MULTI REVIEWING SYSTEM

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

THALLADA AJAY KUMAR 1602-20-737-122

NUVVULA RAHUL DATTA 1602-20-737-154

BANDARU CHANDRA KIRAN 1602-20-737-129

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY CANDIDATE

We, **THALLADA AJAY KUMAR, NUVVULA RAHUL DATTA ,** and **BANDARU CHANDRA KIRAN ,** bearing hall ticket numbers, **1602-20-737-122, 1602-20-737-154, 1602-20-737-129** hereby declare that the project and report entitled **"MULTI REVIEWING SYSTEM "**, is submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report has not been submitted to any other university or institute for the award of any other degree or diploma.

THALLADA AJAY KUMAR 1602-20-737-122

NUVVULA RAHUL DATTA 1602-20-737-154

BANDARU CHANDRA KIRAN 1602-20-737-129

(Faculty In-Charge) (Head, Deptof IT)

ACKNOWLEDGEMENT

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators C.SIREESHA and N . DAVID RAJU , for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.

ABSTRACT

A lot of people have a desire to be unique at everything . But few of them don’t have complete knowledge about that particular topic in which they want to be perfect . For Example people want to look nice for parties , other occasions which they attend . But they don’t know which wardrobe suits and make them unique . Similarly when they want to buy any newly launching products they don’t know what are the features does the product contain so that it will meet our needs . And few people want a review on the art work they have done .

The Main objective of this reviewing system is to get a review for wardrobe , Art and design and any of newly launching electronic products from an expert who have knowledge about that product.

For this the user just needs to login and upload the image he want to getreview . In the same way reviewer will also logins in and gives his review (opinion) the system analyses the review into positive , negative and neutral reviews . Again When user ask for result it displays pie chart representation of the review result . .

TABLE OF CONTENTS:

INTRODUCTION

The main objective of this project is to design reviewing system using Python. Python has a vast number of built-in libraries . Python is a super cool language we can use that built in libraries directly by importing them .because it provides the usability to write cross platform code and we can use already existing code . The main objective of the project is to allow users to get review on wardrobe , art and design , newly launching electronic products. To be engaging for users, the application has to have a simple but beautiful interface.

The diverse application of the Python Language is a result of the combination of the features which gives this language an edge over others. Python has a clean object oriented design and provides enhanced process control helped us throughout the coding process.

PURPOSE

In day to day life every person want to get and be the best in every thing . Suppose a person want to buy a laptop but he don't know the features it should have so that it will be useful for his need . So , he need an advice or review on the laptop he wanted to buy . Similarly , few people don't know the perfect pair they can wear for the parties . so that they look unique and more attractive in fests.

Our reviewing system help to overcome the above limitations and this is the sole purpose for the idea behind this project.

In our reviewing system user will upload the image on which he want a review , then reviewers will download that image and they mention their reviews . By analysing all the reviews the system will display the result to the user in form of pie chart . when the user asked for the result of image he uploaded for review .

TECHNOLOGY

To implement any project successfully, there will be technological requirements which can either be software or hardware requirements .

Software Requirements

Since our project was supposed to be based on the Python programming language, it is a bare necessity to have the knowledge over syntax of the language and a proper interpreter and text editor to run and write the programs .

- Data base : Sqlite3 package
- GUI: Tkinter package
- Manipulating image files : PIL package
- Stop words , Sentiment analysis – Sentiment analysis , NLP package
- Plotting pie chart – Matplotib package

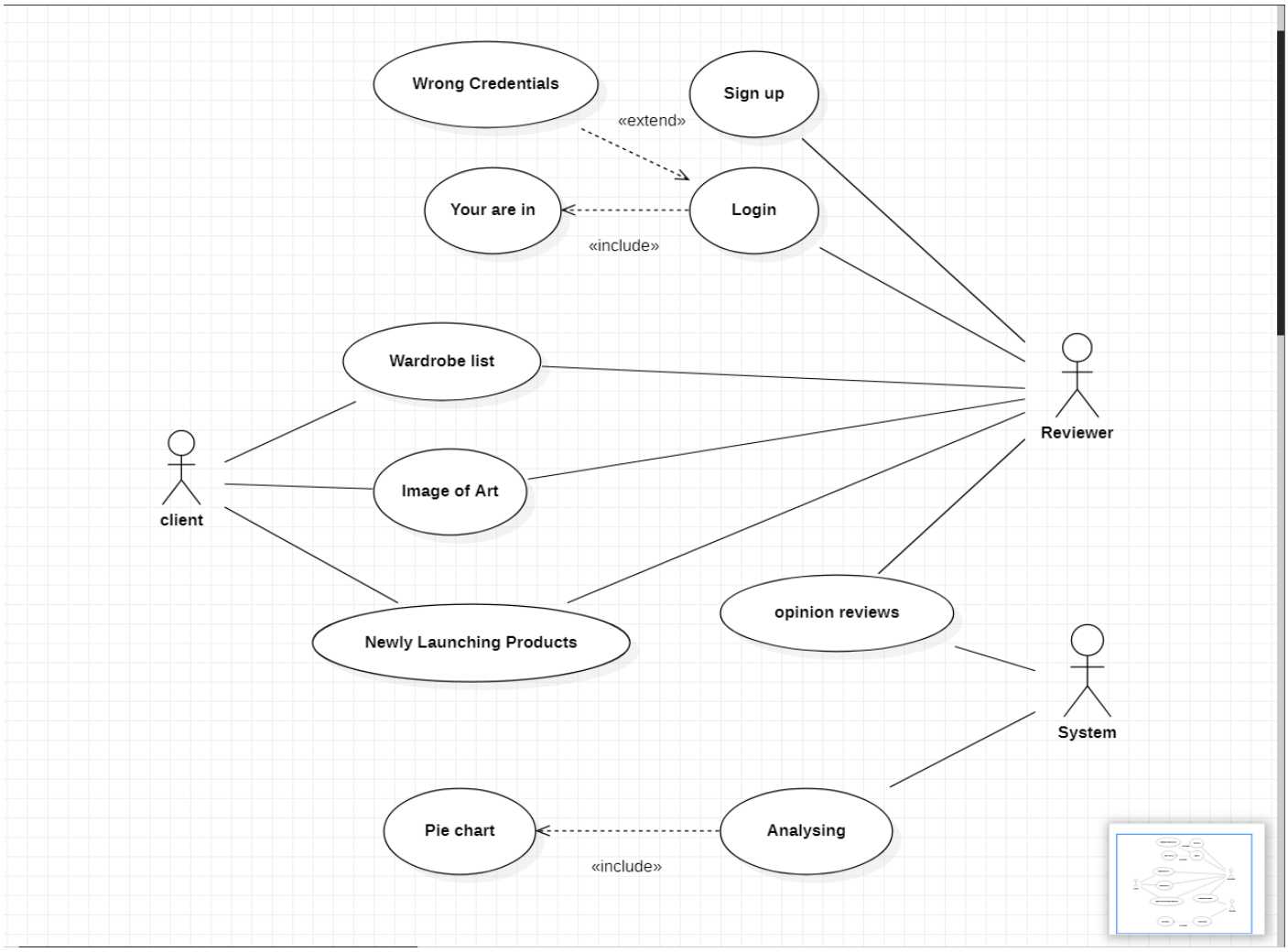
Hardware Requirements

The hardware requirements are quite low and there is no specific hardware required to run this program

- Operating System: Windows 9 or above
- RAM: 4GB and above

PROPOSED WORK

USE CASE DIAGRAM



USE CASES

There are 7 Use Cases:

1. UC01 – Register
2. UC02 – Login
3. UC03 – upload image
4. UC04 – download image
5. UC05 – reviewing
6. UC06 – Analysis
7. UC07 – Fetch result

1. Use Case ID: UC01

   Name: Register

   Actors: User , System

   Description: Allows new user/reviewer to register for a profile

   Pre-conditions: None

   Post-conditions: A profile is created for the user

2. Use Case ID: UC02

   Name: Login

   Actors: User , System

   Description: Allows registered user/reviewer to login

   Pre-conditions: User should be registered with the system

   Post-conditions: User logs in and all the options are displayed on the screen

3. Use Case ID: UC03

   Name: uploading image

   Actors: User , System

   Description: Allows registered user to upload image of wardrobe/art/products .

   Pre-conditions: User should be registered with the system

   Post-conditions: image will be added into database.

4. Use Case ID: UC04

   Name: download image

   Actors: Reviewer , System

   Description: Allows registered reviewer to download image

   Pre-conditions: Reviewer should be registered with the system

   Post-conditions: image will be downloaded into specified location

5. Use Case ID: UC05

   Name: reviewing

   Actors: Reviewer , System

   Description: Allows registered reviewer reviews the wardrobe/art/product

   Pre-conditions: User should be registered with the system

   Post-conditions: Review will be saved

6. Use Case ID: UC06

   Name: Analysis

   Actors: System

   Description: system analyses the review entered by the reviewer

   Pre-conditions: Reviewer should write a review

   Post-conditions: The analysed result saved in database .

7. Use Case ID: UC07

   Name: Fetch result

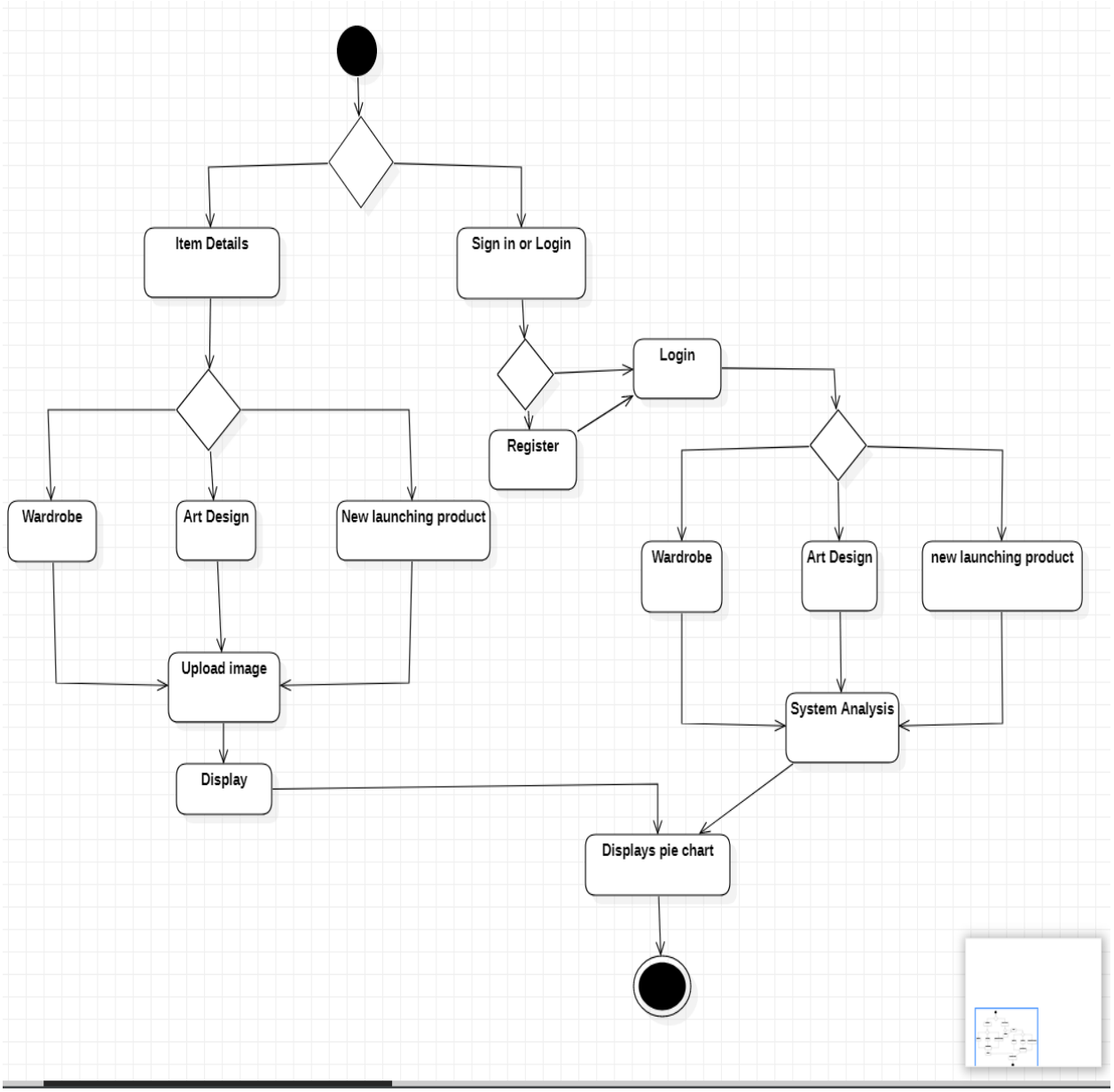
   Actors: User , System

   Description: System displays the result in form of pie chart .

