```
#####################################################################################
#Importing
from tkinter import *
from tkinter import ttk , messagebox as msg
from threading import Thread
from time import sleep
from mysql import connector as ms
import tkinter as tk
import pygame
import random
import time
import os
from datetime import datetime
from PIL import Image , ImageTk
#####################################################################################
root  = Tk()
root.resizable(width = False , height = False)
root.wm_state('zoomed')
root.title("Appstore By AVIONICS and RRAJJ")
frame = Frame(root, bg='grey')
#####################################################################################
root.iconbitmap('Photos/Icon.ico')
back_lg1             = PhotoImage(file = r'Photos/back small.png')
home_photo           = PhotoImage(file = r'Photos/home small.png')
appstore_lg2         = PhotoImage(file = r'Photos/Appstore Small.png')
front_photo_variable = PhotoImage(file = r'Photos/applicationmanagement.png')
person1              = Image.open('Photos/person_1.jpeg')
person2              = Image.open('Photos/person_2.jpg')
person_1             = person1.resize((300 , 300) , Image.ANTIALIAS)
person_2             = person2.resize((300 , 300) , Image.ANTIALIAS)
person1_photo        = ImageTk.PhotoImage (person_1)
person2_photo        = ImageTk.PhotoImage(person_2)
pas = ''
#####################################################################################
##Classes
class Calculator(Toplevel):

    def __init__(self):
        super().__init__()
        self.geometry('300x350')
        self.title("Calculator")
        self.iconbitmap('Photos/calculator_icon.ico')
        self.bgcolor = '#c5c5c5'
        self.config(bg = self.bgcolor)
        self.resizable(width = False , height = False)
        self.scvalue = tk.StringVar()
        self.scvalue.set("0")
        self.screen = tk.Entry(self , text = self.scvalue , font = 'helvatica 19 bold'
,relief = 'sunken' , width = 300)
        self.screen.pack(pady = 6 , padx = 3 , ipadx = 9 , ipady = 9)
        self.protocol("WM_DELETE_WINDOW" , self.close)

        #This save variable is to check wether the last thing done was root , equal ,
int or log
        #If yes then we have to clear the screen of entry for the next thing to be
entered
        self.save = bool  #Used in the click function
        self.create_buttons()

    def close(self):
        self.destroy()
        global root
        root.wm_state('zoomed')

    #Creating buttons
    def create_buttons(self):
        self.lst = [['<--' , 'CE', 'C' , 'root' , 'log'] , ['7' , '8' , '9' , '/' ,
'%'] , ['4', '5' , '6' , '*' , '1/x'], ['1' , '2' , '3' , '-' , '='] , ['0' , '.' , '+'
, 'int']]
        tk.Label(self , text = '' , bg = self.bgcolor).pack(pady = 4)
        for i in self.lst :
            self.frame = tk.Frame(self , bg = self.bgcolor)
            for j in i :
                if j == '0' :
                    self.button = tk.Button(self.frame , text = j , width = 10 , height
= 2 , relief = 'raised')
                    self.button.pack(side = 'left' , anchor = 'nw' , pady = 5 , padx =
5 , ipadx = 10)
                    self.button.bind("<Button-1>" , self.click)
                else :
                    self.button = tk.Button(self.frame , text = j , width = 5 , height
= 2 , relief = 'raised')
                    self.button.pack(side = 'left' , anchor = 'nw' , pady = 5 , padx =
5)
                    self.button.bind("<Button-1>" , self.click)
            self.frame.pack()

    #If the given string to this fucntion is digit then it will return True else False
    def digit(self , string):
        if string.isdigit() :
            return True
        try :
            float(string)
            return True
        except :
            return False

    #All the click events here
    def click(self , event):
        self.text = event.widget.cget("text")
        if self.scvalue.get() == '0' or self.scvalue.get() == "Something went wrong" or
self.save == True:
            self.save = False
            self.scvalue.set("")
            self.screen.update()
            self.result()

        else :
            self.result()

    #Change in the entry widget with the result output is done here
    def result(self) :
        if self.text == 'root' :
            self.save = True
            from math import sqrt
            try :
                self.scvalue.set(sqrt(float(self.scvalue.get())))
            except :
                self.scvalue.set("Something went wrong")
            self.screen.update()

        elif self.text == 'int' :
            self.save = True
            self.scvalue.set(int(float(self.scvalue.get())))
            self.screen.update()

        elif self.text == 'log' :
            self.save = True
            from math import log
            try :
                self.scvalue.set(log(float(self.scvalue.get())))
            except :
                self.scvalue.set("Something went wrong")

        elif self.text == '=' :
            self.save = True
            try :
                self.scvalue.set(eval(self.scvalue.get()))
                self.screen.update()
            except :
                self.scvalue.set("Something went wrong")
                self.screen.update()

        #To clear the screen
        elif self.text == 'C' or self.text == 'CE':
            self.scvalue.set("0")
            self.screen.update()

        #Using this as backspace
        elif self.text == '<--' :
            val = self.scvalue.get()
            if len(val) == 1 or len(val) == 0:
                self.scvalue.set('0')
                self.screen.update()
            else :
                res = ''
                for i in range(len(val) - 1):
                    res += val[i]
                self.scvalue.set(res)
                self.screen.update()

        else :
            self.scvalue.set(str(self.scvalue.get()) + self.text)
            self.screen.update()
#######################################################################################
class Game(Toplevel):
    def __init__(self):
        super().__init__()
        self.title("Tic-Tac-Toe  -App Store")
        self.resizable(width = False , height = False)
        self.click = True
        self.count = 0
        self.reset_bool = False
        self.winner = False
        self.protocol("WM_DELETE_WINDOW" , self.close)
        self.menus()
        self.create_buttons()

    def close(self):
        self.destroy()
        global root
        root.wm_state('zoomed')

    def create_buttons(self):
        self.new_frame = tk.Frame(self)
        if self.reset_bool == True:
            self.click = True
            self.reset_bool = False

        #Creating buttons
        self.b1 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b1))
        self.b2 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b2))
```

```
        self.b3 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b3))

        self.b4 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b4))
        self.b5 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b5))
        self.b6 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b6))

        self.b7 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b7))
        self.b8 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b8))
        self.b9 = tk.Button(self.new_frame , text = ' ' , font = 'helvatica 13' ,
height = 3 , width =6 , command = lambda : self.clicked(self.b9))

       #Gridding
        self.new_frame.grid()
        self.b1.grid(row = 0 , column = 0)
        self.b2.grid(row = 0 , column = 1)
        self.b3.grid(row = 0 , column = 2)

        self.b4.grid(row = 1 , column = 0)
        self.b5.grid(row = 1 , column = 1)
        self.b6.grid(row = 1 , column = 2)

        self.b7.grid(row = 2 , column = 0)
        self.b8.grid(row = 2 , column = 1)
        self.b9.grid(row = 2 , column = 2)

    def reset(self):
        self.reset_bool = True
        self.count = 0
        self.winner = False
        self.new_frame.destroy()
        self.create_buttons()

    def menus(self):
        self.Mainmenu = tk.Menu(self)

        self.menu = tk.Menu(self.Mainmenu , tearoff = False)
        self.menu.add_command(label = 'Reset game' , command = self.reset)
        self.Mainmenu.add_cascade(label = 'Options' , menu = self.menu)
        self.config(menu = self.Mainmenu)

    #If the button is clicked then this will run
    def clicked(self , b):
        if b['text'] == ' ' and self.click == True:
            b.config(text = 'X')
            self.click = False
            self.count += 1
            if self.count >= 5:
                self.check_won("X")

        elif b['text'] == ' ' and self.click == False:
            b.config(text = 'O')
            self.click = True
            self.count += 1
            if self.count >= 5:
                self.check_won("O")

        else :
            self.disable()
            if msg.showerror(title = "Tic-Tac-Toe  -App Store" , message= f'That box is
already taken by {b["text"]}\nPlease click a box that is not clicked') == 'ok' :
                self.enable()

    def check_won(self , who_clicked):
        global root

        if self.b1['text'] == who_clicked and self.b2['text'] == who_clicked and
self.b3['text'] == who_clicked :
            self.b2.config(bg = 'red')
            self.b1.config(bg = 'red')
            self.b3.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message=
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        elif self.b4['text'] == who_clicked and self.b5['text'] == who_clicked and
self.b6['text'] == who_clicked :
            self.b4.config(bg = 'red')
            self.b5.config(bg = 'red')
            self.b6.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message=
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        elif self.b7['text'] == who_clicked and self.b8['text'] == who_clicked and
self.b9['text'] == who_clicked :
            self.b7.config(bg = 'red')
            self.b8.config(bg = 'red')
            self.b9.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message=
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        elif self.b1['text'] == who_clicked and self.b4['text'] == who_clicked and
self.b7['text'] == who_clicked :
            self.b4.config(bg = 'red')
            self.b1.config(bg = 'red')
            self.b7.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message=
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        elif self.b5['text'] == who_clicked and self.b2['text'] == who_clicked and
self.b8['text'] == who_clicked :
            self.b2.config(bg = 'red')
            self.b5.config(bg = 'red')
            self.b8.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message=
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        elif self.b3['text'] == who_clicked and self.b6['text'] == who_clicked and
self.b9['text'] == who_clicked :
            self.b3.config(bg = 'red')
            self.b6.config(bg = 'red')
            self.b9.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message=
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
```

```
            self.winner = True

        elif self.b1['text'] == who_clicked and self.b5['text'] == who_clicked and 
self.b9['text'] == who_clicked :
            self.b1.config(bg = 'red')
            self.b5.config(bg = 'red')
            self.b9.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message= 
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        elif self.b5['text'] == who_clicked and self.b3['text'] == who_clicked and 
self.b7['text'] == who_clicked :
            self.b3.config(bg = 'red')
            self.b5.config(bg = 'red')
            self.b7.config(bg = 'red')
            self.disable()
            msg.showinfo(title= "Tic-Tac-Toe  -App Store" , message= 
f'CONGRATULATIONS!! \n{who_clicked} Won!!!')
            self.winner = True

        if self.winner == False and self.count == 9 :
            yesno = msg.askyesno(title = "Tic-Tac-Toe  -App Store" , message= "The game 
ends with a tie\n Do you wan't to restart the game??")
