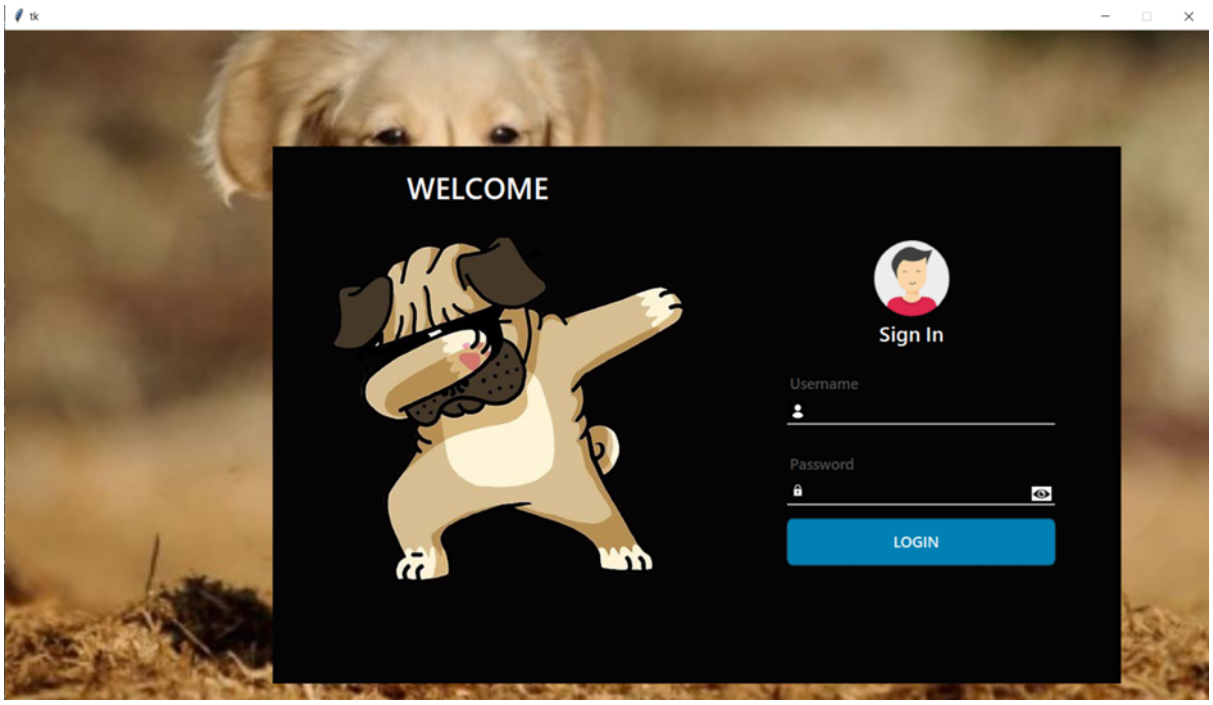
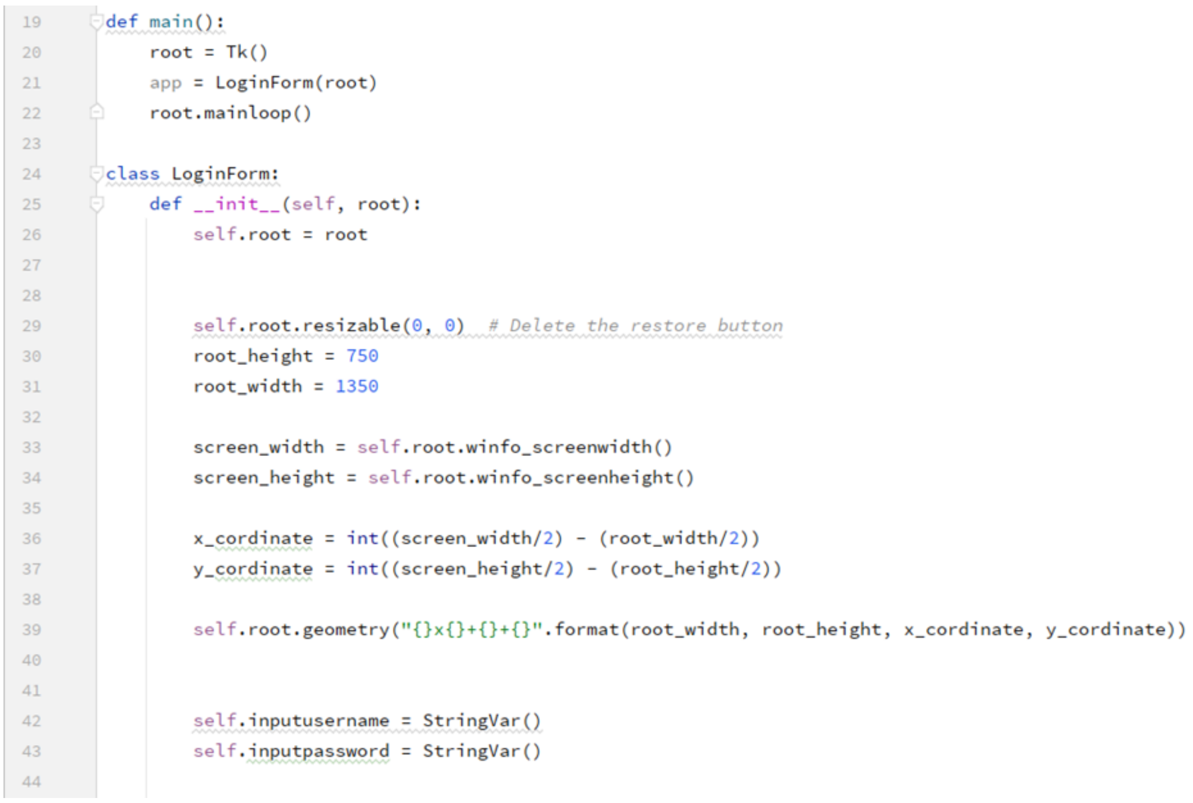
Login Page(Sulaiman(MainCoding),Kazuki(Convert format and combine))

The figure above shows that it is login page in our desktop application, we put an entry box for both username and password. For the password part it has the ability to seen and unseen the password. After filling in the username and password, click the button below the password. For now, you may have notice that there's a few images in the login page, further explanation is in the coding section.



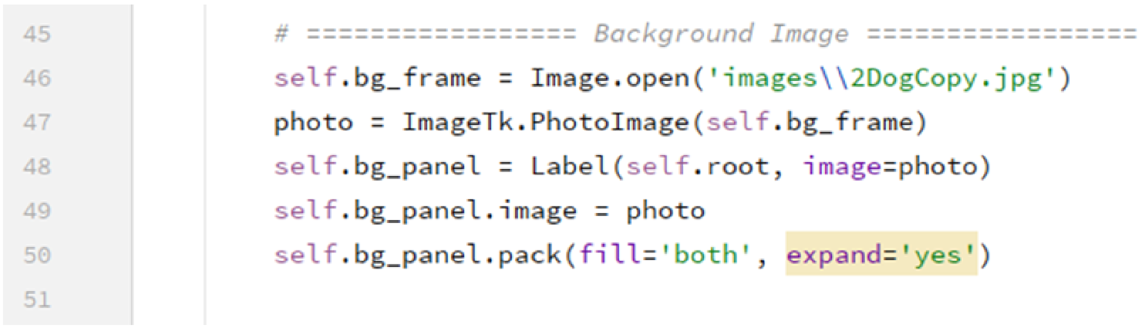
Login Page (Code)

In this login page, the design and implement of the code was contribute by me and Chan Kok Han.



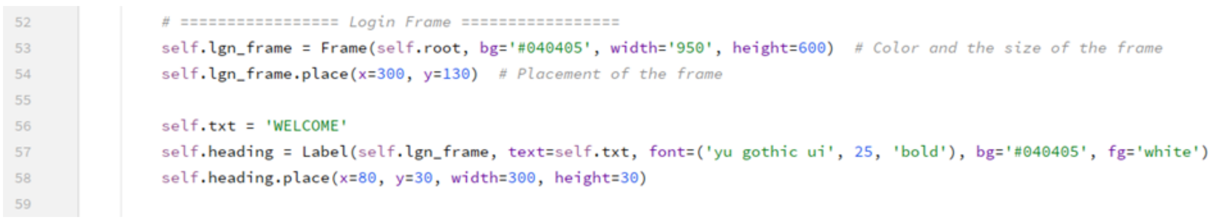
```python
def main():
    root = Tk()
    app = LoginForm(root)
    root.mainloop()

class LoginForm:
    def __init__(self, root):
        self.root = root


        self.root.resizable(0, 0)  # Delete the restore button
        root_height = 750
        root_width = 1350

        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        x_cordinate = int((screen_width/2) - (root_width/2))
        y_cordinate = int((screen_height/2) - (root_height/2))

        self.root.geometry("{}x{}+{}+{}".format(root_width, root_height, x_cordinate, y_cordinate))


        self.inputusername = StringVar()
        self.inputpassword = StringVar()

```

In this part, I create the login window and the size and placement of the window. It also has the initialization part for the input of username and password.



