tkinter as tk
from tkinter import messagebox, ttk
#Thanks To Mark Arvin (https://code-projects.org/simple-inventory-system-in-python-with-source-code/)
#for providing source code for study and reference for making this project possible

#THE SIZE THAT WE ARE GOING TO WORK WITH (IN PIXEL)
HEIGHT = 700
WIDTH = 800

#MAKING THE DATABASE TO SAVE THE INVENTORY DATA, USING THE MODULE FROM SQLITE3
def database():
    global conn , cursor
    conn = sqlite3.connect("inventory_database.db")
    cursor = conn.cursor()
    cursor.execute(
        "CREATE TABLE IF NOT EXISTS 'product' (product_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, product_name TEXT, product_qty TEXT, product_price TEXT)"
    )

#MAKING THE GUI APPLICATION USING THE TKINTER

#MAKING THE CANVAS FOR EASYMODELLING
class Canvas:
    def __init__(self, master):
        canvas = tk.Canvas(root, height=HEIGHT, width=WIDTH)
```

```

        canvas.pack()

#INVENTORY FRAME FOR EASYMODELLING

#TITLE FRAME FOR EASY MODELLING
class Title:
    def __init__(self,master):
        frame_title = tk.Frame(master,bg='#29316C')
        frame_title.place(relx=0, rely=0, relheight=0.1, relwidth=1)

        self.title_label = tk.Label(frame_title, text="I N V E N T O R Y   S Y S T E M", font=("Times New Roman",28), bg='#29316C',fg='white')
        self.title_label.place(relx=0.05, rely=0.25, relheight=0.50, relwidth=0.90)

#THE MAIN CLASS WHERE WE PUT OUR WIDGET AND ITS FUNCTION
class Widgets:
    def __init__(self,master):

        self.SEARCH = tk.StringVar() #VARIABLE TO CONTAIN THE SEARCH RESULT
        frame_other_widgets = tk.Frame(master, bg="#657081") #MAKING FRAME FOR THE RIGHT SIDE OF PROGRAM
        frame_other_widgets.place(relx=0.5, rely=0, relheight=1, relwidth=0.5)

        frame_inventory_list = tk.Frame(master, bg="black") #MAKING FRAME FOR THE LEFT SIDE OF PROGRAM
        frame_inventory_list.place(relx=0, rely=0, relheight=1, relwidth=0.45)

        self.scrollbary = tk.Scrollbar(master,orient='vertical') #MAKING THE SCROLL-BAR FOR THE INVENTORY SECTION

        #MAKING A LISTBOX TO CONTAIN ALL THE DATA FROM THE DATABASE USING TKINTER TREEVIEW , A MORE ADVANCE LISTBOX
        self.inventory_box = ttk.Treeview(frame_inventory_list,columns=("ProductID","Product Name","Product Qty","Product Price"),selectmode='extended',yscrollcommand=self.scrollbary.set)
        self.scrollbary.config(command=self.inventory_box.yview())
        self.scrollbary.place(relx=0.45,relly=0.1,relheight=1)
        self.inventory_box.heading('ProductID',text='ID',anchor='w')
        self.inventory_box.heading('Product Name', text='Product Name', anchor='w')
        self.inventory_box.heading('Product Qty', text='Product Qty', anchor='w')
        self.inventory_box.heading('Product Price', text='Product Price', anchor='w')
        self.inventory_box.column('#0', stretch = 'no',minwidth=0, width=0)
        self.inventory_box.column('#1', stretch = 'no',minwidth=0,width=0)
        self.inventory_box.column('#2', stretch='no', minwidth=0, width=126)
        self.inventory_box.column('#3', stretch='no', minwidth=0, width=127)
        self.inventory_box.column('#4', stretch='no', minwidth=0, width=127)
        self.inventory_box.place(relx=0, rely=0.1, relheight=1, relwidth=1)
        self.DisplayData() #CALLING THE DISPLAY DATA FUNCTION SO THAT WHEN THE PROGRAM EXECUTE IT WILL SHOW THE INITIAL DATA

        #MAKING THE SEARCH BAR
        self.search_bar = tk.Entry(frame_other_widgets, textvariable=self.SEARCH,font=('times new roman',13), bd=6, bg='grey',fg='white')
        self.search_bar.place(relx=0, rely=0.1, relheight=0.05, relwidth=0.70)
        #MAKING THE SEARCH BUTTON
        self.search_button = tk.Button(frame_other_widgets,bg='#657081',text="Search",bd=6,font=("times new roman",13),fg='white',

```



```

        command=lambda:self.search())
    self.search_button.place(relx=0.75, rely=0.1, relheight=0.05, relwidth=0.23)

    #MAKING THE ADD BUTTON
    self.add_button = tk.Button(frame_other_widgets,
text="Add",bg='#657081',bd=6,font=("times new roman",13),fg='white', command=lambda:
self.showAddNew())
    self.add_button.place(relx=0.75, rely=0.16, relheight=0.05, relwidth=0.23)

    #MAKING THE REFRESH BUTTON
    self.refresh_button = tk.Button(frame_other_widgets, text="Re-
fresh",bd=6,bg='#657081',font=("times new roman",13),fg='white',
        command=lambda: self.refresh())
    self.refresh_button.place(relx=0.75, rely=0.22, relheight=0.05, rel-
width=0.23)

