**D.A.V. SENIOR SECONDARY SCHOOL BOYS MOGAPPAIR CH-50**



**COMPUTER SCIENCE PROJECT**

**ON**

# GRAPH EQUATION SIMULATOR

Name:

Class and Section: XII B

Year 2023-24

**BONAFIDE CERTIFICATE**

Register Number:

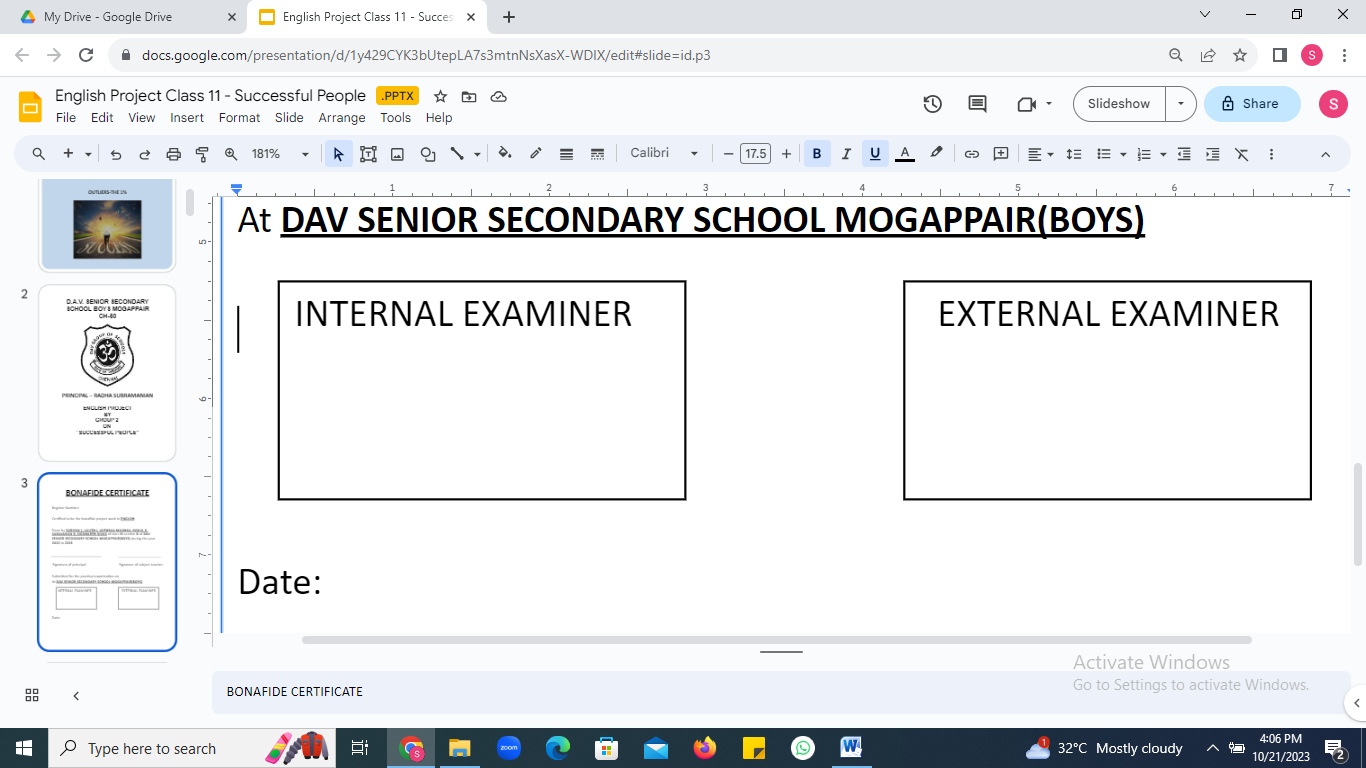
Certified to be the Bonafide project work in **Computer science**. Done by **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  of class **XII** section **B** of **DAV SENIOR SECONDARY SCHOOL MOGAPPAIR (BOYS)** during the year **2023** to **2024.**

Signature of principal                                      Signature of subject teacher

SCHOOL SEAL

Submitted for the practical examination on

At **DAV SENIOR SECONDARY SCHOOL MOGAPPAIR(BOYS)**



Date:

**CONTENT**

1 Acknowledgement

2 Introduction

3 Concepts used

4 Modules imported

5 Source Code

6 Outputs

7 Bibliography

**Acknowledgement**

I would like to express my gratitude and sincere thanks to my Computer science teacher Mrs. Praveena for her able guidance and support in completing this project.

I would also like to thank our Principal Mrs. Bhuvaneshwari for giving us this golden opportunity of doing this project.

Lastly I would like to thank my parents and friends for their constant support and valuable guidance.

**Introduction**

Graph equation simulator is a platform use by students and teachers to draw graphs for mathematical equations and shapes. It runs a menu driven program with options to draw specific graphs. It gives the graph of the equation in entered by the user in a two dimensional plot. Facility for user to be able to create a user id and login through it is provided. User can select the required options to get their graph.

**Concepts Used**

Binary file operation

Tkinter

Graph plotting operation

**Modules Imported**

**Tkinter:** tkinter package (Tk interface) is the standard Python interface to the Tk GUI toolkit.

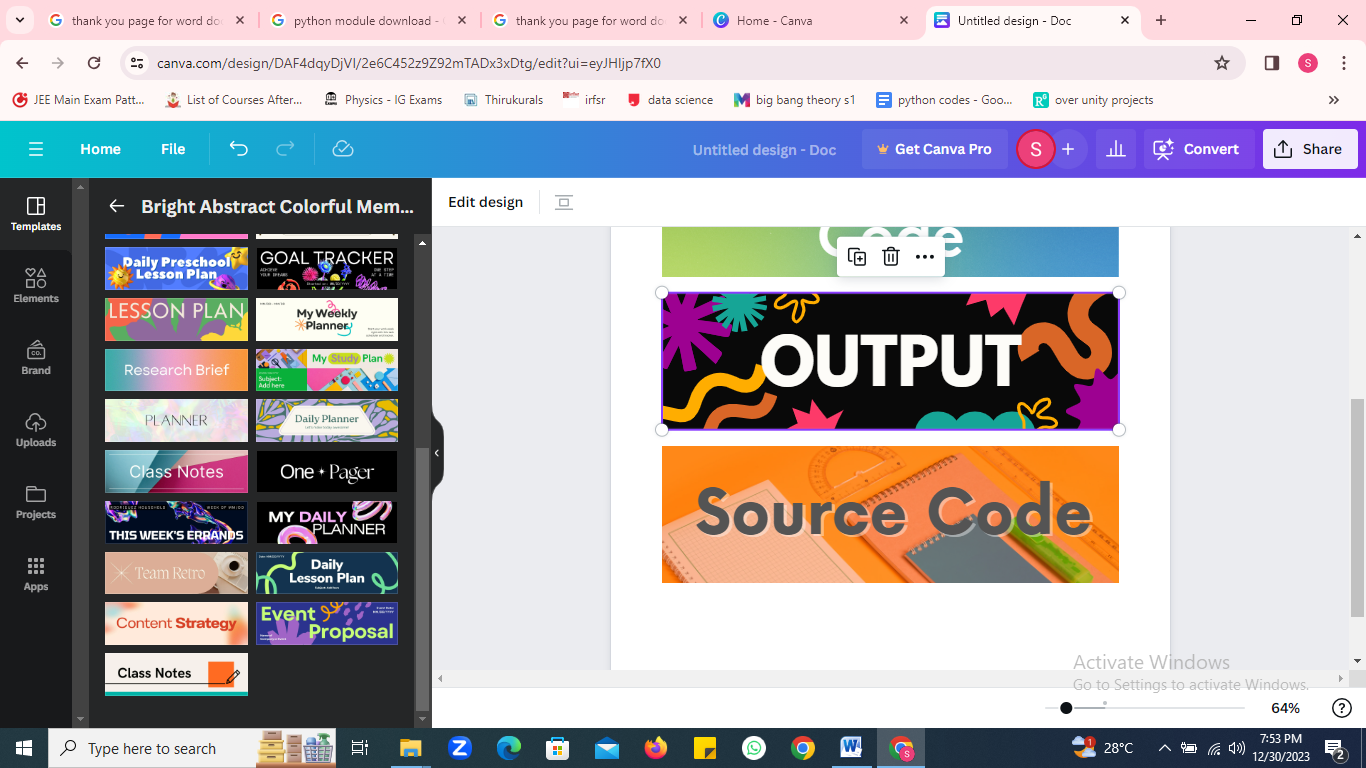
**Time**- this module supplies classes for manipulating time.

**Matplotlib**- it is a cross platform data visualization and graphical plotting library for Python and its numerical extension

**NumPy**- it is used to perform a wide variety of high-level mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices.

**Pickle**- it can transform a complex object into a byte stream and it can transform the byte stream into an object with the same internal structure.