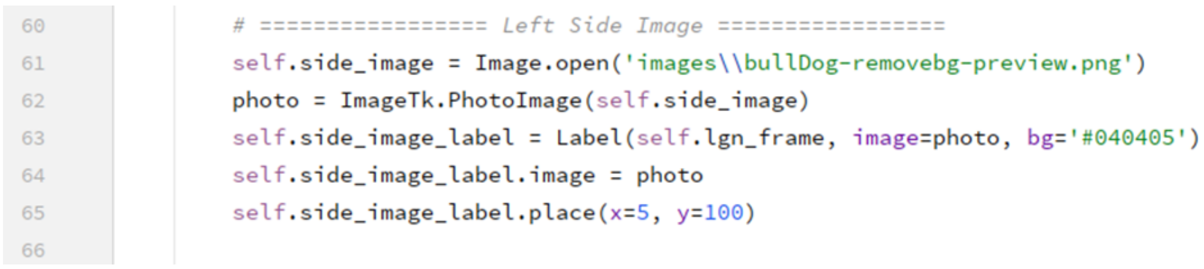
```python
        # ================= Background Image =================
        self.bg_frame = Image.open('images\\2DogCopy.jpg')
        photo = ImageTk.PhotoImage(self.bg_frame)
        self.bg_panel = Label(self.root, image=photo)
        self.bg_panel.image = photo
        self.bg_panel.pack(fill='both', expand='yes')

```

In this part, an image is chosen and use to paste the background of the login window.



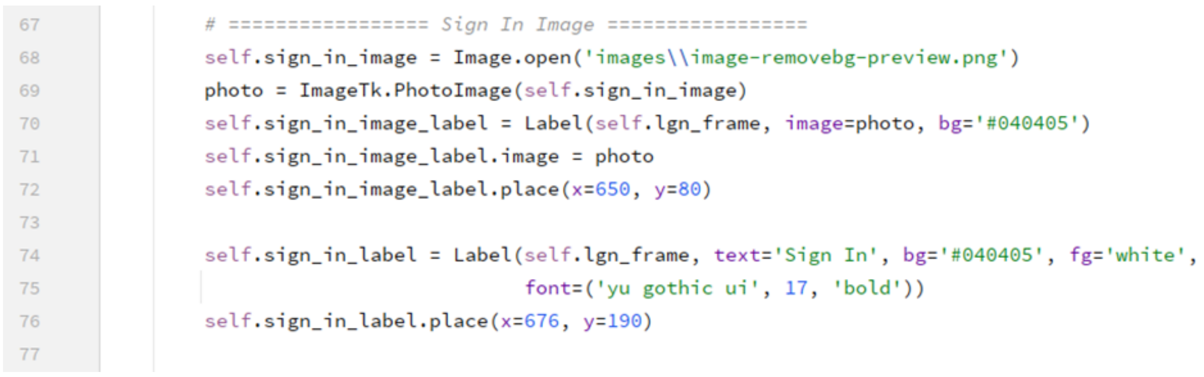
```python
# ================= Login Frame =================
self.lgn_frame = Frame(self.root, bg='#040405', width='950', height=600)  # Color and the size of the frame
self.lgn_frame.place(x=300, y=130)  # Placement of the frame

self.txt = 'WELCOME'
self.heading = Label(self.lgn_frame, text=self.txt, font=('yu gothic ui', 25, 'bold'), bg='#040405', fg='white')
self.heading.place(x=80, y=30, width=300, height=30)

```

It is the login frame; it is the black colour frame that is in the login page. In here we determine the colour and the size of the frame. And display a text that state “WELCOME” in the frame.



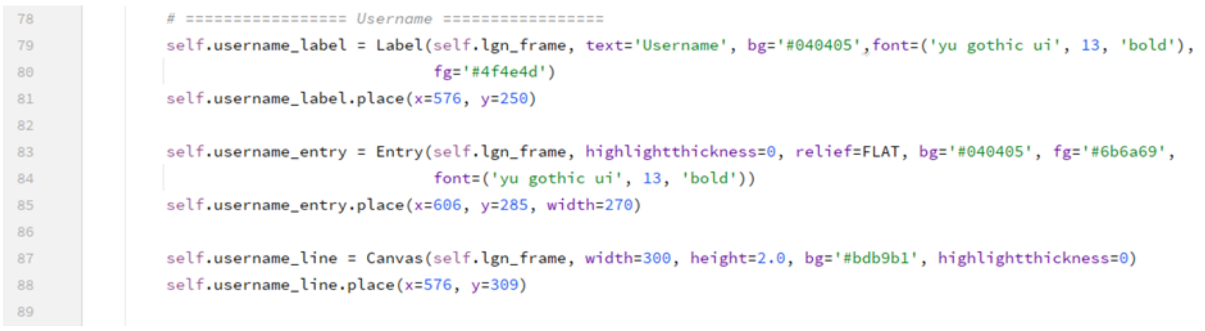
```python
# ================= Left Side Image =================
self.side_image = Image.open('images\\bullDog-removebg-preview.png')
photo = ImageTk.PhotoImage(self.side_image)
self.side_image_label = Label(self.lgn_frame, image=photo, bg='#040405')
self.side_image_label.image = photo
self.side_image_label.place(x=5, y=100)

```

It displays the bulldog that was position on the left side of the frame and change the background of the bulldog to black.



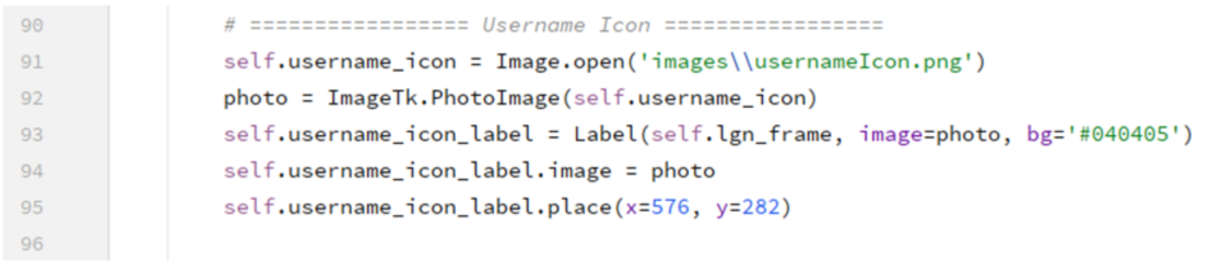
```python
# ================= Sign In Image =================
self.sign_in_image = Image.open('images\\image-removebg-preview.png')
photo = ImageTk.PhotoImage(self.sign_in_image)
self.sign_in_image_label = Label(self.lgn_frame, image=photo, bg='#040405')
self.sign_in_image_label.image = photo
self.sign_in_image_label.place(x=650, y=80)

self.sign_in_label = Label(self.lgn_frame, text='Sign In', bg='#040405', fg='white',
                           font=('yu gothic ui', 17, 'bold'))
self.sign_in_label.place(x=676, y=190)

```

insert a sign in image and it is placed above the username and password, also display text underneath the sign in image which are “Sign In”.



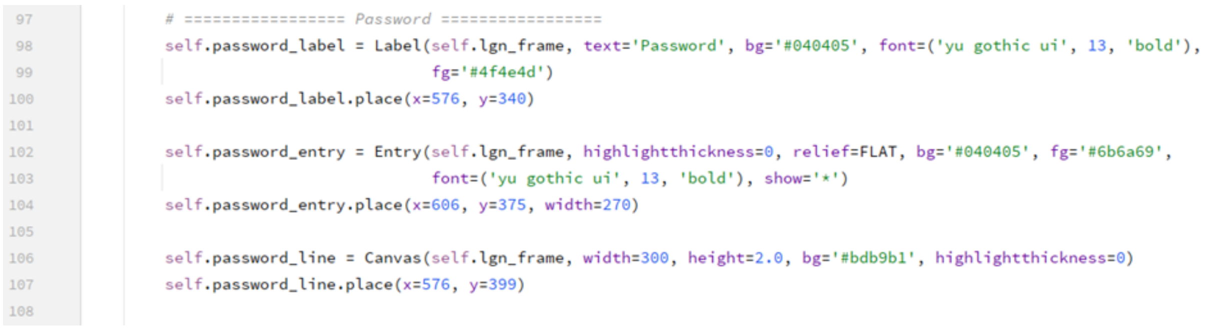
```python
# ================= Username =================
self.username_label = Label(self.lgn_frame, text='Username', bg='#040405',font=('yu gothic ui', 13, 'bold'),
                            fg='#4f4e4d')
self.username_label.place(x=576, y=250)

self.username_entry = Entry(self.lgn_frame, highlightthickness=0, relief=FLAT, bg='#040405', fg='#6b6a69',
                            font=('yu gothic ui', 13, 'bold'))
self.username_entry.place(x=606, y=285, width=270)

self.username_line = Canvas(self.lgn_frame, width=300, height=2.0, bg='#bdb9b1', highlightthickness=0)
self.username_line.place(x=576, y=309)
```

A username entry box is created and was at the right side of the frame.



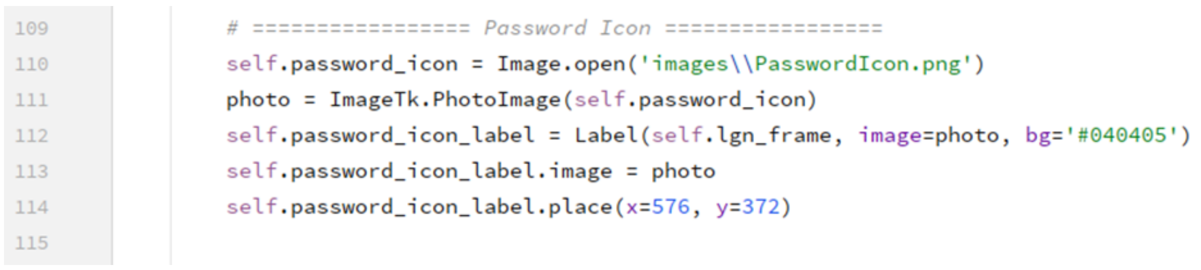
```python
# ================= Username Icon =================
self.username_icon = Image.open('images\\usernameIcon.png')
photo = ImageTk.PhotoImage(self.username_icon)
self.username_icon_label = Label(self.lgn_frame, image=photo, bg='#040405')
self.username_icon_label.image = photo
self.username_icon_label.place(x=576, y=282)
```

Insert an image for the username icon and it was placed beside the username entry box on the left.



