CASE STUDY

This project (Music Synthesizer) aims at creating a realistic virtual platform to create or generate music to provide an alternative to real life pianos and keyboards.

In today’s world where almost everyone listens to music and/or plays an instrument, I thought this would be a fun way to provide easy access to music anywhere and everywhere. The project was inspired by several of my friends who love playing instruments and my brother who plays the keyboard.

Users are required to create accounts along with passwords which they can use to login at further instances. Once they have created their accounts, they are able to play the piano, using the keys on the computer keyboard.

There are several options available to users, which include –

* Recording any music that they play
* Saving above mentioned recordings in the database
* Replaying the saved recordings even in future logins

The application opens with an animated display message and background music. This then moves on to the login page. If the user has not yet created an account, he may click the sign-up button to create a new account and provide a username and password. Keyboard shortcuts such as tab key (or shift tab), delete, arrow keys etc. have been enabled for easy access.

In the background of the home page, there is a timer to indicate the amount of time the user has spent in that page.

Once the user has logged into his account, one octave of the piano is visible along with instructions regarding how to play the keys. Users are able to shift octaves using the up and down arrow keys in order to go to a higher or lower pitch. When the key is pressed, the note is played for as long as the key is pressed. However, there is a cutoff of 10 seconds as it is unlikely that such a long press is intentional. When a note is pressed, the key changes colour so as to indicate which note has been pressed. When the note is released, the note returns to its original colour.

The options available to the user are displayed to the side of the piano. These include – Record, Save, Play, Songs, Logout etc.

The application has been designed in such a way as to maintain proper ratios for both the initial and full screen modes.

Whenever the user chooses to record a song, all the notes that are pressed are stored in a dictionary along with the time at which it is pressed. When the user presses the play button, each note is accessed using threading as multiple notes would not be possible otherwise.

When the user logs out of their account, they are taken back to the home page. Login page of the application.

TABLE STRUCTURE

Table Name: users

+----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+-------+

| username | varchar(30) | NO | PRI | NULL | |

| password | varchar(22) | NO | | NULL | |

+----------+-------------+------+-----+---------+-------+

Table name: currentuser

+-------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------------+-------------+------+-----+---------+-------+

| currentuser | varchar(30) | YES | | NULL | |

| currentsong | text | YES | | NULL | |

+-------------+-------------+------+-----+---------+-------+

Table name: <user>songs (similar table for every user)

+-----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-----------+-------------+------+-----+---------+-------+

| songname | varchar(30) | NO | PRI | NULL | |

| recording | text | YES | | NULL | |

+-----------+-------------+------+-----+---------+-------+

ALGORITHM

MAJOR FUNCTIONS

1. keys()
2. Get height and width of window.
3. Divide the window into 10 parts with equal spacing in order to place white keys
4. Find the average width of each key
5. Create list containing pygame rectangle objects for both white keys and black keys
6. Draw the rectangle objects on the screen using a loop.
7. Include conditions to check if the note is being pressed or not.
8. Do v and vi for both white and black keys
9. play\_note()
10. Get Amplitude, frequency, duration and sample rate of the sine wave
11. Define the sine wave using above values
12. Create numpy array using sine wave
13. Convert the array to a pygame sound object
14. Return the sound object.
15. press\_action()
16. Pass required arguments such as the key and the color needed to change to.
17. Play the sound corresponding to the given note from the note\_sounds dictionary
18. If it is being recorded then add the keyname along with the timestamp to the time\_record dictionary as a key value pair
19. lift\_action()
20. Does the opposite of press action.
21. The key color is changed back to original
22. The sound is stopped
23. The ending time/duration is noted
24. multiwork()
25. Using the dictionary containing the notes and their timestamps the recorded song is to be played
26. This is done with the help of threading to enable multiple notes being played simultaneously based on their timestamps and durations
27. b()
28. Used to create buttons.
29. Pass the x and y coordinates to the functions along with the text to be displayed and the state(pressed or lifted).
30. The button is drawn on the screen and if it pressed then action will be taken. This is based on position of mouse pointer.

Flowchart of Function: play\_note

START

PASS ARGUMENTS NOTE(DEFAULT A4), DURATION D(DEFAULT 10), AMPLITUDE A(DEFAULT 6000)

Set the following

A = A

f = nf[note]

t1 = array of equally spaced values from 0 to d

ω = angular frequency of wave

y = sin wave representing the sound

Sound = create an array from the sin wave (y)

Sound = redefine sound as pygame sound object

RETURN SOUND (PYGAME SOUND OBJECT)

END

PROGRAM CODE

MODULE NAME: Homepagescl.py

import pygame

import os

import time

import threading

import mysql.connector as M

from Constants import \*

#SOUNDS

#https://soundcloud.com/ashamaluevmusic/sets/instrumental-background-music-for-videos

#https://www.ashamaluevmusic.com/summer-music

pygame.init()

#USEFUL CONSTANTS

D = pygame.display

S = D.set\_mode((800,600),pygame.RESIZABLE)

Sh\_org = S.get\_height()

Sw\_org = S.get\_width()