   Pre-conditions: User enters image name and ask for result

   Post-conditions: displays the result

DESIGN (ACTIVITY DIAGRAM)



b. IMPLEMENTATION

```
from tkinter import *
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import tkinter as tk
from PIL import ImageTk,Image
import sqlite3
import re
from nltk.corpus import stopwords
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.stem import PorterStemmer
from tkinter import ttk
import matplotlib.pyplot as plt
root=Tk()
root.title("Reviewing System")
root.iconbitmap()
root.geometry("200x200")
global ward
ward=0
#create a database or connect to one
conn=sqlite3.connect('Details.db')

#create cursor
c=conn.cursor()

root.geometry("500x600")
frame = Frame(root,bg='red')
frame.grid(row=20,column=20)

msg = Label(text = "MULTI REVIEWING SYSTEM",font = ("Algerian",30),bg =
'LightSkyBlue2')
msg.place(x=500,y=10)

#function for displaying result
def final_ward():
    global resulter
    resulter=Tk()
    resulter.title("Fetch Result")
    resulter.iconbitmap()
    resulter.geometry("400x250")
    global res_label
```

```
    global res_box
    res_label=Label(resulter,text="Enter Name ")
    res_label.place(relx=0.3,rely=0.4)
    res_box=Entry(resulter,width=30)
    res_box.place(relx=0.5,rely=0.4)

res_btn=Button(resulter,text="Fetch",command=final_searchward,fg="Green",activeba
ckground = "black")
    res_btn.place(relx=0.45,rely=0.85)
def create_charts_ward():
    pier= tk.Tk()
    canvas1 = tk.Canvas(pier, width = 100, height = 40)
    canvas1.pack()
    pier.title("Wardrobe result pie chart")
    label1 = tk.Label(pier, text='Graphical User Interface')
    label1.config(font=('Arial', 20))
    global x1
    global x2
    global x3
    global bar1
    global pie2
    x1 = float(res_zero)
    x2 = float(res_one)
    x3 = float(res_two)

    figure2 = Figure(figsize=(4,3), dpi=100)
    subplot2 = figure2.add_subplot(111)
    labels2 = 'Negative', 'Positive', 'Neutral'
    pieSizes = [float(x1),float(x2),float(x3)]
    my_colors2 = ['lightblue','lightsteelblue','silver']
    explode2 = (0, 0.1, 0)
    subplot2.pie(pieSizes, colors=my_colors2, explode=explode2, labels=labels2,
autopct='%1.1f%%', shadow=True, startangle=90)
    subplot2.axis('equal')
    pie2 = FigureCanvasTkAgg(figure2, pier)
    pie2.get_tk_widget().pack()
    pier.mainloop()

def clear_charts():
    bar1.get_tk_widget().pack_forget()
    pie2.get_tk_widget().pack_forget()


def final_searchward():
    global res_res
```

```
    res_res=res_box.get()
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    query_res= f"SELECT * FROM Wardrobe WHERE name='{res_res}';"
    c.execute(query_res)
    res_record=c.fetchone()
    global res_zero
    global res_one
    global res_two
    res_zero=res_record[4]
    res_one=res_record[3]
    res_two=res_record[5]
    slices=[res_two,res_one,res_zero]#the final neutral,positive,negative value shoud be
passed here
    outputs=['negative','positive','neutral']
    cols=['c','m','b']

plt.pie(slices,labels=outputs,colors=cols,startangle=90,shadow=True,explode=(0,0.1,0),
autopct='%1.1f%%')
    plt.title("Final result")
    plt.show()
    create_charts_ward()
def prop(n):
    return 360.0 * n / 1000


#function for uploading image into db for client side
def ward_upload():
    global ward_editor_upload
    ward_editor_upload=Tk()
    ward_editor_upload.title("To Upload Wardrobe Image")
    ward_editor_upload.iconbitmap()
    ward_editor_upload.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global val_label_1
    global val_box_1
    global name_label_1
    global name_box_1
    global image_address_1
    name_label_1=Label(ward_editor_upload,text="Enter name :")
    name_label_1.grid(row=1,column=3)
```

```
    name_box_1=Entry(ward_editor_upload,width=30)
    name_box_1.grid(row=1,column=5)
    val_label_1=Label(ward_editor_upload,text="Enter location image present:")
    val_label_1.grid(row=3,column=3)
    val_box_1=Entry(ward_editor_upload,width=30)
    val_box_1.grid(row=3,column=5)
    image_address_1=val_box_1.get()
    print(image_address_1)
    '''
    #image_name_1=name_box_1.get("1.0","end-1c")
    ttk.Label(ward_editor_upload, text="Enter your Review :",
              font=("Times New Roman", 15)).place(relx=0.15,rely=0.75)
    # Text Widget
    global tl
    tl = Text(ward_editor_upload, width=100, height=6)
    tl.place(relx=0.3,rely=0.7)
    tl.focus()'''

upld_btn=Button(ward_editor_upload,text="UPLOAD",command=retrieve_input_1,fg=
"Green",activebackground = "black")
    upld_btn.grid(row=5,column=4)
#function to retrive input from client
def retrieve_input_1():
    #image_address_1=tl.get("1.0","end-1c")
    global image_name_1
    global image_address_1
    image_name_1=name_box_1.get()
    image_address_1=val_box_1.get()
    print(image_name_1)
    print(image_address_1)
    upld_2_btn=Button(ward_editor_upload,text="Confirm UPLOAD
",command=insertWard(image_name_1,image_address_1),fg="Green",activebackgroun
d = "black")
    upld_2_btn.grid(row=6,column=5)
# Function for Convert Binary Data
# to Human Readable Format
def convertToBinaryData(filename):

    # Convert binary format to images
    # or files data
    with open(filename, 'rb') as file:
        blobData = file.read()
    return blobData
```

```
def insertWard(name, photo):
    try:

        # Using connect method for establishing
        # a connection
        sqliteConnection = sqlite3.connect('Details.db')
        cursor = sqliteConnection.cursor()
        print("Connected to SQLite")

        # insert query
        sqlite_insert_blob_query = """ INSERT INTO Wardrobe
                            (name,image,ones,zeros,twos) VALUES (?,?,0,0,0)"""

        # Converting human readable file into
        # binary data
        empPhoto = convertToBinaryData(photo)

        # Convert data into tuple format
        data_tuple = (name, empPhoto)

        # using cursor object executing our query
        cursor.execute(sqlite_insert_blob_query, data_tuple)
        sqliteConnection.commit()
        print("Image and file inserted successfully as a BLOB into a table")
        global success_label
        success_label=Label(ward_editor_upload,text="Image and file inserted
successfully ..")
        success_label.grid(row=9,column=7)
        cursor.close()

    except sqlite3.Error as error:
        print("Failed to insert blob data into sqlite table", error)
        global fail_label
        fail_label=Label(ward_editor_upload,text="Failed to insert")
        fail_label.grid(row=9,column=7)
    finally:
        if sqliteConnection:
            sqliteConnection.close()
            print("the sqlite connection is closed")
#insertBLOB("Smith", "D:\Internship Tasks\GFG\images\One.png")

#function for reviewer side wardrobe
def ward_list_items():
    global ward_editor
    ward_editor=Tk()
```

```
    ward_editor.title("To Review Wardrobe List")
    ward_editor.iconbitmap()
    ward_editor.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global n

    global val_label
    global val_box
    val_label=Label(ward_editor,text="Enter location where you want to download:")
    val_label.place(relx=0.15,rely=0.35)
    val_box=Entry(ward_editor,width=30)
    val_box.place(relx=0.35,rely=0.35)
    n=val_box.get()

down_btn=Button(ward_editor,text="DOWNLOAD",command=ward_list,fg="Green",
activebackground = "black")
    down_btn.place(relx=0.45,rely=0.4)

def ward_list():
    try:

        # Using connect method for establishing
        # a connection
        con = sqlite3.connect('Details.db')
        cursor = con.cursor()
        print("Connected Successfully")

        query2=f"SELECT ward_image from Login_details WHERE userid='{username}'
AND passcode ='{password}';"

        cursor.execute(query2)
        global last_image_ward
        global ward_oid
        last_image_ward=cursor.fetchone()
        print(last_image_ward)
        global last_ward
        last_ward=int(last_image_ward[0])
        last_ward+=1
        # Search from table query
        query = f"SELECT * FROM Wardrobe WHERE oid={last_ward}"

        # using cursor object executing our query
```

```
    cursor.execute(query)

    # fectching all records from cursor object
    records = cursor.fetchall()
    ward_oid=last_ward
    # using for loop retrieving one by one
    # rows or data

    for row in records:

        # storing row[0] in name variable
        name = row[0]
        #print(row)
        #present_ones=row[3]
        # printing name variable
        print("Image Name = ", name)

        # storing image (currently in binary format)
        image = row[1]

        # calling above convert_data() for converting
        # binary data to human readable
        convert_data(image, n + name + ".png")
        print("Yeah!! We have successfully retrieved values from database")

        # If we don't have any records in our database,
        # then print this
        if len(records) == 0:
            print("Sorry! Please Insert some data before reading from the database.")


    # print exception if found any during program
# is running
except sqlite3.Error as error:
    print(format(error))

# using finally, closing the connection
# (con) object
finally:
    if con:
        con.close()
        print("SQLite connection is closed")

#ward_list_items()
global ward_num
```

```
    limit=0
    ward_num=0
    global val1_label
    val1_label=Label(ward_editor,text="IMAGE "+name)
    val1_label.place(relx=0.3,rely=0.6)
    ttk.Label(ward_editor, text="Enter your Review :",
              font=("Times New Roman", 15)).place(relx=0.15,rely=0.75)
    # Text Widget
    global t
    t = Text(ward_editor, width=100, height=6)
    t.place(relx=0.3,rely=0.7)
    t.focus()

down_btn=Button(ward_editor,text="Submit",command=lambda:retrieve_input(),fg="
Green",activebackground = "black")
    down_btn.place(relx=0.45,rely=0.85)
#command=lambda: retrieve_input()

def retrieve_input():
    #print("hello")
    n=val_box.get()
    #print(n)
    inputValue=t.get("1.0","end-1c")
    result=senti(inputValue)
    global final_ward_label

    final_ward_label=Label(ward_editor,text="Review Has been saved , click on close ")
    final_ward_label.place(relx=0.5,rely=0.9)
    #we need to add data after creating wardrobe table
    t.delete(1.0,END)
    if(result==1):
        #present_ones+=1
        conn=sqlite3.connect('Details.db')

        #create cursor
        c=conn.cursor()
        query1= f"SELECT ones FROM Wardrobe WHERE oid={ward_oid}"
        c.execute(query1)
        present_ones=c.fetchone()
        print(present_ones)
        present_one=int(present_ones[0])
        present_one+=1
        print(present_one)
        c.execute(f"""UPDATE Wardrobe SET
            ones=:ones
```

```
        WHERE oid={ward_oid}""",
                {
                    'ones':present_one
                }
                )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
elif(result==0):
    #present_ones+=1
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    query1= f"SELECT zeros FROM Wardrobe WHERE oid={ward_oid}"
    c.execute(query1)
    global present_zeros
    present_zeros=c.fetchone()
    print(present_zeros)
    present_zero=int(present_zeros[0])
    present_zero+=1
    print(present_zero)
    c.execute(f"""UPDATE Wardrobe SET
        zeros=:zeros

        WHERE oid={ward_oid}""",
                {
                    'zeros':present_zero
                }
                )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
elif(result==2):
    #present_ones+=1
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    query1= f"SELECT twos FROM Wardrobe WHERE oid={ward_oid}"
    c.execute(query1)
```

```
        global present_twos
        present_twos=c.fetchone()
        print(present_twos)
        present_two=int(present_twos[0])
        present_two+=1
        print(present_two)
        c.execute(f"""UPDATE Wardrobe SET
            twos=:twos
            WHERE oid={ward_oid}""",
                {
                    'twos':present_two
                }
                )
        #commit changes
        conn.commit()

        #close connection
        conn.close()
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    c.execute(f"""UPDATE Login_details SET
        ward_image=:ward_image

        WHERE userid={username} AND passcode={password}""",
            {
                'ward_image':last_ward
            }
            )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
    ward_editor.mainloop()


    #ward_editor.destroy()
    #ward_editor.destroy()
#function to upload image into database
# Function for Convert Binary
# Data to Human Readable Format
def convert_data(data, file_name):
```

```
        # Convert binary format to
        # images or files data
        with open(file_name, 'wb') as file:
                file.write(data)
        img = Image.open(file_name)
        print(img)

#sentimental analysis

def senti(inputvalue):
    data=inputvalue
    stop_words = set(stopwords.words('english'))
    stop_words.remove("not")
    data.lower()
    dat=list(data.split())
    #word_tokens = word_tokenize(data)
    corpus=[]
    ps = PorterStemmer()
    print(data)
    for i in range(0,len(dat)):
        review = re.sub('[^a-zA-Z]',' ', dat[i])
        if(review not in set(stop_words)):
            corpus.append(review)
    print(corpus)
    sid = SentimentIntensityAnalyzer()
    pos_word_list=[]
    neu_word_list=[]
    neg_word_list=[]

    for word in corpus:
        print(sid.polarity_scores(word))
        if (sid.polarity_scores(word)['compound']) >= 0.4:
            pos_word_list.append(word)
        elif (sid.polarity_scores(word)['compound']) <= -0.4:
            neg_word_list.append(word)
        else:
            neu_word_list.append(word)

    if(len(pos_word_list)>len(neg_word_list)):
        return 1
    elif(len(pos_word_list)<len(neg_word_list)):
        return 0

    else:
```

```
        return 2

def update():
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()

    #record_id=user.get()
    c.execute("""UPDATE Login_details SET
        userid=:userid,
        passcode=:passcode,
        name=:name,
        phone=:phone,
        address=:address

        WHERE userid=:userid AND passcode=:passcode""",
            {
                'userid':userid_editor1.get(),
                'passcode':passcode_editor1.get(),
                'name':name_editor1.get(),
                'phone':phone_editor1.get(),
                'address':address_editor1.get()
            }
           )


    #commit changes
    conn.commit()
    #close connection
    conn.close()
    editor.destroy()
    #delete_box.delete(0,END)

def edit_details():
    global name_login_box
    global passcode_login_box
    #create textboxes
    name_login_box=Entry(root,width=30)
    name_login_box.place(relx=0.4,rely=0.4)
    #passcode text code
    passcode_login_box=Entry(root,width=30,show='*')
    passcode_login_box.place(relx=0.4,rely=0.5)

    #label for userid
```

```
    name_login_label=Label(root,text="USER ID")
    name_login_label.place(relx=0.3,rely=0.4)
    #label for passcode in login
    passcode_login_label=Label(root,text="passcode")
    passcode_login_label.place(relx=0.3,rely=0.5)

    #create Login button
    login_btn2=Button(root,text="submit",command=edit)
    login_btn2.place(relx=0.47,rely=0.55)
    name_login_box.delete(0,END)
    passcode_login_box.delete(0,END)
#create an edit function to update a record
def edit():
    global record_id
    global record_passcode
    record_id=name_login_box.get()
    record_passcode=passcode_login_box.get()
    global editor
    editor=Tk()
    editor.title("Update A Record")
    editor.iconbitmap()
    editor.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()

    #Query the database
    statement=f"SELECT * FROM Login_details WHERE userid='{record_id}' AND
passcode = '{record_passcode}'"
    c.execute(statement)
    records=c.fetchall()
    #print("hi")

    #create global variables for text box names
    global name_editor1
    global phone_editor1
    global address_editor1