            self.disable()
            if yesno :
                self.enable()
                self.reset()
            else :
                root.wm_state('zoomed')
                self.destroy()

        if self.winner == True :
            if msg.askyesno(title = "Tic-Tac-Toe  -App Store" , message= "Do you want 
to try it again") == True :
                self.reset()
                self.enable()
            else :
                root.wm_state('zoomed')
                self.destroy()

    def enable(self):
        for but in self.new_frame.grid_slaves():
            but['state'] = tk.NORMAL

    def disable(self):
        for but in self.new_frame.grid_slaves():
            if type(but) == Button:
                but.config(state = DISABLED)
########################################################################################
class Snake():

    def __init__(self):
        pygame.mixer.init()
        pygame.mixer.music.load('Game/Faded.mp3')
        pygame.mixer.music.play(-1)
        pygame.init()

        #Colors
        self.white = (225 ,225 ,225)
        self.dark_red = (200 , 0 , 0)
        self.red = (225 , 0 ,0)
        self.black = (0, 0 , 0)
        self.green = (0 , 128 ,0)
        self.light_green = (0 , 225 ,0)
        self.grey = (128 , 128 ,128)
        self.blue = (0 ,0 , 225)
        self.pink = (225, 0 , 225)
        self.yellow = (200, 200 ,0)

        #Global variables
        self.game_width = 600
        self.game_height = 600
        self.game_window = pygame.display.set_mode((self.game_width ,
self.game_height))
        pygame.display.set_caption('Snake With Roshan')
        self.clock = pygame.time.Clock()
        self.count = 0
        self.score_lst = []
        self.fps = 30
        self.esc_exit = False
        self.Ones =
['','First','Second','Third','Fourth','Fifth','Sixth','Seventh','Eighth','Ninth']
        self.Tens =
['Tenth','Eleventh','Twelveth','Thirteenth','Fourteenth','Fifteenth','Sixteenth','Seven
teenth','Eigtheenth','Nineteenth']
        self.Multiple_of_ten =
['','','Twenty','Thirty','Fourty','Fifty','Sixty','Seventy','Eigthy','Ninty']
        self.Power_of_ten = ['','Hundred','Thousand','Lakh','Crore']
        #self.WelcomeScreen()

    #Funtions
    def ExitScreen(self):

        pygame.mixer.music.load('Game/Ahrix.mp3')
        pygame.mixer.music.play(-1)
        exit_game = False
        space = True
        while not exit_game:
            self.game_window.fill(self.white)
            bgimg = pygame.image.load('Game/ExitScreen.jpg')
            bgimg = pygame.transform.scale(bgimg , (600 , 600)).convert_alpha()
            self.game_window.blit(bgimg , (0 , 0))

            self.text_screen(self.game_window , "Made by Roshan Raj" , self.pink , 400,
580 , 25)

            if self.esc_exit :
                if self.count-1 == 0 :
                    self.text_screen(self.game_window , "You exited the game in your
running first game" , self.red , 100 , 260 , 30)
                elif self.count-1 == 1 :
                    self.text_screen(self.game_window , "You played one game" ,
self.black , 200 , 200 , 30 )
                    self.text_screen(self.game_window , "Score :" +
str(self.score_lst[0]) , self.black , 200 , 230 , 30 )
                    self.text_screen(self.game_window , "You exited the game in your
running second game" , self.red , 100 , 260 , 30)
                else :
                    self.text_screen(self.game_window , "You played " + str(self.count-
1) + " games" , self.yellow , 5, 5 , 30)
                    self.text_screen(self.game_window , "Your scores are the following
:" , self.yellow , 5 , 30 , 30)
                    i = 0
                    while i < len(self.score_lst) :
                        self.text_screen(self.game_window , self.numbers_to_words(i+1)
+": " + str(self.score_lst[i]) , self.yellow , 5 , 60 + (i* 30), 30)
                        i += 1
                    self.text_screen(self.game_window , "You exited the game in your
running\n" + self.numbers_to_words(i+1) + " game" , self.yellow , 5 , 60 + (i* 30) ,
30)

            else :
                if self.count == 0 :
                    self.text_screen(self.game_window , "You haven't tried the game" ,
self.black , 190, 200 , 30)
                    self.text_screen(self.game_window , "You should try it" ,
self.black , 200, 230 , 30)
                elif self.count == 1 :
                    self.text_screen(self.game_window , "You played one game" ,
self.black , 200 , 200 , 30 )
                    self.text_screen(self.game_window , "Score :" +
str(self.score_lst[0]) , self.black , 200 , 230 , 30 )
                else :
                    self.text_screen(self.game_window , "You played " + str(self.count)
+ " games" , self.yellow , 5, 5 , 30)
                    self.text_screen(self.game_window , "Your scores are the following
:" , self.yellow , 5 , 30 , 30)
                    for i in range(len(self.score_lst)) :
                        self.text_screen(self.game_window , self.numbers_to_words(i+1)
+": " + str(self.score_lst[i]) , self.yellow , 5 , 60 + (i* 30), 30)

            pygame.display.update()
            for event in pygame.event.get():
                if event.type == pygame.QUIT :
                    exit_game = True
                    break

                elif event.type == pygame.KEYDOWN :
                    if event.key == pygame.K_SPACE and space:
                        pygame.mixer.music.pause()
                        space = False
                    elif event.key == pygame.K_SPACE and not space :
                        pygame.mixer.music.unpause()
                        space = True

                    if event.key == pygame.K_ESCAPE :
                        exit_game = True
                        break

                    elif event.key == pygame.K_r:
                        self.esc_exit = False
                        self.GameLoop()

                    elif event.key == pygame.K_s :
                        self.shortcuts()
        pygame.quit()
        exit()

    def GameLoop(self) :

        pygame.mixer.music.load('Game/back.mp3')
        pygame.mixer.music.play(-1)

        #Variables
        self.count +=1
        game_over = False
        exit_game = False
        snake_x = 295
        snake_y = 295
        snake_size = 10
        velocity = 5
        velocity_x = 0
        velocity_y = 0
        food_size = snake_size
        food_x = random.randint(0 + snake_size , self.game_width - snake_size)
        food_y = random.randint(0 + snake_size , self.game_height - snake_size)
        big_food_size = 15
        big_food_x = random.randint(40 + snake_size ,self.game_width - snake_size - 30)
        big_food_y = random.randint(60 , self.game_width - snake_size - 30)
        score = 0
        snake_length = 1
        snake_list = []
        last_key = 0
        big_count = False
        space = True
        start_time = 0
        end_time = 0

        if not os.path.exists('Game/HighScore.txt') :
            with open('Game/HighScore.txt' , 'w') as f :
                f.write("0")
        with open("Game/HighScore.txt" , 'r') as f :
            HighScore = f.read()
########################################################################################
        #GameLoop
        while not exit_game :
            if game_over :
                with open('Game/HighScore.txt' , 'w') as f :
                    f.write(str(HighScore))
                self.game_window.fill(self.white)
                game_bgimg = pygame.image.load('Game/GameOver.jpg')
                game_bgimg = pygame.transform.scale(game_bgimg , (600 ,
600)).convert_alpha()
                self.game_window.blit(game_bgimg , (0 , 0))
                self.text_screen(self.game_window , "Game Over! press enter to restart"
, self.red ,135 , self.game_height/2-60 , 35)
                self.text_screen(self.game_window , "Esc to exit the game" , self.red
,180 , self.game_height/2-25 , 35)
                self.text_screen(self.game_window , "Made by Roshan Raj" , self.pink ,
400, 580 , 25)
                for event in pygame.event.get():
                    if event.type == pygame.QUIT :
                        self.ExitScreen()

                    elif event.type == pygame.KEYDOWN :

                        if event.key == pygame.K_SPACE and space:
                            pygame.mixer.music.pause()
                            space = False
                        elif event.key == pygame.K_SPACE and not space :
                            pygame.mixer.music.unpause()
                            space = True

                        if event.key == pygame.K_RETURN or event.type == pygame.K_r :
                            self.GameLoop()
                        elif event.key == pygame.K_ESCAPE:
                            self.ExitScreen()
                        elif event.key == pygame.K_s :
                            self.shortcuts()

            else :
                for event in pygame.event.get():
                    if event.type == pygame.QUIT :
                        exit_game = True
                        self.esc_exit = True
                        self.ExitScreen()
                    elif event.type == pygame.MOUSEBUTTONDOWN :
                        if event.pos[0] <= 50 and event.pos[1] <= 50:
                            self.shortcuts()
                        elif (event.pos[0] >= 60 and event.pos[0] <= 110) and
event.pos[1] <= 50:
                            unpause = False
                            while not unpause :
                                for i in pygame.event.get():
                                    if i.type == pygame.QUIT :
                                        unpause = True
                                        exit_game = True
                                        break
                                    elif (i.type == pygame.KEYDOWN and i.key ==
pygame.K_p):
                                        unpause = True
                                        break
                                    elif i.type == pygame.MOUSEBUTTONDOWN and
((i.pos[0] >= 120 and i.pos[0]) and i.pos[1]<= 50):
                                        unpause = True
                                        break

                    elif event.type == pygame.KEYDOWN :

                        if event.key == pygame.K_SPACE and space:
                            pygame.mixer.music.pause()
                            space = False
                        elif event.key == pygame.K_SPACE and not space :
                            pygame.mixer.music.unpause()
                            space = True

                        if event.key == pygame.K_ESCAPE :
                            exit_game = True
                            self.esc_exit = True
                            self.ExitScreen()

                        elif event.key == pygame.K_s:
                            self.shortcuts()
                        elif event.key == pygame.K_r :