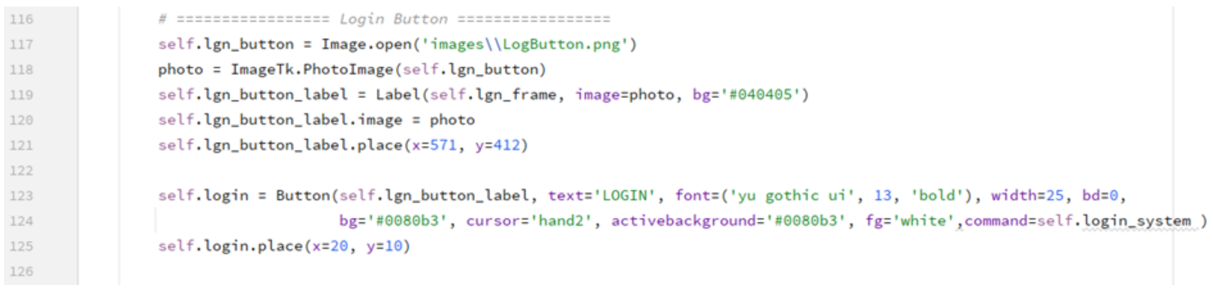
```python
# ================= Password =================
self.password_label = Label(self.lgn_frame, text='Password', bg='#040405', font=('yu gothic ui', 13, 'bold'),
                            fg='#4f4e4d')
self.password_label.place(x=576, y=340)

self.password_entry = Entry(self.lgn_frame, highlightthickness=0, relief=FLAT, bg='#040405', fg='#6b6a69',
                            font=('yu gothic ui', 13, 'bold'), show='*')
self.password_entry.place(x=606, y=375, width=270)

self.password_line = Canvas(self.lgn_frame, width=300, height=2.0, bg='#bdb9b1', highlightthickness=0)
self.password_line.place(x=576, y=399)
```

A password entry box is created and was placed on the right side of the frame and underneath the username entry box.



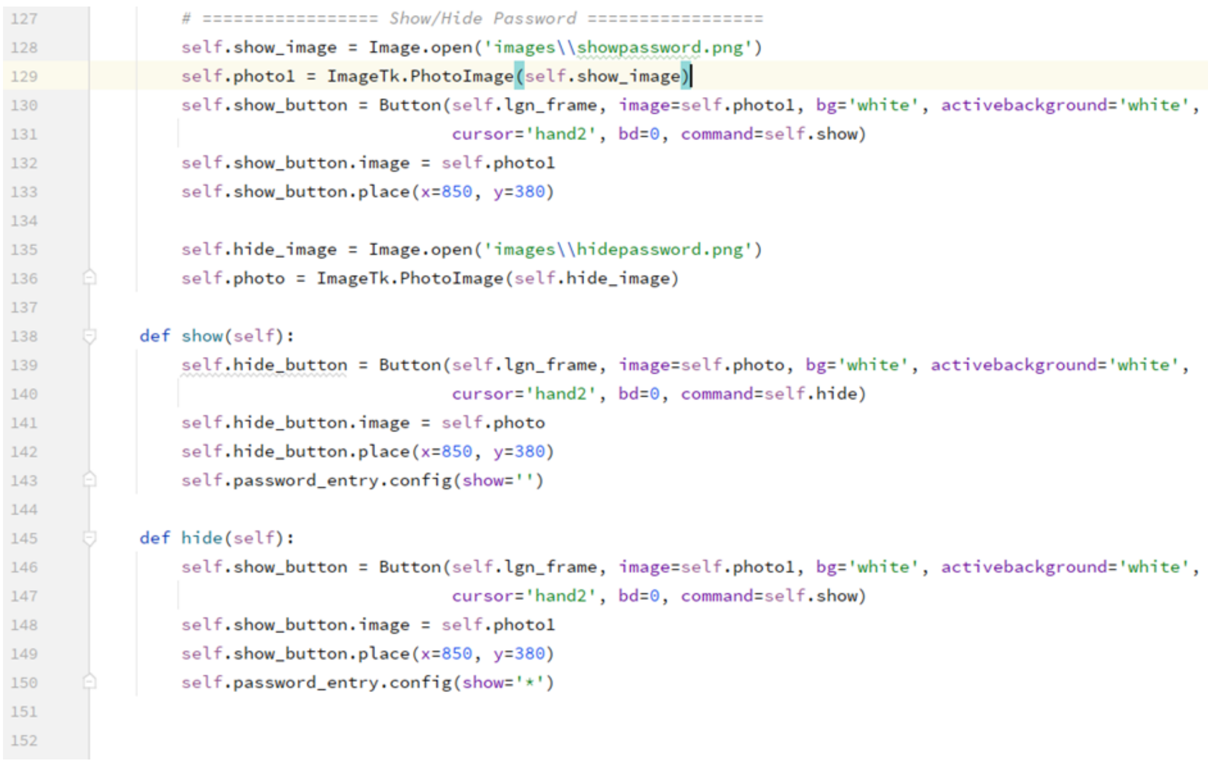
```python
# ================= Password Icon =================
self.password_icon = Image.open('images\\PasswordIcon.png')
photo = ImageTk.PhotoImage(self.password_icon)
self.password_icon_label = Label(self.lgn_frame, image=photo, bg='#040405')
self.password_icon_label.image = photo
self.password_icon_label.place(x=576, y=372)
```

Insert an image as a password icon and was placed beside the password entry box on the left.



```python
# ================= Login Button =================
self.lgn_button = Image.open('images\\LogButton.png')
photo = ImageTk.PhotoImage(self.lgn_button)
self.lgn_button_label = Label(self.lgn_frame, image=photo, bg='#040405')
self.lgn_button_label.image = photo
self.lgn_button_label.place(x=571, y=412)

self.login = Button(self.lgn_button_label, text='LOGIN', font=('yu gothic ui', 13, 'bold'), width=25, bd=0,
                    bg='#0080b3', cursor='hand2', activebackground='#0080b3', fg='white',command=self.login_system )
self.login.place(x=20, y=10)
```

Create a login button and was placed underneath the password entry box, and the button its functional and will move to the next page.