Sympy: it is used to generate reference values for unit tests and some code generation.It can solve equations and simplify expressions.



import tkinter as tk

from tkinter import messagebox

import pickle

import time

import sys

import numpy as np

from matplotlib.patches import Ellipse

import matplotlib.pyplot as plt

import math

from sympy import \*

from sympy.plotting import plot, plot\_implicit, plot\_parametric

class GraphSimulator:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Graph Equation Simulator")

self.root.geometry("500x500")

# Account list to store username and password

global accountlist

accountlist = [ ("suri", "done"),]

with open("Account4",'ab') as f:

pickle.dump(accountlist, f)

# Create widgets

self.heading\_label = tk.Label(root, text="Graph Equation Simulator", font=('Helvetica', 16, 'bold'), fg='dark blue')

self.welcome\_message = tk.Label(root, text="Welcome to graph equation simulator. \n"

"This program allows user to \n"

"input any 2 variable equation and then give output as its graph. \n"

"You have to use decimals instead of fractions.", justify='left')

self.old\_user\_button = tk.Button(root, text="Old User Login", command=self.old\_user\_login)

self.new\_user\_button = tk.Button(root, text="New User Signup", command=self.new\_user\_signup)

# Place widgets on the grid

self.heading\_label.grid(row=0, column=0, columnspan=2, pady=10)

self.welcome\_message.grid(row=1, column=0, columnspan=2, pady=10, padx=20)

self.old\_user\_button.grid(row=2, column=0, pady=10, padx=20)

self.new\_user\_button.grid(row=2, column=1, pady=10, padx=20)

def old\_user\_login(self):

# Create a new window for login

login\_window = tk.Toplevel(self.root)

login\_window.title("Old User Login")

login\_window.geometry("500x500")

# Create login widgets

username\_label = tk.Label(login\_window, text="Username:")

password\_label = tk.Label(login\_window, text="Password:")

self.username\_entry = tk.Entry(login\_window)

self.password\_entry = tk.Entry(login\_window, show="\*")

login\_button = tk.Button(login\_window, text="Login", command=self.check\_login\_credentials)

# Place login widgets on the grid

username\_label.grid(row=0, column=0, pady=5, padx=20)

password\_label.grid(row=1, column=0, pady=5, padx=20)

self.username\_entry.grid(row=0, column=1, pady=5, padx=20)

self.password\_entry.grid(row=1, column=1, pady=5, padx=20)

login\_button.grid(row=2, column=0, columnspan=2, pady=10)

def check\_login\_credentials(self):

# Check if the entered credentials are in the account list

username = self.username\_entry.get()

password = self.password\_entry.get()

data = []

try:

with open("Accounts4.dat",'rb') as f:

data = pickle.load(f)

except FileNotFoundError:

print("File not found.")

global accountlist

print(accountlist)

print(data)

if (username, password) in accountlist:

messagebox.showinfo("Login Successful", "Login successful!")

self.empty\_command()

else:

messagebox.showerror("Login Failed", "Invalid username or password. Please try again.")

def new\_user\_signup(self):

# Create a new window for signup

signup\_window = tk.Toplevel(self.root)

signup\_window.title("New User Signup")

signup\_window.geometry("500x500")

# Create signup widgets

new\_username\_label = tk.Label(signup\_window, text="New Username:")

new\_password\_label = tk.Label(signup\_window, text="New Password:")

strong\_password\_label = tk.Label(signup\_window, text="Set a strong password of at least eight characters")

self.new\_username\_entry = tk.Entry(signup\_window)

self.new\_password\_entry = tk.Entry(signup\_window, show="\*")

signup\_button = tk.Button(signup\_window, text="Signup", command=self.save\_new\_user)

# Place signup widgets on the grid

new\_username\_label.grid(row=0, column=0, pady=5, padx=20)

new\_password\_label.grid(row=1, column=0, pady=5, padx=20)

strong\_password\_label.grid(row=2, column=0, columnspan=2, pady=5, padx=20)

self.new\_username\_entry.grid(row=0, column=1, pady=5, padx=20)

self.new\_password\_entry.grid(row=1, column=1, pady=5, padx=20)

signup\_button.grid(row=5, column=0, columnspan=2, pady=10)

def save\_new\_user(self):

# Save new user credentials to the account list

new\_username = self.new\_username\_entry.get()

new\_password = self.new\_password\_entry.get()

if len(new\_password) < 8:

messagebox.showerror("Weak Password", "Please set a strong password of at least eight characters.")

return

global accountlist

accountlist.append((new\_username, new\_password))

with open("Account4.dat",'ab') as f:

pickle.dump(accountlist, f)

print(accountlist)

messagebox.showinfo("Signup Successful", "Signup successful!")

# Perform additional actions after successful signup (empty command for now)

self.empty\_command()

def empty\_command(self):

# Placeholder for additional actions after successful login/signup

import tkinter as tk

def makeline():

def putline():

slope=int(slope\_entry.get())

intercept=int(intercept\_entry.get())

print(slope)

print(intercept)

#to add graph

if slope!=0:

fig, ax = plt.subplots()

plt.plot([0, (100-intercept)/slope], [intercept, 100], marker='o', color='black',linestyle='-', linewidth=1)

ax.set\_aspect('equal', adjustable='box')

ax.set\_xlim(0, 100)

ax.set\_ylim(0, 100)

plt.title("Line Plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.grid(True)

plt.show()

rootline=tk.Tk()

rootline.title("Straight Line Equation")

rootline.geometry("600x300")

line\_frame = tk.Frame(rootline)

line\_frame.pack(pady=20)

tk.Label(line\_frame, text="Lines are in the form y=mx+c. Enter your slope value and y intercept value in the dialog box").grid(row=0, column=0, padx=10, pady=10)

line2\_frame = tk.Frame(rootline)

line2\_frame.pack(pady=20)

tk.Label(line2\_frame, text=" Y = ").grid(row=1, column=0, padx=10, pady=10)

slope\_entry = tk.Entry(line2\_frame)

slope\_entry.grid(row=1, column=1, padx=10, pady=10)

tk.Label(line2\_frame, text=" X +").grid(row=1, column=2, padx=10, pady=10)

intercept\_entry = tk.Entry(line2\_frame)

intercept\_entry.grid(row=1, column=3, padx=10, pady=10)

tk.Button(line2\_frame, text="Go", command=putline).grid(row=2, column=4, columnspan=2, pady=10)

rootline.mainloop()

pass

def makecircle():

def putcircle():

h=int(h\_entry.get())

k=int(k\_entry.get())

r=int(r\_entry.get())

if r!=0:

plt.scatter(h, k, s=r\*3200, facecolors='none', edgecolors='black')

plt.xlim(h-5, k+5)

plt.ylim(h-5, k+5)

plt.show()

rootcircle=tk.Tk()

rootcircle.title("Circle Equation")

rootcircle.geometry("800x300")

circle\_frame = tk.Frame(rootcircle)

circle\_frame.pack(pady=20)

tk.Label(circle\_frame, text="Circles are in the form (x-h)^2 + (y-k)^2 = r^2. Enter the respective values in the dialog box").grid(row=0, column=0, padx=10, pady=10)

circle2\_frame = tk.Frame(rootcircle)

circle2\_frame.pack(pady=20)

tk.Label(circle2\_frame, text=" (X-").grid(row=1, column=0, padx=10, pady=10)

h\_entry = tk.Entry(circle2\_frame)

h\_entry.grid(row=1, column=1, padx=10, pady=10)

tk.Label(circle2\_frame, text=" )^2 + (Y-").grid(row=1, column=2, padx=10, pady=10)

k\_entry = tk.Entry(circle2\_frame)

k\_entry.grid(row=1, column=3, padx=10, pady=10)

tk.Label(circle2\_frame, text=" )^2 = ").grid(row=1, column=4, padx=10, pady=10)

r\_entry = tk.Entry(circle2\_frame)

r\_entry.grid(row=1, column=5, padx=10, pady=10)

tk.Button(circle2\_frame, text="Go", command=putcircle).grid(row=2, column=4, columnspan=2, pady=10)

tk.Label(circle2\_frame, text=")^2 ").grid(row=1, column=6, padx=10, pady=10)

rootcircle.mainloop()

h=h\_entry.get()

k=k\_entry.get()

r=r\_entry.get()

#graphing

plt.scatter(h, k, s=r\*3200, facecolors='none', edgecolors='black')

plt.xlim(h-5, k+5)

plt.ylim(h-5, k+5)

plt.show()

pass

def makeparabola():

def putparabola():

a=0

b=0

a=int(a\_entry.get())

b=int(b\_entry.get())

#work only if the other unused space is filled zero

#graphing

if a != 0 or b!=0:

if a !=0:

y = np.linspace(-10, 10, 400)

x = (y\*\*2)/(4\*a)

plt.plot(x, y)

plt.title("Graph of Horizontal parabola")

if b !=0:

x = np.linspace(-10, 10, 400)

#dont use this gives zero error

y = (x\*\*2)/(4\*b)

plt.plot(x, y)

plt.title("Graph of Vertical Parabola")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.grid(True)

plt.show()

rootparabola=tk.Tk()

rootparabola.title("Parabola Equation")

rootparabola.geometry("800x300")

parabola\_frame = tk.Frame(rootparabola)

parabola\_frame.pack(pady=20)

tk.Label(parabola\_frame, text="Standard Parabolas are in the form Y^2 = 4AX (Horizontal) and X^2 = 4AY. Enter the respective values in the dialog box.").grid(row=0, column=0, padx=10, pady=10)

parabola2\_frame = tk.Frame(rootparabola)

parabola2\_frame.pack(pady=20)

tk.Label(parabola2\_frame, text="horizontal parabola-Enter focus").grid(row=1, column=0, padx=10, pady=10)

a\_entry = tk.Entry(parabola2\_frame)

a\_entry.grid(row=1, column=1, padx=10, pady=10)

tk.Label(parabola2\_frame, text="vertical parabola-Enter focus").grid(row=1, column=2, padx=10, pady=10)

b\_entry = tk.Entry(parabola2\_frame)

b\_entry.grid(row=1, column=3, padx=10, pady=10)

tk.Button(parabola2\_frame, text="Go", command=putparabola).grid(row=2, column=4, columnspan=2, pady=10)

pass

def makeellipse():

from matplotlib.patches import Ellipse

def putellipse():

a=int(a\_entry.get())

b=int(b\_entry.get())