                            self.GameLoop()
                        elif event.key == pygame.K_p :
                            unpause = False
                            while not unpause :
                                for i in pygame.event.get():
                                    if i.type == pygame.QUIT :
                                        unpause = True
                                        exit_game = True
                                        break
                                    elif (i.type == pygame.KEYDOWN and i.key ==
pygame.K_p):
                                        unpause = True
                                        break
                                    elif i.type == pygame.MOUSEBUTTONDOWN and
((i.pos[0] >= 120 and i.pos[0]) and i.pos[1]<= 50):
                                        unpause = True
                                        break

                        elif event.key == pygame.K_b :
                            pygame.quit()
                            quit()

                        elif event.key == pygame.K_RIGHT :
                            if last_key == pygame.K_LEFT :
                                continue
                            velocity_x = velocity
                            velocity_y = 0
                            last_key = pygame.K_RIGHT

                        elif event.key == pygame.K_DOWN :
                            if last_key == pygame.K_UP :
                                continue
                            velocity_y = velocity
                            velocity_x = 0
                            last_key = pygame.K_DOWN

                        elif event.key == pygame.K_UP :
                            if last_key == pygame.K_DOWN :
                                continue
                            velocity_y = -velocity
                            velocity_x = 0
                            last_key = pygame.K_UP

                        elif event.key == pygame.K_LEFT :
                            if last_key == pygame.K_RIGHT :
                                continue
                            velocity_x = -velocity
                            velocity_y = 0
                            last_key = pygame.K_LEFT

                snake_x += velocity_x
                snake_y += velocity_y

                if abs(food_x - snake_x) < 7 and abs(food_y - snake_y) < 7 :
                    beepSound = pygame.mixer.Sound('Game/beep.wav')
                    beepSound.play()
                    score += 1
                    velocity += 0.2
                    big_count = True
                    food_x = random.randint(0 + 5 *snake_size ,self.game_width - 5
*snake_size)
                    food_y = random.randint(60 , self.game_height - 5 *snake_size)
                    snake_length += 1
                    if score > int(HighScore) :
                        HighScore = score
                    if score %5 == 0:
                        start_time = time.time()
                        i = 1

                #Setting logo
                self.game_window.fill(self.grey)
                setting = pygame.image.load('Game/setting.png')
                setting = pygame.transform.scale(setting , (50 , 50)).convert_alpha()
                self.game_window.blit(setting , (0 , 0))

                #Pause logo
                pause = pygame.image.load('Game/pause.png')
                pause = pygame.transform.scale(pause , (50 , 50)).convert_alpha()
                self.game_window.blit(pause , (60 , 0))

                #Resume logo
                resume = pygame.image.load('Game/resume.png')
                resume = pygame.transform.scale(resume , (50 , 50)).convert_alpha()
                self.game_window.blit(resume , (120 , 0))
                #game_bgimg = pygame.image.load('snake.png')
                #game_bgimg = pygame.transform.scale(game_bgimg , (600 ,
600)).convert_alpha()
                #self.game_window.blit(game_bgimg , (0 , 0))
                if score%5 == 0 and big_count:
                    pygame.draw.circle(self.game_window , self.red , [big_food_x
,big_food_y] , big_food_size)
```

```
                    if abs(big_food_x - snake_x) < 12 and abs(big_food_y - snake_y) <
12:
                        big_count = False
                        beepSound = pygame.mixer.Sound('Game/beep_big.wav')
                        beepSound.play()
                        score += 5
                        big_food_x = random.randint(0 + 5 *snake_size , self.game_width
- 5 *snake_size)
                        big_food_y = random.randint(60 , self.game_height - 5
*snake_size)
                        snake_length += 1
                        if score > int(HighScore) :
                            HighScore = score

                    end_time = time.time()
                    if i < 2 :
                        self.text_screen(self.game_window , "Timer : " + str(4 -i) ,
self.green , 5 , 560 , 35)
                    elif i <3:
                        self.text_screen(self.game_window , "Timer : " + str(4 -i) ,
self.yellow , 5 , 560 , 35)
                    elif i < 4:
                        self.text_screen(self.game_window , "Timer : " + str(4 -i) ,
self.red , 5 , 560 , 35)

                    if (end_time - start_time) >= i :
                        i += 1
                    if (end_time - start_time) >= 4:
                        big_count = False

                head = []
                head.append(snake_x)
                head.append(snake_y)
                snake_list.append(head)
                if len(snake_list) > snake_length :
                    del snake_list[0]
                if snake_x>(self.game_width - snake_size) or snake_y>(self.game_height
- snake_size) or snake_x<(snake_size/2) or snake_y<(snake_size/2):
                    game_over = True
                    self.score_lst.append(score)
                    pygame.mixer.music.load('Game/Astronomia.mp3')
                    pygame.mixer.music.play(-1)

                elif head in snake_list[:-1] :
                    game_over = True
                    self.score_lst.append(score)
                    pygame.mixer.music.load('Game/adhi.mp3')
                    pygame.mixer.music.play(-1)
                self.text_screen(self.game_window , "Score : " +str(score) , self.blue
, 260 ,5 , 35)
                self.text_screen(self.game_window , "High Score : " + str(HighScore) ,
self.blue , 400,5 , 35)
                self.text_screen(self.game_window , "Made by Roshan Raj" , self.pink ,
400, 580 , 25)
                pygame.draw.rect(self.game_window , self.red , [food_x , food_y ,
food_size , food_size])
                self.plot_snake(self.game_window , self.black , snake_list ,
snake_size)
            pygame.display.update()
            self.clock.tick(self.fps)
        pygame.quit()

    def numbers_to_words(self ,n):
        s=0
        w=''
        while n>0:
            if s==1:
                r=n%10
                if r!=0:
                    w = self.Ones[r] + ' Hundred ' + w
                n=n//10
            else:
                r=n%100
                x=r%10
                m=r//10
                if m==1:
                    w = self.Tens[x] + ' ' + self.Power_of_ten[s] + ' ' + w
                elif m==0:
                    w= self.Ones[x] + ' ' + self.Power_of_ten[s] + ' ' + w
                else:
                    w =self.Multiple_of_ten[m] + ' ' + self.Ones[x] + ' ' +
self.Power_of_ten[s] + ' ' + w
                n=n//100
            s+=1
        return w

    def plot_snake(self , game_window , color , snake_list , snake_size):
        for x , y in snake_list :
            pygame.draw.rect(game_window , color , [x , y , snake_size, snake_size])
        x , y = snake_list[-1][0] , snake_list[-1][1]
        pygame.draw.rect(game_window , self.light_green , [x , y , snake_size,
snake_size])

    def shortcuts(self):

        exit_game = False
        while not exit_game :
            self.game_window.fill(self.white)
            self.text_screen(self.game_window , "" , self.black , 200 , 200 , 30)
            for event in pygame.event.get():

                if event.type == pygame.QUIT :
                    exit_game = True

                elif event.type == pygame.KEYDOWN :
                    if event.key == pygame.K_ESCAPE or event.key == pygame.K_s or
event.key == pygame.K_RETURN :
                        exit_game = True

            self.text_screen(self.game_window , "Game Shortcuts" ,self.black , 5 ,5 ,
40)
            self.text_screen(self.game_window , "P : Pause/Unpause Game" , self.grey ,5
,50 , 30)
            self.text_screen(self.game_window , "Spacebar : Pause/Unpause Music" ,
self.grey ,5 ,80 , 30)
            self.text_screen(self.game_window , "R : Restart Game" , self.grey ,5 ,110
, 30)
            self.text_screen(self.game_window , "S : Shortcuts" , self.grey ,5 ,140 ,
30)
            self.text_screen(self.game_window , "Esc : Exit Game" , self.grey ,5 ,170 ,
30)

            pygame.display.update()

    def text_screen(self , game_window , text , color , x , y , size) :
        font = pygame.font.SysFont(None , size)
        screen_text = font.render(text , True , color)
        self.game_window.blit(screen_text , [x , y])

    def WelcomeScreen(self):
        exit_game = False
        while not exit_game:
            self.game_window.fill((233,210,229))
            bgimg = pygame.image.load('Game/Welcome.jpg')
            bgimg = pygame.transform.scale(bgimg , (600 , 600)).convert_alpha()
            self.game_window.blit(bgimg , (0 , 0))
            self.text_screen(self.game_window , "Welcome to snakes" , self.black , 200
, 250 , 30)
            self.text_screen(self.game_window , "Press SpaceBar to Play" , self.black ,
180 , 275 , 30)
            self.text_screen(self.game_window , "Made by Roshan Raj" , self.pink , 10,
580 , 25)
            for event in pygame.event.get():
                if event.type == pygame.QUIT :
                    exit_game = True

                if event.type == pygame.MOUSEBUTTONDOWN:
                    self.GameLoop()

                if event.type == pygame.KEYDOWN :
                    if event.key == pygame.K_SPACE :
                        self.GameLoop()

                    elif event.key == pygame.K_ESCAPE :
                        exit_game = True

                    elif event.key == pygame.K_s :
                        self.shortcuts()

            pygame.display.update()
            self.clock.tick(self.fps)
        self.ExitScreen()

#########################################################################################
#Functions
def clock():
    global labl
    cur = datetime.now()
    hour = str(cur.hour)
    minute = str(cur.minute)
    second = str(cur.second)
    labl.config(text = hour + ':' + minute + ':' + second)
    labl.after(1000 , clock)

def close():
    yesno = msg.askyesno(title= "Exit",message= 'Are you sure?')
