**MEMEZEE**

*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

**<D.AKSHITHA ><1602-20-737-003>**

**<K.SAI SHRUTHI><1602-20-737-036>**

**<C.SAMIKSHA><1602-20-737-037>**

**A close-up of a logo

Description automatically generated with medium confidence**

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2022**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Hyderabad-500 031**

**Department of Information Technology**

A close-up of a logo

Description automatically generated with medium confidence

**DECLARATION BY THE CANDIDATE**

We, **D.AKSHITHA, K.SAI SHRUTHI, C.SAMIKSHA,** bearing hall ticket number, **1602-20-737-003, 1602-20-737-036, 1602-20-737-037** , hereby declare that the project report entitled **“MEMEZEE”** Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment 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 me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**D.AKSHITHA( 1602-20-737-003)**

**K.SAI SHRUTHI(1602-20-737-036)**

**C.SAMIKSHA(1602-20-737-037)**

L.DIVYA Dr. K. RAM MOHAN RAO

(Faculty In-Charge) (IT HOD)

**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 L.Divya and Rajyalaxmi, 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**

The aim of our project is to generate a meme for the user, where the user needs to choose the specifications like image or meme tamplete. After choosing the user is provided with options for text editing and adding at spcified location, drawing on the meme and then the user can save the meme created. Our project is build by using built-in-methods . We used the tkinter module to implement graphical user interface.

**CONTENTS**

1 INTRODUCTION

1.1 OVERVIEW OF THE PROJECT. . . . . . . . . . . . . . . . . . . . . . . . .

1.2 FEATURES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.3 SCOPE. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2 TECHNOLOGY

2.1 SOFTWARE REQUIREMENTS. . . . . . . . . . . . . . . . . . . . . . . . . .

2.2 HARDWARE REQUIREMENTS. . . . . . . . . . . . . . . . . . . . . . . . .

3 PROPOSED WORK

3.1 DESIGN. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 IMPLEMENTATION. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4 RESULTS

5 CONCLUSION AND FUTURE WORK

6 REFERENCES

**1.INTRODUCTON**

**1.1 OVERVIEW OF THE PROJECT**

The project’s objective is to develop an application for generating a Meme according to the specifications entered by the user.

**1.2 FEATURES**

1. Encoding the given specification into a Meme

2. Storing the generated Meme

**1.3 SCOPE**

MemeZee is a toolkit used to create basic memes. Here the user can create memes with one or two picture grid. The user will be getting different options like to add text, draw, some photo editing can also be done. For meme with two pictures the images are resized inside the layout automatically. After creating a meme you get an option to save the image.

In this project, we present a methodology and generate Memes according to the users preference.

**2.TECHNOLOGY**

**2.1 SOFTWARE REQUIREMENTS**

**1.Windows 8 or latest**

**2.Processor speed minimum x64 Processor : 1.4GHz**

**3.Runtime Environment : PyCharm**

**2.2 HARDWARE REQUIREMENTS**

**None**

**3.PROPOSED WORK**

**3.1 DESIGN**

**USE CASES**

1. **Generate a meme**
2. **Generate horizontal meme**
3. **Generate vertical meme**

**USE CASE 1**

**Name :** Choose a single image

**Actors :** User

**Description :** Allowing the user to give specifications for the meme

**Precondition :** None

**Postcondition :** Meme is generated for the given specifications

|  |  |
| --- | --- |
| User | System |
| **-**Chooses the image and make changes according to his/her preference | **-**Meme is generated according to the given specifications |

**USE CASE 2**

**Name :**

**Actors :** User

**Description :** Allowing the user to give specifications for the meme

**Precondition :** None

**Postcondition :** Meme is generated for the given specifications

|  |  |
| --- | --- |
| User | System |
| **-**Chooses two image and make changes according to his/her preference | **-**Both the images are merged side by sideas one and a meme is generated according to the given specifications |

**USE CASE 3**

**Name :**

**Actors :** User

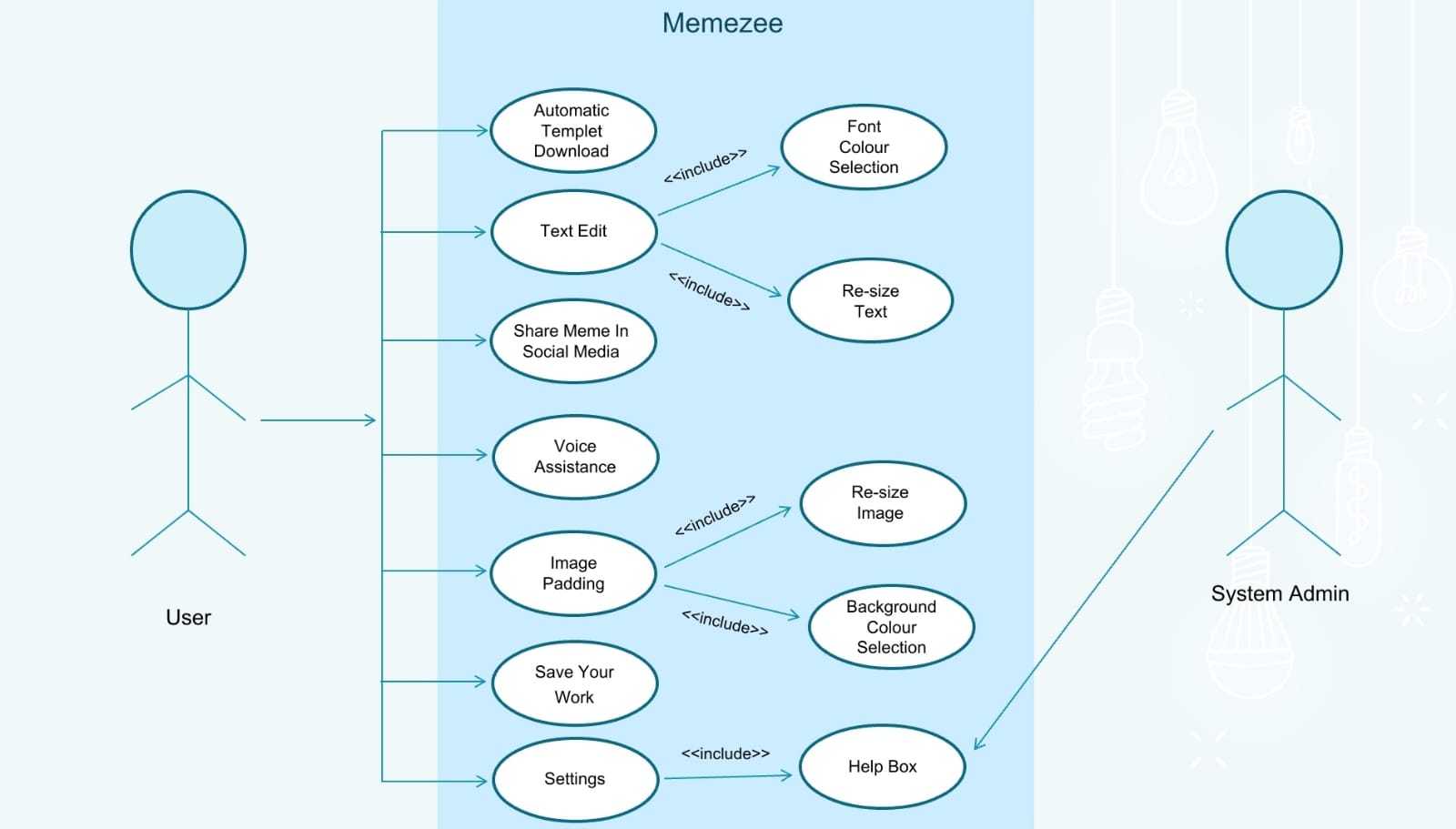
**Description :** Allowing the user to give specifications for the meme