#graphing

fig,ax = plt.subplots()

ellipse = Ellipse((0,0), a, b, angle=0, fill=False, color='black')

ax.add\_patch(ellipse)

ax.set\_aspect('equal', adjustable='box')

ax.set\_xlim(-10, 10)

ax.set\_ylim(-10, 10)

plt.title("Ellipse")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.grid(True)

plt.show()

rootellipse=tk.Tk()

rootellipse.title("Ellipse Equation")

rootellipse.geometry("800x300")

ellipse\_frame = tk.Frame(rootellipse)

ellipse\_frame.pack(pady=20)

tk.Label(ellipse\_frame, text="Standard Ellipse are in the form (X/A)^2 + (Y/B)^2 = 1. Enter the respective values in the dialog box").grid(row=0, column=0, padx=10, pady=10)

ellipse2\_frame = tk.Frame(rootellipse)

ellipse2\_frame.pack(pady=20)

tk.Label(ellipse2\_frame, text="Enter width or A value").grid(row=1, column=0, padx=10, pady=10)

a\_entry = tk.Entry(ellipse2\_frame)

a\_entry.grid(row=1, column=1, padx=10, pady=10)

tk.Label(ellipse2\_frame, text="Enter ehight or B value").grid(row=1, column=2, padx=10, pady=10)

b\_entry = tk.Entry(ellipse2\_frame)

b\_entry.grid(row=1, column=3, padx=10, pady=10)

tk.Button(ellipse2\_frame, text="Go", command=putellipse).grid(row=2, column=4, columnspan=2, pady=10)

pass

def makehyperbola():

def puthyperbola():

a=int(a\_entry.get())

b=int(b\_entry.get())

#graphing

x, y = symbols('x y')

plot\_implicit(Eq(x\*\*2/a - y\*\*2/b, 1),(x,-10,10))

roothyperbola=tk.Tk()

roothyperbola.title("Hyperbola Equation")

roothyperbola.geometry("800x300")

hyperbola\_frame = tk.Frame(roothyperbola)

hyperbola\_frame.pack(pady=20)

tk.Label(hyperbola\_frame, text="Standard hyperbolas are in the form (X/A)^2 - (Y/B)^2 = 1. Enter the respective values in the dialog box").grid(row=0, column=0, padx=10, pady=10)

hyperbola2\_frame = tk.Frame(roothyperbola)

hyperbola2\_frame.pack(pady=20)

tk.Label(hyperbola2\_frame, text="Enter A value").grid(row=1, column=0, padx=10, pady=10)

a\_entry = tk.Entry(hyperbola2\_frame)

a\_entry.grid(row=1, column=1, padx=10, pady=10)

tk.Label(hyperbola2\_frame, text="Enter B value").grid(row=1, column=2, padx=10, pady=10)

b\_entry = tk.Entry(hyperbola2\_frame)

b\_entry.grid(row=1, column=3, padx=10, pady=10)

tk.Button(hyperbola2\_frame, text="Go", command=puthyperbola).grid(row=2, column=4, columnspan=2, pady=10)

pass

def makegeneral():

def putgeneral():

a=int(a\_entry.get())

b=int(b\_entry.get())

c=int(c\_entry.get())

#

x = np.linspace(-10, 10, 400)

y = a \* x\*\*2 + b \* x + c

plt.plot(x, y, label=f'{a}x^2 + {b}x + {c}')

# Add labels and title

plt.xlabel('x')

plt.ylabel('y')

plt.title('Quadratic Equation Graph')

# Add a legend

plt.legend()

# Show the plot

plt.grid(True)

plt.axhline(0, color='black',linewidth=0.5)

plt.axvline(0, color='black',linewidth=0.5)

plt.show()

rootgeneral=tk.Tk()

rootgeneral.title("General Quadratic Equation")

rootgeneral.geometry("800x300")

general\_frame = tk.Frame(rootgeneral)

general\_frame.pack(pady=20)

tk.Label(general\_frame, text="Quadratic equations are in the form AX^2 + BX + C. Enter the respective values in the dialog box").grid(row=0, column=0, padx=10, pady=10)

general2\_frame = tk.Frame(rootgeneral)

general2\_frame.pack(pady=20)

tk.Label(general2\_frame, text="Enter A value").grid(row=1, column=0, padx=10, pady=10)

a\_entry = tk.Entry(general2\_frame)

a\_entry.grid(row=1, column=1, padx=10, pady=10)

tk.Label(general2\_frame, text="Enter B value").grid(row=1, column=2, padx=10, pady=10)

b\_entry = tk.Entry(general2\_frame)

b\_entry.grid(row=1, column=3, padx=10, pady=10)

tk.Label(general2\_frame, text="Enter C value").grid(row=1, column=4, padx=10, pady=10)

c\_entry = tk.Entry(general2\_frame)

c\_entry.grid(row=1, column=5, padx=10, pady=10)

tk.Button(general2\_frame, text="Go", command=putgeneral).grid(row=2, column=4, columnspan=2, pady=10)

pass

def exitapp():

print("EXITING")

messagebox.showinfo("You will be exited in 3 seconds","You will be exited in 3 seconds")

print("You will be exited in 3 seconds")

time.sleep(3)

root2.destroy()

sys.exit()

root2 = tk.Tk()

root2.title("GRAPH EQUATION SIMULATOR")

root2.geometry("800x700")

main\_frame = tk.Frame(root2)

main\_frame.pack(pady=20)

tk.Label(main\_frame, text="GRAPH EQUATION SIMULATOR", bg="orange", font='20').grid(row=0, column=0, columnspan=8, padx=10, pady=10)

tk.Label(main\_frame, text="Choose any of the following buttons to proceed and give your equation").grid(row=1, column=0, columnspan=8, padx=10, pady=10)

buttons = [

("Straight Line", makeline),

("Circle", makecircle),

("Parabola", makeparabola),

("Ellipse", makeellipse),

("Hyperbola", makehyperbola),

("General Quadratic", makegeneral),

("Exit Application", exitapp)

]

row\_index = 3

for button\_text, command in buttons:

tk.Button(main\_frame, text=button\_text, command=command).grid(row=row\_index, column=buttons.index((button\_text, command)) + 1, padx=10, pady=10)

root2.mainloop()

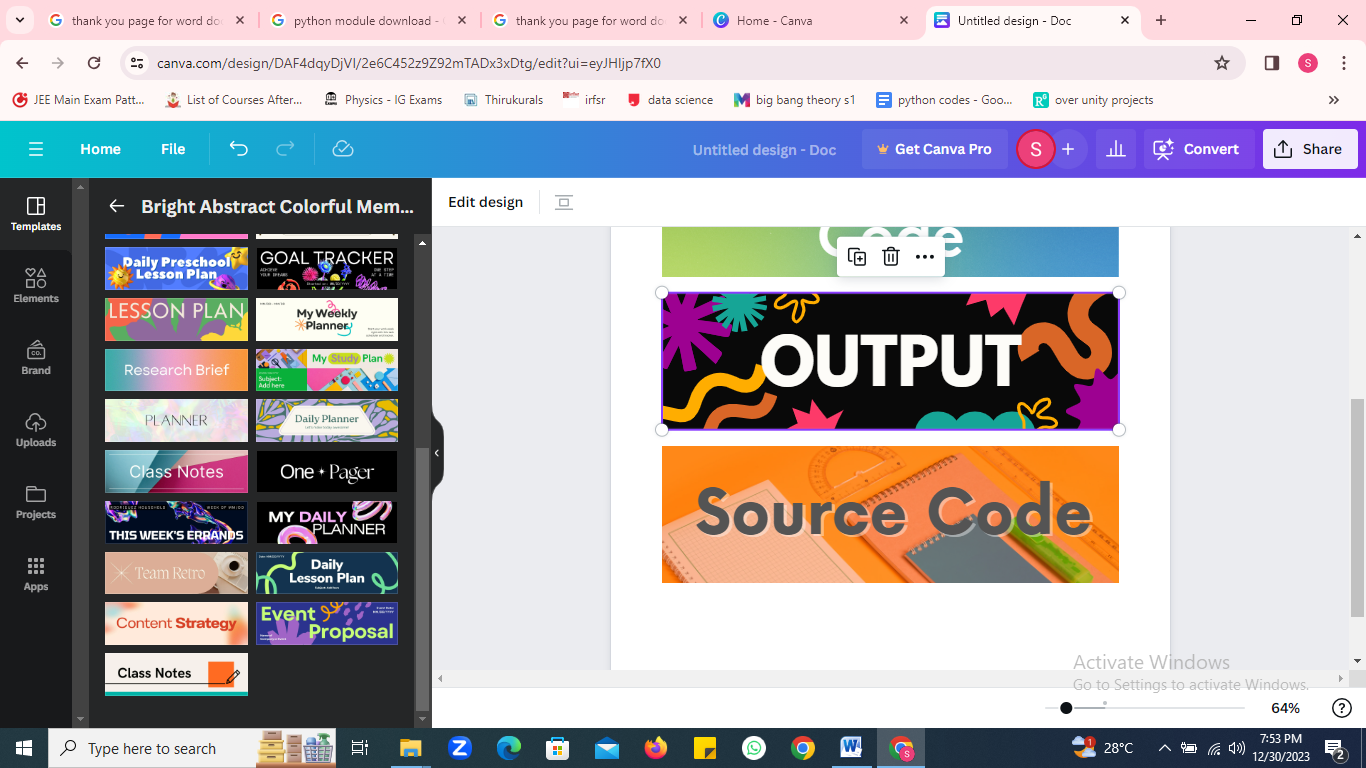
if \_\_name\_\_ == "\_\_main\_\_":