    global userid_editor1
    global passcode_editor1

    #create text boxes
    userid_editor1=Entry(editor,width=30)
    userid_editor1.grid(row=1,column=1)
```

```
    passcode_editor1=Entry(editor,width=30)
    passcode_editor1.grid(row=2,column=1)
    name_editor1=Entry(editor,width=30)
    name_editor1.grid(row=3,column=1,padx=20,pady=(10,0))
    phone_editor1=Entry(editor,width=30)
    phone_editor1.grid(row=4,column=1)
    address_editor1=Entry(editor,width=30)
    address_editor1.grid(row=5,column=1)


    #create TextBox Labels
    userid_label=Label(editor,text="User id")
    userid_label.grid(row=1,column=0)
    passcode_label=Label(editor,text="passcode")
    passcode_label.grid(row=2,column=0)
    name_label=Label(editor,text="Name")
    name_label.grid(row=3,column=0,pady=(10,0))
    phone_label=Label(editor,text="Phone Number")
    phone_label.grid(row=4,column=0)
    address_label=Label(editor,text="Address")
    address_label.grid(row=5,column=0)
    #create a Save button to save edited record

     #Loop thru results
    for record in records:
        userid_editor1.insert(0,record[0])
        passcode_editor1.insert(0,record[1])
        name_editor1.insert(0,record[2])
        phone_editor1.insert(0,record[3])
        address_editor1.insert(0,record[4])

    edit_btn=Button(editor,text="Save Record",command=update)
    edit_btn.grid(row=14,column=1,columnspan=2,pady=10,padx=10,ipadx=145)
    #name_login_box.delete(0,END)
    #passcode_login_box.delete(0,END)
    #commit changes
    conn.commit()
    #close connection
    conn.close()


#function for adding record into database
def submit():
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
```

```
    #create cursor
    c=conn.cursor()
    #insert into table
    c.execute("INSERT INTO Login_details VALUES
(:userid,:passcode,:name,:phone,:address,:ward_image,:design_image,:new_image)",
    {

        'userid': userid_editor.get(),
        'passcode':passcode_editor.get(),
        'name':name_editor.get(),
        'phone':phone_editor.get(),
        'address':address_editor.get(),
        'ward_image':'0',
        'design_image':'0',
        'new_image':'0'
    })

    #commit changes
    conn.commit()
    #close connection
    conn.close()

    #clear the text boxes
    userid_editor.delete(0,END)
    passcode_editor.delete(0,END)
    name_editor.delete(0,END)
    phone_editor.delete(0,END)
    address_editor.delete(0,END)
    adder.destroy()

def final_design():
    global resulter1
    resulter1=Tk()
    resulter1.title("Fetch Result")
    resulter1.iconbitmap()
    resulter1.geometry("400x250")
    global res_label
    global res_box
    res_label=Label(resulter1,text="Enter Name ")
    res_label.place(relx=0.3,rely=0.4)
    res_box=Entry(resulter1,width=30)
    res_box.place(relx=0.5,rely=0.4)

res_btn=Button(resulter1,text="Fetch",command=final_searchdesign,fg="Green",active
background = "black")
```

```
    res_btn.place(relx=0.45,rely=0.85)

def create_charts_design():
    pier= tk.Tk()

    canvas1 = tk.Canvas(pier, width = 100, height = 40)
    canvas1.pack()

    label1 = tk.Label(pier, text='Graphical User Interface')
    label1.config(font=('Arial', 20))
    global x1
    global x2
    global x3
    global bar1
    global pie2
    x1 = float(res_zero)
    x2 = float(res_one)
    x3 = float(res_two)

    figure2 = Figure(figsize=(4,3), dpi=100)
    subplot2 = figure2.add_subplot(111)
    labels2 = 'Negative', 'Positive', 'Neutral'
    pieSizes = [float(x1),float(x2),float(x3)]
    my_colors2 = ['lightblue','lightsteelblue','silver']
    explode2 = (0, 0.1, 0)
    subplot2.pie(pieSizes, colors=my_colors2, explode=explode2, labels=labels2,
autopct='%1.1f%%', shadow=True, startangle=90)
    subplot2.axis('equal')
    pie2 = FigureCanvasTkAgg(figure2, pier)
    pie2.get_tk_widget().pack()
    pier.mainloop()

def clear_charts():
    bar1.get_tk_widget().pack_forget()
    pie2.get_tk_widget().pack_forget()


def final_searchdesign():
    global res_res
    res_res=res_box.get()
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    query_res= f"SELECT * FROM Design WHERE name='{res_res}';"
    c.execute(query_res)
```

```
    res_record=c.fetchone()
    global res_zero
    global res_one
    global res_two
    res_zero=res_record[4]
    res_one=res_record[3]
    res_two=res_record[5]
    slices=[res_two,res_one,res_zero]#the final neutral,positive,negative value shoud be
passed here
    outputs=['negative','positive','neutral']
    cols=['c','m','b']

plt.pie(slices,labels=outputs,colors=cols,startangle=90,shadow=True,explode=(0,0.1,0),
autopct='%1.1f%%')
    plt.title("Final result")
    plt.show()
    create_charts_design()

def prop(n):
    return 360.0 * n / 1000


#function for uploading image into db for client side
def design_upload():
    global design_editor_upload
    design_editor_upload=Tk()
    design_editor_upload.title("To Upload Wardrobe Image")
    design_editor_upload.iconbitmap()
    design_editor_upload.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global val_label_12
    global val_box_12
    global name_label_12
    global name_box_12
    global image_address_12
    name_label_12=Label(design_editor_upload,text="Enter name :")
    name_label_12.grid(row=1,column=3)
    name_box_12=Entry(design_editor_upload,width=50)
    name_box_12.grid(row=1,column=5)
    val_label_12=Label(design_editor_upload,text="Enter location image present:")
    val_label_12.grid(row=3,column=3)
    val_box_12=Entry(design_editor_upload,width=50)
```

```
    val_box_12.grid(row=3,column=5)
    image_address_12=val_box_12.get()
    print(image_address_12)
    '''
    #image_name_1=name_box_1.get("1.0","end-1c")
    ttk.Label(ward_editor_upload, text="Enter your Review :",
              font=("Times New Roman", 15)).place(relx=0.15,rely=0.75)
    # Text Widget
    global tl
    tl = Text(ward_editor_upload, width=100, height=6)
    tl.place(relx=0.3,rely=0.7)
    tl.focus()'''

upld_btn=Button(design_editor_upload,text="UPLOAD",command=retrieve_input_12,
fg="Green",activebackground = "black")
    upld_btn.grid(row=5,column=4)
#function to retrive input from client
def retrieve_input_12():
    #image_address_1=tl.get("1.0","end-1c")
    global image_name_12
    global image_address_12
    image_name_12=name_box_12.get()
    image_address_12=val_box_12.get()
    print(image_name_12)
    print(image_address_12)
    upld_2_btn=Button(design_editor_upload,text="Confirm UPLOAD
",command=insertdesign(image_name_12,image_address_12),fg="Green",activebackgr
ound = "black")
    upld_2_btn.grid(row=6,column=5)

def insertdesign(name, photo):
    try:

        # Using connect method for establishing
        # a connection
        sqliteConnection = sqlite3.connect('Details.db')
        cursor = sqliteConnection.cursor()
        print("Connected to SQLite")

        # insert query
        sqlite_insert_blob_query = """ INSERT INTO Design
                            (name,image,ones,zeros,twos) VALUES (?,?,0,0,0)"""

        # Converting human readable file into
        # binary data
```

```
        empPhoto = convertToBinaryData(photo)

        # Convert data into tuple format
        data_tuple = (name, empPhoto)

        # using cursor object executing our query
        cursor.execute(sqlite_insert_blob_query, data_tuple)
        sqliteConnection.commit()
        print("Image and file inserted successfully as a BLOB into a table")
        global success_label_1
        success_label_1=Label(design_editor_upload,text="Image and file inserted
successfully ..")
        success_label_1.grid(row=9,column=7)
        cursor.close()

    except sqlite3.Error as error:
        print("Failed to insert blob data into sqlite table", error)
        global fail_label_1
        fail_label_1=Label(design_editor_upload,text="Failed to insert")
        fail_label_1.grid(row=9,column=7)
    finally:
        if sqliteConnection:
            sqliteConnection.close()
            print("the sqlite connection is closed")
#insertBLOB("Smith", "D:\Internship Tasks\GFG\images\One.png")

#function for reviewer side wardrobe
def design_list_items():
    global design_editor
    design_editor=Tk()
    design_editor.title("To Review Art And Design")
    design_editor.iconbitmap()
    design_editor.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global n

    global val_label
    global val_box
    val_label=Label(design_editor,text="Enter location where you want to download:")
    val_label.place(relx=0.15,rely=0.35)
    val_box=Entry(design_editor,width=30)
    val_box.place(relx=0.35,rely=0.35)
```

```
    n=val_box.get()

down_btn=Button(design_editor,text="DOWNLOAD",command=design_list,fg="Gree
n",activebackground = "black")
    down_btn.place(relx=0.45,rely=0.4)

def design_list():
    try:

        # Using connect method for establishing
        # a connection
        con = sqlite3.connect('Details.db')
        cursor = con.cursor()
        print("Connected Successfully")

        query2=f"SELECT design_image from Login_details WHERE
userid='{username}' AND passcode ='{password}';"

        cursor.execute(query2)
        global last_image_design
        global design_oid
        last_image_design=cursor.fetchone()
        print(last_image_design)
        global last_design
        last_design=int(last_image_design[0])
        last_design+=1
        # Search from table query
        query = f"SELECT * FROM Design WHERE oid={last_design}"

        # using cursor object executing our query
        cursor.execute(query)

        # fectching all records from cursor object
        records = cursor.fetchall()
        design_oid=last_design
        # using for loop retrieving one by one
        # rows or data

        for row in records:

            # storing row[0] in name variable
            name = row[0]
            #print(row)
            #present_ones=row[3]
            # printing name variable
```

```
            print("Image Name = ", name)

            # storing image (currently in binary format)
            image = row[1]

            # calling above convert_data() for converting
            # binary data to human readable
            convert_data(image, n + name + ".png")
            print("Yeah!! We have successfully retrieved values from database")

            # If we don't have any records in our database,
            # then print this
            if len(records) == 0:
                print("Sorry! Please Insert some data before reading from the database.")


        # print exception if found any during program
    # is running
    except sqlite3.Error as error:
        print(format(error))

    # using finally, closing the connection
    # (con) object
    finally:
        if con:
            con.close()
            print("SQLite connection is closed")

    #ward_list_items()
    global design_num
    limit=0
    design_num=0
    global val1_label
    val1_label=Label(design_editor,text="Art And Design IMAGE")
    val1_label.place(relx=0.3,rely=0.6)
    ttk.Label(design_editor, text="Enter your Review :",
              font=("Times New Roman", 15)).place(relx=0.15,rely=0.75)
    # Text Widget
    global t1
    t1= Text(design_editor, width=100, height=6)
    t1.place(relx=0.3,rely=0.7)
    t1.focus()

down_btn=Button(design_editor,text="Submit",command=lambda:retrieve_input_desig
n(),fg="Green",activebackground = "black")
```

```
    down_btn.place(relx=0.45,rely=0.85)
#command=lambda: retrieve_input()

def retrieve_input_design():
    #print("hello")
    n=val_box.get()
    #print(n)
    inputValue=t1.get("1.0","end-1c")
    result=senti(inputValue)
    global final_design_label

    final_design_label=Label(design_editor,text="Review Has been saved , click on
close ")
    final_design_label.place(relx=0.5,rely=0.9)
    #we need to add data after creating wardrobe table
    t1.delete(1.0,END)
    if(result==1):
        #present_ones+=1
        conn=sqlite3.connect('Details.db')

        #create cursor
        c=conn.cursor()
        query1= f"SELECT ones FROM Design WHERE oid={design_oid}"
        c.execute(query1)
        present_ones_1=c.fetchone()
        print(present_ones_1)
        present_one_1=int(present_ones_1[0])
        present_one_1+=1
        print(present_one_1)
        c.execute(f"""UPDATE Design SET
            ones=:ones
            WHERE oid={design_oid}""",
                {
                    'ones':present_one_1
                }
                )
        #commit changes
        conn.commit()

        #close connection
        conn.close()
    elif(result==0):
        #present_ones+=1
        conn=sqlite3.connect('Details.db')
```

```
    #create cursor
    c=conn.cursor()
    query1= f"SELECT zeros FROM Design WHERE oid={design_oid}"
    c.execute(query1)
    global present_zeros_1
    present_zeros_1=c.fetchone()
    print(present_zeros_1)
    present_zero_1=int(present_zeros_1[0])
    present_zero_1+=1
    print(present_zero_1)
    c.execute(f"""UPDATE Design SET
        zeros=:zeros

        WHERE oid={design_oid}""",
                {
                    'zeros':present_zero_1
                }
                )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
elif(result==2):
    #present_ones+=1
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    query1= f"SELECT twos FROM Design WHERE oid={design_oid}"
    c.execute(query1)
    global present_twos_1
    present_twos_1=c.fetchone()
    print(present_twos_1)
    present_two_1=int(present_twos_1[0])
    present_two_1+=1
    print(present_two_1)
    c.execute(f"""UPDATE Design SET
        twos=:twos
        WHERE oid={design_oid}""",
                {
                    'twos':present_two_1
                }
                )
    #commit changes
```

```
        conn.commit()

        #close connection
        conn.close()
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    c.execute(f"""UPDATE Login_details SET
        design_image=:design_image

        WHERE userid={username} AND passcode={password}""",
                {
                    'design_image':last_design
                }
                )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
    design_editor.mainloop()

def final_new():
    global resulter2
    resulter2=Tk()
    resulter2.title("Fetch Result")
    resulter2.iconbitmap()
    resulter2.geometry("400x250")
    global res_label
    global res_box
    res_label=Label(resulter2,text="Enter Name ")
    res_label.place(relx=0.3,rely=0.4)
    res_box=Entry(resulter2,width=30)
    res_box.place(relx=0.5,rely=0.4)

res_btn=Button(resulter2,text="Fetch",command=final_searchnew,fg="Green",activeba
ckground = "black")
    res_btn.place(relx=0.45,rely=0.85)

def create_charts_new():
    pier= tk.Tk()

    canvas1 = tk.Canvas(pier, width = 100, height = 40)
```

```
    canvas1.pack()

    label1 = tk.Label(pier, text='Graphical User Interface')
    label1.config(font=('Arial', 20))
    global x1
    global x2
    global x3
    global bar1
    global pie2
    x1 = float(res_zero)
    x2 = float(res_one)
    x3 = float(res_two)

    figure2 = Figure(figsize=(4,3), dpi=100)
    subplot2 = figure2.add_subplot(111)
    labels2 = 'Negative', 'Positive', 'Neutral'
    pieSizes = [float(x1),float(x2),float(x3)]
    my_colors2 = ['lightblue','lightsteelblue','silver']
    explode2 = (0, 0.1, 0)
    subplot2.pie(pieSizes, colors=my_colors2, explode=explode2, labels=labels2,
autopct='%1.1f%%', shadow=True, startangle=90)
    subplot2.axis('equal')
    pie2 = FigureCanvasTkAgg(figure2, pier)
    pie2.get_tk_widget().pack()
    pier.mainloop()

def clear_charts():
    bar1.get_tk_widget().pack_forget()
    pie2.get_tk_widget().pack_forget()


def final_searchnew():
    global res_res
    res_res=res_box.get()
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    query_res= f"SELECT * FROM PRODUCT WHERE name='{res_res}';"
    c.execute(query_res)
    res_record=c.fetchone()
    global res_zero
    global res_one
    global res_two
    res_zero=res_record[4]
    res_one=res_record[3]
```

```
    res_two=res_record[5]