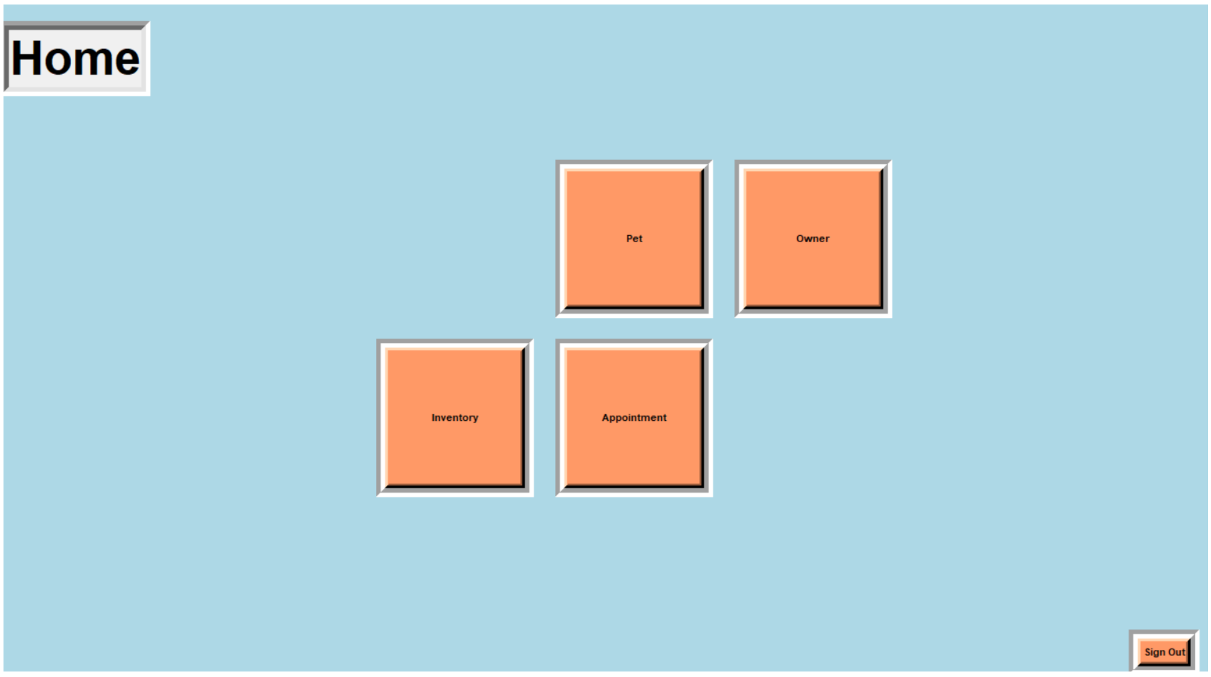


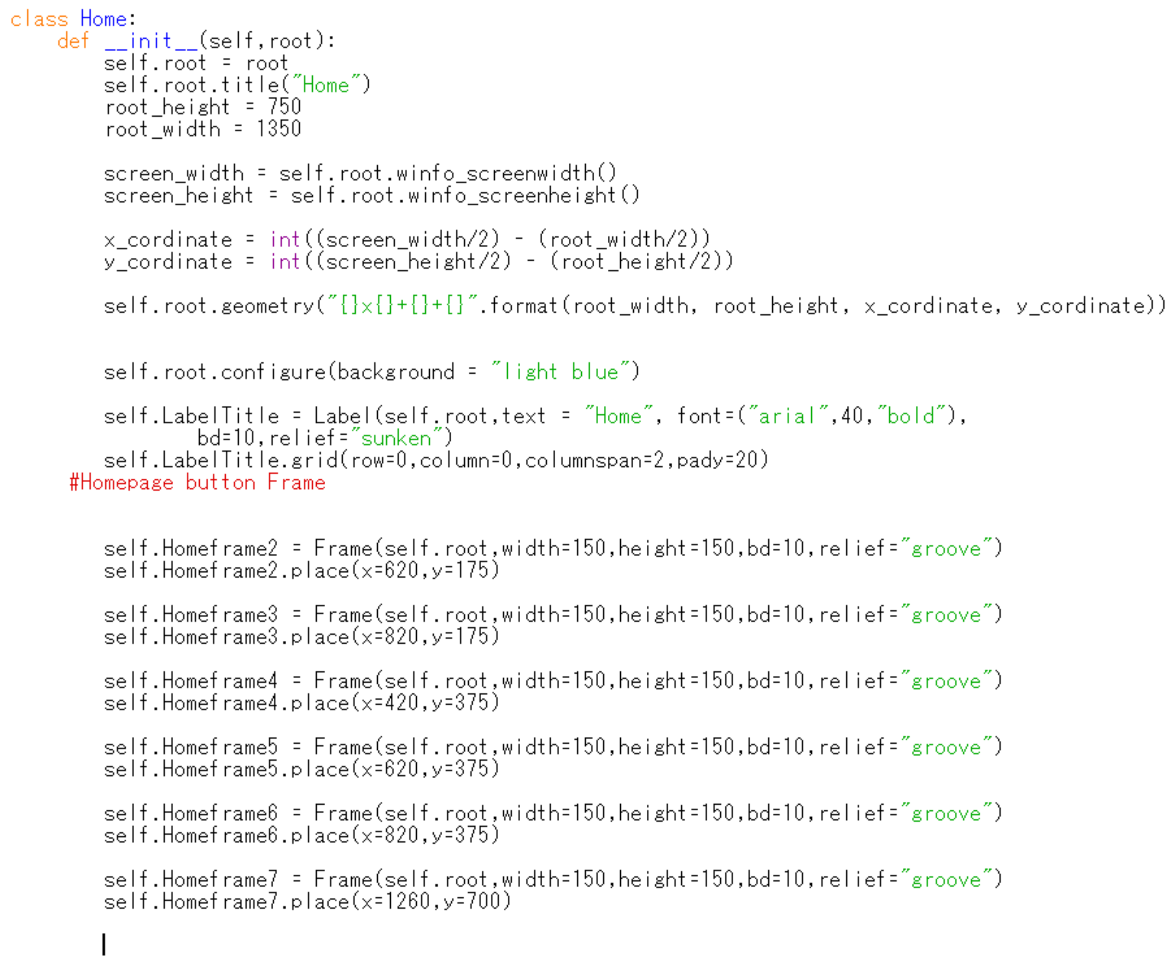
```
        # ================= Show/Hide Password =================
        self.show_image = Image.open('images\\showpassword.png')
        self.photo1 = ImageTk.PhotoImage(self.show_image)
        self.show_button = Button(self.lgn_frame, image=self.photo1, bg='white', activebackground='white',
                                  cursor='hand2', bd=0, command=self.show)
        self.show_button.image = self.photo1
        self.show_button.place(x=850, y=380)

        self.hide_image = Image.open('images\\hidepassword.png')
        self.photo = ImageTk.PhotoImage(self.hide_image)

    def show(self):
        self.hide_button = Button(self.lgn_frame, image=self.photo, bg='white', activebackground='white',
                                  cursor='hand2', bd=0, command=self.hide)
        self.hide_button.image = self.photo
        self.hide_button.place(x=850, y=380)
        self.password_entry.config(show='')

    def hide(self):
        self.show_button = Button(self.lgn_frame, image=self.photo1, bg='white', activebackground='white',
                                  cursor='hand2', bd=0, command=self.show)
        self.show_button.image = self.photo1
        self.show_button.place(x=850, y=380)
        self.password_entry.config(show='*')
```

Insert the show and hide password in the window and the function for show and hide is created.

UI System

(Chan Kok Han(InitialCoding),Kazuki(Modified and finalize the code)





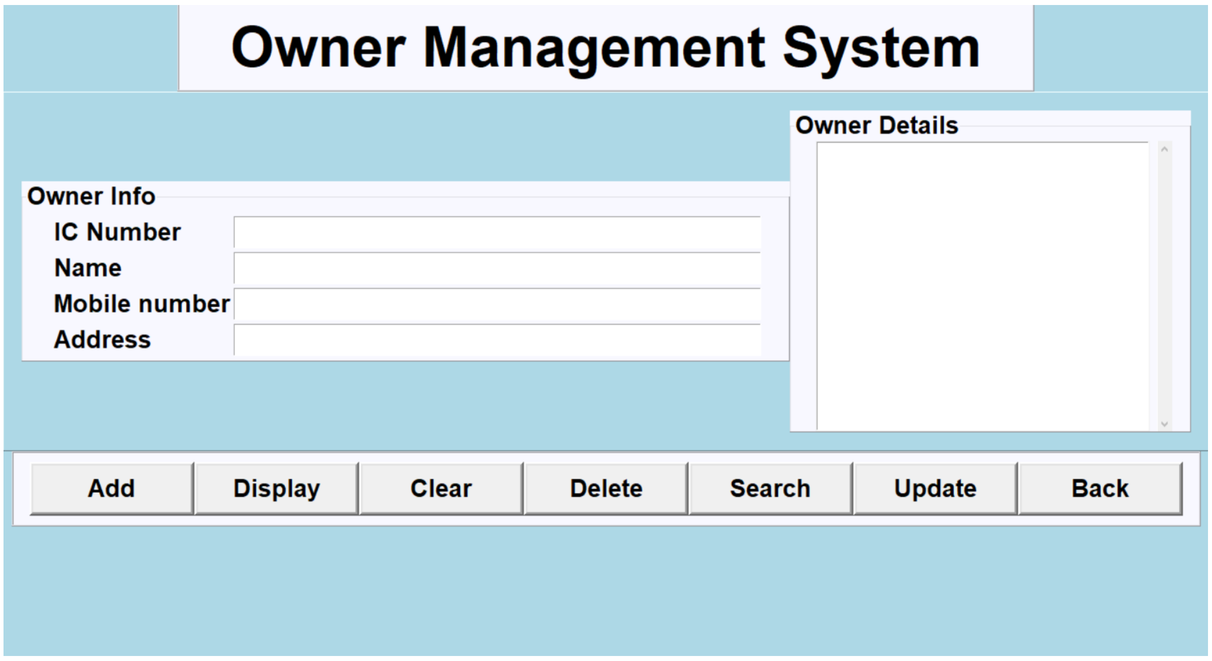
```python
class Home:
    def __init__(self,root):
        self.root = root
        self.root.title("Home")
        root_height = 750
        root_width = 1350

        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        x_cordinate = int((screen_width/2) - (root_width/2))
        y_cordinate = int((screen_height/2) - (root_height/2))

        self.root.geometry("{}x{}+{}+{}".format(root_width, root_height, x_cordinate, y_cordinate))


        self.root.configure(background = "light blue")

        self.LabelTitle = Label(self.root,text = "Home", font=("arial",40,"bold"),
                bd=10,relief="sunken")
        self.LabelTitle.grid(row=0,column=0,columnspan=2,pady=20)
     #Homepage button Frame


        self.Homeframe2 = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.Homeframe2.place(x=620,y=175)

        self.Homeframe3 = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.Homeframe3.place(x=820,y=175)

        self.Homeframe4 = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.Homeframe4.place(x=420,y=375)

        self.Homeframe5 = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.Homeframe5.place(x=620,y=375)

        self.Homeframe6 = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.Homeframe6.place(x=820,y=375)

        self.Homeframe7 = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.Homeframe7.place(x=1260,y=700)
```

First part is set the sizeof the window and set where to place the application on the window when open the application.

The Basic Home page UI are created by ‘Chan Kok Han’ and I improvise and modify it for more user friendly.