    if yesno:
        root.destroy()

def gridding():                      #Anything to be added in body_right will be done
here
    global body_right , labl
    homepage_frame = Frame(body_right, height=100, width=100, bg='#606060',
highlightbackground="black", highlightthickness=1)

    homepage_title = Label(homepage_frame, text='ATOMIC ENERGY CENTRAL SCHOOL - 1,
JADUGODA', bg='#00b050', font='Bebas 40', height= 2, fg='White')
    homepage_title.grid(row=0, column=0, columnspan=3, sticky='NEW', padx= 10, pady=10)

    homepage_subtitle1 = Label(homepage_frame, text='COMPUTER PROJECT', bg='#92D050',
font='Bebas 25', height= 2)
    homepage_subtitle1.grid(row=1, column=0, columnspan=3, sticky='NEW', padx= 10)

    BOX1 = Frame(homepage_frame, bg='#606060')
    BOX1.grid(row=2, column=0, sticky='NEWS', padx=10, pady=10)
    homepage_frame.rowconfigure(2, weight=1)
    BOX1.columnconfigure(0, weight=1)

    project_name_label = Label(BOX1, text='Project Name',bg='#262626', fg='white', 
height=3, font='Roboto') ;project_name_label.grid(row=0, sticky='WE', pady=5)
    name_of_student_label = Label(BOX1, text='Name of the Students',bg='#262626', 
fg='white', height=3, font='Roboto'); name_of_student_label.grid(row=1, sticky='WE', 
pady=5)
    name_of_teacher_label = Label(BOX1, text='Name of the Teacher',bg='#262626', 
fg='white', height=3, font='Roboto'); name_of_teacher_label.grid(row=2, sticky='WE', 
pady=5)

    BOX2 = Frame(homepage_frame, bg='#606060')
    BOX2.grid(row=2, column=1, sticky='NEWS', padx=10, pady=10, columnspan=2)
    BOX2.columnconfigure(1, weight=1)
    BOX2.columnconfigure(2, weight=1)

    project_name= Label(BOX2, text='Application Management', fg='white', height=3, 
bg='#262626', font='Roboto') ; project_name.grid(row=0, columnspan=3, sticky='NEWS', 
pady=5)
    name_of_student= Label(BOX2, text='Avitesh Murmu \n Roshan Raj', fg='white', 
height=3, bg='#262626', font='Roboto') ; name_of_student.grid(row=1, columnspan=3, 
sticky='NEWS', pady=5)
    name_of_teacher= Label(BOX2, text='S. K. Mukherjee', fg='white', height=3, 
bg='#262626', font='Roboto') ; name_of_teacher.grid(row=2, columnspan=3, sticky='NEWS', 
pady=5)


    for i in range(3):
        homepage_frame.columnconfigure(i, weight=1)

    homepage_frame.grid(row = 2 , column = 0, sticky='EW', padx=35, pady=10)
    body_right.rowconfigure(2, weight=1)

def validate_login():                #Login works here
    global body_right
    mydb = ms.connect(host = 'localhost' , user = 'root' , passwd = pas , database = 
'project')
    mycursor = mydb.cursor()

    def already_exists():
        global result_label
        usd = username.get()
        pw = password.get()
        mycursor = mydb.cursor()
        mycursor.execute("Select * from login")
        res = mycursor.fetchall()

        for i in range(len(res)) :
            if res[i][0].lower() == usd.lower():
                if res[i][1] == pw :
                    result_label.config(text = 'Already an account available\nSignined 
to that id')
                    result_label.grid(row =1000 , column = 0)
                else :
                    if res[i][1].lower() == pw.lower():
                        result_label.config(text = 'This Username already 
exists\nPassword is wrong\nYou can sign in')
                        result_label.grid(row = 1000 , column = 0)
                return False
        return True

    def threads():
        try :
            username.set("")
            password.set("")
            home_command()
        except :
            pass

    def logout():
        global sign_up , header , logined
        logined = False
        sign_in.config(text = 'Sign In' , command = signin_window, font='Roboto 9')
        sign_up =Button(header, text='Sign Up', command=signup_window ,height=
3,width=7, bg='#385723', relief=GROOVE , fg='white', font='Roboto 9')
        sign_up.grid(column=4,row=0, sticky=(E), padx=10, pady=5)
        header.columnconfigure(2, weight=1)
        home_command()

    def login():
        global mycursor , result_label , logined
        usd = username.get()
        passWord = password.get()
        mycursor = mydb.cursor()
        mycursor.execute("Select * from login")
        res = mycursor.fetchall()

        try :
            result_label.config()
        except :
            result_label = Label(body_right , font = 'Robot 20 bold' , bg = '#262626')

        if valididty(usd , passWord) == False:
            count = False
            for i in range(len(res)) :
                if res[i][0].lower() == usd.lower():
                    count = True
                    if res[i][1] == passWord :
                        result_label.config(text = "Sign in done" , fg = '#28e84a')
                        result_label.grid(row = 1000 , column = 0)
                        logined = True
                        sign_up.destroy()
                        sign_in.config(text = 'Logout' , command = logout)

                    else :
                        if res[i][1].lower() == passWord.lower():
                            result_label.config(text = "Please check the lower
case/upper case in the password" , fg = "#f1ec14")
                            result_label.grid(row = 1000 , column = 0)
                        else :
                            result_label.config(text = "Sorry wrong password" , fg =
'red',)
                            result_label.grid(row = 1000 , column = 0)
                    break

            if not count:
                result_label.config(text = f"You should first sign up\nNo id named :
{usd}" , fg = '#70d1cc')
                result_label.grid(row = 1000 , column = 0)

    def valididty(user , passwrd):
        global result_label

        try :
            result_label.config()
        except :
            result_label = Label(body_right , font = 'Robot 20 bold' , bg = '#262626')
        if (user == '' or user.isspace() or user.isalpha() == False) and (passwrd == ''
or passwrd.isspace() or passwrd.isalpha() == False):
```

```
            result_label.config(text = "Sorry not a valid username and password" , fg = 
'red') ; result_label.grid(row = 1000 , column = 0)
        elif user == '' or user.isspace()  or user.isalpha() == False:
            result_label.config(text = "Sorry not a valid username" , fg = 'red') ; 
result_label.grid(row = 1000 , column = 0)
        elif passwrd == '' or passwrd.isspace()  or passwrd.isalpha() == False:
            result_label.config(text = "Sorry not a valid password" , fg = 'red') ; 
result_label.grid(row = 1000 , column = 0)

        elif len(passwrd) < 8 and len(user) < 8 :
                result_label.config(text = 'Sorry not a valid Username and 
Password\nUsername Password too small\nMust have 8 letters' , fg = 'red')
                result_label.grid(row = 1000 , column = 0)
        elif len(passwrd) < 8 :
                result_label.config(text = 'Sorry not a valid Password\nPassword too 
small \nMust have 8 letters' , fg = 'red')
                result_label.grid(row = 1000 , column = 0)
        elif len(user) < 8 :
                result_label.config(text = 'Sorry not a valid Username\nUsername too 
small \nMust have 8 letters' , fg = 'red')
                result_label.grid(row = 1000 , column = 0)
        else :
                return False

    def makeId():
        global result_label , logined
        usd = username.get()
        pw = password.get()
        try :
            result_label.config()
        except :
            result_label = Label(body_right , font = 'Robot 20 bold' , bg = '#262626')

        if valididty(usd , pw) == False :
            mycursor = mydb.cursor()
            if already_exists() == True :
                s = 'insert into login values(%s , %s);'
                tup = (usd , pw )
                mycursor.execute(s , tup)
                mycursor.execute("commit")
                result_label.config(text = "Sign up done" , fg = '#28e84a')
                result_label.grid(row = 1000 , column = 0)
                logined = True
                sign_in.config(text = 'Logout' , command = logout)
                sign_up.destroy()


    if sup :
        makeId()
    elif sin :
        login()
    if logined :
        Thread(target= threads).start()

def clear_entry():
    username.set("")
    password.set("")

def destroy_everything():
    for i in body_right.grid_slaves():
        i.destroy()
    for i in range(200):
        body_right.rowconfigure(i, weight=0)

def signup_window():
        global body_right , password , username , login_frame , sin , history_list ,
sup
        if history_list[-1] != 'signup' :
            history_list.append('signup')
        sup = True
        sin = False
        clear_entry()
        destroy_everything()
        login_frame = Frame(            body_right   , bg='#606060', height=200 ,
highlightbackground="black", highlightthickness=1)
        login_label_heading = Label(    login_frame , text = 'Sign Up', bg='#262626',
font='Bebas 25', height=2 , fg='White')
        login_label = Label(            login_frame , text = 'Username', bg='#222222',
fg='white', font='Roboto')
        login_field = Entry(            login_frame , textvariable = username)
        password_label = Label(         login_frame , text='Password', bg='#222222',
fg='white', font='Roboto')
        password_field = Entry(         login_frame , textvariable = password , show
="\u2022")
        submit_button = Button(         login_frame , text='Submit', width=20 , command
= validate_login, font='Roboto')
        #sign_in_reference = Button(login_frame, text='Already Have')

        login_frame.grid(row = 33,column=0,sticky='NEW',            padx=35,
pady=10)
        login_label_heading.grid(row = 0, column=0,sticky='WE', padx=5, pady=5,
columnspan=2)
        login_label.grid(row =1,column=0,sticky='WE',           padx=5, pady=5,
ipadx=10, ipady=10)
        login_field.grid(row =1,column=1,sticky='WE',           padx=5, pady=5,
ipadx=10, ipady=10)
        password_label.grid(row =2,column=0 ,sticky='WE',       padx=5,pady=5,
ipadx=10, ipady=10)
        password_field.grid(row =2,column=1,sticky='WE',        padx=5, pady=5,
ipadx=10, ipady=10)
        submit_button.grid(row =3,column=0,columnspan=2,        pady=5)

        body_right.rowconfigure(33, weight=1)           #Expanding right body's (login
frame) vertically
        login_frame.columnconfigure(0, weight=1)        #Expanding login frame's (login
label) horizontally; column one
        login_frame.columnconfigure(1, weight=1)        #Expanding login frame's (login
label) horizontally; colmn two

def signin_window():
        global body_right , password , username , login_frame , sin , history_list ,
sup
        if history_list[-1] != 'signin' :
            history_list.append('signin')
        sin = True
        sup = False
        clear_entry()
        destroy_everything()
        login_frame = Frame(            body_right   , bg='#606060', height=200 ,
highlightbackground="black", highlightthickness=1)
        login_label_heading = Label(    login_frame , text = 'Sign In', bg='#262626',