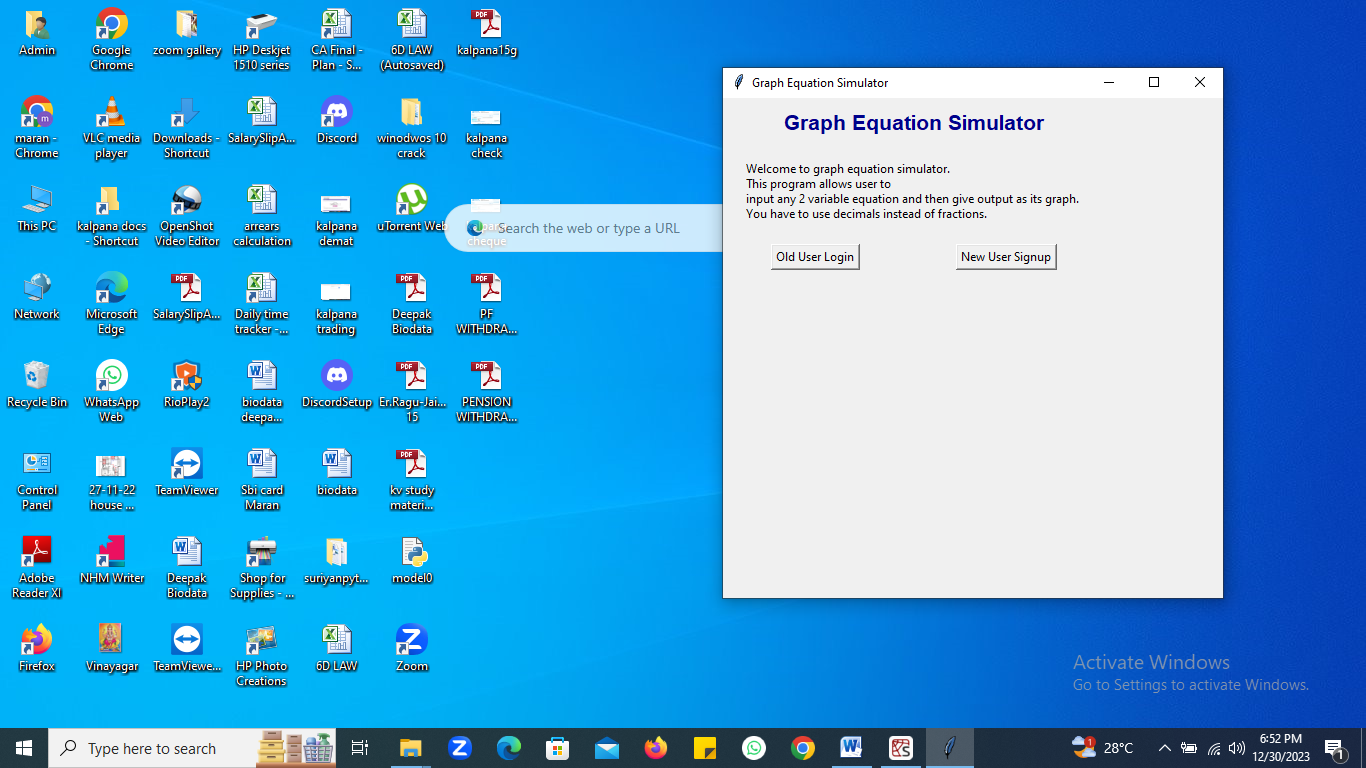
accountlist=[]

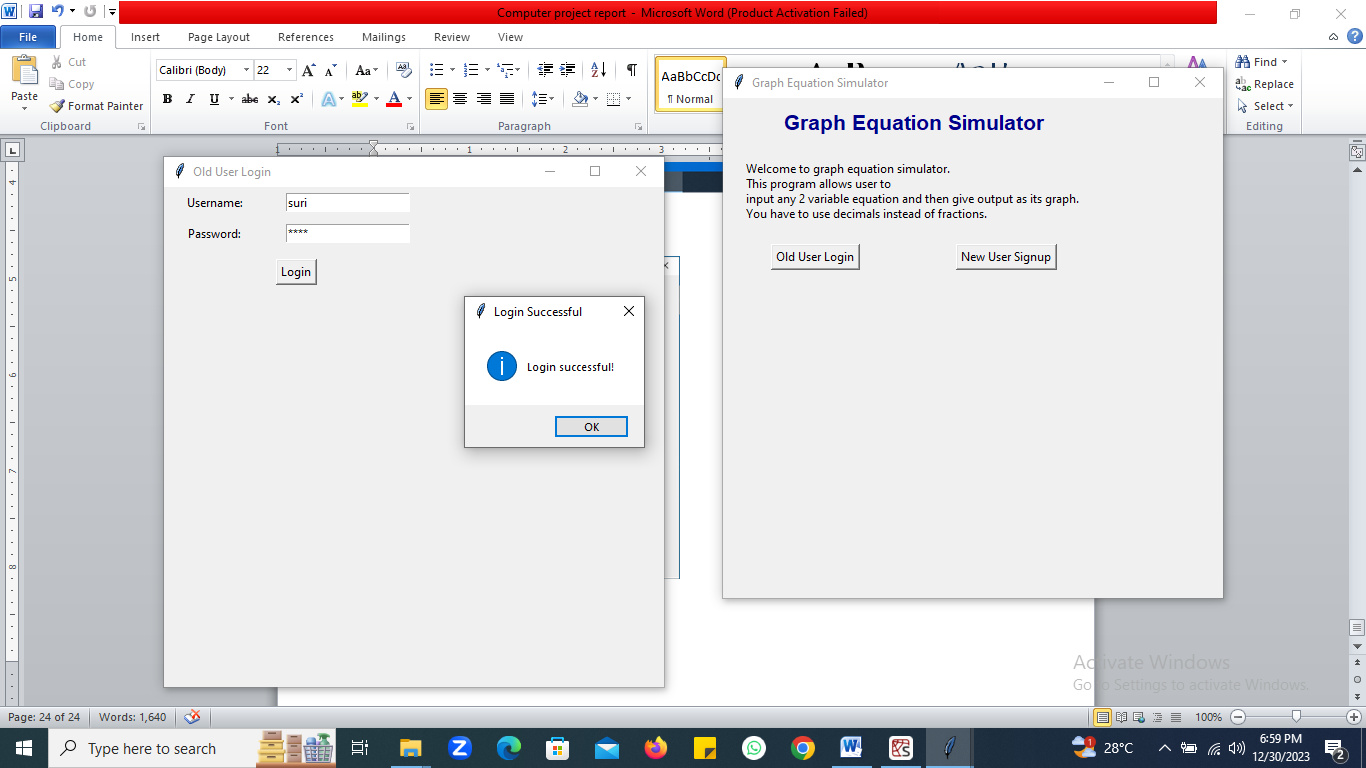
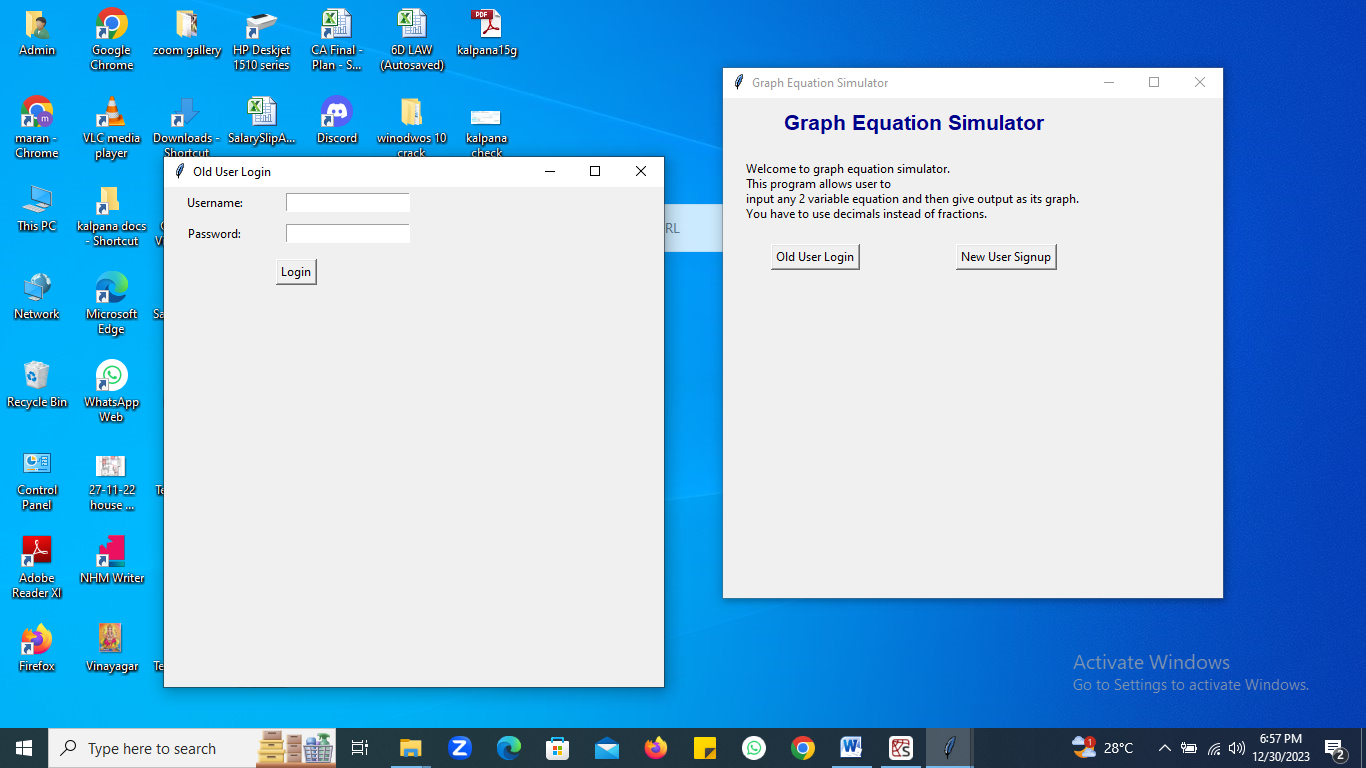
root = tk.Tk()