    slices=[res_two,res_one,res_zero]#the final neutral,positive,negative value shoud be
passed here
    outputs=['negative','positive','neutral']
    cols=['c','m','b']

plt.pie(slices,labels=outputs,colors=cols,startangle=90,shadow=True,explode=(0,0.1,0),
autopct='%1.1f%%')
    plt.title("Final result")
    plt.show()
    global pier
    pier=Tk()
    pier.title("PIE")
    pier.iconbitmap()
    tk.Label(pier, text='Pie Chart').pack()
    c =tk.Canvas(width=154, height=154)
    c.grid(row=5,column=5)
    c.create_arc((2,2,152,152), fill="#FAF402", outline="#FAF402", start=prop(0),
extent = prop(res_zero))
    c.create_arc((2,2,152,152), fill="#2BFFF4", outline="#2BFFF4",
start=prop(res_zero), extent = prop(res_one))
    c.create_arc((2,2,152,152), fill="#E00022", outline="#E00022",
start=prop(res_zero+res_one), extent = prop(res_two))
    pier.mainloop()
    create_charts_new()

#function for uploading image into db for client side
def new_upload():
    global new_editor_upload
    new_editor_upload=Tk()
    new_editor_upload.title("To Upload PRODUCT Image")
    new_editor_upload.iconbitmap()
    new_editor_upload.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global val_label_13
    global val_box_13
    global name_label_13
    global name_box_13
    global image_address_13
    name_label_13=Label(new_editor_upload,text="Enter name :")
    name_label_13.grid(row=1,column=3)
    name_box_13=Entry(new_editor_upload,width=30)
```

```
    name_box_13.grid(row=1,column=5)
    val_label_13=Label(new_editor_upload,text="Enter location image present:")
    val_label_13.grid(row=3,column=3)
    val_box_13=Entry(new_editor_upload,width=30)
    val_box_13.grid(row=3,column=5)
    image_address_13=val_box_13.get()
    print(image_address_13)
    '''
    #image_name_1=name_box_1.get("1.0","end-1c")
    ttk.Label(ward_editor_upload, text="Enter your Review :",
              font=("Times New Roman", 15)).place(relx=0.15,rely=0.75)
    # Text Widget
    global tl
    tl = Text(ward_editor_upload, width=100, height=6)
    tl.place(relx=0.3,rely=0.7)
    tl.focus()'''

upld_btn=Button(new_editor_upload,text="UPLOAD",command=retrieve_input_13,fg
="Green",activebackground = "black")
    upld_btn.grid(row=5,column=4)
#function to retrive input from client
def retrieve_input_13():
    #image_address_1=tl.get("1.0","end-1c")
    global image_name_13
    global image_address_13
    image_name_13=name_box_13.get()
    image_address_13=val_box_13.get()
    print(image_name_13)
    print(image_address_13)
    upld_2_btn=Button(new_editor_upload,text="Confirm UPLOAD
",command=insertnew(image_name_13,image_address_13),fg="Green",activebackgrou
nd = "black")
    upld_2_btn.grid(row=6,column=5)

def insertnew(name, photo):
    try:

        # Using connect method for establishing
        # a connection
        sqliteConnection = sqlite3.connect('Details.db')
        cursor = sqliteConnection.cursor()
        print("Connected to SQLite")

        # insert query
        sqlite_insert_blob_query = """ INSERT INTO PRODUCT
```

```
                    (name,image,ones,zeros,twos) VALUES (?,?,0,0,0)"""

        # Converting human readable file into
        # binary data
        empPhoto = convertToBinaryData(photo)

        # Convert data into tuple format
        data_tuple = (name, empPhoto)

        # using cursor object executing our query
        cursor.execute(sqlite_insert_blob_query, data_tuple)
        sqliteConnection.commit()
        print("Image and file inserted successfully as a BLOB into a table")
        global success_label_2
        success_label_2=Label(new_editor_upload,text="Image and file inserted
successfully ..")
        success_label_2.grid(row=9,column=7)
        cursor.close()

    except sqlite3.Error as error:
        print("Failed to insert blob data into sqlite table", error)
        global fail_label_2
        fail_label_2=Label(new_editor_upload,text="Failed to insert")
        fail_label_2.grid(row=9,column=7)
    finally:
        if sqliteConnection:
            sqliteConnection.close()
            print("the sqlite connection is closed")
#insertBLOB("Smith", "D:\Internship Tasks\GFG\images\One.png")

#function for reviewer side wardrobe
def new_list_items():
    global new_editor
    new_editor=Tk()
    new_editor.title("To Review Newly Launching Products List")
    new_editor.iconbitmap()
    new_editor.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global n

    global val_label
    global val_box
```

```
    val_label=Label(new_editor,text="Enter location where you want to download:")
    val_label.place(relx=0.15,rely=0.35)
    val_box=Entry(new_editor,width=30)
    val_box.place(relx=0.35,rely=0.35)
    n=val_box.get()

down_btn=Button(new_editor,text="DOWNLOAD",command=new_list,fg="Green",ac
tivebackground = "black")
    down_btn.place(relx=0.45,rely=0.4)

def new_list():
    try:

        # Using connect method for establishing
        # a connection
        con = sqlite3.connect('Details.db')
        cursor = con.cursor()
        print("Connected Successfully")

        query2=f"SELECT new_image from Login_details WHERE userid='{username}'
AND passcode ='{password}';"

        cursor.execute(query2)
        global last_image_new
        global new_oid
        last_image_new=cursor.fetchone()
        print(last_image_new)
        global last_new
        last_new=int(last_image_new[0])
        last_new+=1
        # Search from table query
        query = f"SELECT * FROM PRODUCT WHERE oid={last_new}"

        # using cursor object executing our query
        cursor.execute(query)

        # fectching all records from cursor object
        records = cursor.fetchall()
        new_oid=last_new
        # using for loop retrieving one by one
        # rows or data

        for row in records:

            # storing row[0] in name variable
```

```
        name = row[0]
        #print(row)
        #present_ones=row[3]
        # printing name variable
        print("Student Name = ", name)

        # storing image (currently in binary format)
        image = row[1]

        # calling above convert_data() for converting
        # binary data to human readable
        convert_data(image, n + name + ".png")
        print("Yeah!! We have successfully retrieved values from database")

        # If we don't have any records in our database,
        # then print this
        if len(records) == 0:
            print("Sorry! Please Insert some data before reading from the database.")


    # print exception if found any during program
# is running
except sqlite3.Error as error:
    print(format(error))

# using finally, closing the connection
# (con) object
finally:
    if con:
        con.close()
        print("SQLite connection is closed")

#ward_list_items()
global new_num
limit=0
new_num=0
global val2_label
val2_label=Label(new_editor,text="Newly Launching Product")
val2_label.place(relx=0.3,rely=0.6)
ttk.Label(new_editor, text="Enter your Review :",
            font=("Times New Roman", 20)).place(relx=0.15,rely=0.75)
# Text Widget
global t2
t2= Text(new_editor, width=100, height=6)
t2.place(relx=0.3,rely=0.7)
```

```
    t2.focus()

down_btn=Button(new_editor,text="Submit",command=lambda:retrieve_input_2(),fg=
"Green",activebackground = "black")
    down_btn.place(relx=0.45,rely=0.85)
#command=lambda: retrieve_input()

def retrieve_input_2():
    #print("hello")
    n=val_box.get()
    #print(n)
    inputValue=t2.get("1.0","end-1c")
    result=senti(inputValue)
    global final_new_label

    final_new_label=Label(new_editor,text="Review Has been saved , click on close ")
    final_new_label.place(relx=0.5,rely=0.9)
    #we need to add data after creating wardrobe table
    t2.delete(1.0,END)
    if(result==1):
        #present_ones+=1
        conn=sqlite3.connect('Details.db')

        #create cursor
        c=conn.cursor()
        query1= f"SELECT ones FROM PRODUCT WHERE oid={new_oid}"
        c.execute(query1)
        present_ones_2=c.fetchone()
        print(present_ones_2)
        present_one_2=int(present_ones_2[0])
        present_one_2+=1
        print(present_one_2)
        c.execute(f"""UPDATE PRODUCT SET
            ones=:ones
            WHERE oid={new_oid}""",
                {
                    'ones':present_one_2
                }
                )
        #commit changes
        conn.commit()

        #close connection
        conn.close()
    elif(result==0):
```

```
    #present_ones+=1
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    query1= f"SELECT zeros FROM PRODUCT WHERE oid={new_oid}"
    c.execute(query1)
    global present_zeros_2
    present_zeros_2=c.fetchone()
    print(present_zeros_2)
    present_zero_2=int(present_zeros_2[0])
    present_zero_2+=1
    print(present_zero_2)
    c.execute(f"""UPDATE PRODUCT SET
        zeros=:zeros

        WHERE oid={new_oid}""",
            {
                'zeros':present_zero_2
            }
            )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
elif(result==2):
    #present_ones+=1
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    query1= f"SELECT twos FROM PRODUCT WHERE oid={new_oid}"
    c.execute(query1)
    global present_twos_2
    present_twos_2=c.fetchone()
    print(present_twos_2)
    present_two_2=int(present_twos_2[0])
    present_two_2+=1
    print(present_two_2)
    c.execute(f"""UPDATE PRODUCT SET
        twos=:twos
        WHERE oid={new_oid}""",
            {
                'twos':present_two_2
```

```
                }
                )
        #commit changes
        conn.commit()

        #close connection
        conn.close()
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')

    #create cursor
    c=conn.cursor()
    c.execute(f"""UPDATE Login_details SET
        new_image=:new_image

        WHERE userid={username} AND passcode={password}""",
            {
                'new_image':last_new
            }
            )
    #commit changes
    conn.commit()

    #close connection
    conn.close()
    new_editor.mainloop()

#function for sign in
def addintodb():
    global adder
    adder=Tk()
    adder.title("Create A Record")
    adder.iconbitmap()
    adder.geometry("400x250")
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()

    #create global variables for text box names
    global name_editor
    global phone_editor
    global address_editor
    global userid_editor
    global passcode_editor
```

```
    #create text boxes
    userid_editor=Entry(adder,width=30)
    userid_editor.grid(row=7,column=1)
    passcode_editor=Entry(adder,width=30)
    passcode_editor.grid(row=8,column=1)
    name_editor=Entry(adder,width=30)
    name_editor.grid(row=4,column=1,padx=20,pady=(10,0))
    phone_editor=Entry(adder,width=30)
    phone_editor.grid(row=5,column=1)
    address_editor=Entry(adder,width=30)
    address_editor.grid(row=6,column=1)

    #create TextBox Labels
    userid_label=Label(adder,text="User id")
    userid_label.grid(row=7,column=0)
    passcode_label=Label(adder,text="passcode")
    passcode_label.grid(row=8,column=0)
    name_label=Label(adder,text="Name")
    name_label.grid(row=4,column=0,pady=(10,0))
    phone_label=Label(adder,text="Phone Number")
    phone_label.grid(row=5,column=0)
    address_label=Label(adder,text="Address")
    address_label.grid(row=6,column=0)
    userid_label=Label(adder,text="User id")
    userid_label.grid(row=7,column=0)
    passcode_label=Label(adder,text="passcode")
    passcode_label.grid(row=8,column=0)

    #create a Save button to save edited record

    #create button
    submit_btn=Button(adder,text="Add Record to Database",command=submit)
    submit_btn.grid(row=9,column=1,columnspan=2,pady=10,padx=10,ipadx=100)

    #commit changes
    conn.commit()
    #close connection
    conn.close()
#function for differnt items list
def item_list_res():
    wadrobe_btn=Button(root,text="of WADROBE LIST",command=final_ward)
    wadrobe_btn.place(relx=0.45,rely=0.75)
    design_btn=Button(root,text="of ART AND DESIGN",command=final_design)
    design_btn.place(relx=0.4455,rely=0.85)
```

```
    prod_btn=Button(root,text="of NEWLY LAUNCHING
PRODUCTS",command=final_new)
    prod_btn.place(relx=0.425,rely=0.95)

#function for differnt items list
def item_list():
    wadrobe_btn=Button(root,text="WADROBE LIST",command=ward_upload)
    wadrobe_btn.place(relx=0.45,rely=0.75)
    design_btn=Button(root,text="ART AND DESIGN",command=design_upload)
    design_btn.place(relx=0.4455,rely=0.85)
    prod_btn=Button(root,text="NEWLY LAUNCHING
PRODUCTS",command=new_upload)
    prod_btn.place(relx=0.425,rely=0.95)

def item_list_give():
    wadrobe_btn=Button(root,text="For WADROBE LIST",command=ward_list_items)
    wadrobe_btn.place(relx=0.45,rely=0.75)
    design_btn=Button(root,text="For ART AND
DESIGN",command=design_list_items)
    design_btn.place(relx=0.4455,rely=0.85)
    prod_btn=Button(root,text="For NEWLY LAUNCHING
PRODUCTS",command=new_list_items)
    prod_btn.place(relx=0.425,rely=0.95)
#function for entering details
def login_details():
    global name_login_box
    global passcode_login_box
    #create textboxes
    name_login_box=Entry(root,width=30)
    name_login_box.place(relx=0.4,rely=0.4)
    #passcode text code
    passcode_login_box=Entry(root,width=30,show='*')
    passcode_login_box.place(relx=0.4,rely=0.5)
    #label for userid
    name_login_label=Label(root,text="USER ID")
    name_login_label.place(relx=0.3,rely=0.4)
    #label for passcode in login
    passcode_login_label=Label(root,text="passcode")
    passcode_login_label.place(relx=0.3,rely=0.5)

    #create Login button
    login_btn2=Button(root,text="submit",command=login)
    login_btn2.place(relx=0.47,rely=0.55)

    #print("hel",username,password)
```

```
    #name_login_box.delete(0,END)
    #passcode_login_box.delete(0,END)
#function for login
def login():
    #create a database or connect to one
    conn=sqlite3.connect('Details.db')
    #create cursor
    c=conn.cursor()
    global username
    global password
    username=name_login_box.get()
    password=passcode_login_box.get()
    #print("hel",username,password)
    statement = f"SELECT userid from Login_details WHERE userid='{username}'
AND passcode ='{password}';"
    c.execute(statement)
    record=c.fetchone()
    if not record:  # An empty result evaluates to False.
        fail=Label(text="Enter valid details...")
        fail.place(relx=0.37,rely=0.6)
        name_login_box.delete(0,END)
        passcode_login_box.delete(0,END)
    else:
        #print("Welcome")
        passed_label=Label(text="Logged in successfully..")
        passed_label.place(relx=0.37,rely=0.6)
        login3_btn=Button(root,text="GIVE REVIEW",command=item_list_give)
        login3_btn.place(relx=0.35,rely=0.65)
        login3_1_btn=Button(root,text="GET REVIEW",command=item_list)
        login3_1_btn.place(relx=0.45,rely=0.65)
        login3_2_btn=Button(root,text="GET RESULT",command=item_list_res)
        login3_2_btn.place(relx=0.55,rely=0.65)
        name_login_box.delete(0,END)
        passcode_login_box.delete(0,END)
    #commit changes
    conn.commit()

    #close connection
    conn.close()
    #name_login_box.delete(0,END)
    #passcode_login_box.delete(0,END)


'''
#create table
```

```
c.execute("""CREATE TABLE Login_details(
        userid text,
        passcode text,
        name text,
        phone text,
        address text,
        ward_image text,
        design_image text,
        new_image text
    )""")
'''
#frame 1

login1_btn=Button(root,text="LOGIN",command=login_details,fg="blue",activebackgr
ound = "black")
#login1_btn.grid(row=20,column=90)
login1_btn.place(relx=0.45, rely=0.25, anchor=CENTER)

sign_btn=Button(root,text="SIGN
UP",command=addintodb,fg="Green",activebackground = "black")
sign_btn.place(relx=0.55, rely=0.25, anchor=CENTER)

makechange_btn=Button(root,text="EDIT
INFORMATION",command=edit_details,fg="indigo",activebackground = "red")
makechange_btn.place(relx=0.5, rely=0.3, anchor=CENTER)



#commit changes
conn.commit()

#close connection
conn.close()

root.mainloop()
```