font='Bebas 25', height=2 , fg='White')
        login_label = Label(            login_frame , text = 'Username', bg='#222222',
fg='white', font='Roboto')
        login_field = Entry(            login_frame , textvariable = username)
        password_label = Label(         login_frame , text='Password', bg='#222222',
fg='white', font='Roboto')
        password_field = Entry(         login_frame , textvariable = password , show
="\u2022")
```

```
        submit_button = Button(         login_frame , text='Submit', width=20 , command
= validate_login, font='Roboto')
        #sign_in_reference = Button(login_frame, text='Already Have')

        login_frame.grid(row = 34,column=0,sticky='NEW',            padx=35,
pady=10)
        login_label_heading.grid(row = 0, column=0,sticky='WE', padx=5, pady=5,
columnspan=2)
        login_label.grid(row =1,column=0,sticky='WE',           padx=5, pady=5,
ipadx=10, ipady=10)
        login_field.grid(row =1,column=1,sticky='WE',           padx=5, pady=5,
ipadx=10, ipady=10)
        password_label.grid(row =2,column=0 ,sticky='WE',       padx=5,pady=5,
ipadx=10, ipady=10)
        password_field.grid(row =2,column=1,sticky='WE',        padx=5, pady=5,
ipadx=10, ipady=10)
        submit_button.grid(row =3,column=0,columnspan=2,        pady=5)

        body_right.rowconfigure(34, weight=1)           #Expanding right body's (login
frame) vertically
        login_frame.columnconfigure(0, weight=1)        #Expanding login frame's (login
label) horizontally; column one
        login_frame.columnconfigure(1, weight=1)        #Expanding login frame's (login
label) horizontally; colmn two

def check_connection_with_mysql(check = False):
        #Checking that project database is their or not if not then will create that
        my = ms.connect(host = 'localhost' , user = 'root' , passwd = pas)
        mycur = my.cursor()
        mycur.execute("Show databases;" )
        databases = mycur.fetchall()

        if ('project',) not in databases :
            mycur.execute("Create database project;")

        #Checking that login table is in the database if not then creating that
        global mydb ; mydb = ms.connect(host = 'localhost' , user = 'root' , passwd =
pas , database = 'project')
        mycur = mydb.cursor()
        mycur.execute("Show tables ;")
        res = mycur.fetchall()
        if ('login',) not in res :
            mycur.execute("Create table login(Id varchar(30) , PassWord varchar(20))
;")

        if check == True :
            try :
                for line in open("APPS.sql"):
                    mycur.execute(line)
                mycur.execute('commit')
            except :
                pass

def back_command():
        global login_frame , history_list ,app_list_frame , contact_us_frame ,
delete_frame , bonus_zone
        try :
            app_list_frame.destroy()
            result_label.destroy()
        except :
            pass

        try :
            login_frame.destroy()
            result_label.destroy()
        except :
            pass

        try :
            edit_apps_frame.destroy()
            result_label.destroy()
        except :
            pass

        try :
            contact_us_frame.destroy()
        except :
            pass

        try :
            delete_update_frame.destroy()
        except :
            pass

        try :
            bonus_zone.destroy()
        except :
            pass

        if len(history_list) > 1:
            del history_list[-1]
            if history_list[-1]  == 'main':
                gridding()
            elif history_list[-1] == 'signin' :
                del history_list[-1]
                signin_window()
            elif history_list[-1] == 'signup' :
                del history_list[-1]
                signup_window()
            elif history_list[-1] == 'apps' :
                del history_list[-1]
                all_apps_command()
            elif history_list[-1] == 'entertainment' :
                del history_list[-1]
                entertainment_command()
            elif history_list[-1] == 'games' :
                del history_list[-1]
                games_command()
            elif history_list[-1] == 'addapps' :
                del history_list[-1]
                add_apps_command()
            elif history_list[-1] == 'contact' :
                del history_list[-1]
                contact_us_command()
            elif history_list[-1] == 'delete' :
                del history_list[-1]
                app_delete_command()
            elif history_list[-1] == 'bonus' :
                del history_list[-1]
                bonus_zone_command()

def home_command():
    global login_frame , app_list_frame , history_list , edit_apps_frame
    history_list = ['main']
    try :
        app_list_frame.destroy() ;result_label.destroy() ; gridding()
    except :
        destroy_everything() ; gridding()

    try :
        login_frame.destroy() ; destroy_everything() ; gridding()
    except :
        destroy_everything() ; gridding()

    try :
        edit_apps_frame.destroy() ; destroy_everything() ; gridding()
    except :
        destroy_everything() ; gridding()

    try :
        delete_update_frame.destroy() ; gridding()
    except :
        destroy_everything() ; gridding()

    try :
        bonus_zone.destroy() ; gridding()
    except :
        destroy_everything() ; gridding()

def show_apps(reslt = None):
    global my_tree
    if reslt == None:
        mydb = ms.connect(host = 'localhost' , user = 'root' , passwd = pas , database
= 'project')
        mycur = mydb.cursor()
        if entertainment_bool :
            mycur.execute("Select * from apps where category = 'entertainment' order by
appid;")
        elif games_bool:
            mycur.execute("Select * from apps where category = 'games' order by
appid;")
        else :
            mycur.execute('SELECT * FROM apps order by appid;')
        records = mycur.fetchall()
        return records

    else :
        clear_apps()
        my_tree.tag_configure("oddrow", background='#859bbc',foreground='black')
        my_tree.tag_configure("evenrow",background='#414141',foreground='white')

        for i in range(len(reslt)):
            if i%2 == 0:
                my_tree.insert(parent= '' , index= 'end' , iid=i , text = 'clear',
values = reslt[i] , tags=('evenrow',))
            elif i%2 != 0:
                my_tree.insert(parent= '' , index= 'end' , iid=i , text = 'clear',
values = reslt[i] , tags=('oddrow',))

def clear_apps():
    global my_tree
    for i in my_tree.get_children():
        my_tree.delete(i)

def searching_all_apps():
    global result_label
    entered_name = search_entry.get()
    if entered_name == '' or entered_name.isspace():
        show_apps()

    else :
        mydb = ms.connect(host = 'localhost' , user = 'root' , passwd = pas , database
= 'project')
        mycur = mydb.cursor()
        if entertainment_bool :
            mycur.execute(f"Select * from apps where name LIKE '{entered_name}%' and 
category = 'entertainment' order by appid")
        elif games_bool :
            mycur.execute(f"Select * from apps where name LIKE '{entered_name}%' and 
category = 'Games' order by appid")
        else :
            mycur.execute(f"Select * from apps where name LIKE '{entered_name}%' order 
by appid")
        res = mycur.fetchall()
        if len(res) > 0:
            show_apps(reslt= res)
        else :
            if entertainment_bool:
                try :
                    result_label.config(text = f"Sorry, No App Available With 
'{entered_name}' Name\n In Entertainment category" , fg = '#ed4242')
                    result_label.grid(row = 1000 , column = 0)
                except :
                    result_label = Label(body_right , text = f"Sorry, No App Available 
With '{entered_name}' Name\n In Entertainment category" , fg = '#ed4242' ,font = 'Robot 
20 bold' , bg = '#262626')
                    result_label.grid(row = 1000 , column = 0)
            elif games_bool :
                try :
                    result_label.config(text = f"Sorry, No App Available With 
'{entered_name}' name\n In Games Gategory" , fg = '#ed4242')
                    result_label.grid(row = 1000 , column = 0)
                except :
                    result_label = Label(body_right , text = f"Sorry, No App Available 
With '{entered_name}' name\n In Games Category" , fg = '#ed4242' ,font = 'Robot 20 
bold' , bg = '#262626')
                    result_label.grid(row = 1000 , column = 0)
            else :
                try :
                    result_label.config(text = f"Sorry, No App Available With 
'{entered_name}' name" , fg = '#ed4242')
                    result_label.grid(row = 1000 , column = 0)
                except :
                    result_label = Label(body_right , text = f"Sorry, No App Available 
With '{entered_name}' Name" , fg = '#ed4242' ,font = 'Robot 20 bold' , bg = '#262626')
                    result_label.grid(row = 1000 , column = 0)

def apps_command():
    global body_right , history_list , app_list_frame , searchforallapps , search_entry 
, entertainment_bool , games_bool , search_entry_var , my_tree

    destroy_everything()
    if entertainment_bool :
        if history_list[-1] != 'entertainment' :
            history_list.append('entertainment')
        games_bool = False
    elif games_bool :
        if history_list[-1] != 'games' :
            history_list.append('games')
        entertainment_bool = False
    else :
        if history_list[-1] != 'apps' :
            history_list.append('apps')

    result = show_apps(reslt= None)
    app_list_frame = Frame(body_right , highlightbackground="black", 
highlightthickness=1, bg='#606060')

    # my_scrollbar = Scrollbar(app_list_frame)
    # my_scrollbar.grid(row = 1 , column = 1 , sticky = 'NES')
    my_tree = ttk.Treeview(app_list_frame)  #, yscrollcommand = my_scrollbar.set
    search_bar = Frame(app_list_frame)
    search_bar.grid(        row = 0 , column = 0  , sticky='EW' , padx = 10 , pady =
(10,0), ipadx = 10)
    search_bar.columnconfigure(0, weight=8)
    search_bar.columnconfigure(1, weight=1)

    search_entry_var = StringVar()
    search_entry =          Entry(search_bar , bg = '#dedede' , textvariable =
search_entry_var , highlightbackground = 'WHITE' , font = ("Roboto 19") , fg = 'black'
, relief = SUNKEN ,borderwidth = 1)
    search_entry.grid(      row = 0 , column = 0 , sticky = 'NEWS' , padx = (0,0))

    searchforallapps =      Button(search_bar, bg = '#606060' , fg = 'white' , height =
2 , width = 12,relief = RAISED, text='Search',activebackground='#4F7942',
activeforeground='white' , font='Roboto', command = searching_all_apps)
    searchforallapps.grid(  row = 0 , column = 1, sticky='NEWS')


    # Creating columns
    my_tree['column'] = ('App
IDs','Category','Name','Developer','Size','Views','Description')
    my_tree.column('#0' ,            minwidth = 0   , width = 0)
    my_tree.column('App IDs' ,       minwidth = 74  , width = 74  , anchor = W)