    #MAKING THE DELETE BUTTON
    self.delete_button = tk.Button(frame_other_widgets,bg='#657081',bd=6,
text="Delete",font=("times new roman",13),fg='white',command= lambda:self.delete()
        )
    self.delete_button.place(relx=0.75, rely=0.28, relheight=0.05, relwidth=0.23)

    #MAKING A FUNCTION THAT TAKE THE DATA FROM THE SQLITE3 DATABASE AND DISPLAY IT IN
THE DISPLAYBOX
    def DisplayData(self):
        database()
        cursor.execute("SELECT * FROM `product`")
        fetch = cursor.fetchall()
        for data in fetch:
            self.inventory_box.insert('', 'end', values=data)
        cursor.close()
        conn.close()

    #MAKING A FUNCTION THAT WHEN EXECUTE WILL POP UP A WINDOW FOR INPUTTING NEW
PRODUCT TO BE STORED
    def showAddNew(self):
        global addnewForm,PRODUCT_NAME,PRODUCT_QTY,PRODUCT_PRICE
        PRODUCT_NAME = tk.StringVar()
        PRODUCT_QTY = tk.IntVar()
        PRODUCT_PRICE = tk.IntVar()
        addnewForm = tk.Toplevel(width=600,height=500)
        addnewForm.title("Inventory System / Add new")
        title_frame = tk.Label(addnewForm,text="Add Product",an-
chor='n',font=("arial",18))
        title_frame.place(relx=0,rely=0.01,relheight=0.2,relwidth=1)

        product_name_label = tk.Label(addnewForm,text="Product Name:"
,font=("arial",25))
        product_name_label.place(relx=0.05,rely=0.2,relheight=0.1,relwidth=0.4)
        product_name_entry = tk.Entry(addnewForm,font=("arial",20),textvariable=PROD-
UCT_NAME)
        product_name_entry.place(relx=0.45,rely=0.2,relheight=0.1,relwidth=0.5)

        product_quantity_label = tk.Label(addnewForm,text="Product
Qty:",font=("arial",25))
        product_quantity_label.place(relx=0.05,rely=0.33,relheight=0.1,relwidth=0.4)
        product_quantity_entry = tk.Entry(addnewForm,font=("arial",20),textvaria-
ble=PRODUCT_QTY)

```

```

        product_quantity_entry.place(relx=0.45,relly=0.33,relheight=0.1,relwidth=0.5)

        product_price_label = tk.Label(addnewForm,text="Product
Price:",font=("arial",25))
        product_price_label.place(relx=0.05,relly=0.46,relheight=0.1,relwidth=0.4)
        product_price_entry = tk.Entry(addnewForm,font=("arial",20),textvaria-
ble=PRODUCT_PRICE)
        product_price_entry.place(relx=0.45,relly=0.46,relheight=0.1,relwidth=0.5)

        add_product_button = tk.Button(addnew-
Form,text="Save",font=("arial",18),bg="powder blue",command=self.addNew)
        add_product_button.place(relx=0.3,relly=0.7,relwidth=0.5,relheight=0.2)

    #A FUNCTION THAT IS STORING THE NEW DATA TO THE SQLITE3 DATABASE
    def addNew(self):
        database()
        cursor.execute("INSERT INTO `product` (product_name, product_qty, prod-
uct_price) VALUES(?, ?, ?)",
                        (str(PRODUCT_NAME.get()), int(PRODUCT_QTY.get()), int(PROD-
UCT_PRICE.get()))
        conn.commit()
        PRODUCT_NAME.set("")
        PRODUCT_PRICE.set("")
        PRODUCT_QTY.set("")
        cursor.close()
        conn.close()

    #FUNCTION TO REFRESH THE INVENTORY LISTBOX DATA
    def refresh(self):
        for i in self.inventory_box.get_children():
            self.inventory_box.delete(i)
        self.DisplayData()

    #FUNCTION TO SEARCH WHATS INSIDE THE INVENTORY
    def search(self):
        if self.SEARCH.get() != "":
            self.inventory_box.delete(*self.inventory_box.get_children())
            database()
            cursor.execute("SELECT * FROM `product` WHERE `product_name` LIKE ?",
('%' + str(self.SEARCH.get()) + '%',))
            fetch = cursor.fetchall()
            for data in fetch:
                self.inventory_box.insert('', 'end', values=(data))
            cursor.close()
            conn.close()

    #FUNCTION TO DELETE SOMETHING INSIDE THE INVENTORY DATA
    def delete(self):
        if not self.inventory_box.selection():
            messagebox.showinfo('Inventory System',"ERROR, Item not found")
        else:
            result = messagebox.askquestion('Inventory System', 'Are you sure you
want to delete this record?',
                                            icon="warning")

            if result == 'yes':
                curItem = self.inventory_box.focus()
                contents = (self.inventory_box.item(curItem))
                selecteditem = contents['values']

```

```
        self.inventory_box.delete(curItem)
        database()
        cursor.execute("DELETE FROM `product` WHERE `product_id` = %d" % selecteditem[0])
        conn.commit()
        cursor.close()
        conn.close()

#CALLING EACH CLASS SO THAT THE PROGRAM CAN BE RUN
root = tk.Tk()
root.title("Inventory System")
canvas = Canvas(root)
widget= Widgets(root)
title = Title(root)

root.mainloop()
```