GITHUB LINK

https://github.com/ajay-63/Multi-Reviewing-System

TESTING

Test Case ID: TC01		Use case ID: UC01
Test Case Title: New User Registration		Register
Test Case Description: Check the response when new user registration details given.		
Test Steps	Expected Result	Actual Result
1. Enter user name, number 2. Enter address 3. Enter user id , passcode 4. Click on add record	A message Successfully Registered is shown	Successfully Registered message is displayed

Test Case ID: TC02		Use case ID: UC02
Test Case Title: Login		Login
Test Case Description: Check the details in database .		
Test Steps	Expected Result	Actual Result
1. Enter user id 2. Enter Passcode 3. Click on Submit 4. Select category on which you need review	A message Successfully logged is shown	Message of Successfully logged in is displayed

Test Case ID: TC03		Use case ID: UC03
Test Case Title: image uploading		Uploading image
Test Case Description: it saves the image of specified location into database		
Test Steps	Expected Result	Actual Result
1. Enter image name 2. Enter image location 3. Click on Upload , Confirm upload	A message of image uploaded successfully is shown .	Message of image uploaded successfully is displayed .

Test Case ID: TC04	Use case ID: UC01
Test Case Title: New User Registration	

Test Case Description: Check the response when new user registration details given.		Register
Test Steps	Expected Result	Test Steps
1.Enter user name, number 2.Enter address 3.Enter user id , passcode 4.Click on add record	A message Successfully Registered is shown	Successfully registered message displayed .

Test Case ID: TC05		Use case ID: UC02
Test Case Title: Login		Login
Test Case Description: Checks the details in database .		
Test Steps	Expected Result	Test Steps
1.Enter user id , passcode 2.Click on submit.	A message Successfully logged in is shown	Successfully logged in message displayed .

Test Case ID: TC06		Use case ID: UC04
Test Case Title: Image Download		Downloading image
Test Case Description: it saves the image from database to specified location		
Test Steps	Expected Result	Actual Result
1. Enter image location 2. Click on Download	Asks for review of image with its name .	Asked for review of image with its name .

Test Case ID: TC07		Use case ID: UC05
Test Case Title: reviewing		Reviewing
Test Case Description: it saves the review and gives to the system		
Test Steps	Expected Result	Actual Result

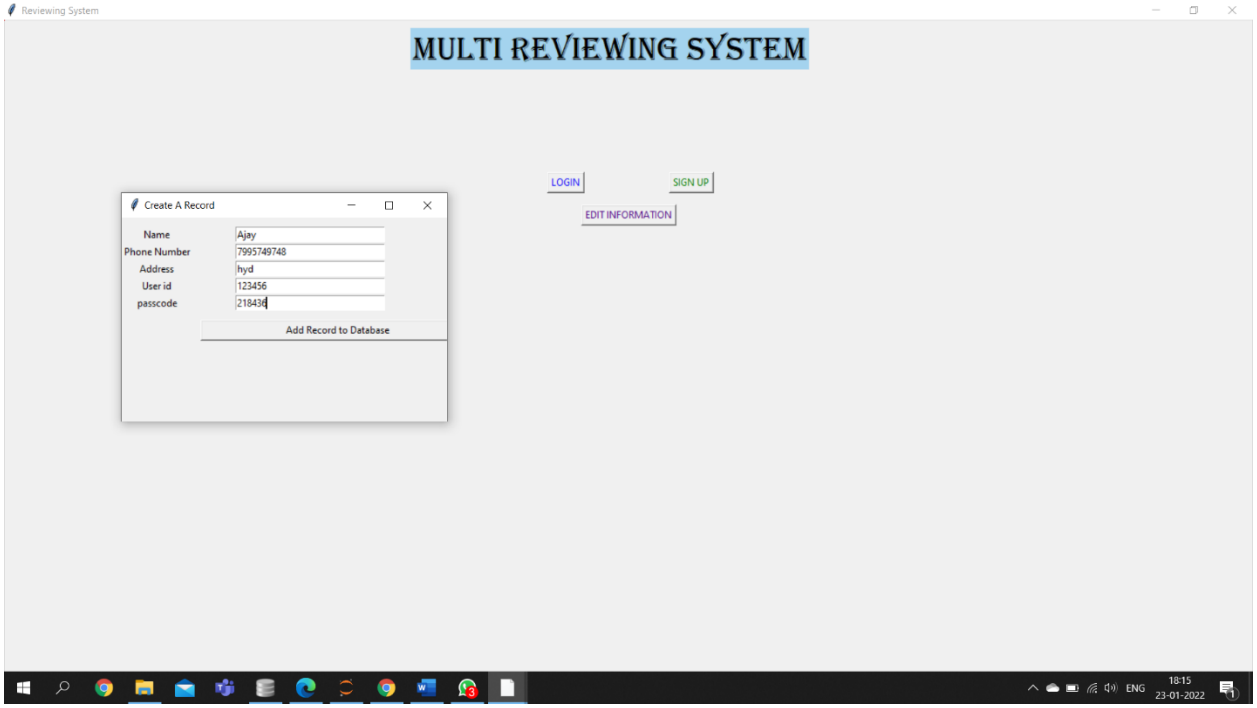
1. Enter review 2. Click on submit .	A message of review submitted successfully is shown .	Message of review submitted successfully is displayed .

Test Case ID: TC08		Use case ID: UC06
Test Case Title: Analysing review		Analysis
Test Case Description: it analyses the review and adds into database		
Test Steps	Expected Result	Actual Result
1. Review is taken and given to system 2. System analyses it.	A positive or negative or neutral review count increases	A positive or negative or neutral review count increased.

Test Case ID: TC09		Use case ID: UC07
Test Case Title: image uploading		Fetch result
Test Case Description: it fetches the number of positive , negative, neutral comments and displays pie chart		
Test Steps	Expected Result	Actual Result
1.Enter image name 2.Click on fetch	Pie chart will be shown.	Pie chart will be displayed .