4.2.2:Owner Management page

(Kazuki(Creation of UI , MainCoding))



```
def clearData():
    self.textICNumber.delete(0,END)
    self.textname.delete(0,END)
    self.textMbNumber.delete(0,END)
    self.textaddress.delete(0,END)

def addData():
    if(len(IC_Number.get())!=0):
       RGST.addowndata(IC_Number.get(),Name.get(),Mobile_no.get(),Address.get())
       ownerlist.delete(0,END)
       ownerlist.insert(END,(IC_Number.get(),Name.get(),Mobile_no.get(),Address.get()))

def DisplayData():
    ownerlist.delete(0,END)
    for row in RGST.viewData():
        ownerlist.insert(END,row,str(""))

def OwnerRec(event):
    global sd
    searchowner = ownerlist.curselection()[0]
    sd = ownerlist.get(searchowner)

    self.textICNumber.delete(0,END)
    self.textICNumber.insert(END,sd[0])
    self.textname.delete(0,END)
    self.textname.insert(END,sd[1])
    self.textMbNumber.delete(0,END)
    self.textMbNumber.insert(END,sd[2])
    self.textaddress.delete(0,END)
    self.textaddress.insert(END,sd[3])
```

ClearData: clear the data keyin in the entry box

AddData: Add data to database

DisplayData: Recall the data added in database and display

OwnerRec: call the owner record in database to use in Search function

```python
def DeleteData():
    if(len(IC_Number.get())!=0):
        RGST.deleteData(sd[0])
        clearData()
        DisplayData()

def searchData():
    ownerlist.delete(0,END)
    for row in RGST.search(IC_Number.get(),Name.get(),Mobile_no.get(),Address.get()):
        ownerlist.insert(END,row,str(""))

def updateData():
    if(len(IC_Number.get())!=0):
       RGST.deleteData(sd[0])
    if(len(IC_Number.get())!=0):
       RGST.addowndata(IC_Number.get(),Name.get(),Mobile_no.get(),Address.get())
       ownerlist.delete(0,END)
       ownerlist.insert(IC_Number.get(),Name.get(),Mobile_no.get(),Address.get())

def backtohome():
    self.root.destroy()#Destroy window
```

DeleteData: Delete data from database

searchData: Search data from database and display

updateDate: Update data saved in database

Backtohome: Destroy(close) the current window back to homepage

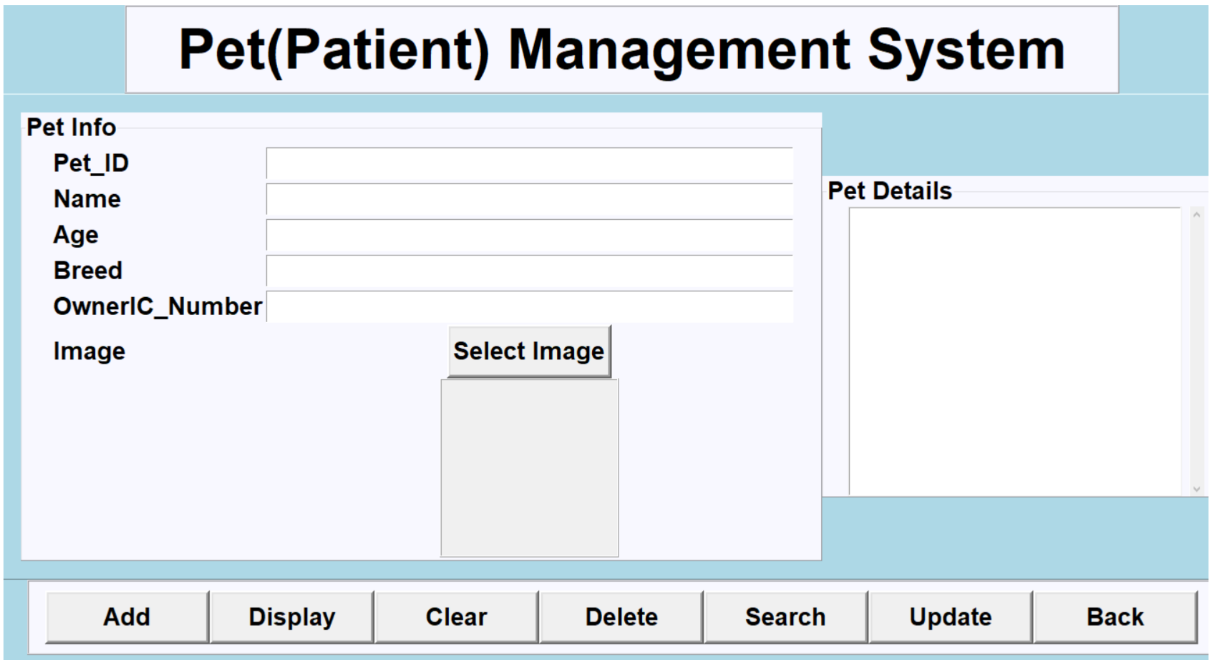
```python
scrollbar = Scrollbar(DataFrameRIGHT)
scrollbar.grid(row=0,column=1,sticky='ns')

ownerlist = Listbox(DataFrameRIGHT, width=41, height=16, font=('arial',12,'bold'),yscrollcommand=scrollbar.set)
ownerlist.bind('<<ListboxSelect>>',OwnerRec)
ownerlist.grid(row=0,column=0,padx=8)

scrollbar.config(command = ownerlist.yview)
```

Above is Coding for scrollbar in the system

UI creation, All functions inside are done by me Kazuki.

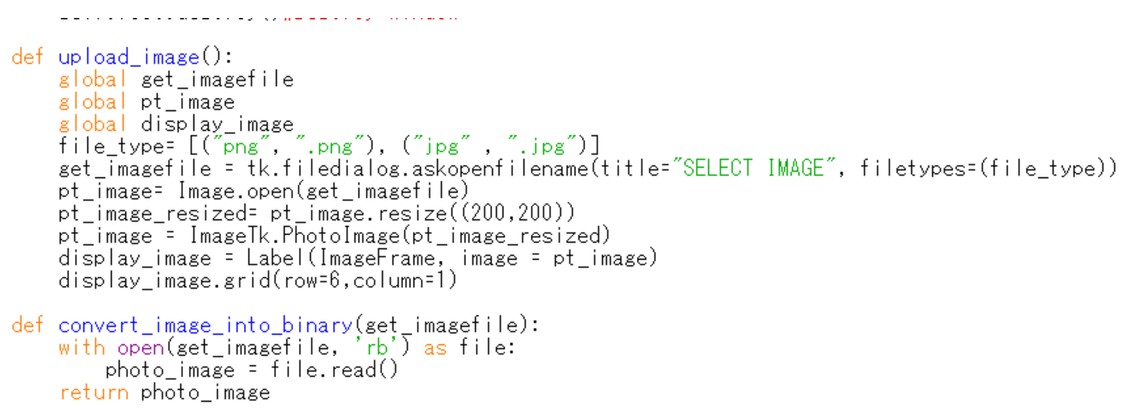
4.2.3:Pet Management Page

(Kazuki(Creation of UI , MainCoding))



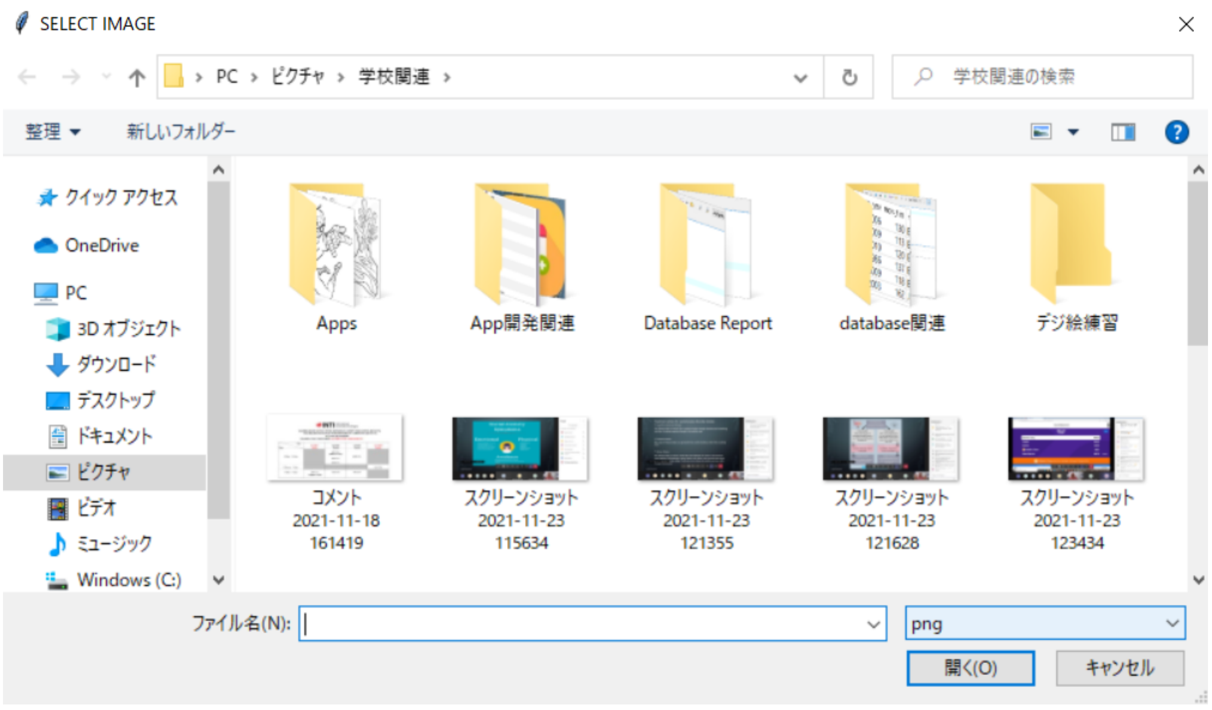
```
def clearData():
    self.textPetID.delete(0,END)
    self.textname.delete(0,END)
    self.textage.delete(0,END)
    self.textbreed.delete(0,END)
    self.textownericnumber.delete(0,END)
    self.textimage.delete(0,END)

def addData():
    if(len(Pet_ID.get())!=0):
        RGST.addpetdata(Pet_ID.get(),Name.get(),Age.get(),Breed.get(),OwnerIC_Number.get(),Image.get())
        petlist.delete(0,END)
        petlist.insert(END,(Pet_ID.get(),Name.get(),Age.get(),Breed.get(),OwnerIC_Number.get(),Image.get()))

def DisplayData():
    petlist.delete(0,END)
    for row in RGST.viewpetData():
        petlist.insert(END,row,str(""))

def PetRec(event):
    global sd
    searchpet = petlist.curselection()[0]
    sd = petlist.get(searchpet)

    self.textPetID.delete(0,END)
    self.textPetID.insert(END,sd[0])
    self.textname.delete(0,END)
    self.textname.insert(END,sd[1])
    self.textage.delete(0,END)
    self.textage.insert(END,sd[2])
    self.textbreed.delete(0,END)
    self.textbreed.insert(END,sd[3])
    self.textownericnumber.delete(0,END)
    self.textownericnumber.insert(END,sd[4])
    self.textimage.delete(0,END)
    self.textimage.insert(END,sd[5])
```

```
def DeleteData():
    if(len(Pet_ID.get())!=0):
        RGST.deletepetData(sd[0])
        clearData()
        DisplayData()

def searchData():
    petlist.delete(0,END)
    for row in RGST.searchpet(Pet_ID.get(),Name.get(),Age.get(),Breed.get(),OwnerIC_Number.get(),Image.get()):
        petlist.insert(END,row,str(""))

def updateData():
    if(len(Pet_ID.get())!=0):
       RGST.deletepetData(sd[0])
    if(len(Pet_ID.get())!=0):
       RGST.addpetdata(Pet_ID.get(),Name.get(),Age.get(),Breed.get(),OwnerIC_Number.get(),Image.get())
       petlist.delete(0,END)
       petlist.insert(Pet_ID.get(),Name.get(),Age.get(),Breed.get(),OwnerIC_Number.get(),Image.get())

def backtohome():
    self.root.destroy()#Destroy window
```



```
def upload_image():
    global get_imagefile
    global pt_image
    global display_image
    file_type= [("png", ".png"), ("jpg" , ".jpg")]
    get_imagefile = tk.filedialog.askopenfilename(title="SELECT IMAGE", filetypes=(file_type))
    pt_image= Image.open(get_imagefile)
    pt_image_resized= pt_image.resize((200,200))
    pt_image = ImageTk.PhotoImage(pt_image_resized)
    display_image = Label(ImageFrame, image = pt_image)
    display_image.grid(row=6,column=1)

def convert_image_into_binary(get_imagefile):
    with open(get_imagefile, 'rb') as file:
        photo_image = file.read()
    return photo_image
```