**Precondition :** None

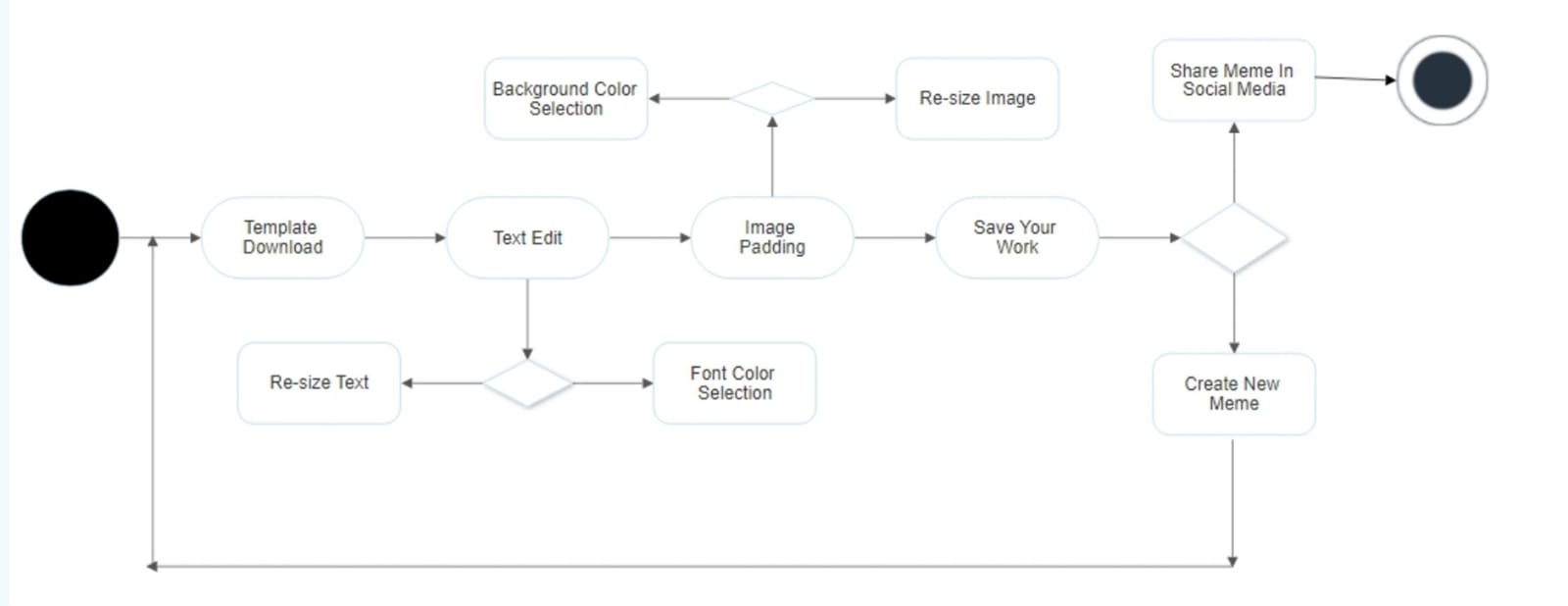
**Postcondition :** Meme is generated for the given specifications

|  |  |
| --- | --- |
| User | System |
| **-**Chooses two image and make changes according to his/her preference | **-**Both the images are merged one below the other and a meme is generated according to the given specifications |

**USE CASE DIAGRAM**

****

**ACTIVITY DIAGRAM**

****

**3.2 IMPLEMENTATION**

**-CODE**

# Importing required modules

from tkinter import \*

from tkinter import filedialog, ttk

from PIL import Image, ImageTk, ImageEnhance, ImageOps, ImageDraw, ImageFont

import os

from tkinter.filedialog import askopenfilename, asksaveasfilename

# Creating main window

root = Tk()

root.title('MemeZee')

root.geometry('1600x700')

# FIRST FRAME

# Adding background to main frame

load = Image.open('images\\logO.jpg')

bg\_temp = ImageTk.PhotoImage(load)

bg = Label(root, image=bg\_temp)

bg.place(x=0, y=0)

# Adding text to main frame

img1 = PhotoImage(file='images\\t1.png')

t1 = Label(root, image=img1, bg='#00015F')

t1.place(x=300, y=190)

def helpbox():

global Img, img, img\_path

newWindow3 = Toplevel(root)

newWindow3.title("About")

newWindow3.geometry("1200x700")

bg1 = Label(newWindow3, image=bg\_temp)

bg1.place(x=0, y=0)

message = '''

Dear User

Thank you for using Memezee.

Memezee is an application that helps us to create memes easily.

Memezee provides you features like:

-image editing

-text editing

-saving

It is a very easy and minimal.

'''

text\_box = Text(

newWindow3,

height=12,

width=100,

bg='lightblue'

)

text\_box.pack(expand=True)

text\_box.insert('end', message)

text\_box.config(state='disabled')

# Second Window

def Window(nw, img\_path):

# Creating horizontal bar in NewWindow

# DoubleVar holds a float

# Scale - used to select from a range of values, provides a sliding bar

v1 = DoubleVar()

s1 = Scale(nw, variable=v1, from\_=0, to=550, orient=HORIZONTAL, length=550, width=20, sliderlength=10,

tickinterval=100)

s1.place(x=100, y=550)

# Creating vertical bar in NewWindow

v2 = DoubleVar()

s2 = Scale(nw, variable=v2, from\_=0, to=400, orient=VERTICAL, length=400, width=20, sliderlength=10,

tickinterval=100)

s2.place(x=30, y=150)

# Creating canvas in Choose Image Window

global canvas1

canvas1 = Canvas(nw, width=550, height=400, bg='#00015F')

canvas1.place(x=100, y=150)

# removes the garbage value

Img = None

img6 = None

img8 = None

img10 = None

img12 = None

# Function for adjusting brightneses of an image

def brightness(event):

global img\_path, img5, img6, imgg

img = Image.open(img\_path)

img.thumbnail((550, 400))

# ImageEnhance.Brightness method is used to controll brightness of an image

# Creating an object of brightness class

imgg = ImageEnhance.Brightness(img)

# Showing the resultant image

img5 = imgg.enhance((float(bright\_combo.get())))

img6 = ImageTk.PhotoImage(img5)

canvas1.create\_image(275, 200, image=img6)

canvas1.image = img

img5.save(img\_path)

# Brightness label

bright = Label(nw, text="Brightness:", font=("ariel 15 bold"))

bright.place(x=670, y=250)

values1 = [1, 1.5, 2.0, 2.2, 2.4, 2.6]

bright\_combo = ttk.Combobox(nw, values=values1, font=('ariel 10 bold'))

bright\_combo.place(x=790, y=257)

bright\_combo.bind("<<ComboboxSelected>>", brightness)

# Function to rotate the image

def rotate\_image(event):

global img\_path, img7, img8, img5

img = Image.open(img\_path)

img.thumbnail((550, 400))

# .rotate - rotates the image by the specified value

img7 = img.rotate(int(rotate\_combo.get()))

img8 = ImageTk.PhotoImage(img7)

canvas1.create\_image(275, 200, image=img8)

canvas1.image = img8

img7.save(img\_path)

# Rotate label

rotate = Label(nw, text="Rotate:", font=("ariel 15 bold"))

rotate.place(x=1000, y=250)

values = [0, 90, 180, 270, 360]

rotate\_combo = ttk.Combobox(nw, values=values, font=('ariel 10 bold'))

rotate\_combo.place(x=1080, y=257)

rotate\_combo.bind("<<ComboboxSelected>>", rotate\_image)

def image\_border(event):

global img\_path, img9, img10, img5

img = Image.open(img\_path)

img.thumbnail((550, 400))

# ImageOps.expand() - adds a border to the image according to the specified values

img9 = ImageOps.expand(img, border=int(border\_combo.get()), fill=(borderr\_combo.get()))

img10 = ImageTk.PhotoImage(img9)

canvas1.create\_image(275, 200, image=img10)

canvas1.image = img10

img9.save(img\_path)

# Border label

border = Label(nw, text="Add border:", font=("ariel 15 bold"))

border.place(x=855, y=370)

values2 = [i for i in range(10, 45, 5)]

border\_combo = ttk.Combobox(nw, values=values2, font=("ariel 10 bold"))

border\_combo.place(x=980, y=375)

border\_combo.bind("<<ComboboxSelected>>", image\_border)

# Border Colour label

borderr = Label(nw, text="BorderColour:", font=("ariel 14 bold"))

borderr.place(x=830, y=320)

values\_borderr = ['red', 'green', 'black', 'yellow', 'pink', 'white']

borderr\_combo = ttk.Combobox(nw, values=values\_borderr, font=('ariel 10 bold'))

borderr\_combo.place(x=980, y=325)

# Functions to paint

def get\_x\_and\_y(event):

global lasx, lasy

# Returns the current position of the mouse pointer

lasx, lasy = event.x, event.y

def paint(event):

global lasx, lasy, img11, img\_path, img12, img

img = Image.open(img\_path)

img.thumbnail((550, 400))

# Creates a line at the mouse pointer

img11 = canvas1.create\_line((lasx, lasy, event.x, event.y), fill=(draw1\_combo.get()), width=2)

lasx, lasy = event.x, event.y

img12 = ImageTk.PhotoImage(img11)

canvas1.create\_image(275, 200, image=img12)

canvas1.image = img12

# Function to draw

def draw():

# Binds the canvas to the functions

canvas1.bind("<Button-1>", get\_x\_and\_y)

canvas1.bind("<B1-Motion>", paint)

# Button to Draw

# Button to Draw

global img08

img08 = PhotoImage(file='images\\b8.png')

b8 = Button(nw, image=img08, command=draw, bg='#00015F')

b8.place(x=780, y=470)

draw1 = Label(nw, text="Draw Colour:", font=("ariel 14 bold"))

draw1.place(x=910, y=475)

values\_draw1 = ['red', 'green', 'black', 'yellow', 'pink', 'white']

draw1\_combo = ttk.Combobox(nw, values=values\_draw1, font=('ariel 10 bold'))

draw1\_combo.place(x=1050, y=477)

def delete():