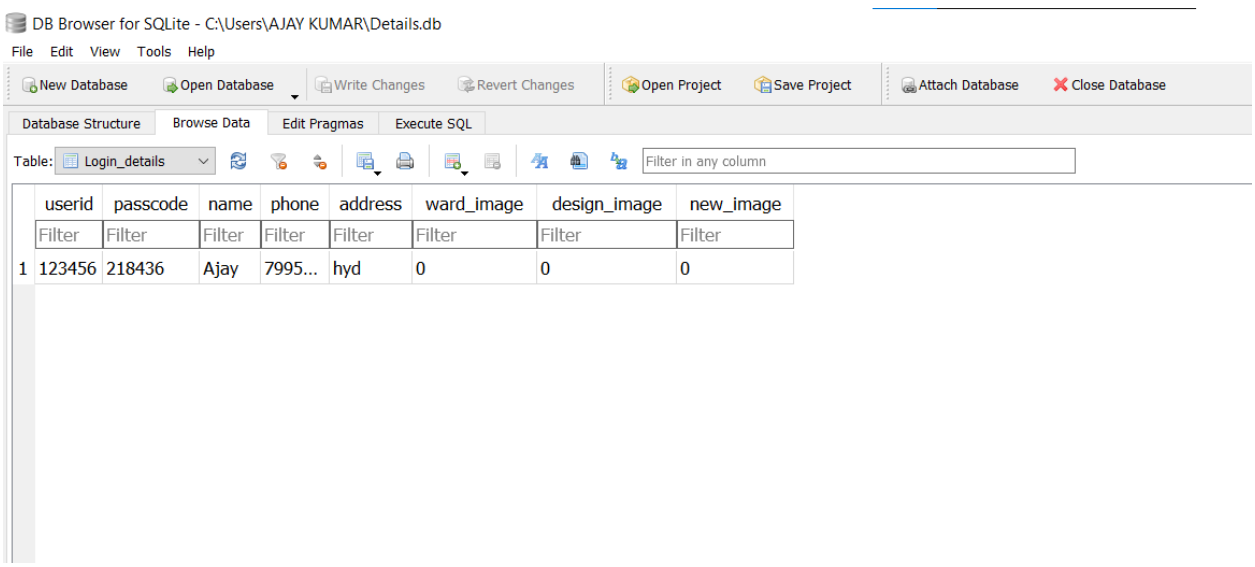
RESULTS

Firstly user who want to get review needs to sign up and he needs to create user id and passcode.



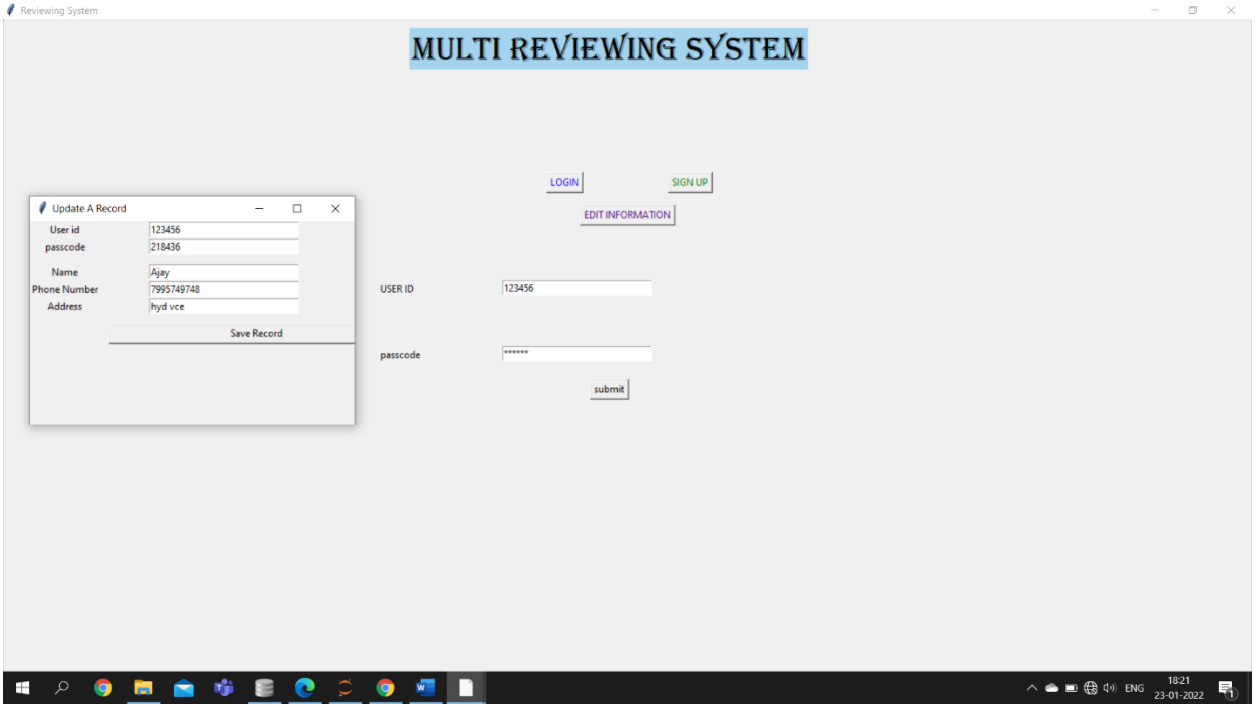
After sign up click on add record to database button the details will be added into database.

Table of login details

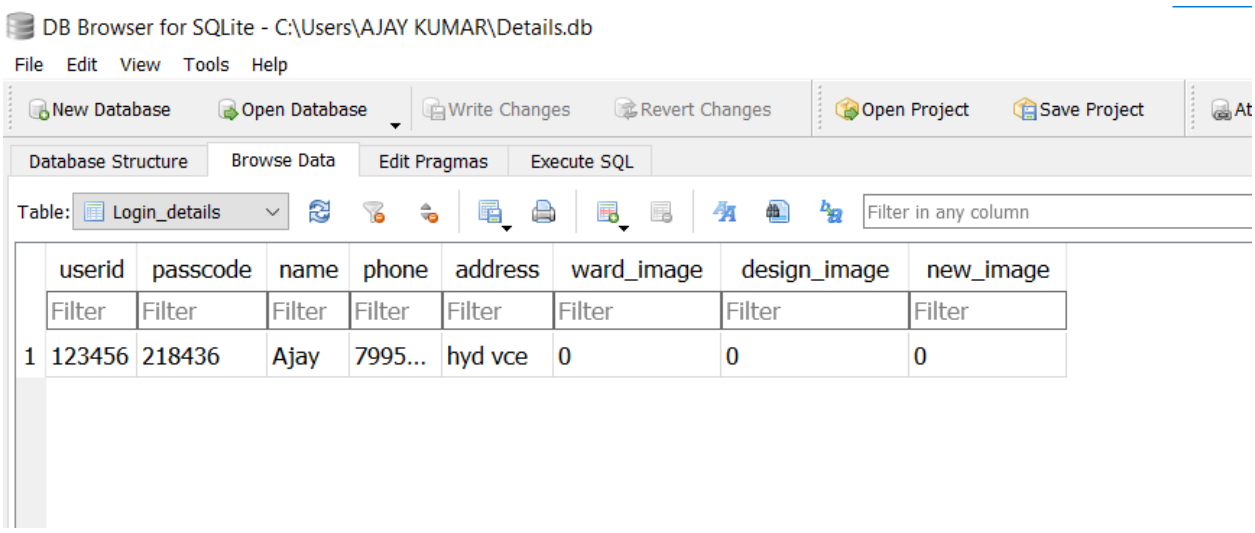


	userid	passcode	name	phone	address	ward_image	design_image	new_image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	123456	218436	Ajay	7995...	hyd	0	0	0

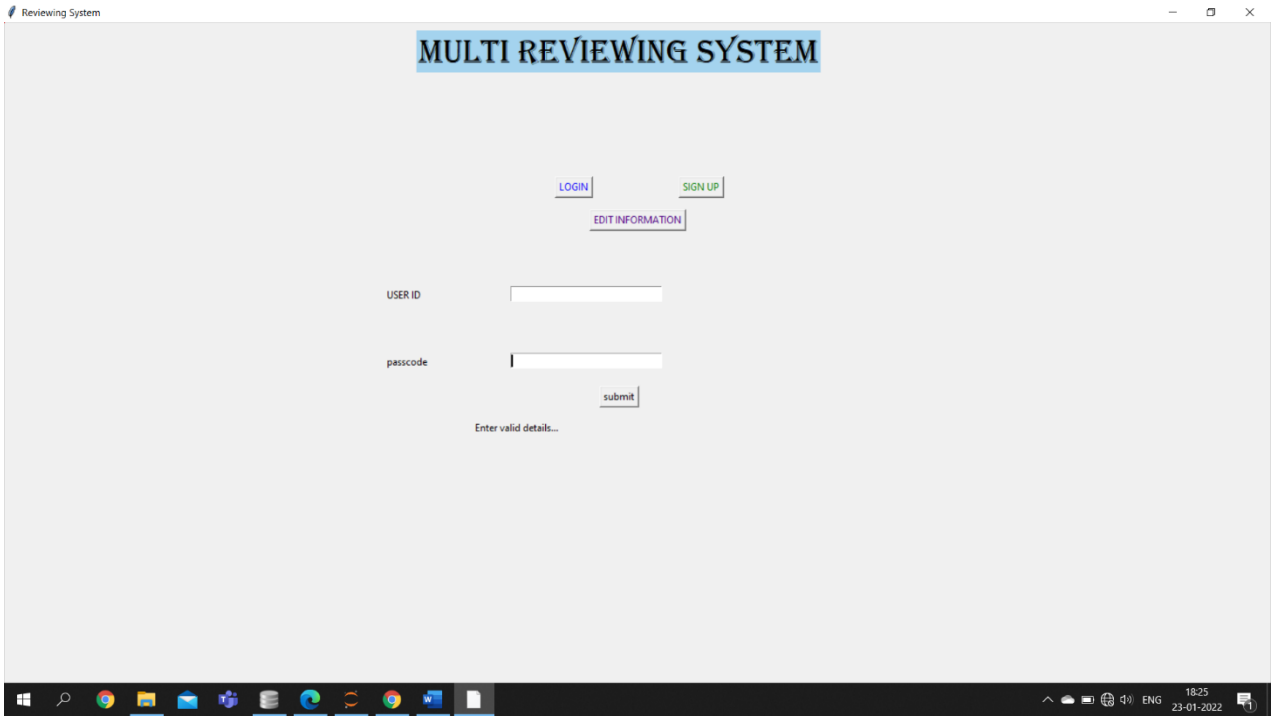
If user want to edit details first he need to click on edit information and then it asks login and passcode . Enter valid details and then it shows window as below to edit details .



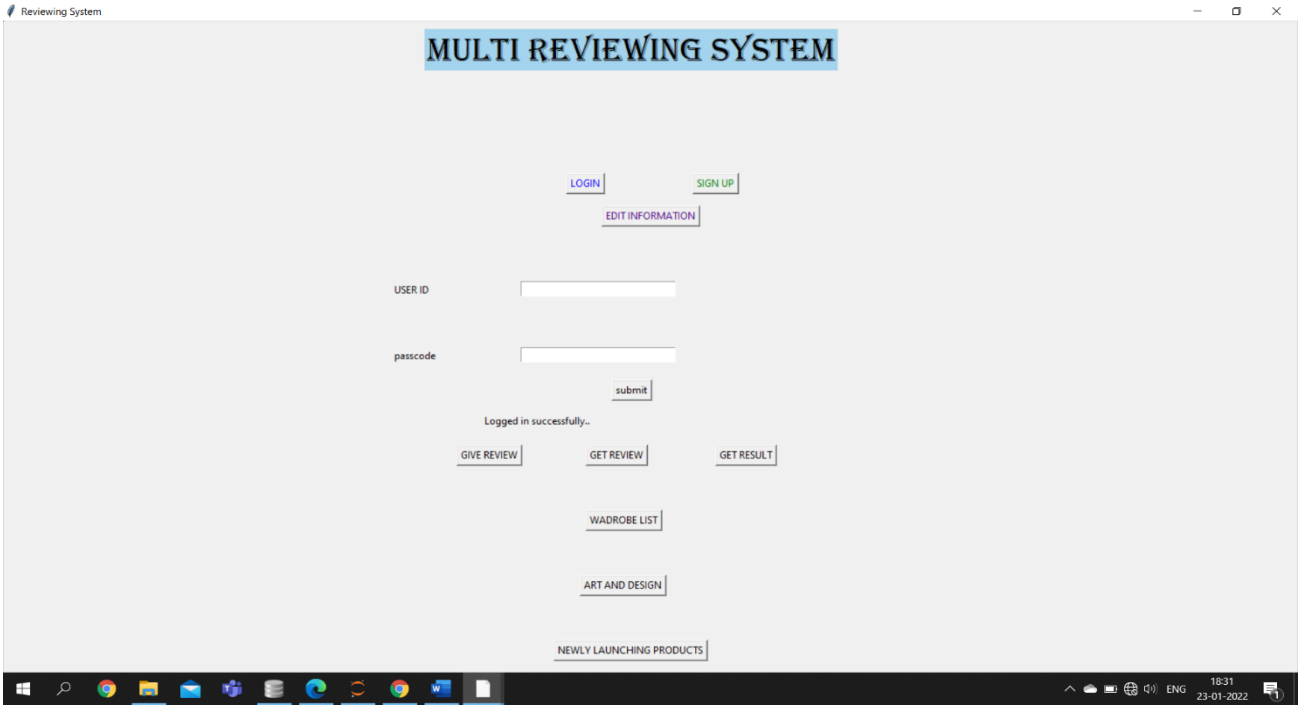
After editing the details click on save record button in database it will be modified.