E = pygame.event

Dr = pygame.draw

def txt(t,font="Corbel",size = 20,color = "#ff0000", italic = False, u = False):

        # defining a font

    try:

        smallfont = pygame.font.Font(font, size)

    except:

        smallfont = pygame.font.SysFont(font, size)

    # rendering a text written in

    # this font

    smallfont.italic = italic

    smallfont.underline = u

    text = smallfont.render( t, True , color)

    return text

def eventcheck(tabslist):

     if event.type == pygame.MOUSEBUTTONDOWN:

            mouse\_pos = pygame.mouse.get\_pos()

            if input\_name.rect.collidepoint(mouse\_pos):

                input\_name.state = "active"

                now\_active = input\_name

                now\_activeindex = tabslist.index(input\_name)

            else:

                input\_name.state = "passive"

            if input\_pwd.rect.collidepoint(mouse\_pos):

                input\_pwd.state = "active"

                now\_active = input\_pwd

                now\_activeindex = tabslist.index(input\_pwd)

            else:

                input\_pwd.state = "passive"

            if input\_pwd.pwd\_rect.collidepoint(mouse\_pos):

                input\_pwd.state = "active"

                if input\_pwd.pwd\_img\_state == "hidden":

                    input\_pwd.pwd\_img\_state = "visible"

                else:

                    input\_pwd.pwd\_img\_state = "hidden"

     if event.type == pygame.MOUSEMOTION:

        mouse\_pos = pygame.mouse.get\_pos()

        if input\_name.rect.collidepoint(mouse\_pos):

            if input\_name.state == "passive":

                input\_name.state = "hover"

        else:

            if input\_name.state == "hover":

                input\_name.state = "passive"

        if input\_pwd.rect.collidepoint(mouse\_pos):

            if input\_pwd.state == "passive":

                input\_pwd.state = "hover"

        else:

            if input\_pwd.state == "hover":

                input\_pwd.state = "passive"

     if event.type == pygame.KEYDOWN:

        gettxt(input\_name)

        gettxt(input\_pwd)

        cursormove(input\_name)

        cursormove(input\_pwd)

        tabaction()

def uservalidation():

    un = input\_name.user\_text

    pwd = input\_pwd.user\_text

    Exe("Select \* from users where username = %s",(un,))

    try:

        L\_details = MyC.fetchall()[0]

    except:

        print("Unavailable now")

        return

    if L\_details[1] == pwd:

        print(1)

        Exe("Delete from currentuser")

        print("Done")

        Exe(f"Insert into currentuser values('{un}',null)")

        MyDB.commit()

        submit1.state = "active"

    else:

        from tkinter import Tk

        from tkinter.messagebox import showerror

        Tk().wm\_withdraw()

        showerror("Error","Invalid username or password")

def usercreation():

    un = input\_name.user\_text

    pwd = input\_pwd.user\_text

    try:

        Exe("Insert into users values(%s,%s)",(un,pwd))

        MyDB.commit()

    except:

        print("Username already exists")

        S.blit(txt("Username already exists", color = "red"),(40,600))

        return

    Exe("Delete from currentuser")

    Exe(f"Insert into currentuser values('{un}',null)")

    MyDB.commit()

    Exe(f"Create table {un}songs(songname varchar(30) primary key, recording text)")

    homepg.state = 0

def gettxt(self):

    if self.state == "active":

        # Check for backspace

        '''if event.key == pygame.K\_BACKSPACE:

            # get text input from 0 to -1 i.e. end.

            ut = self.user\_text

            self.user\_text = ut[:self.cursor\_pos-1] + ut[self.cursor\_pos:]

            self.cursor\_pos -= 1'''

        # Unicode standard is used for string

        # formation

        if len(self.user\_text)<self.maxlen[self.type] :

            if event.key not in (pygame.K\_LEFT ,pygame.K\_RIGHT, pygame.K\_BACKSPACE,pygame.K\_DELETE,pygame.K\_TAB):

                ut = self.user\_text

                if event.unicode.isalnum() or event.unicode in ("\_"):

                    self.user\_text = ut[:self.cursor\_pos] + event.unicode + ut[self.cursor\_pos:]

                    self.cursor\_pos += 1

def backspace(self,t):

    if self.state == "active":

        if t - self.backspace\_time > 0.12:

            if self.cursor\_pos >0:

                # get text input from 0 to -1 i.e. end.

                ut = self.user\_text

                self.user\_text = ut[:self.cursor\_pos-1] + ut[self.cursor\_pos:]

                self.cursor\_pos -= 1

                self.backspace\_time = t

def delete(self,t):

    if self.state == "active":

        if t - self.delete\_time > 0.12:

            # get text input from 0 to -1 i.e. end.

            ut = self.user\_text

            self.user\_text = ut[:self.cursor\_pos] + ut[self.cursor\_pos+1:]

            self.delete\_time = t

def cursormove(self):

    if self.state == "active":

        if event.key == pygame.K\_LEFT:

            if self.cursor\_pos >0:

                self.cursor\_pos -= 1

                print(self.cursor\_pos)

        if event.key == pygame.K\_RIGHT:

            if self.cursor\_pos < len(self.user\_text):

                self.cursor\_pos += 1

                print(input\_name.cursor\_pos)

def tabaction():

    global now\_active,now\_activeindex

    if KEYSTATES[pygame.K\_LSHIFT] or KEYSTATES[pygame.K\_RSHIFT]:

        print("yes")

        if event.key == pygame.K\_TAB:

            if now\_activeindex in (0,None):

                now\_activeindex = len(homepg.tabslist)

            try:

                now\_active.state = "passive"

            except:

                pass

            now\_activeindex -= 1

            now\_active = homepg.tabslist[now\_activeindex]

            if type(now\_active) != type(submit1):

                now\_active.state = "active"

            else:

                now\_active.state = "hover"

    else:

        if event.key == pygame.K\_TAB:

            if now\_activeindex in (len(homepg.tabslist)-1,None):

                now\_activeindex = -1

            try:

                now\_active.state = "passive"

            except:

                pass

            now\_activeindex += 1

            now\_active = homepg.tabslist[now\_activeindex]

            if type(now\_active) != type(submit1):

                now\_active.state = "active"

            else:

                now\_active.state = "hover"

class BGM:

    def \_\_init\_\_(self,track):

        from pygame import mixer

        self.mixer = mixer

        # Starting the mixer

        self.mixer.init()

        # Loading the song

        self.mixer.music.load(track)

        # Setting the volume

        self.mixer.music.set\_volume(0.5)

    def play(self):

        # Start playing the song

        self.mixer.music.play()

    def end(self):

        self.mixer.music.stop()

        pass

#Seawalk\_bgm = BGM("E:\\120204 Ashish Thannickal\\Project Work\\Homepage\\Seawalk.mp3")

Seawalk\_bgm = BGM("D:\\Music Synthesizer\\Homepage\\Seawalk.mp3")

class TextHP:

    def \_\_init\_\_(self,i = 0,hpstate=1):

        global sizeiteration

        self.hpstate = hpstate

        self.th1 = int(60 \* (Sw/Sw\_org))

        if time.time()%0.05 > 0.03:

            sizeiteration += 1

        self.th2 = max(self.th1 - i,40)

        if self.hpstate == 2:

            self.th2 = 35

        self.text = txt("MUSIC SYNTHESIZER",

                        #font = "E:\\120204 Ashish Thannickal\\Project Work\\Homepage\\Another Danger - Demo.otf",

                        font = "D:/Music Synthesizer/Homepage/Another Danger - Demo.otf",

                        size = self.th2,

                        color = "#eeeeee")

        self.th = self.text.get\_height()

        self.tw = self.text.get\_width()

        self.rect = self.text.get\_rect()

        self.TL = ()

        self.TR = ()

        self.BL = ()

        self.BR = ()

    def draw(self):

        self.y = Sh/2-self.th/2

        if self.hpstate == 2:

            self.y = 40

        self.bl = S.blit(self.text ,(Sw/2-self.tw/2,self.y))

        self.TL = self.bl.topleft

        self.TR = self.bl.topright

        self.BL = self.bl.bottomleft

        self.BR = self.bl.bottomright

class underline:

    def \_\_init\_\_(self,len=20,x=20,y=20):

        self.length = len

        self.x = x

        self.y = y

    def draw(self):

        Dr.line(S,"#dddddd",(self.x,self.y),(self.x + self.length,self.y),8)

        pass

class Timer:

    def \_\_init\_\_(self,t):

        self.d = {}

        self.d["ss"] = t-starttime

        self.d["mm"] = int(self.d["ss"]//60)

        self.d["ss"] = round(self.d["ss"]%60,2)

        self.d["hh"] = int(self.d["mm"]//60)

        self.d["mm"] = self.d["mm"]%60

        self.txt = "{} : {} : {}".format(self.d["hh"],self.d["mm"],self.d["ss"])

        self.text = txt(self.txt,size = 40,color = "pink")

    def draw(self):

        self.bl = S.blit(self.text ,(0,0))

        pass

class inputbox:

    def \_\_init\_\_(self,x = 250,y = 100,width = 100, height = 35, state = "passive",\*\*kwargs):

        self.TL = ()

        self.TR = ()

        self.BL = ()

        self.BR = ()

        self.cols = {}

        self.w1 = width

        self.h1 = height

        self.state = state

        self.user\_text = ""

        self.backspace\_time = time.time()

        self.delete\_time = time.time()

        self.text\_tit = ""

        self.tit\_th = height

        self.tit\_tw = 100

        self.text\_rect = ()

        self.x = x

        self.y = y

        self.maxlen = {"Name":30,"pwd":22}

        #pwd

        #self.hidepwd\_img = pygame.image.load(r"E:\120204 Ashish Thannickal\Project Work\Homepage\Hidepwd.png")

        self.hidepwd\_img = pygame.image.load(r"D:\Music Synthesizer\Homepage\Hidepwd.png")

        self.hidepwd\_img = pygame.transform.scale(self.hidepwd\_img,(30,30))