Def upload_image is coding for open the UI below



Def convert_image_into_binary

Is to convert image to binary to put inside the database.

Image upload coding done by chan kok han for first trial

Sulaiman takeover the work

And kazuki fixed the error and connect to database and finalize the code

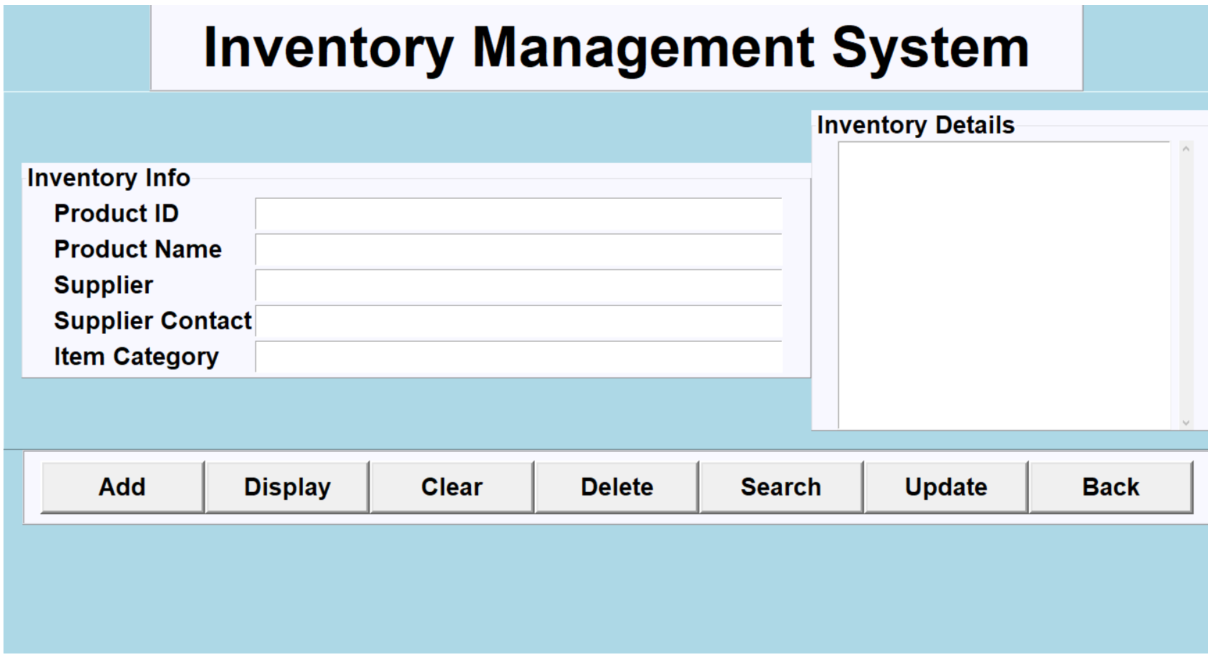
```
#======================List and Scroll=====================================-

        scrollbar = Scrollbar(DataFrameRIGHT)
        scrollbar.grid(row=0,column=1,sticky='ns')

        petlist = Listbox(DataFrameRIGHT, width=41, height=16, font=('arial',12,'bold'),yscrollcommand=scrollbar.set)
        petlist.bind('<<ListboxSelect>>',PetRec)
        petlist.grid(row=0,column=0,padx=8)

        scrollbar.config(command = petlist.yview)
```

Every person in the group is involved in the image upload part.

Code explanations are the same as above.

4.2.4:Inventory Management System

(Kazuki(Creation of UI , MainCoding))

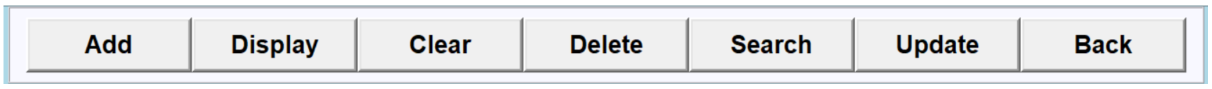


```
    def clearData():
        self.textproductid.delete(0,END)
        self.textproductname.delete(0,END)
        self.textsuppliername.delete(0,END)
        self.textsuppliernumber.delete(0,END)
        self.textcategory.delete(0,END)

    def addData():
        if(len(Product_ID.get())!=0):
           RGST.addivtdata(Product_ID.get(),Product_Name.get(),Supplier_Name.get(),Supplier_Number.get(),Category.get())
           inventorylist.delete(0,END)
           inventorylist.insert(END,(Product_ID.get(),Product_Name.get(),Supplier_Name.get(),Supplier_Number.get(),Category.get()))

    def DisplayData():
        inventorylist.delete(0,END)
        for row in RGST.viewivtData():
            inventorylist.insert(END,row,str(""))

    def InventoryRec(event):
        global sd
        searchinventory = inventorylist.curselection()[0]
        sd = inventorylist.get(searchinventory)

        self.textproductid.delete(0,END)
        self.textproductid.insert(END,sd[0])
        self.textproductname.delete(0,END)
        self.textproductname.insert(END,sd[1])
        self.textsuppliername.delete(0,END)
        self.textsuppliername.insert(END,sd[2])
        self.textsuppliernumber.delete(0,END)
        self.textsuppliernumber.insert(END,sd[3])
        self.textcategory.delete(0,END)
        self.textcategory.insert(END,sd[4])


def DeleteData():
    if(len(Product_ID.get())!=0):
        RGST.deleteivtData(sd[0])
        clearData()
        DisplayData()

def searchData():
    inventorylist.delete(0,END)
    for row in RGST.searchivt(Product_ID.get(),Product_Name.get(),Supplier_Name.get(),Supplier_Number.get(),Category.get()):
        inventorylist.insert(END,row,str(""))

def updateData():
    if(len(Product_ID.get())!=0):
       RGST.deleteivtData(sd[0])
    if(len(Product_ID.get())!=0):
       RGST.addivtdata(Product_ID.get(),Product_Name.get(),Supplier_Name.get(),Supplier_Number.get(),Category.get())
       inventorylist.delete(0,END)
       inventorylist.insert(Product_ID.get(),Product_Name.get(),Supplier_Name.get(),Supplier_Number.get(),Category.get())

def backtohome():
    self.root.destroy()#Destroy window
    '
```

```
#=======================List and Scroll=======================================-

        scrollbar = Scrollbar(DataFrameRIGHT)
        scrollbar.grid(row=0,column=1,sticky='ns')

        inventorylist = Listbox(DataFrameRIGHT, width=41, height=16, font=('arial',12,'bold'),yscrollcommand=scrollbar.set)
        inventorylist.bind('<<ListboxSelect>>',InventoryRec)
        inventorylist.grid(row=0,column=0,padx=8)

        scrollbar.config(command = inventorylist.yview)
```