    my_tree.column('Category' ,      minwidth = 208 , width = 208 , anchor = W)
    my_tree.column('Name' ,          minwidth = 186 , width = 186 , anchor = W)
    my_tree.column('Developer' ,     minwidth = 186 , width = 186 , anchor = W)
    my_tree.column('Size' ,          minwidth = 70  , width = 70  , anchor = W)
    my_tree.column('Views' ,         minwidth = 186 , width = 186 , anchor = W)
    my_tree.column('Description',    minwidth = 280 , width = 280 , anchor = W)


    # Giving the columns heading
    my_tree.heading('#0' ,           text= 'Label' ,        anchor = W)
    my_tree.heading('App IDs' ,      text = 'App IDs' ,     anchor = W)
    my_tree.heading('Category' ,     text = 'Category' ,    anchor = W)
    my_tree.heading('Name' ,         text = 'Name' ,        anchor = W)
    my_tree.heading('Developer' ,    text = 'Developer' ,   anchor = W)
    my_tree.heading('Size' ,         text = 'Size' ,        anchor = W)
    my_tree.heading('Views' ,        text ='Views'  ,       anchor = W)
    my_tree.heading('Description' , text = 'Description' , anchor = W)


    # Styling
    style = ttk.Style()
    style.theme_use('clam')
    style.configure("Treeview", rowheight = 45 , fieldbackground = '#8fb198')
    style.map('Treeview' , background = [('selected' , '#848179')])


    # Giving rows colors
    my_tree.tag_configure("oddrow", background='#859bbc',foreground='black')
    my_tree.tag_configure("evenrow",background='#414141',foreground='white')

    for i in range(len(result)):
        if i%2 == 0:
            my_tree.insert(parent= '' , index= 'end' , iid=i , text = 'clear', values =
result[i] , tags=('evenrow',))
        elif i%2 != 0:
            my_tree.insert(parent= '' , index= 'end' , iid=i , text = 'clear', values =
result[i], tags=('oddrow',))

    # my_scrollbar.config(command = my_tree.yview)
    app_list_frame.grid(padx=40, pady=10, ipady=10, ipadx=10)
    my_tree.grid(padx = 3)
    app_list_frame.columnconfigure(0 , weight = 1)
    app_list_frame.rowconfigure(1 , weight = 1)

def all_apps_command():
    global entertainment_bool , games_bool
    entertainment_bool , games_bool = False , False
    apps_command()

def entertainment_command():
    global entertainment_bool , games_bool
    entertainment_bool = True
    games_bool = False
    apps_command()

def games_command():
    global games_bool , entertainment_bool , search_entry_var
    games_bool = True
    entertainment_bool = False
    apps_command()

def submit():
    global result_label , category_for_storing
    try :
        result_label.config(text = '')
    except :
        result_label = Label(body_right , font = 'Robot 20 bold' , bg = '#262626')

    def valid():
        global appid_for_storing , category_for_storing , name_for_storing ,
developer_for_storing , size_for_storing , views_for_storing , description_for_storing
        global result_label
        if len(str(appid_for_storing.get())) <= 1:
            result_label.config(text = 'Appid too short' , fg = 'red') ;
result_label.grid(row = 1000 )
            return False
        elif name_for_storing.get().isspace() :
            result_label.config(text = 'Name too short' , fg = 'red') ;
result_label.grid(row = 1000 )
            return False
        elif developer_for_storing.get().isspace() :
            result_label.config(text = 'Developer name too short' , fg = 'red') ;
result_label.grid(row = 1000 )
            return False
        elif len(size_for_storing.get()) <= 0 :
            result_label.config(text = "Size of the game can't be this" , fg = 'red') ;
result_label.grid(row = 1000 )
            return False

    if valid() != False :
        global mycur
        result_label.config(text = "Added" , fg = '#28e84a') ; result_label.grid(row =
1000 )
        mydb = ms.connect(host='localhost', user='root',passwd=pas, database='project')
        mycur = mydb.cursor()
        if description_for_storing.get().isspace() or description_for_storing.get() ==
'':
            tup=(appid_for_storing.get() , category_for_storing.get() ,
name_for_storing.get() ,developer_for_storing.get() , size_for_storing.get() ,
views_for_storing)
            s='insert into apps(appid , category , name , developer , size , views)
values( %s , %s , %s , %s , %s , %s);'
        else:
            tup=(appid_for_storing.get() , category_for_storing.get() ,
name_for_storing.get() ,developer_for_storing.get() , size_for_storing.get() ,
views_for_storing , description_for_storing.get())
            s='insert into apps values( %s , %s , %s , %s , %s , %s , %s);'
        try :
            mycur.execute(s, tup)
            mycur.execute("commit")
        except :
            result_label.config(text = "Something went wrong" , fg = 'red')
            result_label.grid(row = 1000 )

def add_apps_command():
    destroy_everything()
    global appid_for_storing , category_for_storing , name_for_storing ,
developer_for_storing , size_for_storing , views_for_storing , description_for_storing
    global history_list , edit_apps_frame
    if history_list[-1] != 'addapps' :
        history_list.append('addapps')

    edit_apps_frame           = Frame(body_right, bg='#606060',
highlightbackground="black", highlightthickness=1)

    name_for_storing =           StringVar()
    category_for_storing =       StringVar()          # Adding combobox drop down list
    developer_for_storing =      StringVar()
    size_for_storing =           StringVar()
    description_for_storing =    StringVar()
    appid_for_storing =          IntVar()
    views_for_storing =          0

    edit_apps_frame_label= Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Bebas 25', text = 'Add Apps')
    idd =                  Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Roboto'  , text = 'Appid')
    categoryy =            Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Roboto'  , text = 'Category')
    namee =                Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Roboto'  , text = 'Name')
    developerr =           Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Roboto'  , text = 'Developer')
    sizee =                Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Roboto'  , text = 'Size')
    descriptionn =         Label(edit_apps_frame, bg = '#222222', fg = 'white',
height=2, font='Roboto'  , text = 'Description')

    iddd =          Entry(       edit_apps_frame, font = 'Roboto 20', textvariable =
appid_for_storing)
    categoryyy =    ttk.Combobox(edit_apps_frame, font = 'Roboto 20', textvariable =
category_for_storing)
    nameee =        Entry(       edit_apps_frame, font = 'Roboto 20', textvariable =
name_for_storing)
    developerrr =   Entry(       edit_apps_frame, font = 'Roboto 20', textvariable =
developer_for_storing)
    sizeee =        Entry(       edit_apps_frame, font = 'Roboto 20', textvariable =
size_for_storing)
    descriptionnn = Entry(       edit_apps_frame, font = 'Roboto 20', textvariable =
description_for_storing)

    sub =           Button(      edit_apps_frame,
text='Submit',command=submit,relief='groove', width=10, height=2)

    body_right.rowconfigure(   5, weight=1)
    edit_apps_frame.grid(      row=5, column=0, sticky='ENSW', padx = 35 , pady = 10)
    edit_apps_frame_label.grid(row=0, column=0, sticky='EW', padx = 10 , pady = (10,0)
, columnspan=2)
```

```
    idd.grid(           row = 1, column=0, sticky='WEN',  padx=10, pady=(10,0),
ipadx=5, ipady=5)
    categoryy.grid(     row = 2, column=0, sticky='WEN',  padx=10, pady=(10,0),
ipadx=5, ipady=5)
    namee.grid(         row = 3, column=0, sticky='WEN',  padx=10, pady=(10,0),
ipadx=5, ipady=5)
    developerr.grid(    row = 4, column=0, sticky='WNE',  padx=10, pady=(10,0),
ipadx=5, ipady=5)
    sizee.grid(         row = 5, column=0, sticky='WNE',  padx=10, pady=(10,0),
ipadx=5, ipady=5)
    descriptionn.grid(  row = 6, column=0, sticky='WNE',  padx=10, pady=(10,0),
ipadx=5, ipady=5)

    iddd.grid(          row = 1, column=1, sticky='WNES', padx=10, pady=10, ipadx=5,
ipady=5)
    categoryyy.grid(    row = 2, column=1 ,sticky='EWNS', padx=10, pady=10, ipadx=5,
ipady=5)
    nameee.grid(        row = 3, column=1, sticky='WNES', padx=10, pady=10, ipadx=5,
ipady=5)
    developerrr.grid(   row = 4, column=1, sticky='WNES', padx=10, pady=10, ipadx=5,
ipady=5)
    sizeee.grid(        row = 5, column=1, sticky='WNES', padx=10, pady=10, ipadx=5,
ipady=5)
    descriptionnn.grid( row = 6, column=1, sticky='WNES', padx=10, pady=10, ipadx=5,
ipady=5)

    sub.grid(           row=7, column=0, sticky='N', columnspan=2, padx=10,
pady=(10,10), ipadx=5, ipady=5)

    edit_apps_frame.columnconfigure(0, weight=1)
    edit_apps_frame.columnconfigure(1, weight=2)

    for i in range(7):
        edit_apps_frame.rowconfigure(i, weight=1)

    # Adding combobox drop down list
    categoryyy['values'] = (' Entertainment',  ' Games' , 'Business'
,'Lifestyle','Music & Audio','Photography','Social','Video Players & Editors')
    categoryyy.current()

def contact_us_command():              ##Contact Us
    global history_list , contact_us_frame
    if history_list[-1] != 'contact' :
        history_list.append('contact')

    destroy_everything()
    contact_us_frame= Frame(body_right, bg='#606060')
    contact_us_label= Label(contact_us_frame, text='Contact Us', bg='#262626',
font='Bebas 25', height=1 , fg='White')

    contact_us_frame.grid(row=10,column=0,sticky='NEWS', padx=40, pady=10) #Contact us
frame
    body_right.rowconfigure(10, weight=1) #contact us frame

    contact_us_frame.columnconfigure(0,weight=1) #expanding 2 columns in frame,
    contact_us_frame.columnconfigure(1,weight=1)

    person1= Frame(contact_us_frame, highlightbackground="black", highlightthickness=1,
height=100,width=100)   #for person 1 frame
    person1.grid(row=1,column=0, sticky='EW', padx=50, pady=20)
    person1.columnconfigure(0, weight=1)

    person1_show_photo=Label(person1,image=person1_photo)   #person 1 photo
    person1_show_photo.grid(row=0, padx=10, pady=10)
    person1_name=Label(person1, text='Avitesh   Murmu', bg='#385723',
fg='white',height=1, font='Bebas 25' )
    person1_name.grid(row=1,sticky='EW')
    person1_description=Label(person1, text='''
    Class   : XII
    Section : A
    Roll No : 36
    ''', justify=LEFT, bg='#92D050', fg='black', height=5, font='Roboto')
    person1_description.grid(row=2, sticky='EW')

    person2= Frame(contact_us_frame, highlightbackground="black", highlightthickness=1,
height=100,width=100)   #for person 2 frame
    person2.grid(row=1,column=1, sticky='EW', padx=50)
    person2.columnconfigure(0, weight=1)

    person2_show_photo=Label(person2,image=person2_photo)   #person 2 photo
    person2_show_photo.grid(row=0, padx=10, pady=10)
    person2_name=Label(person2, text='Roshan   Raj', bg='#385723', fg='white',height=1,
font='Bebas 25')
    person2_name.grid(row=1,sticky='EW')
    person2_description=Label(person2, text='''
    Class   : XII
    Section : A
    Roll No : 13
    ''', justify=LEFT, bg='#92D050', fg='black', height=5, font='Roboto')
    person2_description.grid(row=2, sticky='EW')