        #self.showpwd\_img = pygame.image.load(r"E:\120204 Ashish Thannickal\Project Work\Homepage\Showpwd.png")

        self.showpwd\_img = pygame.image.load(r"D:\Music Synthesizer\Homepage\Showpwd.png")

        self.showpwd\_img = pygame.transform.scale(self.showpwd\_img,(30,30))

        self.pwd\_img\_state = "hidden"

        self.pwd\_imgsrf = self.showpwd\_img.convert()

        self.pwd\_rect = pygame.rect.Rect(0,0,0,0)

        try:

            self.cols["passive"] = kwargs["c1"]

            self.cols["active"] = kwargs["c2"]

            self.cols["hover"] = kwargs["c3"]

        except:

            self.cols["passive"] = "gray"

            self.cols["active"] = "white"

            self.cols["hover"] = "purple"

        self.rect\_w = width

        self.rect\_h = height

        self.rect = pygame.rect.Rect(x,y,self.rect\_w,self.rect\_h)

        self.rect\_bor = pygame.Rect(x,y,self.rect\_w,self.rect\_h)

        #Cursor

        self.cursor\_pos = 0

        self.cursor\_pos\_coords = [0,0]

        self.cursor = pygame.Rect(self.rect.topright, (2, self.rect.height-2))

    def draw(self):

        #Input Box

        self.rect = pygame.rect.Rect(self.x,self.y,self.rect\_w,self.rect\_h)

        self.rect\_bor = pygame.Rect(self.x,self.y,self.rect\_w,self.rect\_h)

        self.bl = Dr.rect(S,color=self.cols[self.state],rect=self.rect)

        #self.bl = Dr.rect(S,color="black",rect=self.rect)

        self.bl\_bor = Dr.rect(S,color="#2949f3",rect=self.rect,width=3)

        self.TL = self.bl.topleft

        self.TR = self.bl.topright

        self.BL = self.bl.bottomleft

        self.BR = self.bl.bottomright

        #Input Text

        if self.type == "pwd":

            self.user\_txt1 = txt("•"\*len(self.user\_text),size = 28)

            self.user\_txt2 = txt("•"\*len(self.user\_text[:self.cursor\_pos]),size = 28)

            if self.pwd\_img\_state == "visible":

                self.user\_txt1 = txt(self.user\_text,size = 28)

                self.user\_txt2 = txt(self.user\_text[:self.cursor\_pos],size = 28)

                self.pwd\_imgsrf = self.hidepwd\_img.convert()

                #self.pwd\_img\_state = "visible"

            else:

                self.pwd\_imgsrf = self.showpwd\_img.convert()

                #self.pwd\_img\_state = "hidden"

            self.pwd\_rect = self.pwd\_imgsrf.get\_rect(topleft = (self.TR[0]+2,self.TL[1]))

            S.blit(self.pwd\_imgsrf,(self.TR[0]+2,self.TL[1]))

        else:

            self.user\_txt1 = txt(self.user\_text,size = 28)

            self.user\_txt2 = txt(self.user\_text[:self.cursor\_pos],size = 28)

        self.bl\_txt = S.blit(self.user\_txt1, (self.TL[0]+10,self.TL[1]+5))

        #print(self.bl\_txt.width+20,self.rect\_w)

        self.rect\_w = max(self.bl\_txt.width+20,self.w1)

        ###self.rect\_h = max(self.bl\_txt.height+20,self.h1)

        # Blit the  cursor

        if time.time() % 1 > 0.5 and self.state == "active":

            # bounding rectangle of the text

            text\_rect = self.user\_txt2.get\_rect(topleft = (self.rect.x+10, self.rect.y + 5))

            # set cursor position

            self.cursor.midleft = text\_rect.midright

            pygame.draw.rect(S, "Black", self.cursor)

        '''self.cursor\_pos\_coords[1] = self.TL[1]

        self.cursor\_pos\_coords[0] = self.TL[0] + \*self.cursor\_pos + 9'''

        #Title of IB

        S.blit(self.text\_tit, (self.TL[0]-self.tit\_tw-20,self.TL[1]))

    def set\_type(self,type="name"):

        if type == "name":

            self.type = "Name"

            self.txt\_tit = "USERNAME :"

            self.text\_tit = txt(self.txt\_tit,size = 38,color = "#abcdef",u = True)

            self.tit\_th = self.text\_tit.get\_height()

            self.tit\_tw = self.text\_tit.get\_width()

        if type == "pwd":

            self.type = "pwd"

            self.txt\_tit = "PASSWORD :"

            self.text\_tit = txt(self.txt\_tit,size = 38,color = "#abcdef",u = True)

            self.tit\_th = self.text\_tit.get\_height()

            self.tit\_tw = self.text\_tit.get\_width()

input\_name = inputbox()

input\_name.set\_type("name")

input\_pwd = inputbox(250,250)

input\_pwd.set\_type("pwd")

class Button:

    def \_\_init\_\_(self,x = 150,y = 400,width = 100, height = 30, state = "passive",text = "LOGIN",\*\*kwargs):

        self.TL = ()

        self.TR = ()

        self.BL = ()