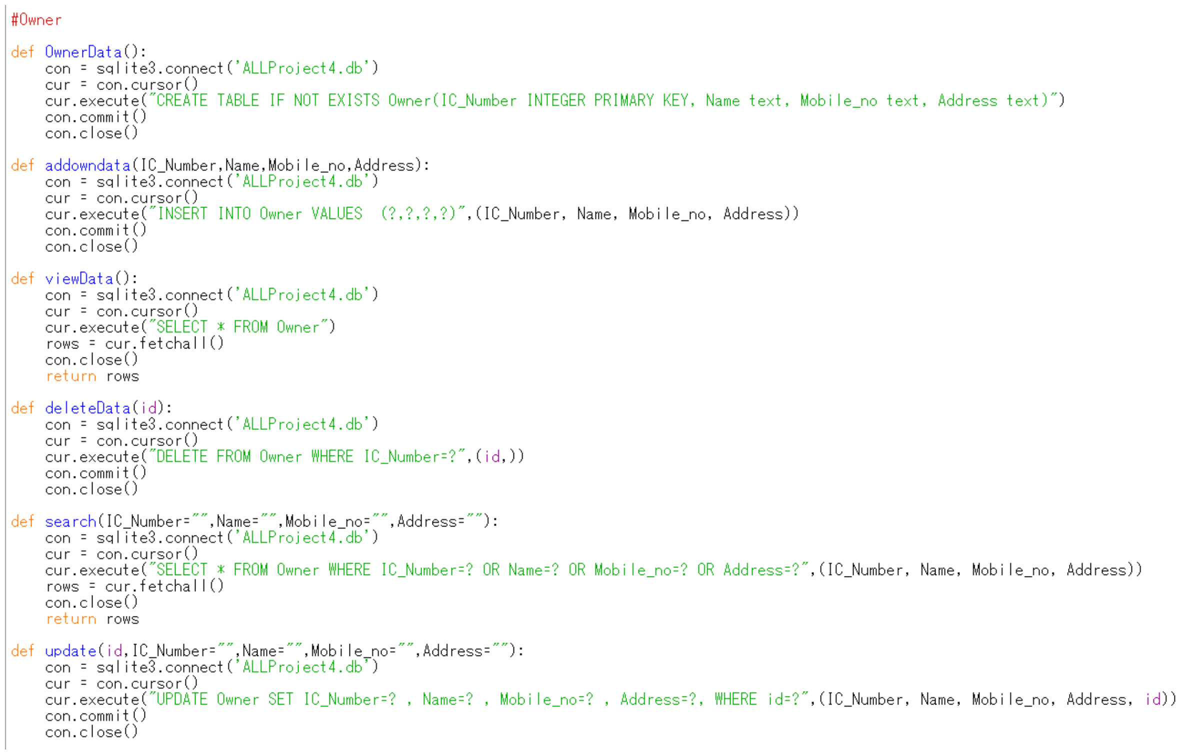
All the code are done by kazuki

4.2.5:Function that involves database



CRUD Function Button.

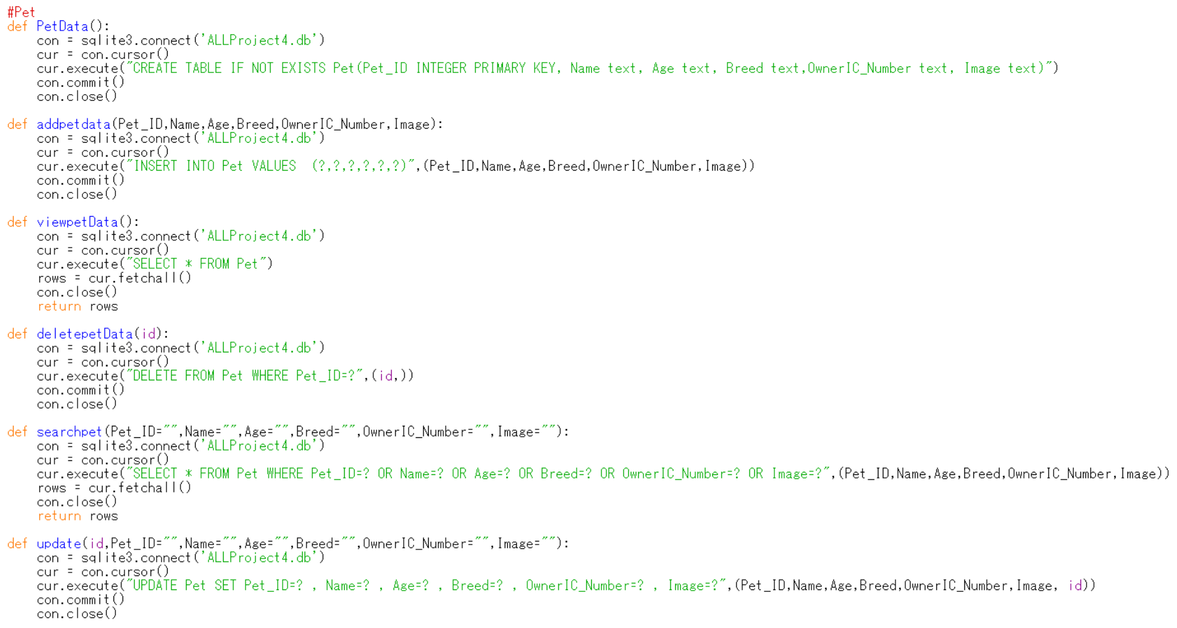
Call the function in the image below.



```
#Owner

def OwnerData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS Owner(IC_Number INTEGER PRIMARY KEY, Name text, Mobile_no text, Address text)")
    con.commit()
    con.close()

def addowndata(IC_Number,Name,Mobile_no,Address):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("INSERT INTO Owner VALUES  (?,?,?,?)",(IC_Number, Name, Mobile_no, Address))
    con.commit()
    con.close()

def viewData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Owner")
    rows = cur.fetchall()
    con.close()
    return rows

def deleteData(id):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("DELETE FROM Owner WHERE IC_Number=?",(id,))
    con.commit()
    con.close()

def search(IC_Number="",Name="",Mobile_no="",Address=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Owner WHERE IC_Number=? OR Name=? OR Mobile_no=? OR Address=?",(IC_Number, Name, Mobile_no, Address))
    rows = cur.fetchall()
    con.close()
    return rows

def update(id,IC_Number="",Name="",Mobile_no="",Address=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("UPDATE Owner SET IC_Number=? , Name=? , Mobile_no=? , Address=?, WHERE id=?",(IC_Number, Name, Mobile_no, Address, id))
    con.commit()
    con.close()
```

Code above is connected with the database and python.

These codes are the base of the CRUD functions.



```
#Pet
def PetData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS Pet(Pet_ID INTEGER PRIMARY KEY, Name text, Age text, Breed text,OwnerIC_Number text, Image text)")
    con.commit()
    con.close()

def addpetdata(Pet_ID,Name,Age,Breed,OwnerIC_Number,Image):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("INSERT INTO Pet VALUES  (?,?,?,?,?,?)",(Pet_ID,Name,Age,Breed,OwnerIC_Number,Image))
    con.commit()
    con.close()

def viewpetData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Pet")
    rows = cur.fetchall()
    con.close()
    return rows

def deletepetData(id):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("DELETE FROM Pet WHERE Pet_ID=?",(id,))
    con.commit()
    con.close()

def searchpet(Pet_ID="",Name="",Age="",Breed="",OwnerIC_Number="",Image=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Pet WHERE Pet_ID=? OR Name=? OR Age=? OR Breed=? OR OwnerIC_Number=? OR Image=?",(Pet_ID,Name,Age,Breed,OwnerIC_Number,Image))
    rows = cur.fetchall()
    con.close()
    return rows

def update(id,Pet_ID="",Name="",Age="",Breed="",OwnerIC_Number="",Image=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("UPDATE Pet SET Pet_ID=? , Name=? , Age=? , Breed=? , OwnerIC_Number=? , Image=?",(Pet_ID,Name,Age,Breed,OwnerIC_Number,Image, id))
    con.commit()
    con.close()
```

This is for the Pet

```python
#Inventory

def InventoryData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS Inventory(Product_ID INTEGER PRIMARY KEY, Product_Name text, Supplier_Name text, Supplier_Number text, Category text)")
    con.commit()
    con.close()

def addivtdata(Product_ID,Product_Name,Supplier_Name,Supplier_Number,Category):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("INSERT INTO Inventory VALUES  (?,?,?,?,?)",(Product_ID,Product_Name,Supplier_Name,Supplier_Number,Category))
    con.commit()
    con.close()

def viewivtData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Inventory")
    rows = cur.fetchall()
    con.close()
    return rows

def deleteivtData(id):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("DELETE FROM Inventory WHERE Product_ID=?",(id,))
    con.commit()
    con.close()

def searchivt(Product_ID="",Product_Name="",Supplier_Name="",Supplier_Number="",Category=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Inventory WHERE Product_ID=? OR Product_Name=? OR Supplier_Name=? OR Supplier_Number=? OR Category=?",(Product_ID,Product_Name,Supplier_Name,Supplier_Number,Category))
    rows = cur.fetchall()
    con.close()
    return rows

def updateivt(id,Product_ID="",Product_Name="",Supplier_Name="",Supplier_Number="",Category=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("UPDATE Inventory SET Product_ID=? , Product_Name=? , Supplier_Name=? , Supplier_Number=?,Category=? WHERE id=?",(Product_ID,Product_Name,Supplier_Name,Supplier_Number,Category, id))
    con.commit()
    con.close()
```