So for getting review user should login by entering the details (valid else it shows to enter valid details) .if it shows logged in successfully then you are in .



If login is successful . Then it shows 3 buttons click on get review button to get review .After clicking on get review then it will show 3 buttons of wardrobe list , Design and newly launching product . select category which you want to get review



Suppose user selects wardrobe then a window opens which have text fields . User needs to enter name he want to give to image and he has to enter address where image is located . Click on upload and then on Confirm upload.

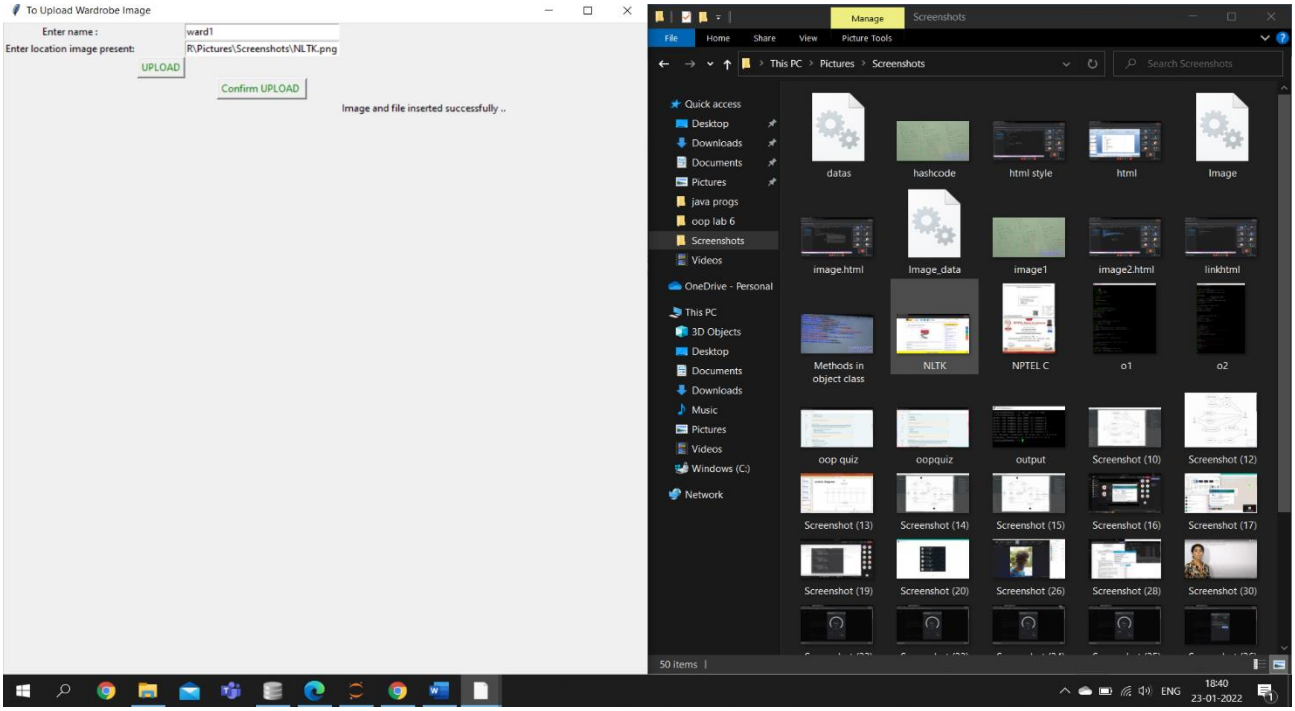
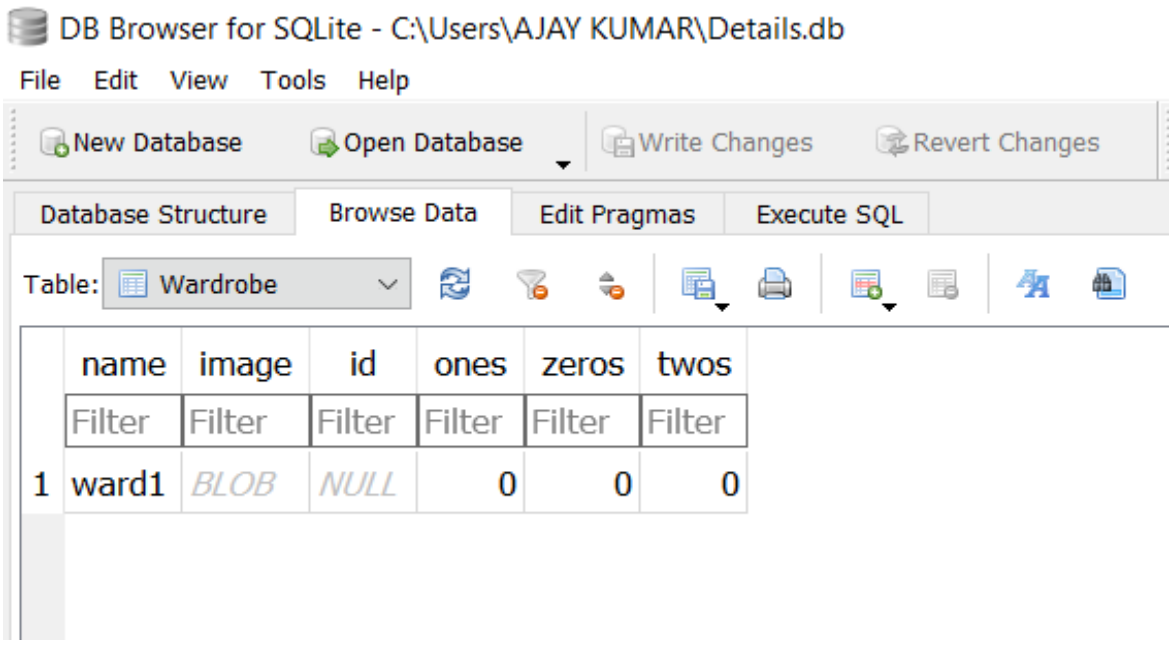
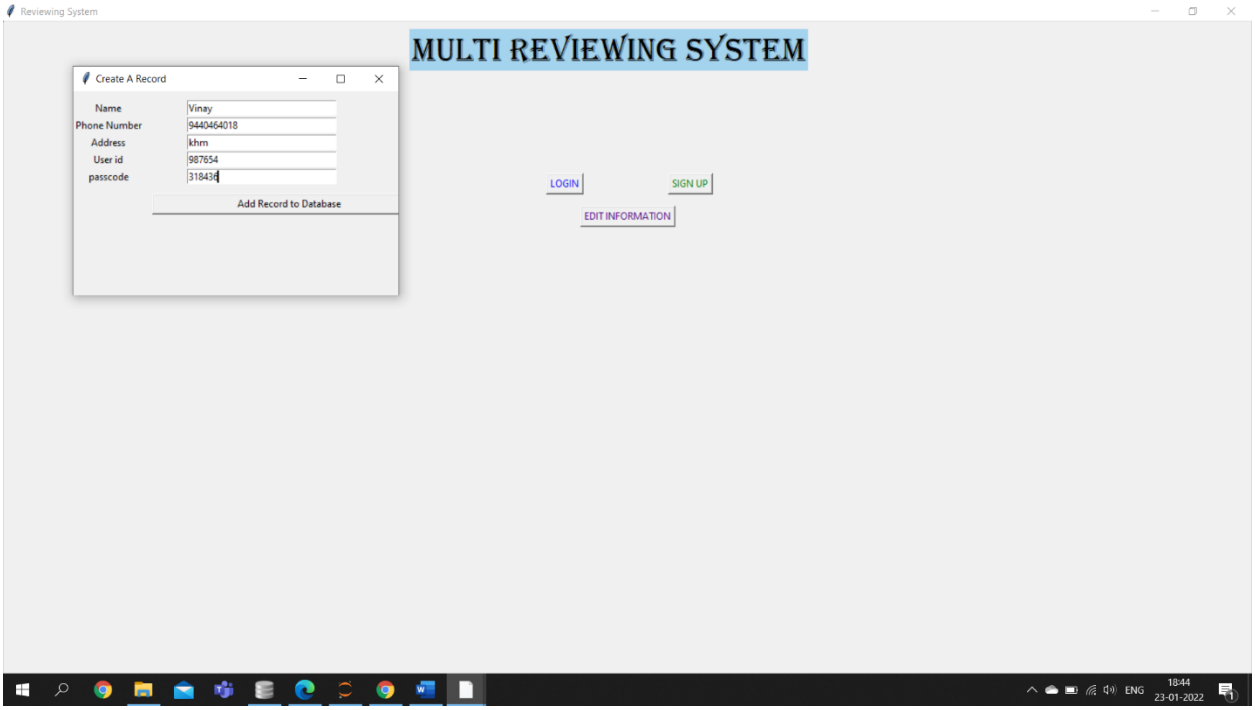


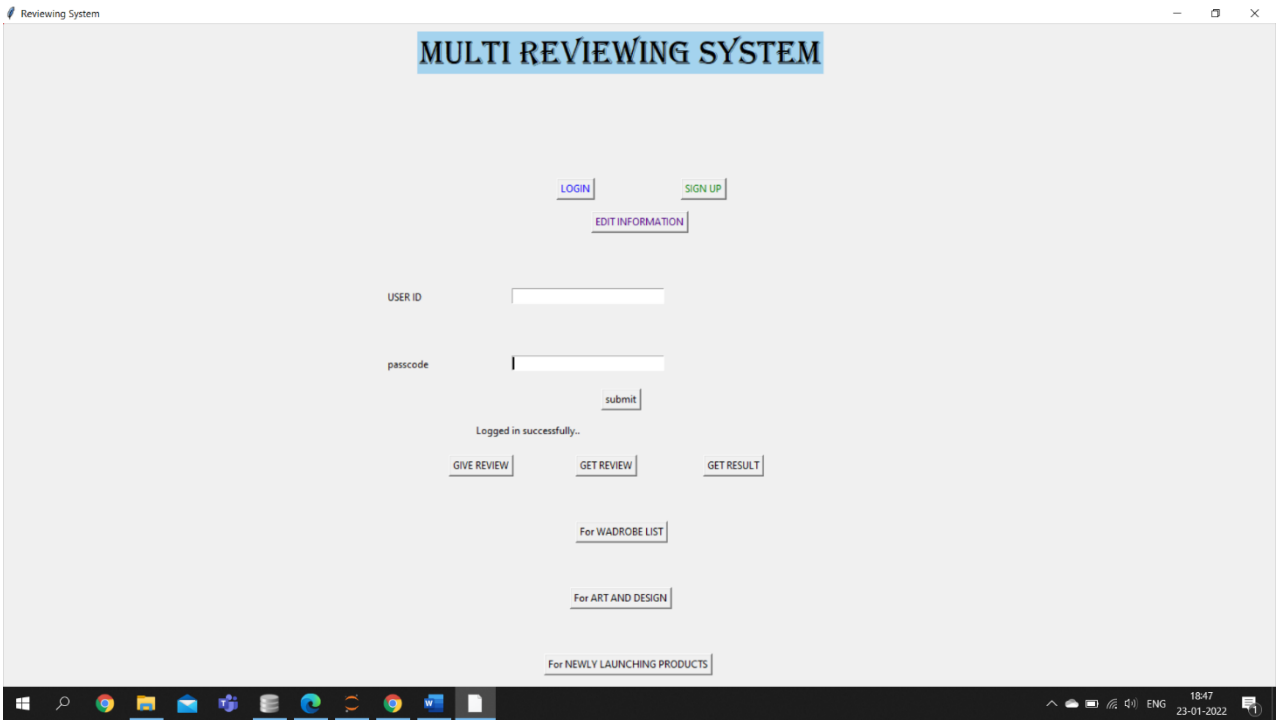
Image has been uploaded into database successfully.