# Cleares the canvas

canvas1.delete("all")

global img010

img010 = PhotoImage(file='images\\b10.png')

b10 = Button(nw, image=img010, bg='#00015F', command=delete)

b10.place(x=1100, y=570)

# Third Window

def next2():

def Window2(nw, img\_path):

global Img, img, img\_path2, imgg, im1

im1 = Image.open(img\_path)

im1.thumbnail((550, 400))

img\_path2 = 'images\\MemeEdit2.jpg'

# Creates a new window for text edit

newWindowT = Toplevel(nw)

newWindowT.title("Text Edit")

newWindowT.geometry("1600x700")

bg1 = Label(newWindowT, image=bg\_temp)

bg1.place(x=0, y=0)

imgg = Image.open(img\_path)

imgg.save(img\_path2)

# Creating horizontal bar in NewWindow

v11 = DoubleVar()

s11 = Scale(newWindowT, variable=v1, from\_=0, to=550, orient=HORIZONTAL, length=550, width=20,

sliderlength=10,

tickinterval=100)

s11.place(x=100, y=550)

# Creating vertical bar in NewWindow

v22 = DoubleVar()

s22 = Scale(newWindowT, variable=v2, from\_=0, to=400, orient=VERTICAL, length=400, width=20,

sliderlength=10,

tickinterval=100)

s22.place(x=30, y=150)

# s2.place(x=90,y=100)

# Creating canvas in Choose Image Window

global canvas2

canvas2 = Canvas(newWindowT, width=550, height=400, bg='#00015F')

canvas2.place(x=100, y=150)

Img = ImageTk.PhotoImage(imgg)

canvas2.create\_image(275, 200, image=Img)

canvas2.image = Img

imgg.save(img\_path2)

# removes the garbage value

Img = None

img3 = None

def Addtext():

global img\_path2, img2, img3, img4, img5

clear1()

img4 = Image.open(img\_path2)

# Image.convert() - Returns a converted copy of this image

img4 = img4.convert('RGB')

img4.thumbnail((550, 400))

text\_to\_add = Text\_entry.get()

font = font\_combo.get()

myFont = ImageFont.truetype(font + '.ttf', int(fontc\_combo.get()))

img2 = ImageDraw.Draw(img4)

img2.text((int(xaxis\_combo.get()), int(yaxis\_combo.get())), text\_to\_add, (colors\_combo.get()),

font=myFont)

# Wait a couple seconds and then show image

textadd.after(2, show\_pic())

img3 = ImageTk.PhotoImage(img4)

canvas2.create\_image(275, 200, image=img3)

canvas2.image = img3

img4.save(img\_path2)

def show\_pic():

# Show New Image

global img, img\_path2

img = PhotoImage(img\_path2)

textadd.config(image=img)

# Clear the entry box

Text\_entry.delete(0, END)

global img09

img09 = PhotoImage(file='images\\b9.png')

b9 = Button(newWindowT, image=img09, bg='#00015F', command=Addtext)

b9.grid(row=730, column=460, padx=670, pady=500)

b9.place(x=875, y=515)

# Text Entry label

textadd = Label(newWindowT, image=img3)

textadd.grid(row=700, column=460, padx=855, pady=470)

# Entry box

Text\_entry = Entry(newWindowT, font=('ariel 15 bold'))

Text\_entry.grid(row=700, column=460, padx=855, pady=470)

# ttk.Combobox - creates a combobox, used for drop down selectioin of values

# .place - places the button/label in the window at specified position

# X axis label

xaxis = Label(newWindowT, text="Xaxis:", font=("ariel 15 bold"))

xaxis.place(x=1000, y=230)

values\_xaxis = [10, 50, 100, 150, 200, 250, 300, 350, 400]

xaxis\_combo = ttk.Combobox(newWindowT, values=values\_xaxis, font=('ariel 10 bold'))

xaxis\_combo.place(x=1070, y=235)

# Y axis label

yaxis = Label(newWindowT, text="Yaxis:", font=("ariel 15 bold"))

yaxis.place(x=1000, y=330)

values\_yaxis = [10, 50, 100, 150, 200, 250, 300, 350, 400]

yaxis\_combo = ttk.Combobox(newWindowT, values=values\_yaxis, font=('ariel 10 bold'))

yaxis\_combo.place(x=1070, y=335)

# TextColour label

colors = Label(newWindowT, text="TextColour:", font=("ariel 15 bold"))

colors.place(x=680, y=180)

values\_colors = ['red', 'green', 'black', 'yellow', 'pink', 'white']

colors\_combo = ttk.Combobox(newWindowT, values=values\_colors, font=('ariel 10 bold'))

colors\_combo.place(x=805, y=185)

# Font type label

font = Label(newWindowT, text="Text Font:", font=("ariel 15 bold"))

font.place(x=680, y=380)

values\_font = ['arial', 'Courier', 'Helvetica', 'Segoe Script', 'Times', 'normal', 'roman', 'italic']

font\_combo = ttk.Combobox(newWindowT, values=values\_font, font=('ariel 10 bold'))

font\_combo.place(x=805, y=385)

# Font Size label

fontc = Label(newWindowT, text="Text Size:", font=("ariel 15 bold"))

fontc.place(x=680, y=280)

values\_fontc = [10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58]

fontc\_combo = ttk.Combobox(newWindowT, values=values\_fontc, font=('ariel 10 bold'))

fontc\_combo.place(x=805, y=285)

def clear1():

global img

img = Image.open(img\_path)

img.save(img\_path)

Img = ImageTk.PhotoImage(img)

canvas2.create\_image(275, 200, image=Img)

canvas2.image = Img

img.save(img\_path2)

def text():

global img1

img1 = Image.open(img\_path2)

img1.save(img\_path)

Img = ImageTk.PhotoImage(img1)

canvas2.create\_image(275, 200, image=Img)

canvas2.image = Img

global img014

img014 = PhotoImage(file='images\\b14.png')

b14 = Button(newWindowT, image=img014, bg='#00015F', command=clear1)

b14.place(x=1100, y=570)

global img013

img013 = PhotoImage(file='images\\b13.png')

b13 = Button(newWindowT, image=img013, bg='#00015F', command=text)

b13.place(x=690, y=560)

def save():

global img\_path2, Img, img2, img3, img4, img5, img6, img7, img8, img9, img10, img11, img12, img

img\_path2 = 'images\\MemeEdit2.jpg'

img = Image.open(img\_path2)

# Returns the last item in image path

ext = img\_path.split(".")[-1]

file = asksaveasfilename(defaultextension=f".{ext}",

filetypes=[("PNG file", ".png"), ("jpg file", ".jpg")])

img.save(file)

global img011

img011 = PhotoImage(file='images\\b11.png')

b11 = Button(newWindowT, image=img011, bg='#00015F', command=save)

b11.place(x=900, y=570)

# Call for Window2

Window2(nw, img\_path)

global img012

img012 = PhotoImage(file='images\\b12.png')

b12 = Button(nw, image=img012, bg='#00015F', command=next2)

b12.place(x=900, y=570)

# Grid for a single image

def gridfor1():

global Img, img, img\_path1, img\_path

img\_path1 = r'C:MemeZee'

img\_path = 'images\\MemeEdit.jpg'

# Creating a new window

newWindow = Toplevel(root)

newWindow.title("Grid for 1")

newWindow.geometry("1600x700")

bg1 = Label(newWindow, image=bg\_temp)

bg1.place(x=0, y=0)

def choose\_image1():

global Img, img, img\_path1, img\_path

# filedialog.askopenfilename - function create an Open dialog and return the selected filename(s) that correspond to existing file(s)

img\_path1 = filedialog.askopenfilename(initialdir=os.getcwd())

# Opens image at the specified path

img = Image.open(img\_path1)

# .thumbnail - method modifies the image to contain a thumbnail version of itself, no larger than the given size

img.thumbnail((550, 400))

Img = ImageTk.PhotoImage(img)

# Displays the selected image on canvas

canvas1.create\_image(275, 200, image=Img)

canvas1.image = Img

# Saves the image at specified path

img.save(img\_path)

# Button for Choose image

global img06

img06 = PhotoImage(file='images\\b6.png')

b6 = Button(newWindow, image=img06, command=choose\_image1, bg='#00015F')

b6.place(x=890, y=150)

Window(newWindow, img\_path)

def gridfor2\_horizontal():

global canvas1, IMG\_H, image1\_1, image1\_2, img\_path1\_1, img\_path1\_2, img\_path

img\_path = 'images\\MemeEdit.jpg'

newWindow1 = Toplevel(root)

newWindow1.title("Grid for 2")

newWindow1.geometry("1600x700")

bg2 = Label(newWindow1, image=bg\_temp)

bg2.place(x=0, y=0)