    contact_us_button.grid( row=100,        column=0,       sticky='NEW',
ipadx=10, ipady=10, pady=5)     #Contact Us button in the left
    contact_us_label.grid(row=0, column=0,padx=10,ipadx=10, ipady=15, pady=10,
sticky='WE', columnspan=2)        #Contact us label/heading in the top
#######################################################################################
def snake_run():
    root.wm_state('iconic')
    try :
        obj = Snake()
        obj.WelcomeScreen()
    except :
        pass
    root.wm_state('zoomed')

def calculator_run():
    root.wm_state('iconic')
    Calculator()

def game_run():
    root.wm_state('iconic')
    Game()

def app_delete_command(appkaappid = None , appname = None):
    global history_list , delete_update_frame , delete_frame , appkaappid_no , appkanam
    if history_list[-1] != 'delete' :
        history_list.append('delete')
        destroy_everything()

    def remove_space(s):
        r = ''
        for i in range(len(s)):
            if s[i] != ' ':
                r += s[i]
        return r

    def delete_app():
        if appkaappid != None and appname != None:
            mydb = ms.connect(host = 'localhost' , user = 'root' , passwd = pas ,
database = 'project')
            mycur = mydb.cursor()
            mycur.execute("Select * from apps ;")
            res = mycur.fetchall()
            if appkaappid == '' and appname == '':
                anymessage_label.config(text = 'Appid and Appname are empty')
                anymessage_label  .grid(row=4, column=0, padx=10, pady=5)
            elif appkaappid == '':
                anymessage_label.config(text = 'Appid is empty')
                anymessage_label  .grid(row=4, column=0, padx=10, pady=5)
            elif appname == '':
                anymessage_label.config(text = 'Appname is empty')
                anymessage_label  .grid(row=4, column=0, padx=10, pady=5)

            else :
                count = False
                for row in res :
                    if row[0] == int(appkaappid) and remove_space(row[2].lower()) ==
remove_space(appname.lower()) :
                        mycur.execute(f'DELETE FROM apps WHERE appid=
{int(appkaappid)}')
                        mycur.execute("commit")
                        count = True
                        anymessage_label.config(text= "Deleting App Done", fg= 'Green')
                        anymessage_label  .grid(row=4, column=0, padx=10, pady=5)
                        break
                    elif row[0] == int(appkaappid) and remove_space(row[2].lower()) !=
remove_space(appname.lower()) :
                        anymessage_label.config(text = 'Something wrong in App Name',
fg = 'red')
                        anymessage_label  .grid(row=4, column=0, padx=10, pady=5)
                        count = True
                        break
                    elif row[0] != int(appkaappid) and remove_space(row[2].lower()) ==
remove_space(appname.lower()) :
                        anymessage_label.config(text = 'Something wrong in App id', fg
= 'red')
                        anymessage_label  .grid(row=4, column=0, padx=10, pady=5)
                        count = True
                        break

                if count == False :
                    anymessage_label.config(text = 'Nothing matches.... You should try
again' , fg = 'red')
                    anymessage_label  .grid(row=4, column=0, padx=10, pady=5)

    delete_update_frame = Frame( body_right, bg='#606060')
    delete_frame        = Frame( delete_update_frame, bg='#606060')
    delete_label        = Label( delete_frame, text='Delete App', bg='#262626',
font='Bebas 25', height=2 , fg='White')
    anymessage_label    = Label( delete_frame, text = "Try deleting any app"
,bg='#262626', fg= 'Green', font='Roboto 18')
    enter_app_id_entry  = Entry( delete_frame, textvariable = appkaappid_no , font =
(17))
    enter_name_entry    = Entry( delete_frame, textvariable = appkanam , font = (17))
    submit2             = Button(delete_frame, text='Submit', height=2 , width=20 ,
command = lambda : app_delete_command(appkaappid = appkaappid_no.get() , appname=
appkanam.get()))

    try :
        delete_app()
    except :
        pass
    delete_label.grid(row=0, sticky='EW', padx=10, pady=10, columnspan=2)

    enter_app_id_label = Label(delete_frame, text='Enter App Id', bg = '#222222', fg =
'white', height=2, font='Roboto')
    enter_app_id_label.grid(row=1, column=0, sticky='EW', padx=10, pady = (0, 5))

    enter_app_name_label = Label(delete_frame, text='Enter App Name', bg = '#222222',
fg = 'white', height=2, font='Roboto')
    enter_app_name_label.grid(row=2, column=0, sticky='EW', padx=10,pady = (5, 0))

    enter_app_id_entry.grid(row=1, column=1, sticky="EWNS", padx=10 , pady = (0, 5))
    enter_name_entry  .grid(row=2, column=1, sticky="EWNS", padx=10 , pady = (5, 0))
    submit2           .grid(row=3, column=0, columnspan=2 , pady=10)
    anymessage_label  .grid(row=4, column=0, pady=5       , padx=10, columnspan=2)

###########################################################################################
    '''
    # Update Apps
    update_frame = Frame(delete_update_frame, bg='#606060')
    update_frame.grid(row=1, sticky='NEWS')      #update Frame
    update_frame.columnconfigure(0, weight=1)

    update_label = Label(update_frame, text='Update App', bg='#262626', font='Bebas
25', height=2 , fg='White')
    update_label.grid(row=1, sticky='EW', padx=10, pady=20, columnspan=2)
    '''
    # Delete and Update Frame
    delete_update_frame.grid(row= 0, sticky='NEWS', padx=35, pady=10)
    body_right.rowconfigure(0, weight=1)
    delete_update_frame.columnconfigure(0, weight=1)

    delete_frame.grid(row=0, sticky='NEWS')      #Delete Frame
    delete_frame.columnconfigure(0, weight=1)
    delete_frame.columnconfigure(1, weight=3)

def bonus_zone_command():
    global history_list , bonus_zone , bonus_snake_logo , bonus_tic_tac_toe_logo ,
bonus_calculator_logo
    if history_list[-1] != 'bonus':
        history_list.append('bonus')

    destroy_everything()
    t=1
    bonus_zone=Frame(body_right, bg='#b8a753')
# bonus Frame in Right
    bonus_zone.grid(row=45, sticky='NEWS', padx=40, pady=10)

    body_right.rowconfigure(45, weight=1)

    bonus_heading = Label(bonus_zone, text='\u272F   '+'Bonus Zone'+'   \u272F',
bg='#262626', font='Bebas 25', height=2 , fg='#c4a502')          # Bonus Heading
    bonus_heading.grid(row=0, column=0,padx=10,ipadx=10, ipady=15, pady=10,
sticky='WE', columnspan=3)



    # Snake Game
    bonus_frame_snake       = Frame(bonus_zone, bg='#8db705')
# Snake Frame
    bonus_snake_logo        = PhotoImage(file= r'Photos/snake_game.png')
    bonus_frame_snake_photo = Button(bonus_frame_snake, image=bonus_snake_logo,
relief=GROOVE , command = snake_run)
    bonus_snake_label       = Button(bonus_frame_snake, text='Play Snake',
bg='#8db705', relief=GROOVE, activeforeground='#8db705', fg='White', font='Roboto' ,
command = snake_run)
```

```
    bonus_frame_snake.grid(       row=t, column=0, padx=(25,0), pady=10, sticky='EW',
ipady=(10))
    bonus_frame_snake_photo.grid( row=0, column=0, pady=(20,10))
    bonus_snake_label.grid(       row=1, sticky='news', padx=10)
    bonus_frame_snake.columnconfigure(0, weight=1)


    # Tic Tac Toe
    bonus_frame_tic_tac_toe = Frame(bonus_zone, height=300, bg='#eb5855')
# Snake Tic Tac Toe
    bonus_tic_tac_toe_logo  = PhotoImage(file = r'Photos/tictactoe.png')
    bonus_tic_tac_toe_photo = Button(bonus_frame_tic_tac_toe,
image=bonus_tic_tac_toe_logo, relief=GROOVE , command = game_run)
    bonus_tic_tac_toe_label       = Button(bonus_frame_tic_tac_toe, text='Play Tic Tac
Toe', bg='#eb5855', relief=GROOVE, activeforeground='#eb5855', fg='White',
font='Roboto' , command = game_run)

    bonus_frame_tic_tac_toe.grid(    row=t, column=1, padx=(25,0), pady=10,
sticky='EW', ipady=10)
    bonus_tic_tac_toe_photo.grid(    row=0, column=0, pady=(20,10))
    bonus_tic_tac_toe_label.grid(    row=1, sticky='news', padx=10)
    bonus_frame_tic_tac_toe.columnconfigure(0, weight=1)


    # Calculator
    bonus_frame_calculator = Frame(bonus_zone, height=300, bg='#b86d33')
# Calculator Frame
    bonus_calculator_logo  = PhotoImage(file = r'Photos/calculator.png')
    bonus_calculator_photo = Button(bonus_frame_calculator,
image=bonus_calculator_logo, relief=GROOVE , command = calculator_run)
    bonus_calculator_label = Button(bonus_frame_calculator, text='Use Calculator',
bg='#b86d33', relief=GROOVE, activeforeground='#b86d33', fg='White', font='Roboto' ,
command = calculator_run)

    bonus_frame_calculator.grid(    row=t, column=2, padx=(25,25), pady=10,
sticky='EW', ipady=10)
    bonus_calculator_photo.grid(    row=0, column=0, pady=(20,10))
    bonus_calculator_label.grid(    row=1, sticky='news', padx=10)
    bonus_frame_calculator.columnconfigure(0, weight=1)

    for i in range(3):
        bonus_zone.columnconfigure(i, weight=1)

def check_connection_and_open_main_py():
    if name_variable.get() == '' and password_variable.get() == '' and
class_variable.get() == '' and rollno_variable.get() == '' :
        anymessage.config(text = 'Please enter the above entries...\nAll entries empty'
, bg = 'red' , fg = 'white')

    elif name_variable.get() != '' and password_variable.get() != '':
        global pas
        try :
            ms.connect(host = 'localhost' , user = 'root' , passwd =
password_variable.get() , database = 'project')
            pas = password_variable.get()
            delete_frame2()
            hello_user_command()
            retrieve_frame1()

        except :
            anymessage.config(text = '!! Wrong !!\nPlease make sure that you entered
your password CORRECTLY' , bg = 'red' , fg = 'white')


    if name_variable.get() == '' and password_variable.get() == '':
        label_1.config(text = 'Your  MySQL  Password   *Required' , fg = 'red')
        label_2.config(text = 'Your  Name   *Required' , fg = 'red')

    elif name_variable.get() != '' and password_variable.get() == '' :
        label_1.config(text = 'Your  MySQL  Password   *Required' , fg = 'red')
        label_2.config(text = 'Your  Name' , fg = 'white')
        anymessage.config(text = '', bg='#262626')

    elif name_variable.get() == '' and password_variable.get() != '':
        label_1.config(text = 'Your  MySQL  Password' , fg = 'white')
        label_2.config(text = 'Your  Name   *Required' , fg = 'red')
        anymessage.config(text = '', bg='#262626')

def start_to_main():
    frame2.grid_forget()
    root.rowconfigure(           1 , weight=0)
    frame.grid(                     sticky=(N, E, W, S), row=0)
    root.rowconfigure(           0 , weight=1)

def delete_frame2():
    frame2.grid_forget()
    root.rowconfigure(           1 , weight=0)

def retrieve_frame1():
    frame.grid(                     sticky=(N, E, W, S), row=0)
    root.rowconfigure(           0 , weight=1);check_connection_with_mysql(check= True)

def delete_signup_and_signin():
    sign_up.grid_forget()
    sign_in.grid_forget()

def hello_user_command():
    if name_variable.get() == '':