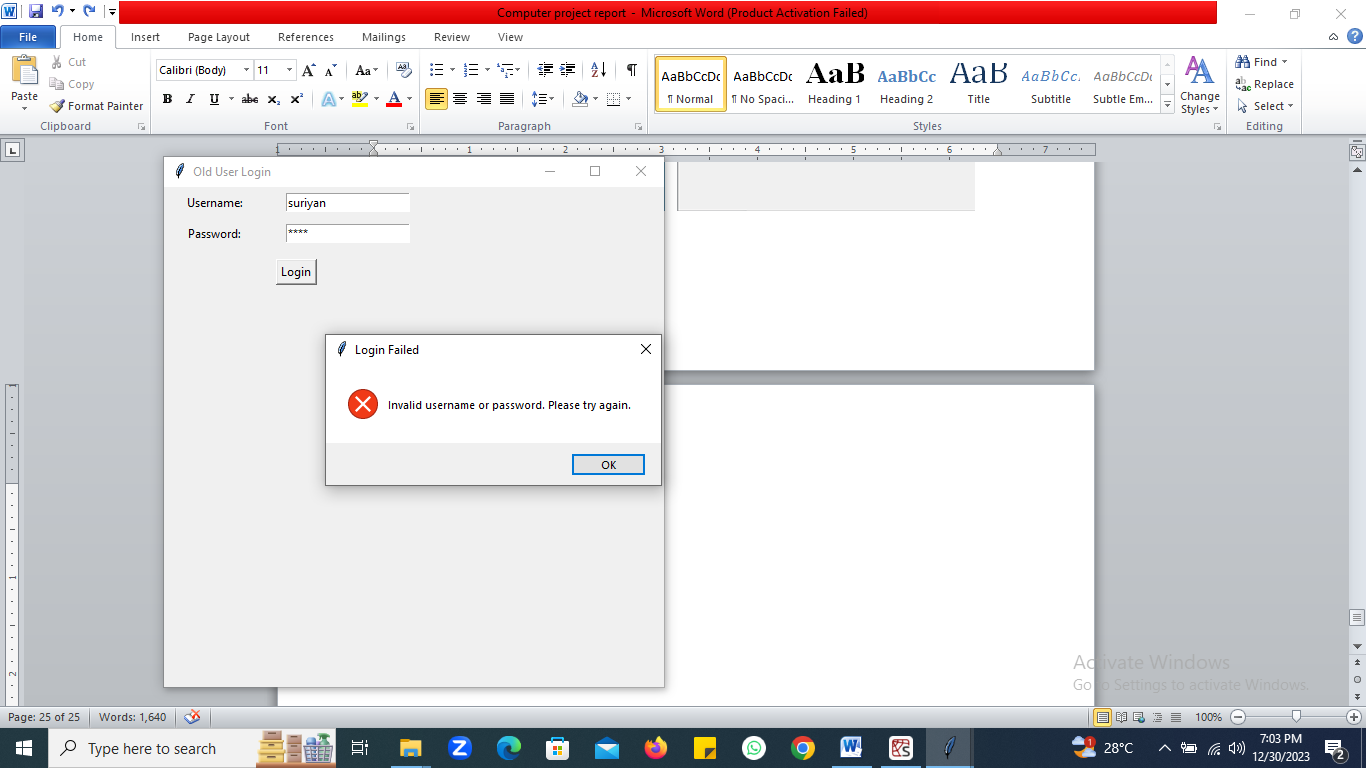
app = GraphSimulator(root)

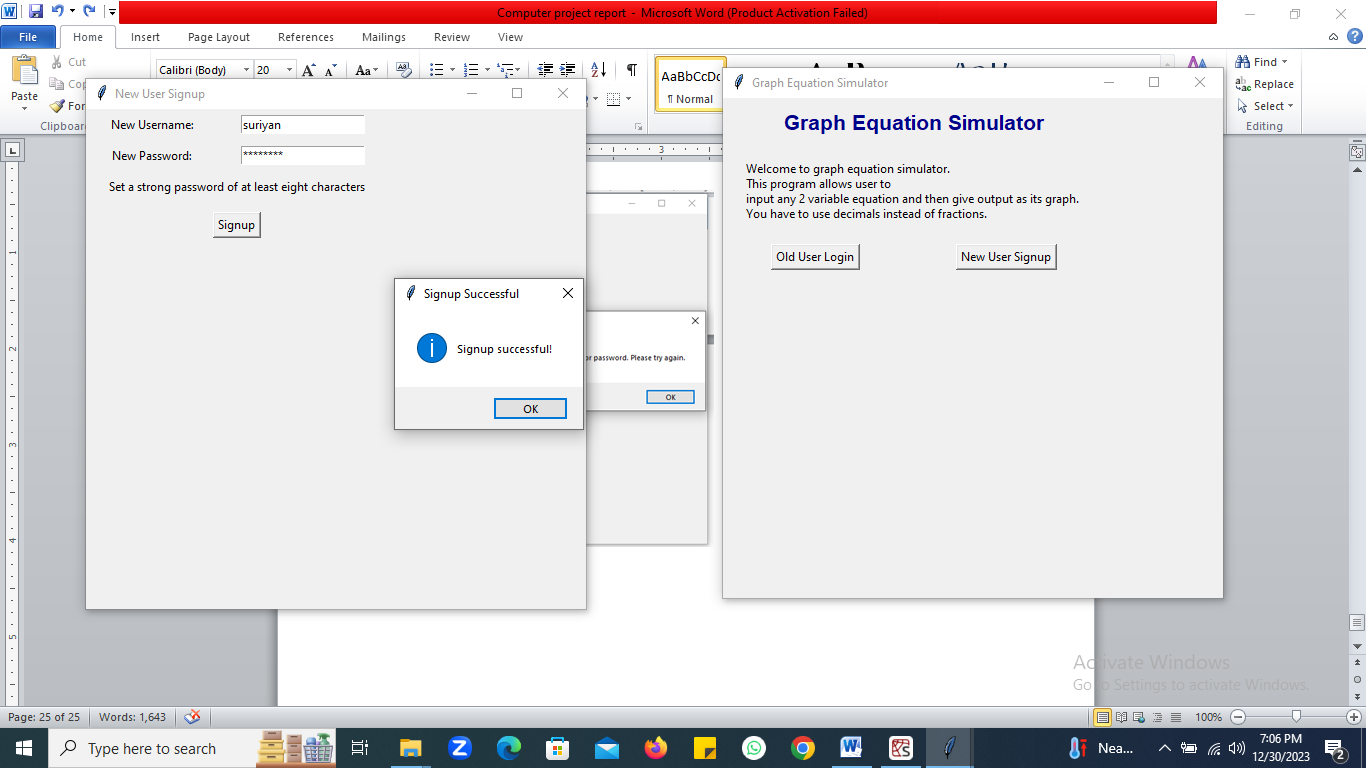
root.mainloop()