This is for the Inventory

```python
#Appointment


def AppointmentData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS Appointment(Booking_ID INTEGER PRIMARY KEY, Doctor_Name text, Booking_Date date, Booking_time time, Booking_Reason text)")
    con.commit()
    con.close()

def addapndata(Booking_ID,Doctor_Name,Booking_Date,Booking_time,Booking_Reason):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("INSERT INTO Appointment VALUES  (?,?,?,?,?)",(Booking_ID,Doctor_Name,Booking_Date,Booking_time,Booking_Reason))
    con.commit()
    con.close()

def viewapnData():
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Appointment")
    rows = cur.fetchall()
    con.close()
    return rows

def deleteapnData(id):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("DELETE FROM Appointment WHERE Booking_ID=?",(id,))
    con.commit()
    con.close()

def searchapn(Booking_ID="",Doctor_Name="",Booking_Date="",Booking_time="",Booking_Reason=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("SELECT * FROM Appointment WHERE Booking_ID=? OR Doctor_Name=? OR Booking_Date=? OR Booking_time=? OR Booking_Reason=?",(Booking_ID,Doctor_Name,Booking_Date,Booking_time,Booking_Reason))
    rows = cur.fetchall()
    con.close()
    return rows

def updateapn(id,Booking_ID="",Doctor_Name="",Booking_Date="",Booking_time="",Booking_Reason=""):
    con = sqlite3.connect('ALLProject4.db')
    cur = con.cursor()
    cur.execute("UPDATE Appointment SET Booking_ID=? , Doctor_Name=? , Booking_Date=? , Booking_time=? , Booking_Reason=? WHERE id=?",(Booking_ID,Doctor_Name,Booking_Date,Booking_time,Booking_Reason, id))
    con.commit()
    con.close()
```

This is for the Appointment.

Connections to the Database are done by Kazuki and Sulaiman.

Research for coding is done by Chan kok han.

Code for function are done by Kazuki

Data Analysis (By Chan kok han, Error fix by Kazuki)

```
class Petgraph_window:
    def __init__(self,root):
        self.root = root
        self.root.title("Analysis")
        root_height = 750
        root_width = 1350

        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        x_cordinate = int((screen_width/2) - (root_width/2))
        y_cordinate = int((screen_height/2) - (root_height/2))

        self.root.geometry("{}x{}+{}+{}".format(root_width, root_height, x_cordinate, y_cordinate))

        self.root.config(bg = "light blue")

        MainFrame = Frame(self.root, bg = "light blue")
        MainFrame.grid()

        TitleFrame = Frame(MainFrame, bd=2, padx=54, pady=8, bg = "Ghost White", relief = RIDGE)
        TitleFrame.pack(side=TOP)

        self.lblTitle = Label(TitleFrame, font = ('arial', 47 , 'bold'), text="Files", bg = "Ghost White")
        self.lblTitle.grid()

        DataFrame = Frame(MainFrame, bd=1,width=1300, height=500, padx=20, pady=20, bg = "light blue", relief = RIDGE)
        DataFrame.pack(side=BOTTOM)

        DataFrameLEFT = LabelFrame(DataFrame, bd=1,width=1300, height=500, padx=20, bg = "Ghost White", relief = RIDGE , font = ('arial', 20 , 'bold'),text="Pet Graph")
        DataFrameLEFT.pack(side=LEFT)

        self.gooutframe = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.gooutframe.place(x=1260,y=700)

        self.showframe = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.showframe.place(x=860,y=700)
```

```
def show_frame(frame):
    frame.tkraise()

def goback():
    self.root.destroy()#Destroy window

goback = Button(self.gooutframe,width=5,padx= 5,bd=5, font=("arail",8, "bold"),
            bg = "#ff9966",  text = "Go back", command =goback)#button detail
goback.pack()#place button

def showGraph():
    database = "ALLProject4.db"
    connection = sql.connect(database)
    query = '''SELECT Name, Breed FROM Pet'''
    df = pd.read_sql_query(query,connection)
    df.head()
    df['Breed'].value_counts().plot(kind='bar', figsize=(6,6))
    plt.show()
showGraph= Button(self.showframe,width=5,padx= 5,bd=5, font=("arail",8, "bold")
            bg = "#ff9966",  text = "Show", command =showGraph)#button detail
showGraph.pack()#place button
```

```
class Ownergraph_window:
    def __init__(self,root):
        self.root = root
        self.root.title("Analysis")
        root_height = 750
        root_width = 1350

        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        x_cordinate = int((screen_width/2) - (root_width/2))
        y_cordinate = int((screen_height/2) - (root_height/2))

        self.root.geometry("{}x{}+{}+{}".format(root_width, root_height, x_cordinate, y_cordinate))

        self.root.config(bg = "light blue")

        MainFrame = Frame(self.root, bg = "light blue")
        MainFrame.grid()

        TitleFrame = Frame(MainFrame, bd=2, padx=54, pady=8, bg = "Ghost White", relief = RIDGE)
        TitleFrame.pack(side=TOP)

        self.lblTitle = Label(TitleFrame, font = ('arial', 47 , 'bold'), text="Files", bg = "Ghost White")
        self.lblTitle.grid()

        DataFrame = Frame(MainFrame, bd=1,width=1300, height=500, padx=20, pady=20, bg = "light blue", relief = RIDGE)
        DataFrame.pack(side=BOTTOM)

        DataFrameLEFT = LabelFrame(DataFrame, bd=1,width=1300, height=500, padx=20, bg = "Ghost White", relief = RIDGE , font = ('arial', 20 , 'bold'),text="Owner Graph")
        DataFrameLEFT.pack(side=LEFT)

        self.gooutframe = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.gooutframe.place(x=1260,y=700)

        self.showframe = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.showframe.place(x=860,y=700)
```

```
def show_frame(frame):
    frame.tkraise()

def goback():
    self.root.destroy()#Destroy window

goback = Button(self.gooutframe,width=5,padx= 5,bd=5, font=("arail",8, "bold"),
            bg = "#ff9966",  text = "Go back", command =goback)#button detail
goback.pack()#place button

def showGraph():
    database = "ALLProject4.db"
    connection = sql.connect(database)
    query = '''SELECT Name, Address FROM Owner'''
    df = pd.read_sql_query(query,connection)
    df.head()
    df['Address'].value_counts().plot(kind='pie', figsize=(6,6))
    plt.show()
showGraph= Button(self.showframe,width=5,padx= 5,bd=5, font=("arail",8, "bold"),
            bg = "#ff9966",  text = "Show", command =showGraph)#button detail
showGraph.pack()#place button
```

```
class Inventorygraph_window:
    def __init__(self,root):
        self.root = root
        self.root.title("Analysis")
        root_height = 750
        root_width = 1350

        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()

        x_cordinate = int((screen_width/2) - (root_width/2))
        y_cordinate = int((screen_height/2) - (root_height/2))

        self.root.geometry("{}x{}+{}+{}".format(root_width, root_height, x_cordinate, y_cordinate))

        self.root.config(bg = "light blue")

        MainFrame = Frame(self.root, bg = "light blue")
        MainFrame.grid()

        TitleFrame = Frame(MainFrame, bd=2, padx=54, pady=8, bg = "Ghost White", relief = RIDGE)
        TitleFrame.pack(side=TOP)

        self.lblTitle = Label(TitleFrame, font = ('arial', 47 , 'bold'), text="Files", bg = "Ghost White")
        self.lblTitle.grid()

        DataFrame = Frame(MainFrame, bd=1,width=1300, height=500, padx=20, pady=20, bg = "light blue", relief = RIDGE)
        DataFrame.pack(side=BOTTOM)

        DataFrameLEFT = LabelFrame(DataFrame, bd=1,width=1300, height=500, padx=20, bg = "Ghost White", relief = RIDGE , font = ('arial', 20 , 'bold'),text="Inventory Graph")
        DataFrameLEFT.pack(side=LEFT)

        self.gooutframe = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.gooutframe.place(x=1260,y=700)

        self.showframe = Frame(self.root,width=150,height=150,bd=10,relief="groove")
        self.showframe.place(x=860,y=700)
```

```
def show_frame(frame):
    frame.tkraise()

def goback():
    self.root.destroy()#Destroy window

goback = Button(self.gooutframe,width=5,padx= 5,bd=5, font=("arail",8, "bold"),
            bg = "#ff9966",  text = "Go back", command =goback)#button detail
goback.pack()#place button

def showGraph():
    database = "ALLProject4.db"
    connection = sql.connect(database)
    query = '''SELECT Product_Name, Category FROM Inventory'''
    df = pd.read_sql_query(query,connection)
    df.head()
    df['Category'].value_counts().plot(kind='pie', figsize=(6,6))
    plt.show()
showGraph= Button(self.showframe,width=5,padx= 5,bd=5, font=("arail",8, "bold"),
            bg = "#ff9966",  text = "Show", command =showGraph)#button detail
showGraph.pack()#place button
```

Connection: connect to database from sqlitestudio
Query: It selects particular variables from the table
Matplotlib inline: used to plot the variables
Value_counts:count the number of particular value or words
kind: it creates graph and charts such bar and pie
plt.show: shows the graph visualisation
Figsize: the size of the pie chart

Errors inside the system fixed by Kazuki.

Code for function are done by Kazuki and Kok Han.

Research for coding is done by Kazuki and Chan kok han.