        hello_user = 'Welcome, Guest'
    else:
        hello_user = 'Welcome \n'+ name_variable.get()
    hello_frame = Frame(time_greetings_signup_signin, bg='#385723')
    hello_frame.grid(row= 0, column=1, sticky='NEWS', ipadx=5, pady=10, padx=(0,10),
ipady=3)
    hello_frame.rowconfigure(0, weight=1)
    user_display = Label(hello_frame, image=user_photo, bg = '#385723')
    user_display.grid(row=0, padx=10)
    hello_show = Label(hello_frame , text = hello_user , height= 3, bg='#385723',
fg='white', font='Roboto 9')
    hello_show.grid(row= 0, column=1, sticky='NEWS')

#####################################################################################
#Global variables
username = StringVar()
password = StringVar()
sup = False ; sin = True
history_list = ['main']
logined = False
entertainment_bool = False
games_bool = False
appkaappid_no = StringVar()
appkanam = StringVar()
user_photo = PhotoImage(file= r'Photos/user.png')

#####################################################################################
#Header
header = Frame(frame, height =5, bg='#00b050') #'#00A86B'
time_greetings_signup_signin = Frame(header,  bg='#00b050')
time_greetings_signup_signin.grid(column=2, sticky='E')
```

```
#####################################
#for_sign_up_and_sign_in
sign_up =Button(time_greetings_signup_signin, text = 'Sign Up',height= 3,width=7,
bg='#385723', relief=GROOVE , fg='white', command=signup_window, font='Roboto 9')
sign_in =Button(time_greetings_signup_signin, text = 'Sign In',height= 3,width=7,
bg='#385723', relief=GROOVE , fg='white', command=signin_window, font='Roboto 9')
labl = tk.Label(time_greetings_signup_signin ,text = '', bg = '#aaff00' , fg = 'black'
, font = 'Roboto 17')

#####################################
#Add Back ,Home, Logo
back_and_home   = Frame(header , bg='#00b051')
back            = Button(back_and_home, image = back_lg1, bg='#339933',
relief=GROOVE, command = back_command)  #"\u00AB BACK"
appstore        = Label(frame, image = appstore_lg2, bg='white')
home            = Button(back_and_home, image = home_photo, bg = '#339933',
relief=GROOVE, command = home_command, height=51, width=51)
light_green     =Frame(header, height=5, bg='#aaff00')
light_green.grid(row=1, columnspan=5,sticky='WE')
######################################
#Body
body            = Frame(frame)
##############################
#Left Body
body_left       = Frame(body,       bg='#404040')

#Categories
category         = Frame(body_left, bg = '#404040')
apps             = Button(category, bg = '#606060' , fg = 'white' , text='All Apps',
activebackground='#4F7942', activeforeground='white' , font='Roboto', command =
all_apps_command)
entertainment    = Button(category, bg = '#606060' , fg = 'white' ,
text='Entertainment',        activebackground='#4F7942', activeforeground='white' ,
font='Roboto', command = entertainment_command)
games            = Button(category, bg = '#606060' , fg = 'white' , text='Games',
activebackground='#4F7942', activeforeground='white' , font='Roboto', command =
games_command)
edit_apps        = Button(category, bg = '#606060' , fg = 'white' , text='Add Apps',
activebackground='#4F7942', activeforeground='white' , font='Roboto', command =
add_apps_command)
contact_us_button= Button(category, bg = '#c4c4c4' , fg = 'black' , text='Contact Us',
activebackground='#4F7942', activeforeground='white' , font='Roboto', command =
contact_us_command)
delete_button    = Button(category, bg = '#606060' , fg = 'white' , text='Delete/Update
\n Apps',activebackground='#4F7942', activeforeground='white' , font='Roboto', command
= app_delete_command)
bonus_zone_button= Button(category, bg = '#FFD700' , fg = 'black' ,
text='\u272F'+'Bonus Zone'+'\u272F',           activebackground='#cfb00c',
activeforeground='white' , font='Roboto' , command = bonus_zone_command)

##############################
#Right body
body_right      = Frame(body, bg='#262626')
result_label    = Label(body_right , font = 'Robot 20 bold' , bg = '#262626')
gridding()
####################################################################       delete and
update Apps
#Footer
footer          = Frame(frame, height=40, bg='#00b050')
credits         = Button(footer,text='Copyright © 2020 AVIONICS &  RRajj Inc. All
rights reserved.' , bg = '#92D050', fg = '#222222', width=80, )
#########################################################################################
#First window starts here
frame2 = Frame(root, bg='#262626')
frame2.columnconfigure(0, weight=1)
```

```
frame2.rowconfigure(101, weight=1)
q=0
#       Heading
heading_label = Label(frame2, text='Please  Enter', bg='#00b050', font='Bebas 30',
fg='White')
heading_label.grid(row=q, column=0, ipadx=30, ipady=15, sticky='EW', columnspan=2)
underline_design = Frame(frame2, height=3, bg='White')
underline_design.grid(row=q+1, sticky='EW', columnspan=2)
#       Logo
front_photo=Label(frame2,image=front_photo_variable)
front_photo.grid(row=2, column=1, rowspan=102, padx=(0,0), pady=0)
#       MySQL Password
label_1    = Label(frame2, text='Your  MySQL  Password', bg='#262626', font='Bebas 17',
fg='White')
label_1   .grid(row=q+2, column=0, padx=30, pady=(10,0), sticky='W')
password_variable=StringVar()
passw = Entry(frame2, font = 'Roboto 15', textvariable = password_variable , show =
'*')
passw.grid(row=q+3, column=0, padx=30, pady=(0,0), sticky='NEWS')
#       Name
label_2    = Label(frame2, text='Your  Name',height= 1, bg='#262626', font='Bebas 17',
fg='White')
label_2   .grid(row=q+11, column=0, padx=30, pady=(30,0), sticky='W')
name_variable=StringVar()
name_entry = Entry(frame2, font = 'Roboto 15', textvariable = name_variable)
name_entry.grid(row=q+12, column=0, padx=30, pady=(0,0), sticky='NEWS')
#       Class
label_3    = Label(frame2, text='Your  Class',height= 1, bg='#262626', font='Bebas 17',
fg='White')
label_3.grid(row=q+21, column=0, padx=30, pady=(30,0), sticky='W')
class_variable=StringVar()
class_chosen = ttk.Combobox(frame2, font = 'Roboto 15', textvariable = class_variable)
#       Adding combobox drop down list
class_chosen['values'] = ('X', 'XI', 'XII', 'None')
class_chosen.current()
class_chosen.grid(row=q+23, sticky='WE', padx=30, pady=(0,30), column=0)
#       RollNO
label_4 = Label(frame2, text='Your  Roll No',height= 1, bg='#262626', font='Bebas 17',
fg='White')
label_4.grid(row=q+31, column=0, padx=30, pady=(0,0), sticky='W')
rollno_variable =StringVar()
rollno_chosen = ttk.Combobox(frame2, font = 'Roboto 15', textvariable =
rollno_variable)
  # Adding combobox drop down list
value = []
for i in range(1,41):
    value.append(i)
value.append('None')
rollno_chosen['values'] = value
rollno_chosen.current()
rollno_chosen.grid(row=q+33, sticky='NEWS', padx=30, pady=(0,30))
#       Section
subm = Button( frame2, text='Submit', command = lambda :
check_connection_and_open_main_py(), height=2, width=20, relief= GROOVE)
subm.grid(row=100, pady=(0,30))
anymessage=Label(frame2, font='Roboto 17', text = '', bg='#262626' , fg = 'white')
anymessage.grid(row=102, padx=30, pady=(0,15), sticky='EWNS')

##########################################################################################
#Griding
root.grid()
frame2.grid(                      sticky=(N, E, W, S), row=1)
#Header
header.grid(            row=0,                          sticky=(N, E, W))   #Griding
Header And its components
```

```
back.grid(              row=0,          column=0,       sticky=(W), padx=10, pady=5)
back_and_home.grid(     row=0,          column=0,       sticky='W')
appstore.grid(          row=0,          column=0,                   padx=10, pady=5)
home.grid(              row=0,          column=1,       sticky='W')
sign_up.grid(           row=0,          column=3,       sticky=(E), padx=10, pady=5)
#Done
sign_in.grid(           row=0,          column=5,       sticky=(E), padx=10, pady=5)
#Done
labl.grid(              row=0,          column=0,       sticky=(E), padx=(0 , 15),
ipadx=15, ipady=9, pady=5)
clock()
#########################################################################################
#Griding Main Body, Row is equal to 1 because it should be in the frames's second row
body.grid(              row=1,          sticky=('NEWS'))
body_left.grid(         row=0,          column=0,       sticky='NEWS')    #Left Body
category.grid(          row=0,          column=0,       sticky='WNE',        padx=10,
pady=10)   #Category
apps.grid(              row=0,          column=0,       sticky='WE',        ipadx=10,
ipady=10, pady=5)
entertainment.grid(     row=1,          column=0,       sticky='WE',        ipadx=10,
ipady=10, pady=5)
games.grid(             row=2,          column=0,       sticky='WE',        ipadx=10,
ipady=10, pady=5)
edit_apps.grid(         row=3,          column=0,       sticky='EW',        ipadx=10,
ipady=10, pady=5)
contact_us_button.grid( row=100,        column=0,       sticky='NEW',       ipadx=10,
ipady=10, pady=5)
delete_button.grid(     row=14,         column=0,       sticky='NEWS')
bonus_zone_button.grid( row=4,          column=0,       sticky='EW',        ipadx=10,
ipady=10, pady=5)
#Body Right
body_right.grid(        row=0,          column=1,       sticky='NEWS')   #Right Body
#footer
footer.grid(            row=2, sticky='WE')
credits.grid(           padx=10, pady=5)
#Configure
#Root Main Window
root.columnconfigure(        0 , weight=1)
root.rowconfigure(           0 , weight=0)
root.rowconfigure(           1 , weight=1)
#       Header
frame.columnconfigure(       0 , weight=1)                 #Header - Column ,and must
not be repeated
frame.rowconfigure(          0 , weight=0)                 #Header - Row
#       Back
header.columnconfigure(      0 , weight=1)                  #Back
header.columnconfigure(      1 , weight=1)                  #AppStore (Heading)
header.columnconfigure(      2 , weight=1)                  #SignUp
#       Body
#       Body Left
frame.columnconfigure(       1 , weight=0)                 #Body   - Column , for
example here it should not be 1
frame.rowconfigure(          1 , weight=1)                 #Body   - Row
body.rowconfigure(           0 , weight=1)                 #Body_Left              -
Row
body.columnconfigure(        0 , weight=1)                 #Body_Left              -
Column
body_left.columnconfigure(   0 , weight=1)                 #Category
body_left.rowconfigure(      0 , weight=1)                 #Category
category.columnconfigure(    0 , weight=1)                 #Entertainment
#    Body Right
body.columnconfigure(        1 , weight=100)               #Body_Right     - 1 for
right Column
body_right.columnconfigure(  0 , weight=1)                 #Expanding right body's
(login frame) horizontally
```

```
body_right.rowconfigure(     0 , weight=1)                  #Expanding right body's
(login frame) vertically
body_right.rowconfigure(     5 , weight=1)
#    Footer
footer.columnconfigure(      0 , weight=1)                  #Credits
#########################################################################################
#   Output
root.protocol("WM_DELETE_WINDOW" , close)
root.mainloop()
```