# Function to choose image

def choose\_image2\_1():

global image1\_1, image\_path1\_1, image1\_2, image\_path1\_2, Img, img\_path

img\_path1\_1 = filedialog.askopenfilename(initialdir=os.getcwd())

image1\_1 = Image.open(img\_path1\_1)

img\_path1\_2 = filedialog.askopenfilename(initialdir=os.getcwd())

image1\_2 = Image.open(img\_path1\_2)

# Function to concat images horizontally

def get\_concat\_h\_resize(image1\_1, image1\_2, resize\_big\_image=True):

global \_image1\_1, \_image1\_2, dst

if image1\_1.height == image1\_2.height:

\_image1\_1 = image1\_1

\_image1\_2 = image1\_2

elif (((image1\_1.height > image1\_2.height) and resize\_big\_image) or

((image1\_1.height < image1\_2.height) and not resize\_big\_image)):

\_image1\_1 = image1\_1.resize((int(image1\_1.width \* image1\_2.height / image1\_1.height), image1\_2.height),

Image.BICUBIC)

\_image1\_2 = image1\_2

else:

\_image1\_1 = image1\_1

\_image1\_2 = image1\_2.resize((int(image1\_2.width \* image1\_1.height / image1\_2.height), image1\_1.height),

Image.BICUBIC)

dst = Image.new('RGB', (\_image1\_1.width + \_image1\_2.width, \_image1\_1.height))

dst.paste(\_image1\_1, (0, 0))

dst.paste(\_image1\_2, (\_image1\_1.width, 0))

return dst

# Concating 2 images and Adding it to canvas which is created here itslef

get\_concat\_h\_resize(image1\_1, image1\_2, resize\_big\_image=True).save('images\\MemeEdit.jpg')

IMG\_H = Image.open('images\\MemeEdit.jpg')

IMG\_H.thumbnail((550, 400))

Img = ImageTk.PhotoImage(IMG\_H)

canvas1.create\_image(275, 200, image=Img)

canvas1.image = Img

IMG\_H.save(img\_path)

# Button for Choose 2 images

global img07

img07 = PhotoImage(file='images\\b7.png')

b7 = Button(newWindow1, image=img07, command=choose\_image2\_1, bg='#00015F')

b7.place(x=890, y=120)

Window(newWindow1, img\_path)

def gridfor2\_vertical():

global canvas1, IMG\_V, im1, im2, img\_path1\_1, img\_path1\_2, Img, img\_path

img\_path = 'images\\MemeEdit.jpg'

newWindow2 = Toplevel(root)

newWindow2.title("Grid for 2")

newWindow2.geometry("1600x700")

bg2 = Label(newWindow2, image=bg\_temp)

bg2.place(x=0, y=0)

# Function to choose image

def choose\_image2\_2():

global im1, image\_path1\_1, im2, image\_path1\_2, img\_path

img\_path1\_1 = filedialog.askopenfilename(initialdir=os.getcwd())

im1 = Image.open(img\_path1\_1)

img\_path1\_2 = filedialog.askopenfilename(initialdir=os.getcwd())

im2 = Image.open(img\_path1\_2)

def get\_concat\_v\_resize(im1, im2, resize\_big\_image=True):

if im1.width == im2.width:

\_im1 = im1

\_im2 = im2

elif (((im1.width > im2.width) and resize\_big\_image) or

((im1.width < im2.width) and not resize\_big\_image)):

\_im1 = im1.resize((im2.width, int(im1.height \* im2.width / im1.width)), Image.BICUBIC)

\_im2 = im2

else:

\_im1 = im1

\_im2 = im2.resize((im1.width, int(im2.height \* im1.width / im2.width)), Image.BICUBIC)

dst = Image.new('RGB', (\_im1.width, \_im1.height + \_im2.height))

dst.paste(\_im1, (0, 0))

dst.paste(\_im2, (0, \_im1.height))

return dst

get\_concat\_v\_resize(im1, im2, resize\_big\_image=True).save('images\\MemeEdit.jpg')

IMG\_V = Image.open('images\\MemeEdit.jpg')

IMG\_V.thumbnail((550, 400))

Img = ImageTk.PhotoImage(IMG\_V)

canvas1.create\_image(275, 200, image=Img)

canvas1.image = Img

IMG\_V.save(img\_path)

# Button for Choose image

global img07

img07 = PhotoImage(file='images\\b7.png')

b7 = Button(newWindow2, image=img07, command=choose\_image2\_2, bg='#00015F')

b7.place(x=890, y=120)

Window(newWindow2, img\_path)

# Adding Buttons to 1st frame

# Adding buttons as image files

img01=PhotoImage(file='images\\b1.png')

img02=PhotoImage(file='images\\b2.png')

img03=PhotoImage(file='images\\b3.png')

img04=PhotoImage(file='images\\b4.png')

img05=PhotoImage(file='images\\b5.png')

b1 =Button(root,image=img01,bg='#00015F',command= gridfor1)

b1.place(x=175,y=350)

b2 =Button(root,image=img02,bg='#00015F',command= gridfor2\_horizontal)

b2.place(x=550,y=350)

b3 =Button(root,image=img03,bg='#00015F',command= gridfor2\_vertical)

b3.place(x=950,y=350)

b4 =Button(root,image=img04,bg='#00015F',command=helpbox)

b4.place(x=860,y=560)

b5 = Button(root,image=img05,bg='#00015F',command=root.destroy)

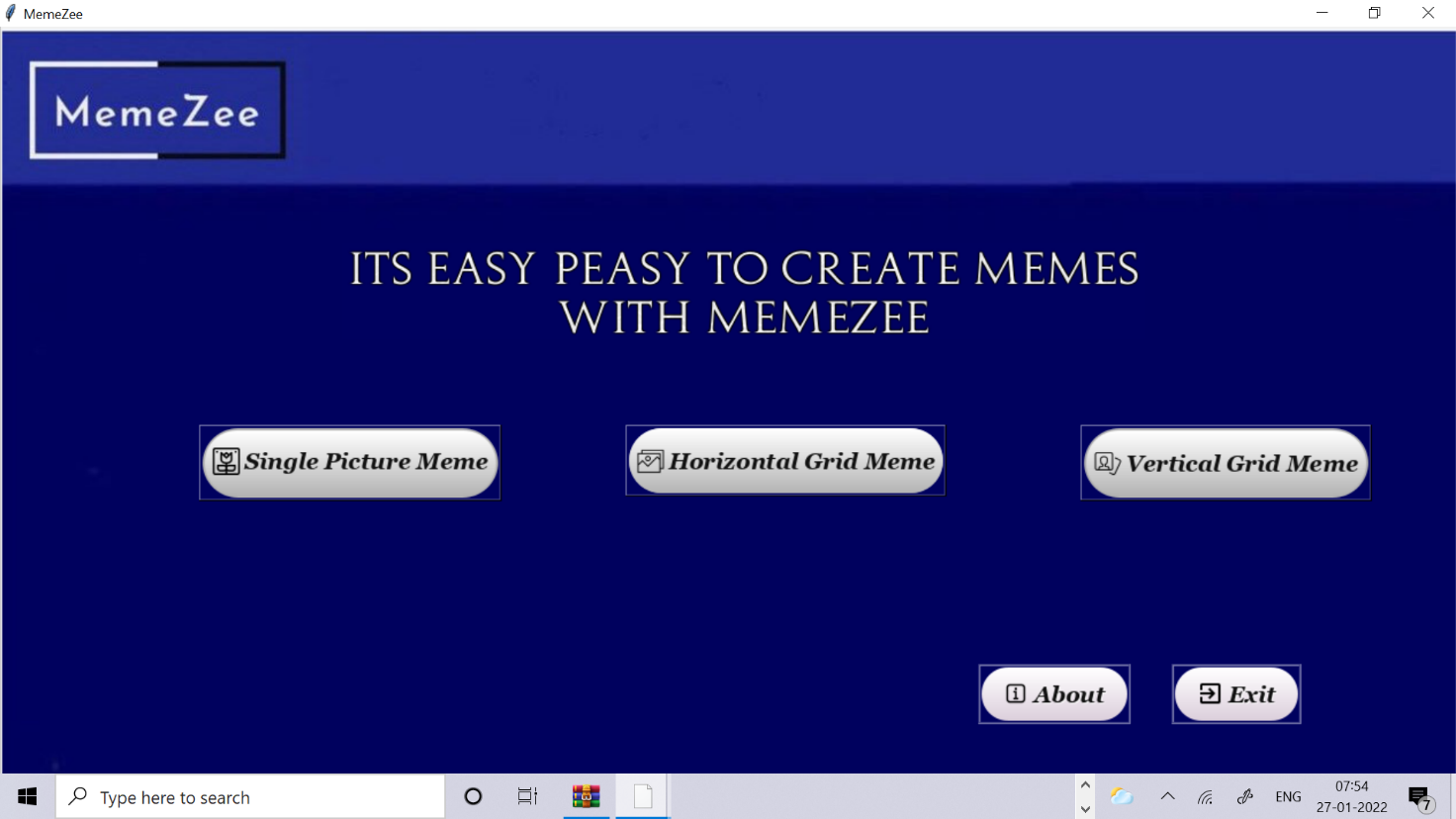
b5.place(x=1030, y=560)

root.mainloop()