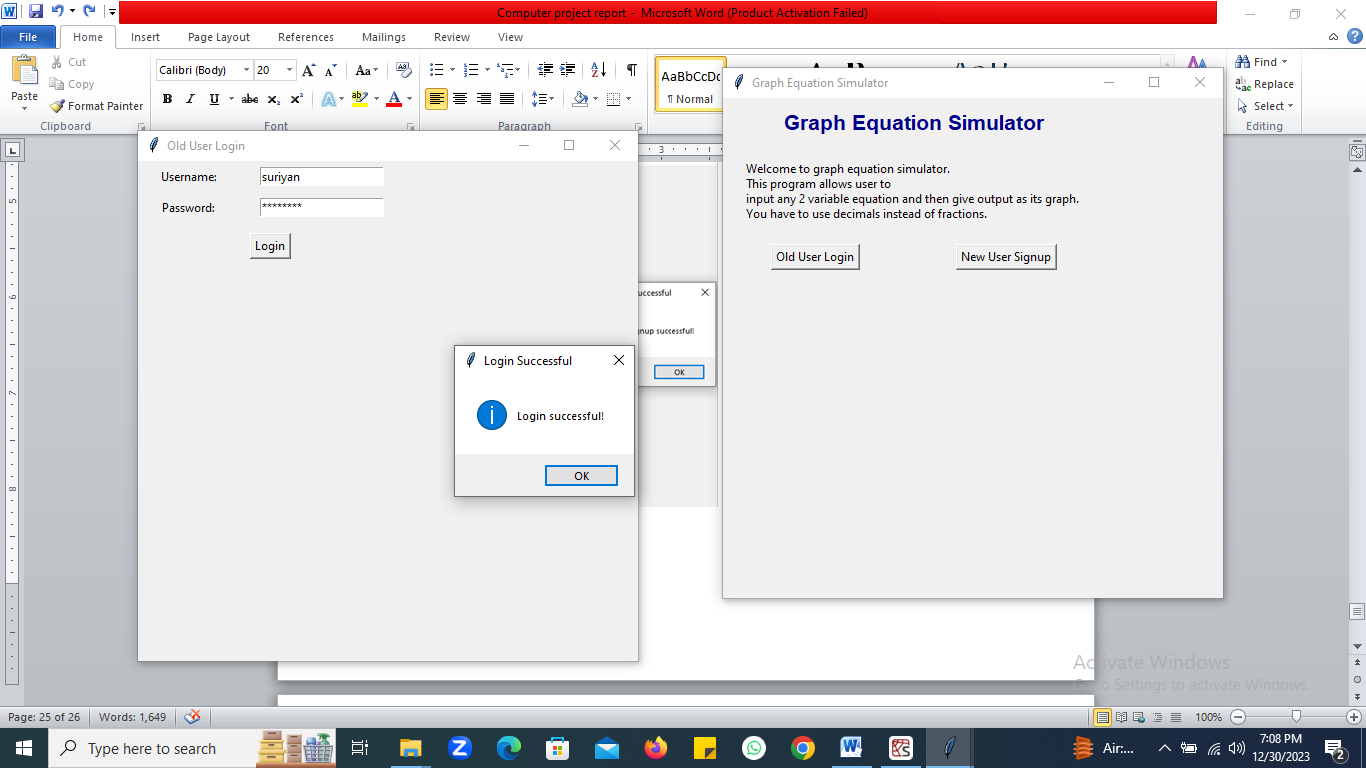


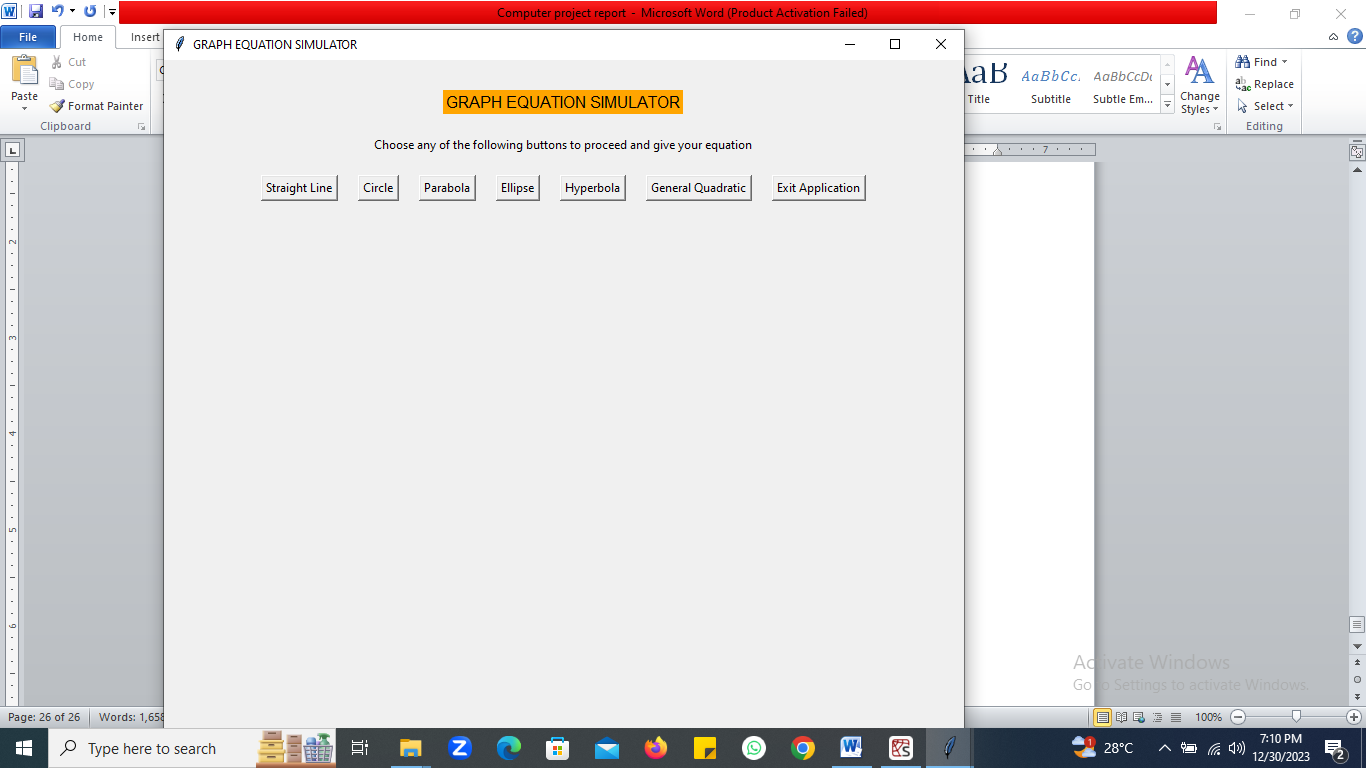


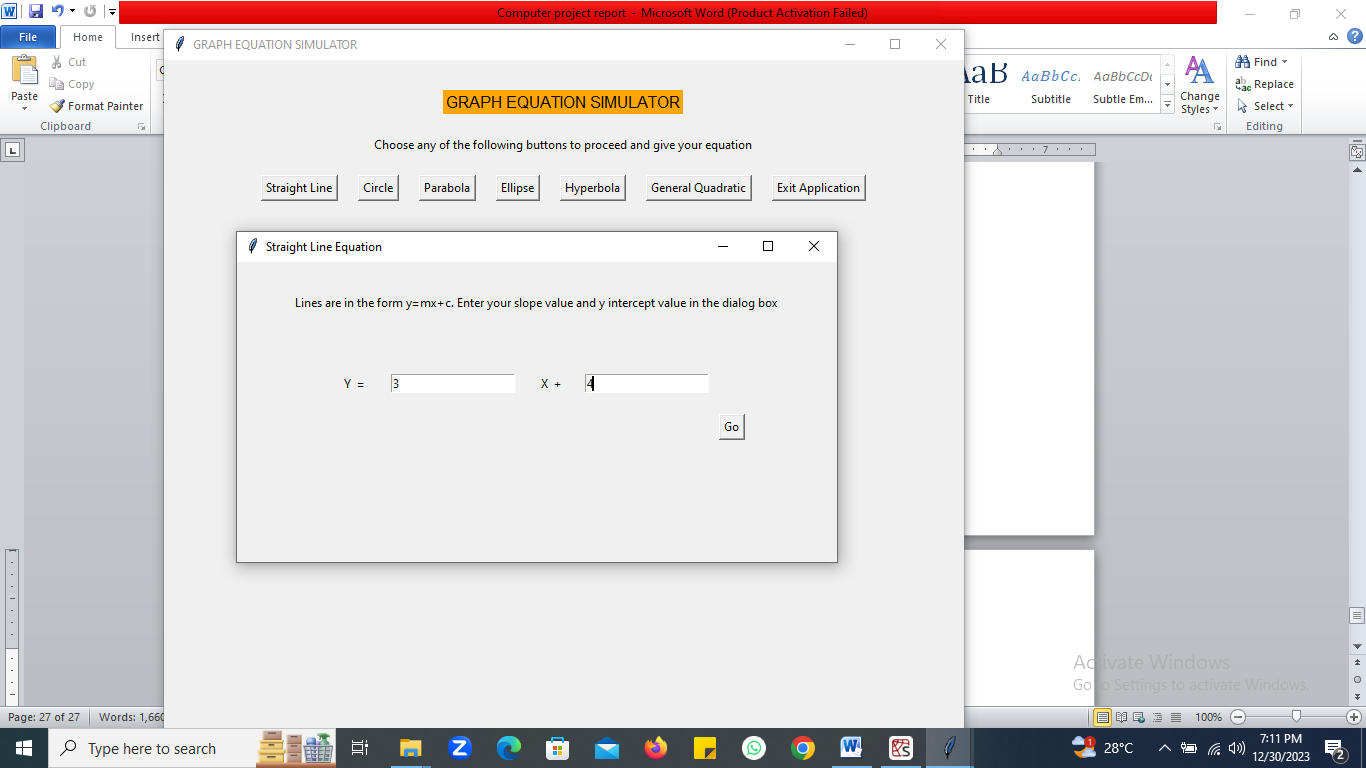


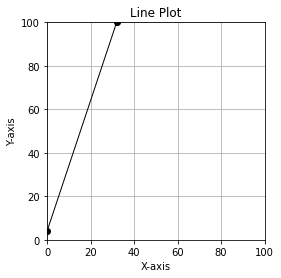
(wrong user id)

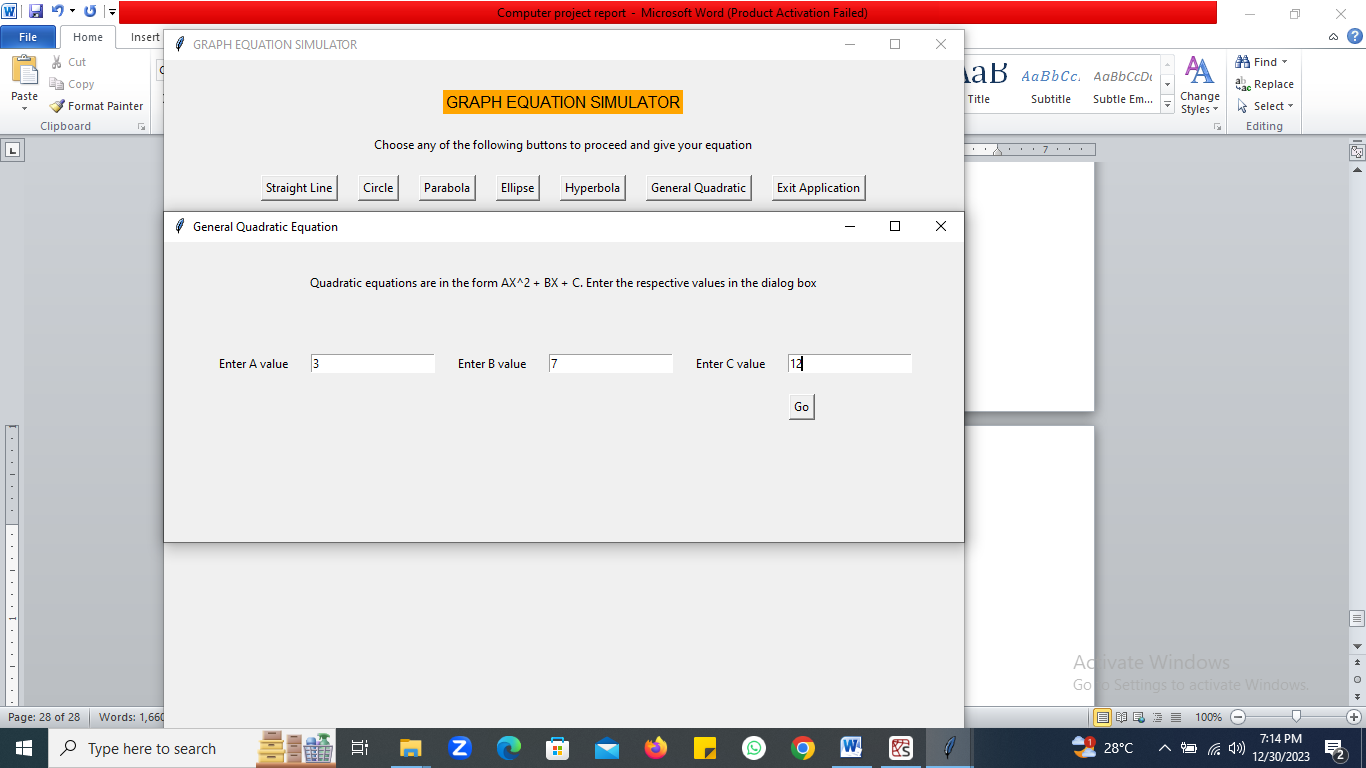
(Sign up page for new user)