        self.BR = ()

        self.cols = {}

        self.w1 = width

        self.h1 = height

        self.state = state

        self.text = text

        #self.txt = txt(self.text,font = r"E:\120204 Ashish Thannickal\Project Work\Homepage\BRADHITC.TTF", color = "black")

        self.txt = txt(self.text,font = r"D:\Music Synthesizer\Homepage\BRADHITC.TTF", color = "black")

        try:

            self.cols["passive"] = kwargs["c1"]

            self.cols["active"] = kwargs["c2"]

            self.cols["hover"] = kwargs["c3"]

        except:

            self.cols["passive"] = "pink"

            self.cols["active"] = "pink"

            self.cols["hover"] = "lavender"

        self.rect\_w = width

        self.rect\_h = height

        self.rect = pygame.rect.Rect(x,y,self.rect\_w,self.rect\_h)

        self.rect\_bor = pygame.Rect(x,y,self.rect\_w,self.rect\_h)

        self.x = x

        self.y = y

    def draw(self):

        #Input Box

        self.rect = pygame.rect.Rect(self.x,self.y,self.rect\_w,self.rect\_h)

        self.rect\_bor = pygame.Rect(self.x,self.y,self.rect\_w,self.rect\_h)

        self.bl = Dr.rect(S,color=self.cols[self.state],rect=self.rect,border\_radius=10)

        self.bl\_bor = Dr.rect(S,color="pink",rect=self.rect,width=3,border\_radius=10)

        self.TL = self.bl.topleft

        self.TR = self.bl.topright

        self.BL = self.bl.bottomleft

        self.BR = self.bl.bottomright

        self.bl\_txt = S.blit(self.txt, (self.TL[0]+10,self.TL[1]+5))

submit1 = Button()

submit2 = Button(text = "SUBMIT")

signup1 = Button(x=260,text = "SIGN UP")

back1 = Button(x = 260, text = "BACK")

class BGimg:

    def \_\_init\_\_(self,img):

        self.img = pygame.image.load(img)

        self.img\_scaled = pygame.transform.scale(self.img,(Sw\_org,Sh\_org))

        pass

    def draw(self):

        self.img\_scaled = pygame.transform.scale(self.img,(Sw,Sh))

        self.imgsrf = self.img\_scaled.convert()

        S.blit(self.imgsrf,(0,0))

bg\_img1 = BGimg(r"D:\Music Synthesizer\Homepage\musicnote.jpg")

class HP:

    def \_\_init\_\_(self):

        self.state = False

        self.tabslist = [input\_name,input\_pwd,submit1,signup1]

    def action(self):

        if self.state == 1:

            TxHP = TextHP(sizeiteration)

            TxHP.draw()

            unTxHP = underline(TxHP.tw,TxHP.BL[0],TxHP.BL[1])

            unTxHP.draw()

            timer1 = Timer(time.time())

            timer1.draw()

        if self.state == 2:

            bg\_img1.draw()

            self.tabslist = [input\_name,input\_pwd,submit1,signup1]

            TxHP = TextHP(hpstate=self.state)

            TxHP.draw()

            input\_name.draw()

            input\_pwd.draw()

            submit1.draw()

            signup1.draw()

            timer1 = Timer(time.time())

            timer1.draw()

        if self.state == 3:

            bg\_img1.draw()

            self.tabslist = [input\_name,input\_pwd,submit2,back1]

            TxHP = TextHP(hpstate=2)

            TxHP.draw()

            input\_name.draw()

            input\_pwd.draw()

            submit2.draw()

            back1.draw()

            timer1 = Timer(time.time())

            timer1.draw()

        if self.state == 0:

            self.txt = f"Welcome {input\_name.user\_text}"

            S.blit(txt(self.txt),(40,40))

            Seawalk\_bgm.end()

            import pygametrial

run = True

starttime = time.time()

homepg = HP()

iteration = 0

sizeiteration = 0

tabslist = [input\_name,input\_pwd,submit1,signup1]

tabstates = {"active":1,"passive":0}

now\_active = None

now\_activeindex = None

Seawalk\_bgm.play()

while run:

    S.fill((0,0,0))

    Sh = S.get\_height()

    Sw = S.get\_width()

    '''x = input\_pwd.hidepwd\_img.convert()

    S.blit(x,(60,140))'''

    #HomePage

    if time.time()-starttime < 2:

        homepg.state = 1

    elif time.time()-starttime < 120:

        if homepg.state != 0:

            homepg.state = max(2,homepg.state)

    else:

        Seawalk\_bgm.end()

    if submit1.state == "active":

        submit1.state="passive"

        homepg.state = 0

    if signup1.state == "active":

        homepg.state = 3

        signup1.state="passive"

    homepg.action()

    try:

        1

        #print(input\_name.user\_text[input\_name.cursor\_pos-1])

    except:

        pass

    KEYSTATES = pygame.key.get\_pressed()

    if KEYSTATES[pygame.K\_BACKSPACE]:

        backspace(input\_name,time.time())

        backspace(input\_pwd,time.time())

    elif KEYSTATES[pygame.K\_DELETE]:

        delete(input\_name,time.time())

        delete(input\_pwd,time.time())

    for event in E.get():

        if event.type == pygame.QUIT:

            run = False

            os.\_exit(0)

        if homepg.state == 2:

            eventcheck(homepg.tabslist)

            if event.type == pygame.MOUSEBUTTONDOWN:

                mouse\_pos = pygame.mouse.get\_pos()

                if submit1.rect.collidepoint(mouse\_pos):

                    now\_active = None

                    now\_activeindex = None

                    uservalidation()

                else:

                    submit1.state = "passive"

                if signup1.rect.collidepoint(mouse\_pos):

                    now\_active = signup1

                    now\_activeindex = homepg.tabslist.index(signup1)

                    homepg.state=3

                    signup1.state = "active"

                else:

                    signup1.state = "passive"

            if event.type == pygame.MOUSEMOTION:

                mouse\_pos = pygame.mouse.get\_pos()

                if submit1.rect.collidepoint(mouse\_pos):

                    if submit1.state == "passive":

                        submit1.state = "hover"

                else:

                    if submit1.state == "hover":

                        submit1.state = "passive"

                if signup1.rect.collidepoint(mouse\_pos):

                    if signup1.state == "passive":

                        signup1.state = "hover"

                else:

                    if signup1.state == "hover":

                        signup1.state = "passive"

        if homepg.state == 3:

            eventcheck(homepg.tabslist)

            if event.type == pygame.MOUSEBUTTONDOWN:

                mouse\_pos = pygame.mouse.get\_pos()

                if submit2.rect.collidepoint(mouse\_pos):

                    now\_active = None

                    now\_activeindex = None

                    usercreation()

                else:

                    submit2.state = "passive"

                if back1.rect.collidepoint(mouse\_pos):

                    homepg.state = 2

                else:

                    back1.state = "passive"

            if event.type == pygame.MOUSEMOTION:

                mouse\_pos = pygame.mouse.get\_pos()

                if submit2.rect.collidepoint(mouse\_pos):

                    if submit2.state == "passive":

                        submit2.state = "hover"

                else:

                    if submit2.state == "hover":

                        submit2.state = "passive"

                if back1.rect.collidepoint(mouse\_pos):

                    if back1.state == "passive":

                        back1.state = "hover"

                else:

                    if back1.state == "hover":

                        back1.state = "passive"

    #print(input\_pwd.pwd\_img\_state)

    #print(homepg.state)

    iteration += 1

    D.update()

MODULE NAME: Constants.py

import mysql.connector as M

from tkinter import \*

from tkinter.messagebox import askyesno,showinfo

try:

    MyDB = M.connect(host="localhost",user="root",password="mysql", database = "Ashishproject")

except:

    MyDB = M.connect(host="localhost",user="root",password="l9hfG&7a", database = "Ashishproject")

MyC = MyDB.cursor()

Exe= MyC.execute

def getuser():

    global UN

    Exe("Select currentuser from currentuser")

    UN = MyC.fetchall()[0][0]

    print(UN)

    return UN

def playsong(UN):

    try:

        Exe(f"update currentuser set currentsong = NULL")

        MyDB.commit()

        Exe(f"Select Songname from {UN}songs")

        l = [i[0] for i in MyC.fetchall()]

        #Create an instance of tkinter frame

        win= Tk()

        #Define the size of window or frame

        #win.geometry("715x250")

        #Set the Menu initially

        menu= StringVar()

        menu.set("Select Song")

        #Create a dropdown Menu

        drop= OptionMenu(win, menu,\*l)

        drop.pack()

        def selectbuttonaction():

            global song

            song = menu.get()

            print(song)

            win.destroy()

            if song != "Select Song":

                Exe(f"update currentuser set currentsong = '{song}'")

            MyDB.commit()

        def delbuttonaction():

            global song

            song = menu.get()

            print(song)

            answer1 = (askyesno("Delete Song","Are you sure you wish to delete?\n"+

            "This action cannot be undone!"))

            if answer1 == True:

                win.destroy()

                if song != "Select Song":

                    Exe(f"delete from {UN}songs where songname = '{song}'")

                MyDB.commit()

        songselbutton = Button(

            win,

            text = "Play Song",

            command = selectbuttonaction

            )

        songdelbutton = Button(

            win,

            text = "Delete Song",

            command = delbuttonaction

        )

        songselbutton.pack()

        songdelbutton.pack()

        win.mainloop()

        try:

            Exe(f"Select recording from {UN}songs where songname = (select currentsong from currentuser)")

            recording = MyC.fetchall()[0]

            Exe(f"update currentuser set currentsong = NULL")

            MyDB.commit()

            return recording

        except:

            pass

    except:

        win.wm\_withdraw()

        showinfo("Message","No Saved Songs")

        win.destroy()