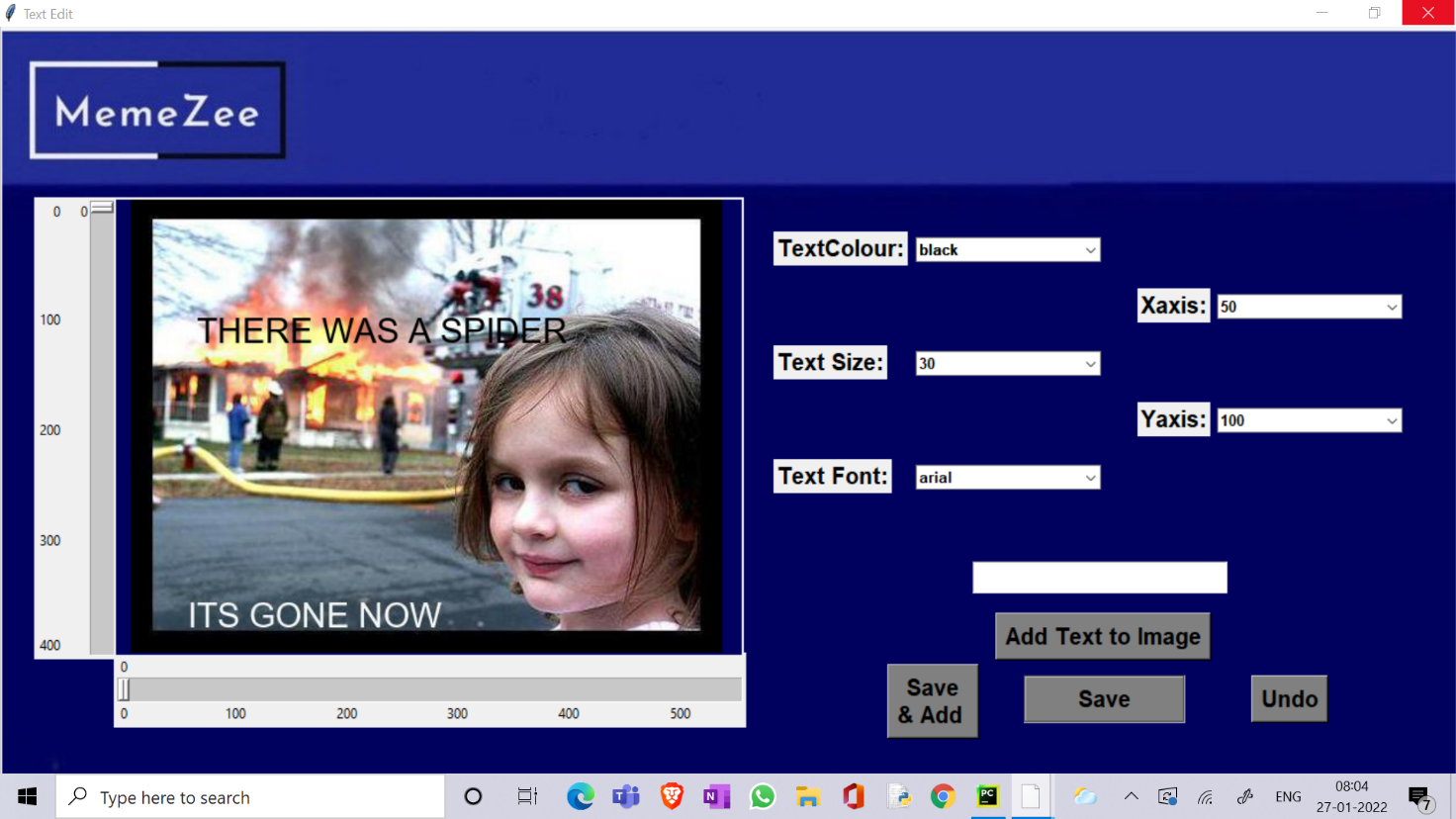
**GIT HUB LINK**

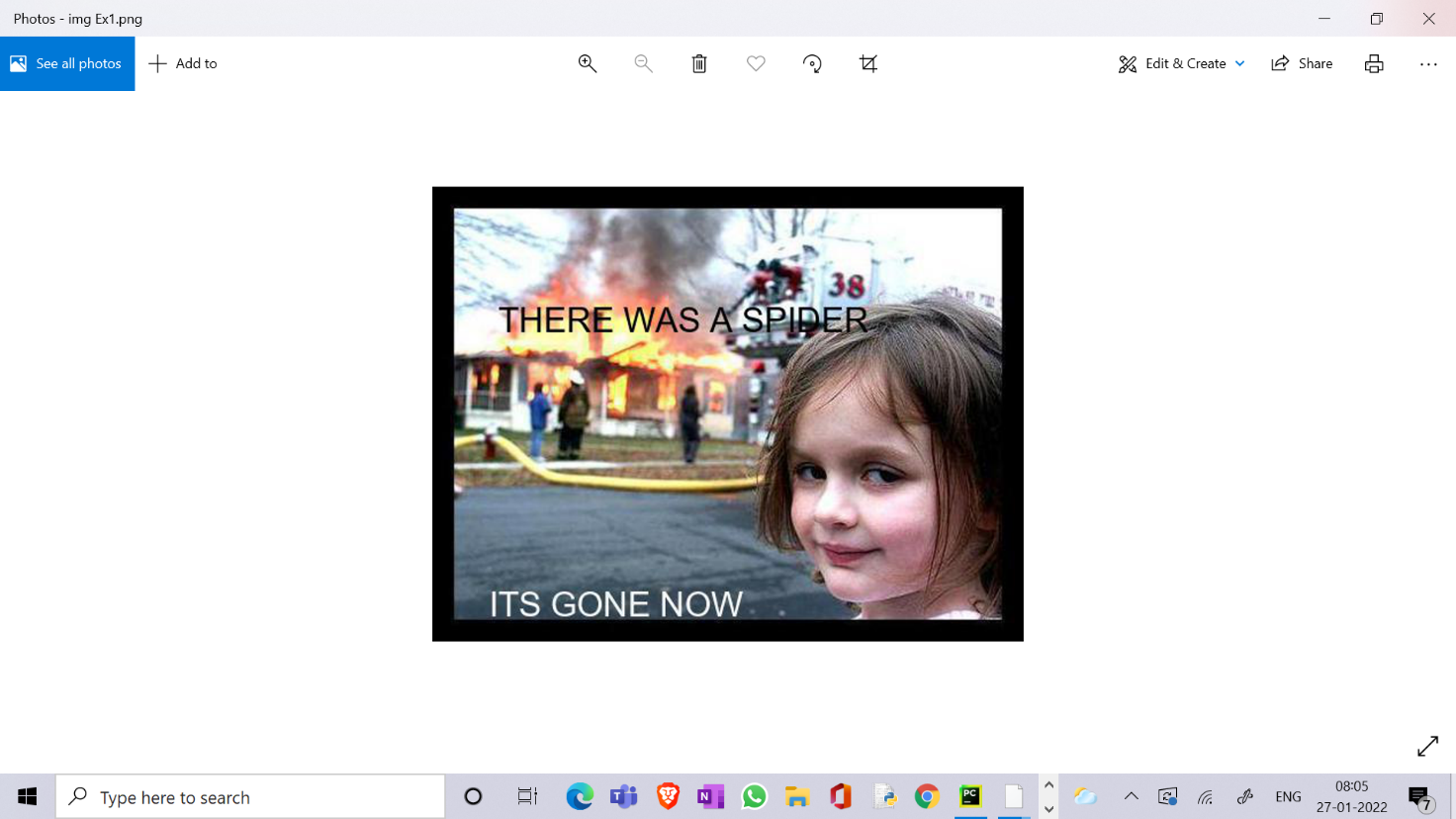
https://github.com/Shruthi-Kovvur/MEMEZEE-MP-1.git