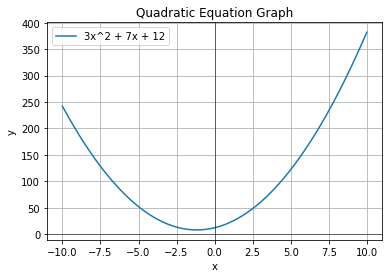
(The new user can login after succesful sign up)

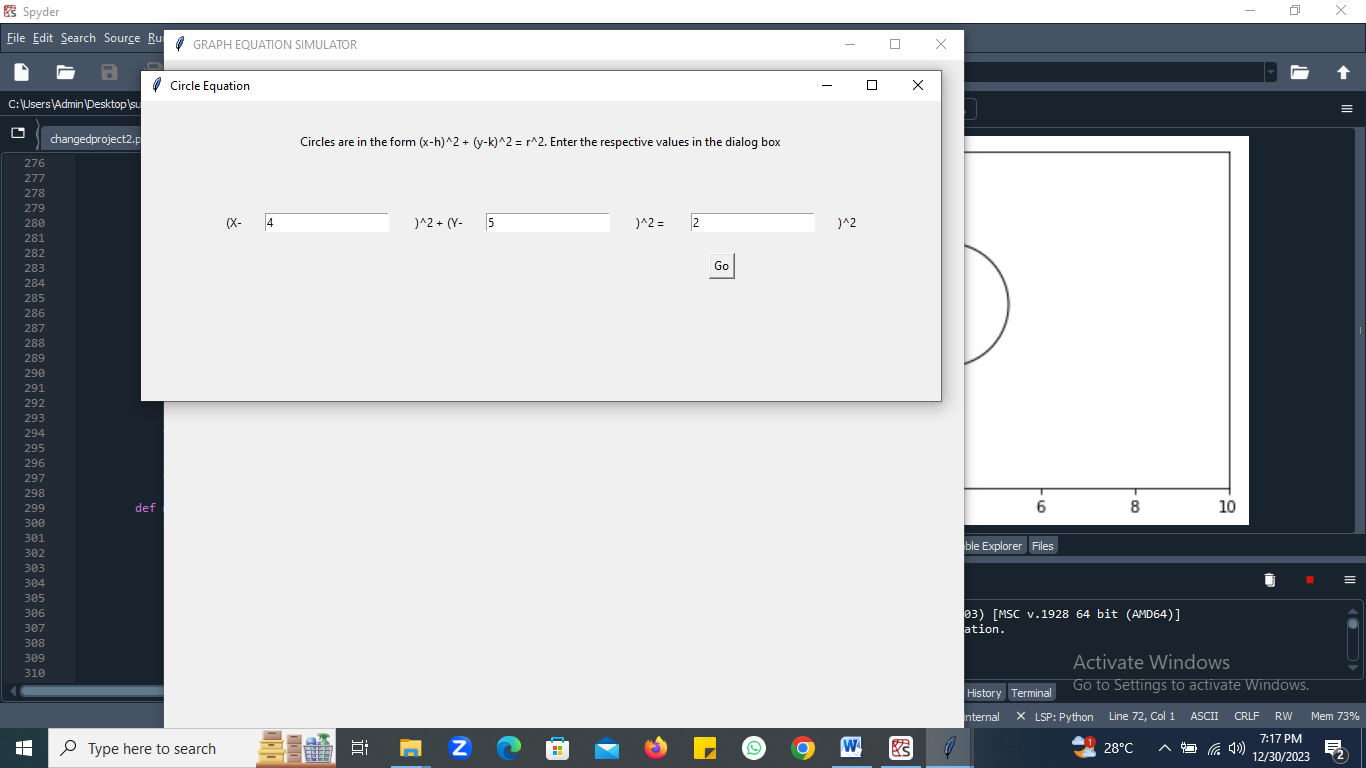
  
(main page)