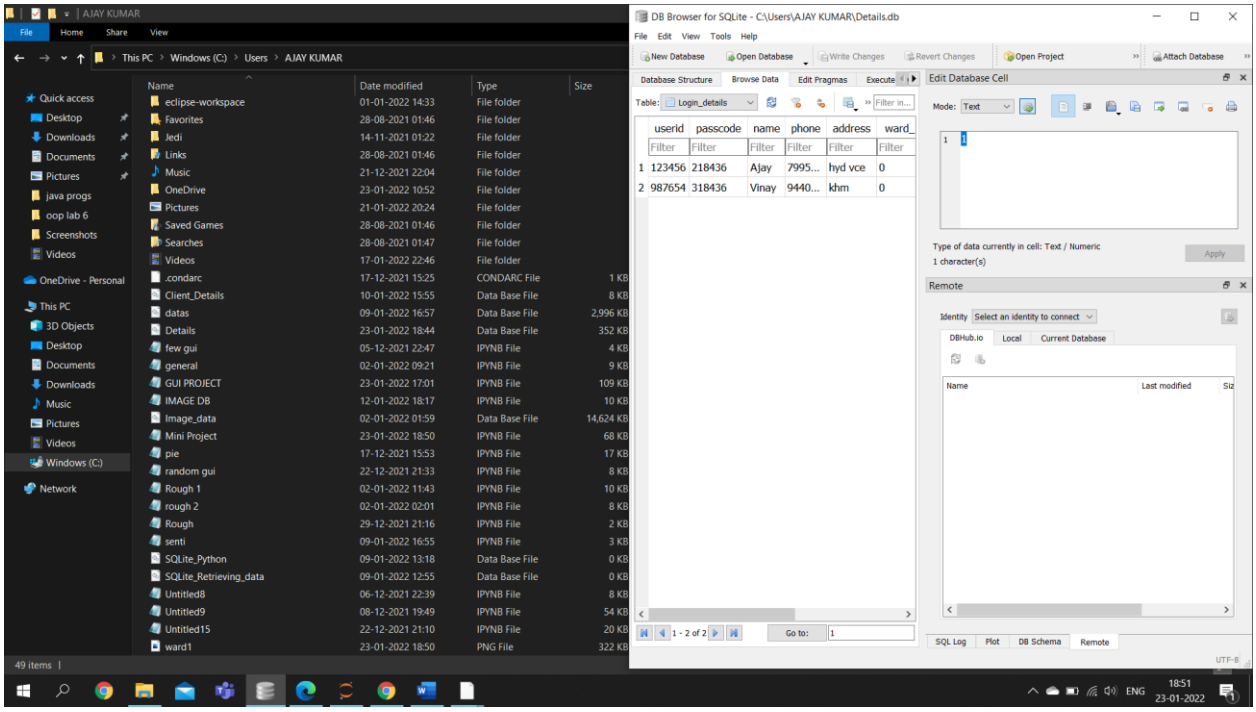
Now reviewer will also login in after successful sign up .



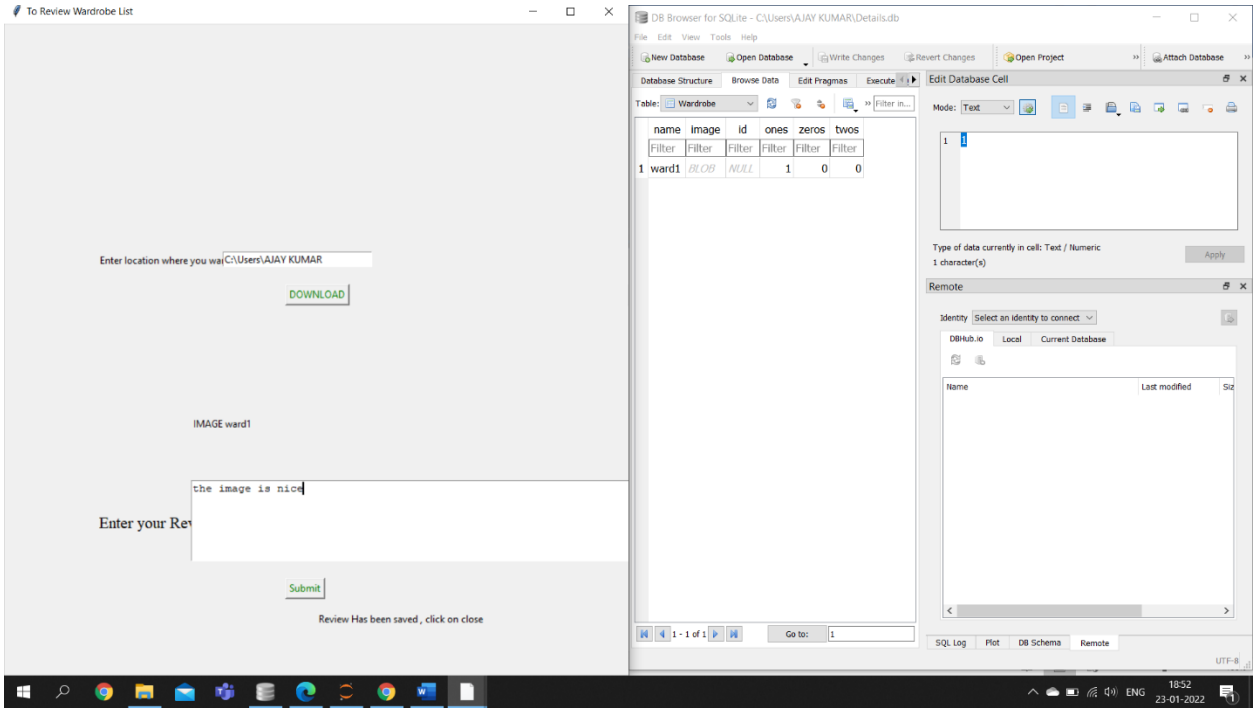
After sign up and successful login reviewer will click on give review button . Then it shows 3 buttons wardrobe , Art , product . Reviewer selects any one of button to review it .



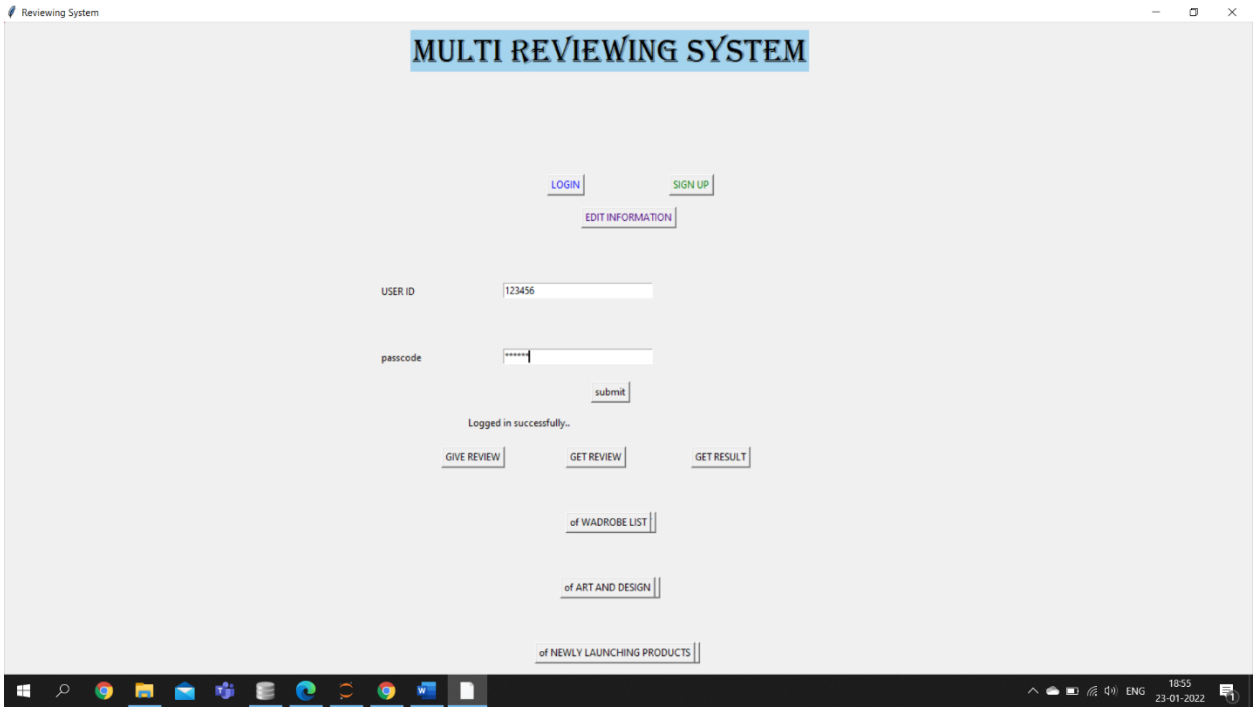
If the reviewer selects wardrobe then it asks for location where you want to download the image . Click on Download it displays text area asks for his review after entering review click on submit it gives confirmation that your review is saved.



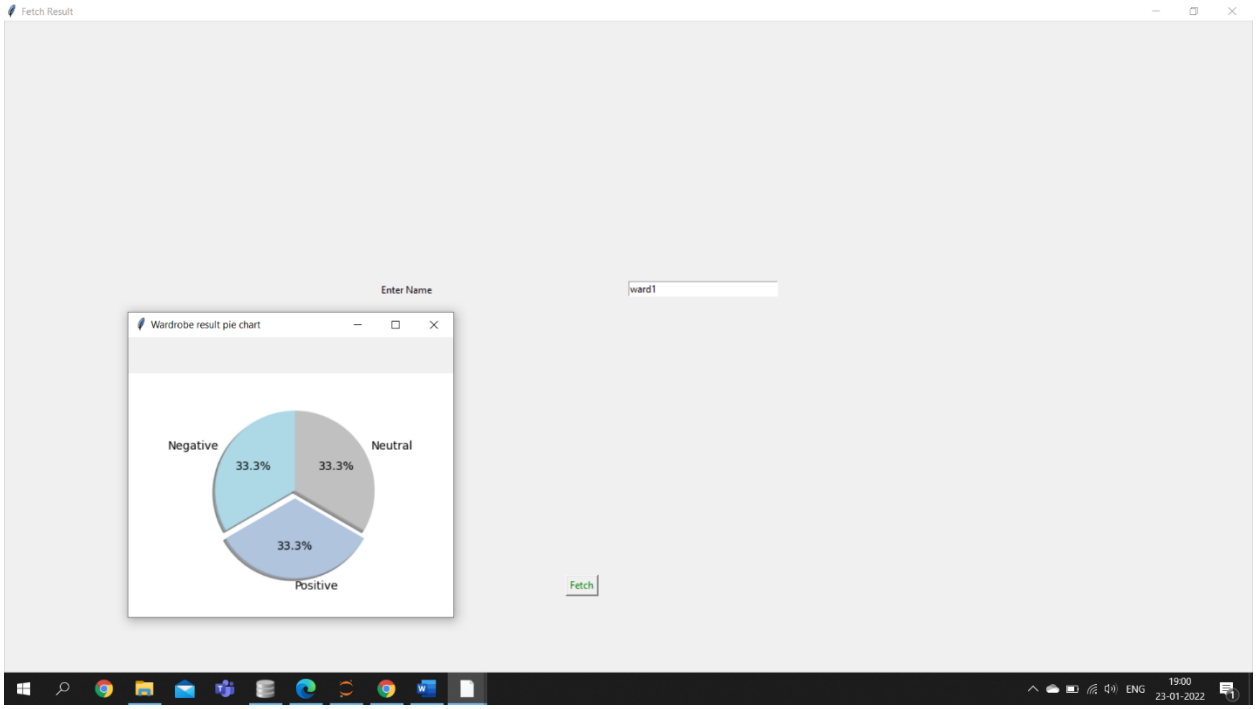
The review entered is a positive review . So ones column marked as 1



After getting few reviews when the user want result of review the user logins in and click on get result it opens window where he needs to enter image name .



After entering image name click on Fetch then it will display pie chart .



Similarly in case of Design and Newly launching Product.

ADDITIONAL KNOWLEDGE GAINED

Apart from the syllabus covered in the course python programming in 2nd semester we have learnt a lot of concepts in Python which helped us in completing our mini project.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. By importing Tkinter module and adding the widgets we can create GUI applications.

PIL (Python Imaging Library) is a free and open source additional library for python programming language that adds support for opening , manipulating , and saving many different image file formats . It is a cross platform library .

NLTK (Natural Language Toolkit) is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the python programming language . It has a special set of words known as stop words which are used to remove the unwanted words for analysing the tokens of a sentence .

SQLite3 can be integrated with Python using sqlite3 module. It need not be installed separately for working. It provides a straightforward and simple-to-use interface for interacting with SQLite databases. The module sqlite3 provides a SQL-like interface to read, query and write SQL databases from Python.

Matplotlib is a cross-platform , data visulaization and graphical plotting library of Python. matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB . It is used to embed plots in GUI applications .

CONCLUSION AND FUTURE WORK

We have developed a reviewing system where a user can get review on art, electronic products . user can get the reviewed result in the form of pie chart in that it displays number of positive reviews , negative and neutral .

We can further continue to develop this by adding a feature of giving review by the system. In this case the system will analyse the image uploaded and it will review it based on the list of comments which are already stored in the database . We can also add the voice recogonization feature near the reviewer side so that he can just give his review without even typing text .

Finally we can also build GUI in more efficient way in which the user can use it easily.

REFERENCES

NLP

- https://www.geeksforgeeks.org/introduction-to-natural-language-processing/

Sqlite3

- https://docs.python.org/3/library/sqlite3.html

GUI

- https://www.geeksforgeeks.org/python-gui-tkinter/

PIL

- https://www.javatpoint.com/python-pillow

NLTK

- https://www.nltk.org/api/nltk.sentiment.util.html

MATPLOTLIB

- https://www.w3schools.com/python/matplotlib_pyplot.asp