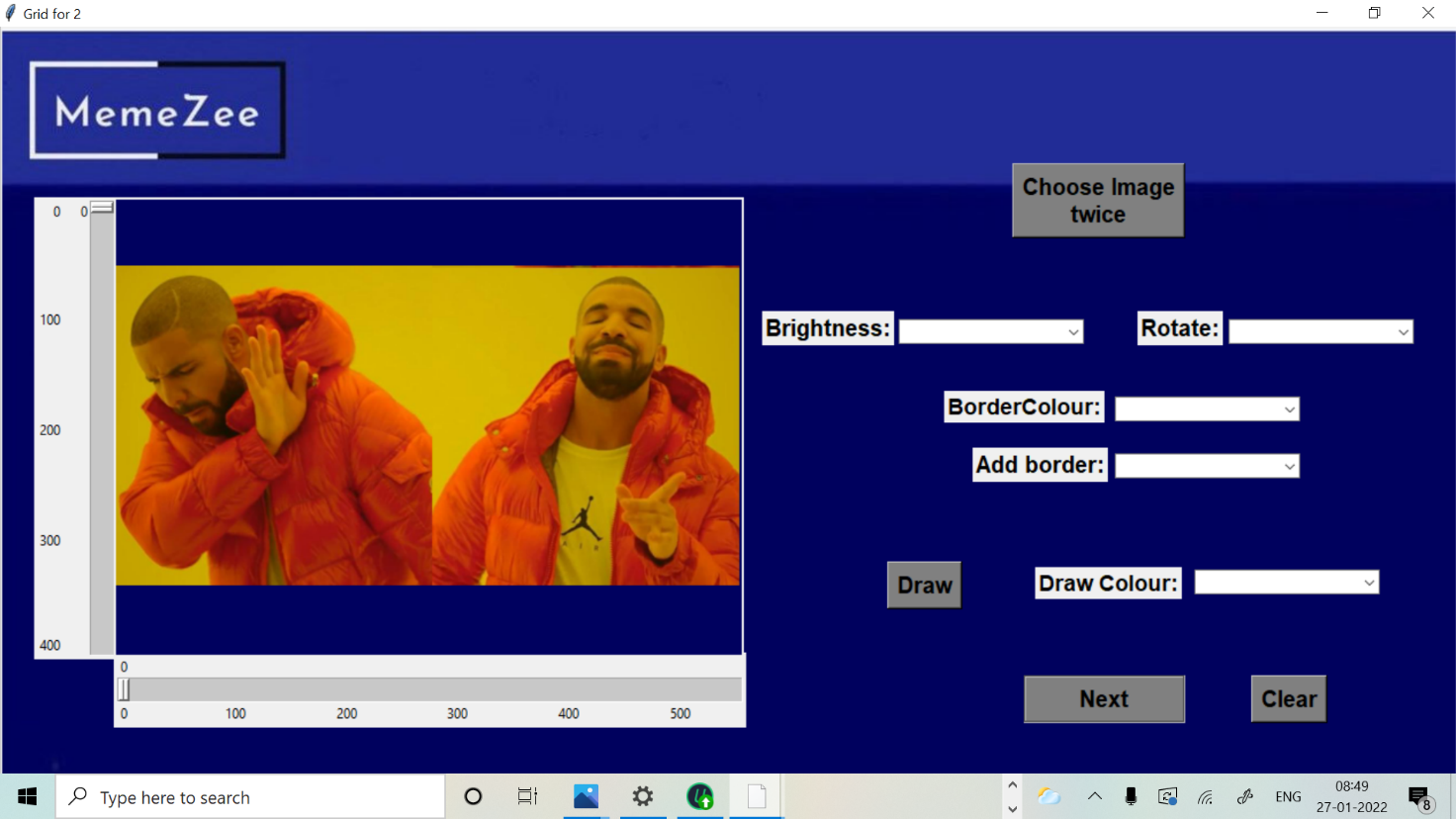
**4.RESULT**

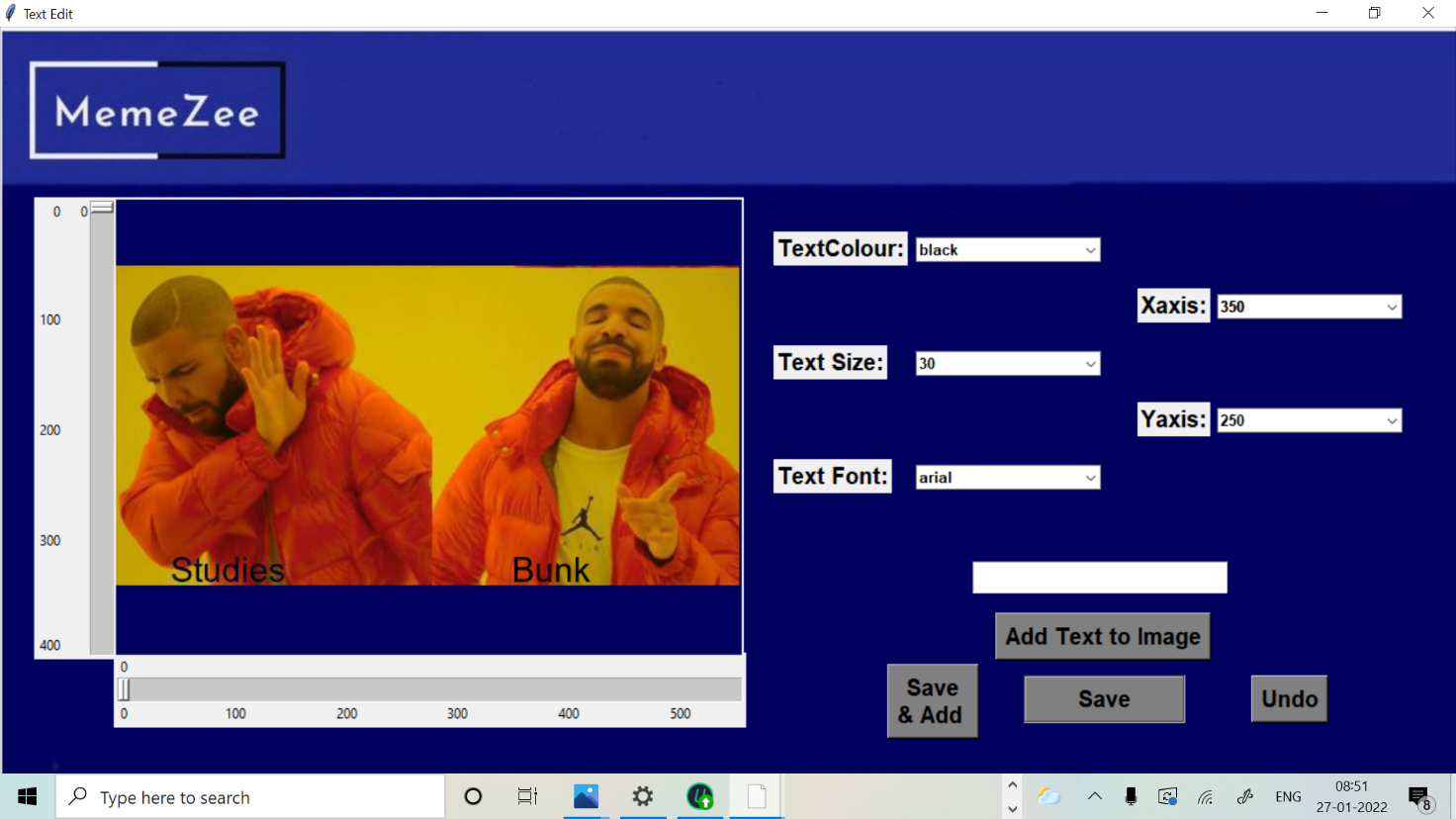
****

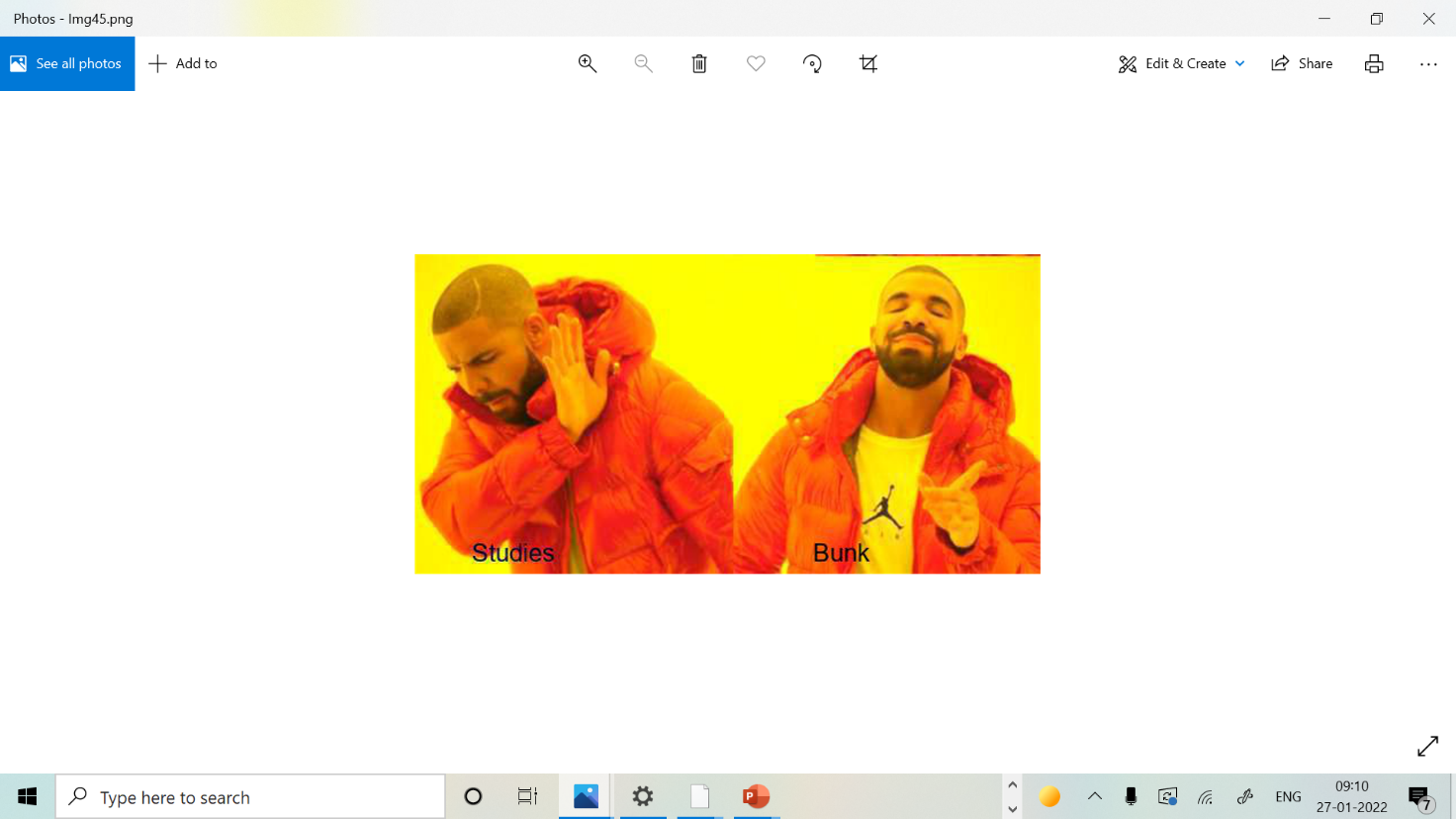
****

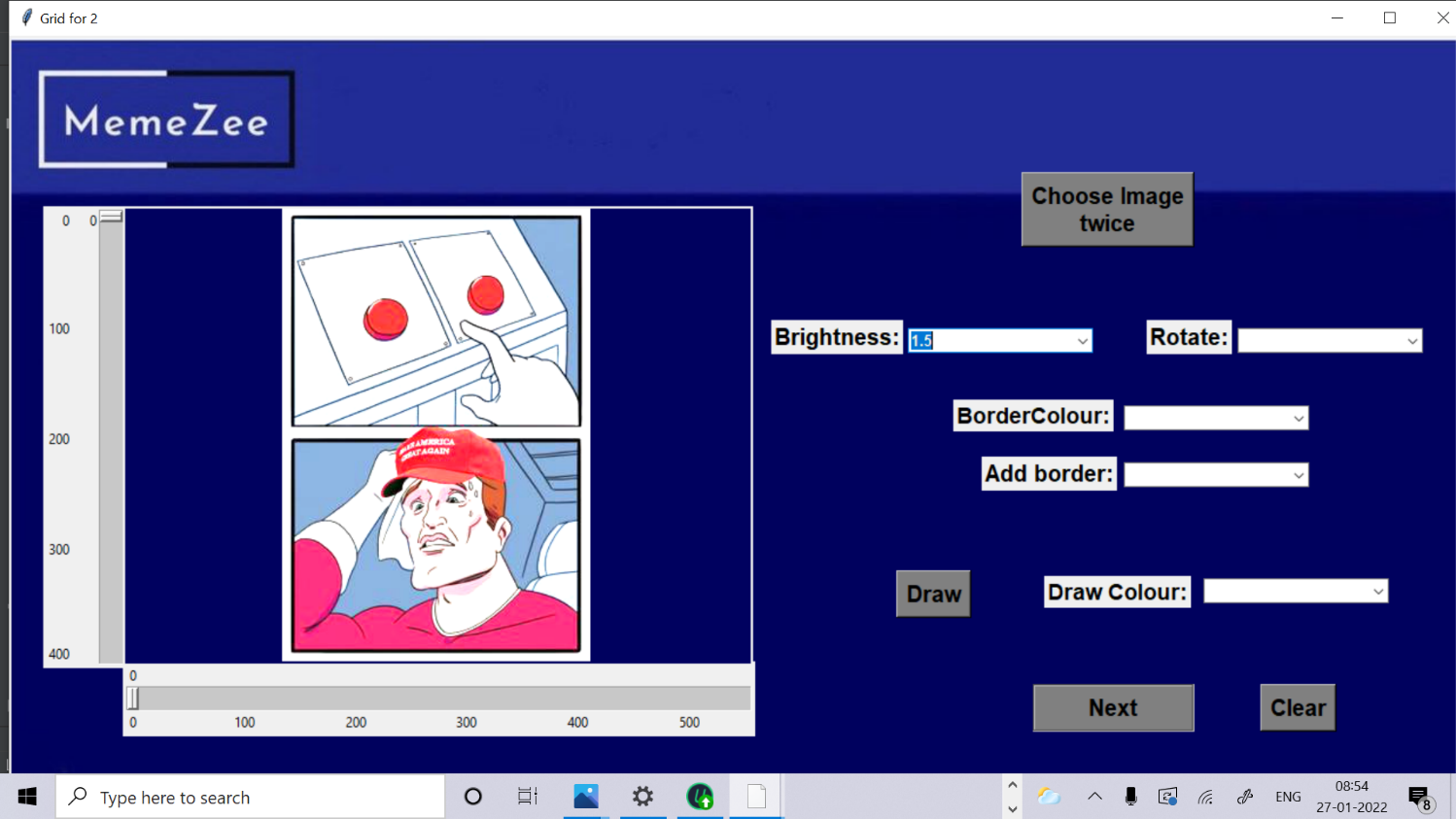
****

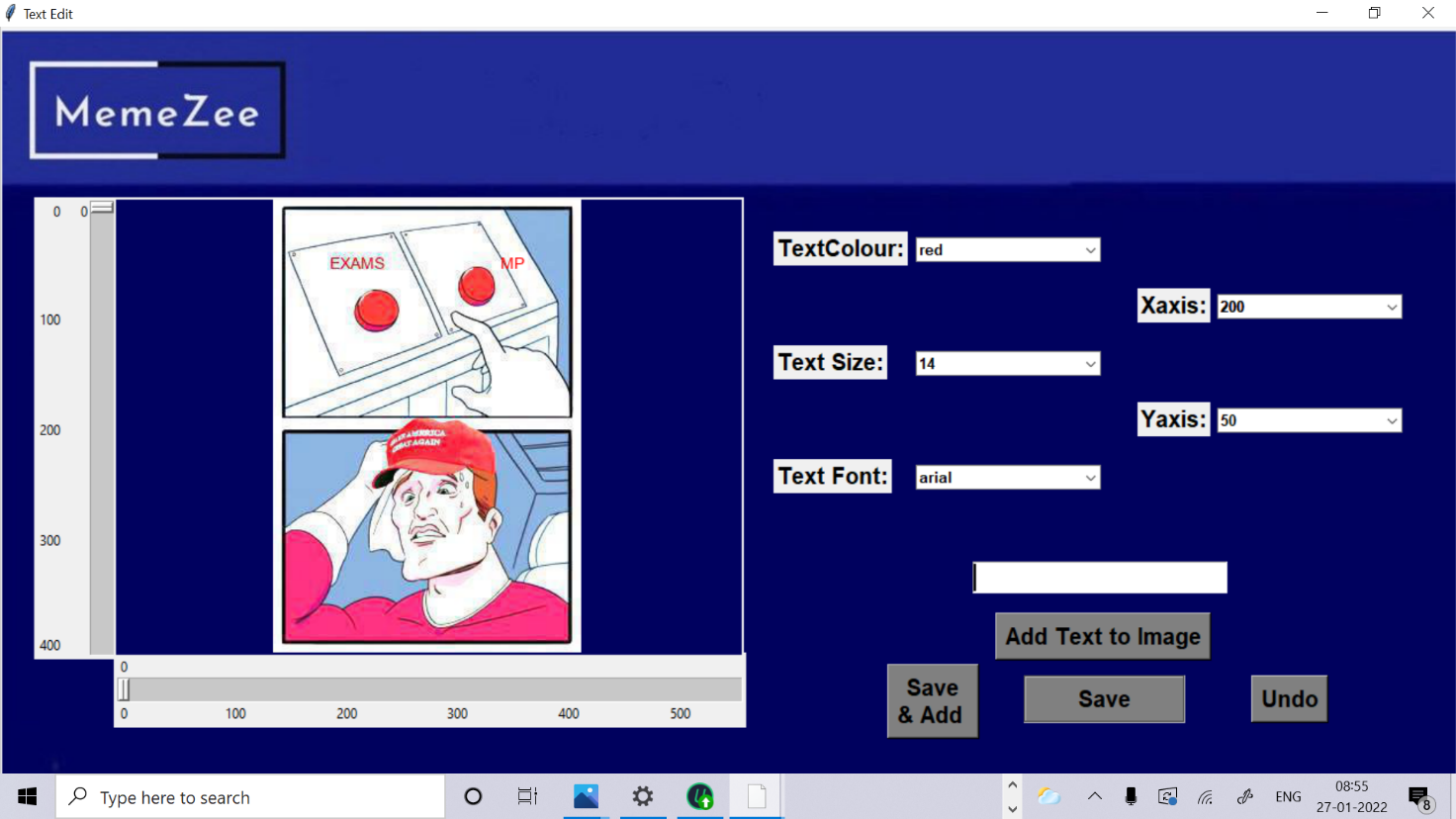
****

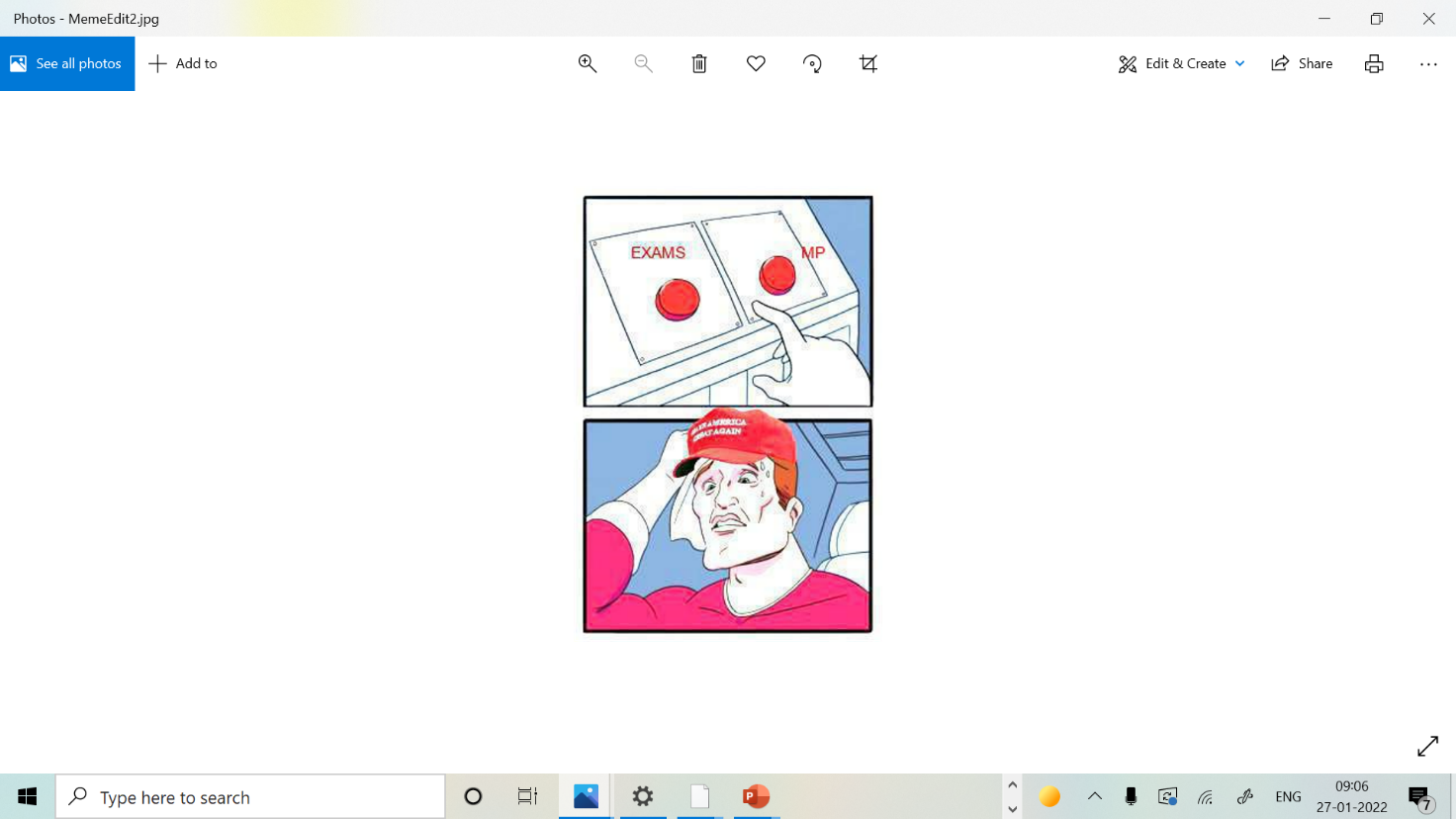
****

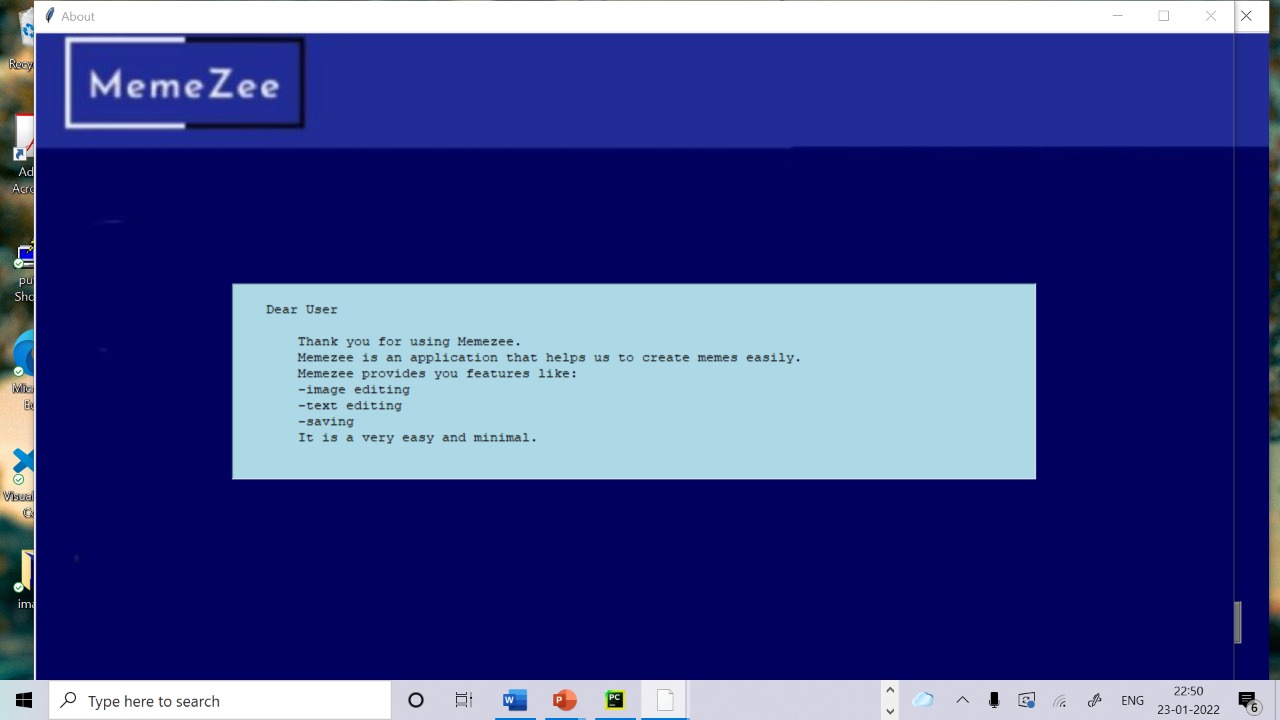
****

****

****

****

****

****

**5.CONCLUSION AND FUTURE WORK**

We learned how to manage time. Though we had lot of quizzes and assignments we somehow managed to pull up. This project helped us to gain interest in coding. From many topics we choose memezee and we went through a lot but as a team we faced them. We had an amazing experience working together.

Teamwork made understanding of our project a lot easier and helped us to be more creative in various steps of its development. We also had to revise a lot of concepts regarding graphical user interface, which made our basics even stronger and also helping us to be even more confident.

We have a lot of plans that we would like to add a lot of elements to our project. We would like add a feature that would allow the user to choose images directly from the web with the given spesification, also add a voice assistance to our project and a feature to allow the user to share the meme created on social media.We would also like to make the code more simpler and easier to understand.

**6. REFERENCES**

* <https://docs.python.org/3/>
* <https://docs.python.org/3/library/tk.html>