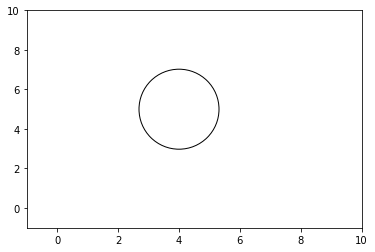


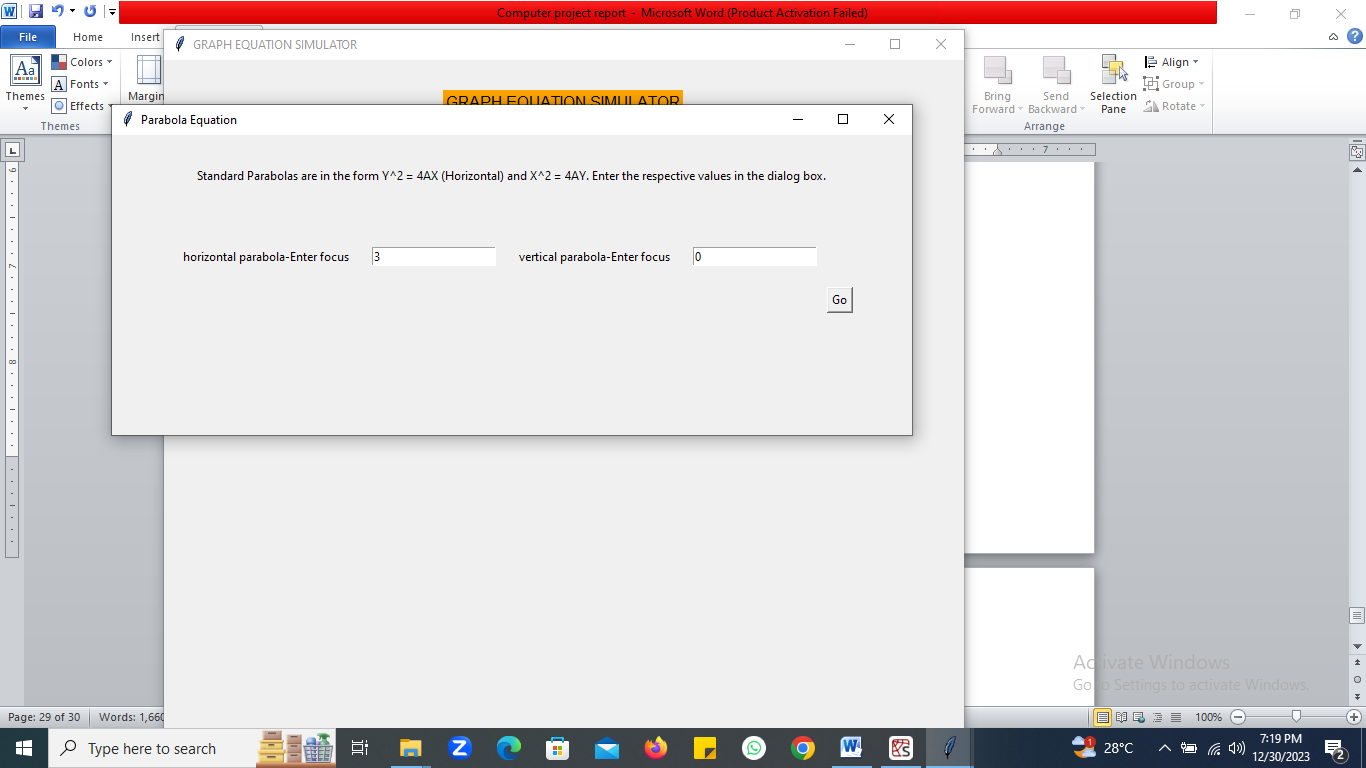


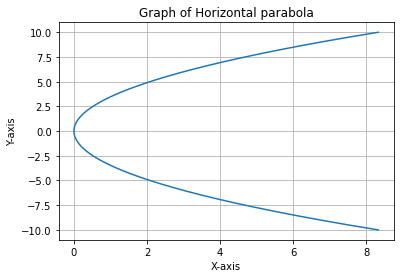


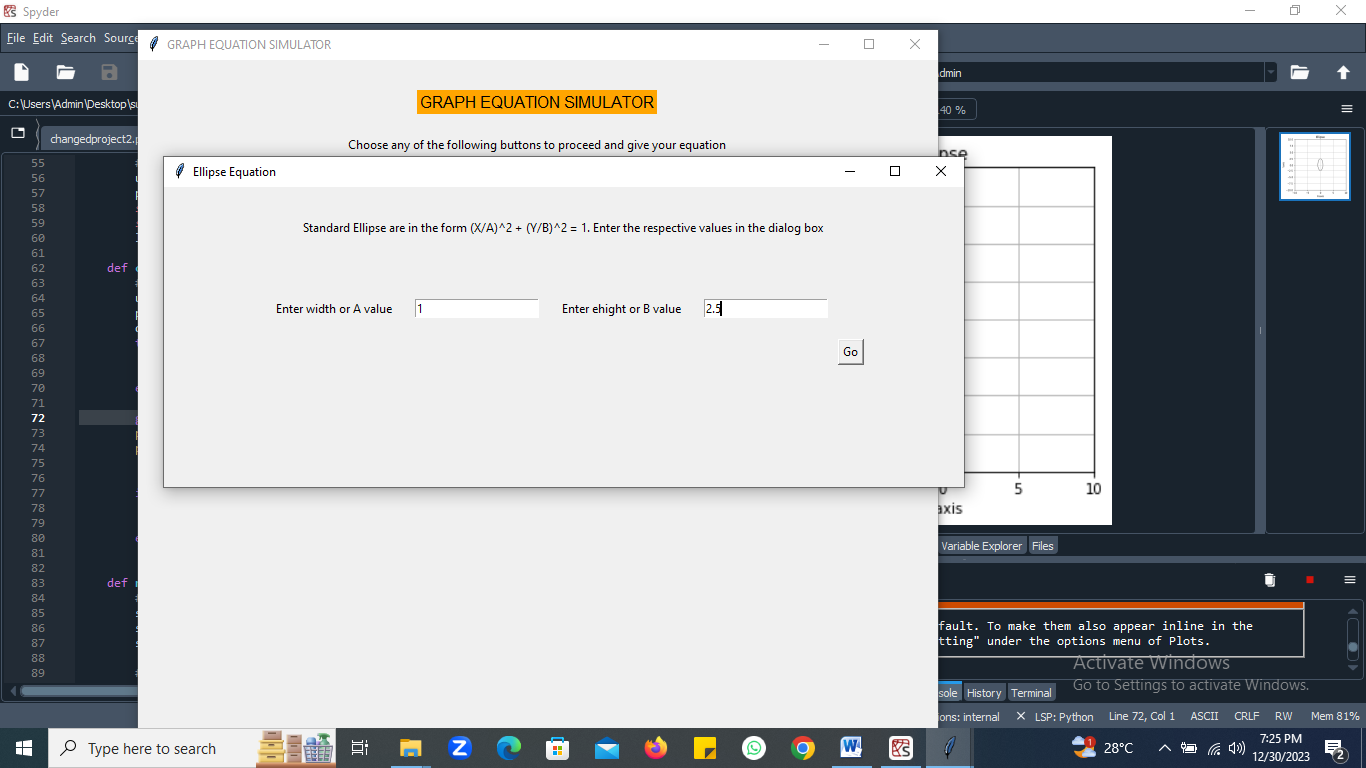


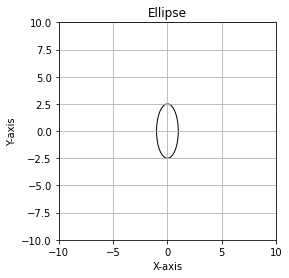


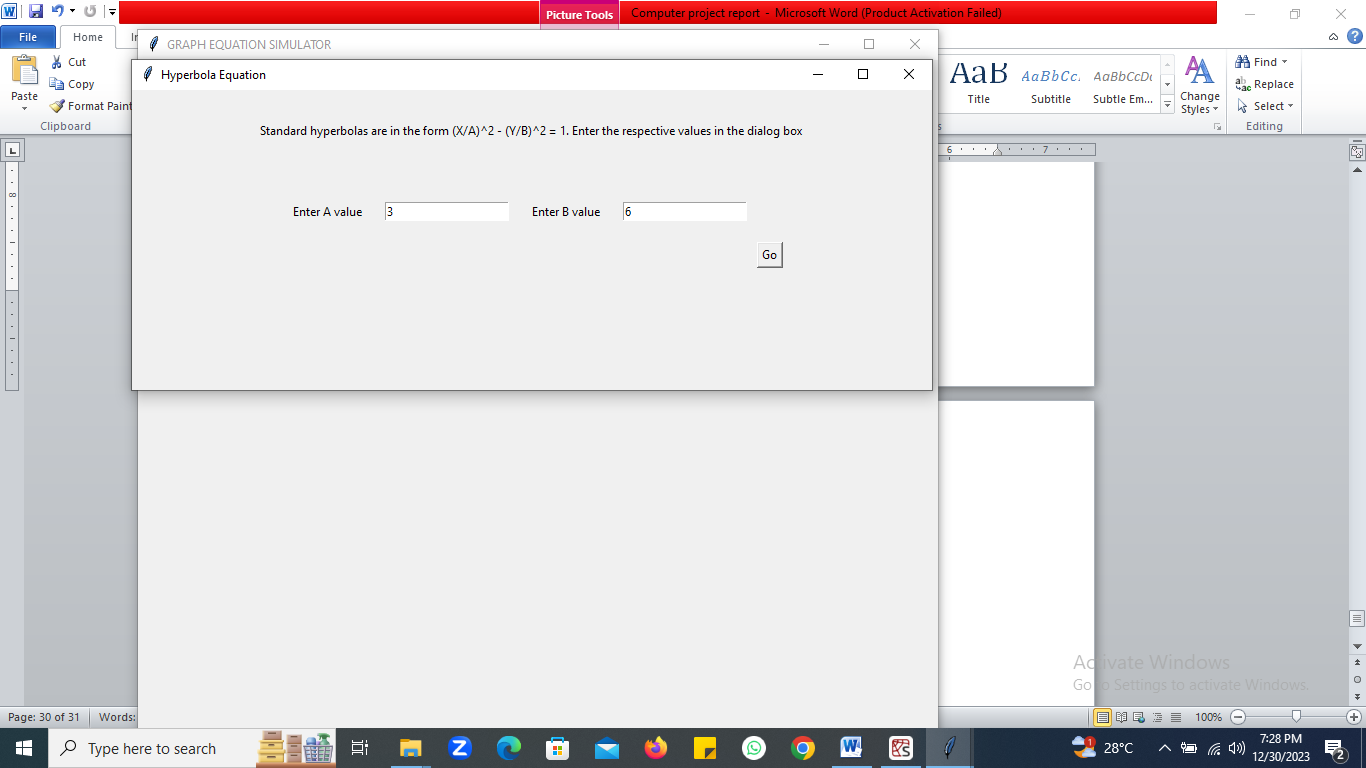


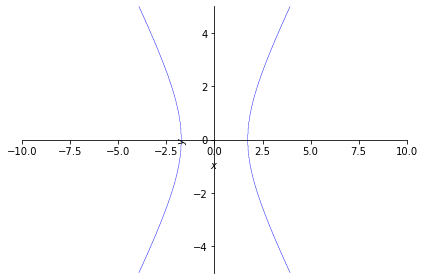


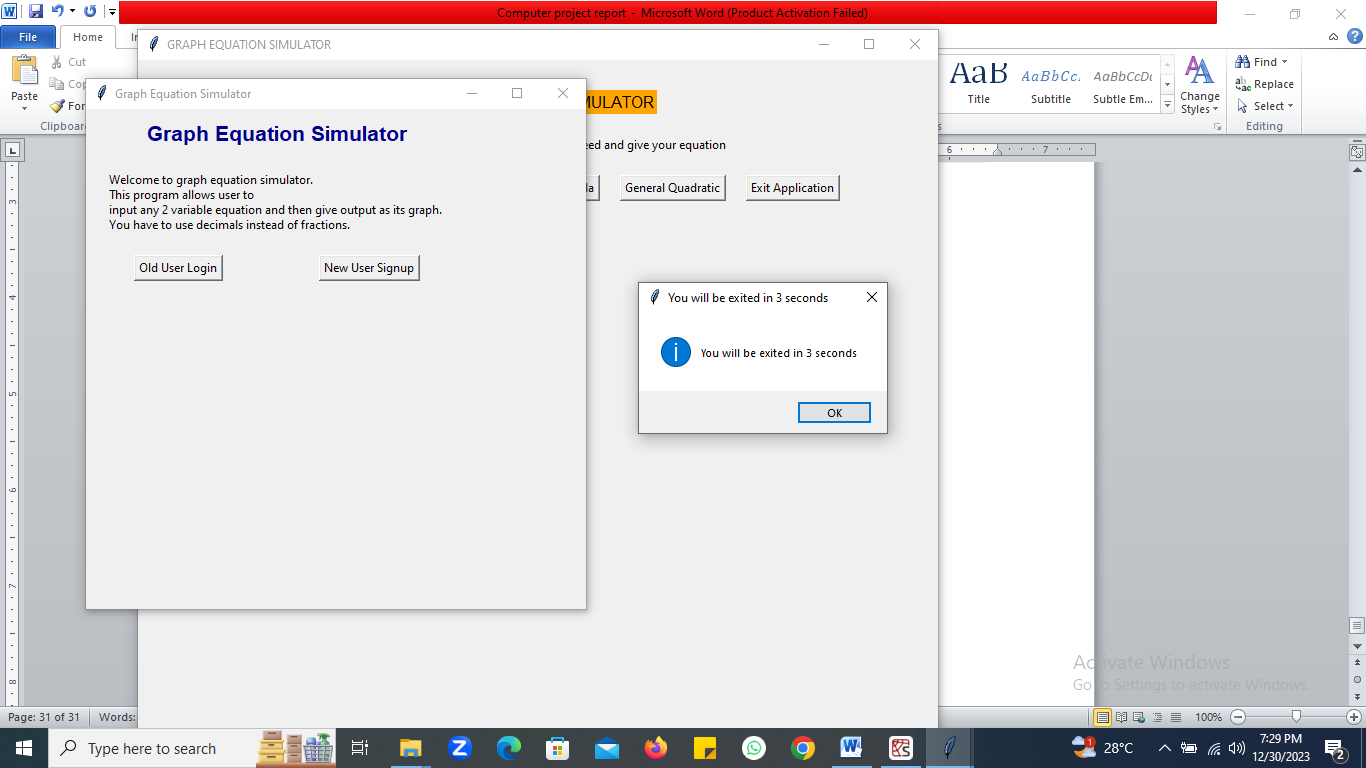










(The program exits)

BIBLIOGRAPHY

[www.geeksforgeeks.orgpython](http://www.geeksforgeeks.orgpython)

<https://www.w3schools.com/python/>

<https://trinket.io/>

<https://packaging.python.org/en/latest/tutorials/installing-packages/>