MODULE NAME: soundmodules.py

piano\_keys = ["A0","a0","B0"]

for i in range (1,8):

    keys = (str(i)+" ").join("CcDdEFfGgAaB ").split()

    piano\_keys += keys

piano\_keys += ["C8"]

base\_freq = 27.5

note\_freqs = {piano\_keys[i]:base\_freq\*pow(2,i/12) for i in range(len(piano\_keys))}

note\_freqs[''] = note\_freqs[" "] = 0.0

MODULE NAME: pygametrial.py

import pygame

import numpy as np

from numpy import sin, pi as π

from soundmodules import piano\_keys as pk, note\_freqs as nf

import time

from threading import \*

import sys

import os

import tkinter as tk

from tkinter import Tk

from tkinter.messagebox import askyesno,showinfo

from tkinter.simpledialog import askstring

from Constants import \*

#https://www.projectrhea.org/rhea/index.php/Fourier\_analysis\_in\_Music

#https://data-flair.training/blogs/deep-surveillance-with-deep-learning-intelligent-video-surveillance-project/

#initialising pygame

pygame.init()

#Variables

D = pygame.display

S = D.set\_mode((800,600), pygame.RESIZABLE)

E = pygame.event

Dr = pygame.draw

def avg(l): return sum(l)/len(l)

record = {1:{"E4":0.5},2:{"E4":0.5},3:{"F4":0.5}}

time\_record = {}

timestamps = {}

dead = False

#Format

color\_active = (255,255,255)

color\_passive = (100,100,100)

color\_current = color\_active

d = {color\_active:color\_passive, color\_passive: color\_active}

def hex\_to\_RGB(hex):

  ''' "#FFFFFF" -> [255,255,255] '''

  # Pass 16 to the integer function for change of base

  return [int(hex[i:i+2], 16) for i in range(1,6,2)]

def RGB\_to\_hex(RGB):

  ''' [255,255,255] -> "#FFFFFF" '''

  # Components need to be integers for hex to make sense

  RGB = [int(x) for x in RGB]

  return "#"+"".join(["0{0:x}".format(v) if v < 16 else

            "{0:x}".format(v) for v in RGB])

def color\_dict(gradient):

  ''' Takes in a list of RGB sub-lists and returns dictionary of

    colors in RGB and hex form for use in a graphing function

    defined later on '''

  return {"hex":[RGB\_to\_hex(RGB) for RGB in gradient],

      "r":[RGB[0] for RGB in gradient],

      "g":[RGB[1] for RGB in gradient],

      "b":[RGB[2] for RGB in gradient]}

def linear\_gradient(start\_hex="#c31432", finish\_hex="#240b36", n=255):

  ''' returns a gradient list of (n) colors between

    two hex colors. start\_hex and finish\_hex

    should be the full six-digit color string,

    inlcuding the number sign ("#FFFFFF") '''

  # Starting and ending colors in RGB form

  s = hex\_to\_RGB(start\_hex)

  f = hex\_to\_RGB(finish\_hex)

  # Initilize a list of the output colors with the starting color

  RGB\_list = [s]

  # Calcuate a color at each evenly spaced value of t from 1 to n

  for t in range(1, n):

    # Interpolate RGB vector for color at the current value of t

    curr\_vector = [

      int(s[j] + (float(t)/(n-1))\*(f[j]-s[j]))

      for j in range(3)

    ]

    # Add it to our list of output colors

    RGB\_list.append(curr\_vector)

  return color\_dict(RGB\_list)

#colors = linear\_gradient("#ad5389","#3c1053",n = 30)['hex']

colors = linear\_gradient("#000000","#FFFFFF",n = 30)['hex']

#print(colors)

def color\_switch(color):

    global color\_current

    #print(color\_current)

    color\_current = d[color]

'''    box = pygame.Rect(10,10,50,50)

    Dr.rect(S,color\_current,[10,10,50,50])'''

def button():

    Dr.rect(S,color\_current,[10,10,50,50])

def txt(t,font="Corbel",size = 20,color = "#ff0000", italic = False, u = False):

        # defining a font

    try:

        smallfont = pygame.font.Font(font, size)

    except:

        smallfont = pygame.font.SysFont(font, size)

    # rendering a text written in

    # this font

    smallfont.italic = italic

    smallfont.underline = u

    text = smallfont.render( t, True , color)

    return text

note\_status = dict.fromkeys(nf,False)

nf\_white = {i:nf[i]  for i in nf if i.isupper()}

#print(nf\_white)

nf\_black = {i:nf[i]  for i in nf if i.islower()}

#print(nf\_black)

#print(note\_status)

c\_list = [i for i in range(len(nf\_white)) if list(nf\_white)[i][0] == "C"]

def keys():

    global white\_keys, black\_keys, width

    h = S.get\_height()

    w = S.get\_width()

    x = np.linspace(2\*w/10,8\*w/10,10)

    width = avg([x[i]-x[i-1] for i in range(1,len(x))])

    start = 2

    if defoctave != 0:

        start = max([i for i in c\_list if i<defoctave\*7])

    white\_keys = {list(nf\_white.keys())[start+i]:pygame.Rect(x[i]-width/2,round(h/2),width,round(h/2)) for i in range(10) if start+i in range (0,len(nf\_white))}

    black\_keys = {i.lower():pygame.Rect(white\_keys[i].left +3\*width/4,round(h/2),width/2,round(6\*h/20)) for i in white\_keys if i.lower() in nf}

    iteration = 0

    for i in white\_keys:

        if note\_status[i]:

            Dr.rect(S,"gray",white\_keys[i],border\_radius=10)

            Dr.rect(S,"#000000",white\_keys[i],border\_radius=10,border\_top\_left\_radius=4,border\_top\_right\_radius=4,width = 1)

        else:

            #Dr.rect(S,colors[iteration],white\_keys[i],border\_radius=10)

            Dr.rect(S,"#F0F0F0",white\_keys[i],border\_radius=10,border\_top\_left\_radius=4,border\_top\_right\_radius=4)

            Dr.rect(S,"#000000",white\_keys[i],border\_radius=10,border\_top\_left\_radius=4,border\_top\_right\_radius=4,width = 1)

            S.blit(txt(i),(white\_keys[i].left,white\_keys[i].bottom - 40))

        iteration += 1

    iteration = 0

    for i in black\_keys:

        if note\_status[i]:

            Dr.rect(S,"pink",black\_keys[i],border\_radius=5)

        else:

            Dr.rect(S,"#222222",black\_keys[i],border\_bottom\_left\_radius=5,border\_bottom\_right\_radius=5)

            Dr.rect(S,"#AAAAAA",black\_keys[i],border\_bottom\_left\_radius=5,border\_bottom\_right\_radius=5, width = 2)

            S.blit(txt(i),(black\_keys[i].left,black\_keys[i].bottom-40))

        iteration += 1

    #print(note\_keys)

#note

samplerate = 44100

defoctave = 4

def play\_note(note = "A4",d = 10,A=6000):

    A = A

    f = nf[note]

    t1 = np.linspace(0,d,int(d\*samplerate))

    ω = 2 \* π \* f

    y = A \* np.sin (ω\*t1[10:-10])

    sound = np.asarray([y,y]).T.astype(np.int16)

    sound = pygame.sndarray.make\_sound(sound.copy())

    return sound

    #write("examplewav",samplerate,y.astype(np.int16))

note\_sounds = { i:play\_note(i) for i in note\_status}

instruments = ["flute","piano","violin"]

inst\_flute = {0 : 1, 1: 9, 2: 4, 3:2, 5: 0.3}

def press\_action(pos,keycolor,timestamp,key):

    #color\_switch(color\_current)

    s = keycolor[pos]

    #print(s, nf[s])

    note\_status[s] = True

    note\_sounds[keycolor[pos]].play()

    if Editable:

        time\_record[timestamp] = [s,0]

        timestamps[key] = timestamp

def lift\_action(pos,keycolor,key,newtimestamp):

    #color\_switch(color\_current)

    s = keycolor[pos]

    note\_status[s] = False

    note\_sounds[keycolor[pos]].stop()

    if Editable:

        time\_record[timestamps[key]][1] = newtimestamp-timestamps[key]

# work function

def work(t,speed,key,color,dur = 4,):

    global dead

    print(t,speed,key,dur)

    timepassed = time.time()-t

    h = S.get\_height()

    left = color[key].left

    playcol = "orange"

    widthn = width

    if color == black\_keys:

        widthn = width/2

        playcol = "pink"

    dist = speed \* timepassed

    bar = pygame.Rect(left,dist,width,100)

    playnote = 0

    timeplayed = playstart = 0

    bar\_height = dur\*speed

    while timeplayed < dur and (not dead):

        playnote += 1

        if playnote == 1:

            playstart = time.time()

            note\_status[key] = True

            note\_sounds[key].play()

        timeplayed = time.time()-playstart

        '''dist = speed \* timepassed

        if bar.bottom >= h/2:

            playnote+=1

            bar = pygame.Rect(left,dist,widthn, h/2 - bar.top)

            Dr.rect(S,playcol,bar,border\_radius = -1)

            print(timeplayed)

            if playnote == 1:

                playstart = time.time()

                note\_sounds[key].play()

            timeplayed = time.time() - playstart

        else:

            bar = pygame.Rect(left,dist,widthn, bar\_height)

            Dr.rect(S,playcol,bar,border\_radius =  4)

        timepassed = time.time() - t'''

    note\_status[key] = False

    note\_sounds[key].stop()

def multiwork(record):

    global Playpress

    st = time.time()

    it = 0

    for i in record:

        it+=   1

        if it == 1:

            st = time.time()

        key = record[i][0]

        col = white\_keys if key[0].isupper() else black\_keys

        ct = time.time()

        #print(ct)

        while time.time() - st < i:

            pass

        else:

            t1 = Thread(target = work, args = (time.time(),500,key,col,record[i][1]))

            t1.start()

    Playpress = False

def b(xcord,ycord,text, pressed):

    #stores the width of the

    # screen into a variable

    width = S.get\_width()

    # stores the height of the

    # screen into a variable

    height = S.get\_height()

    # stores the (x,y) coordinates into

    # the variable as a tuple

    mouse = pygame.mouse.get\_pos()

    # if mouse is hovered on a button it

    # changes to lighter shade

    if xcord <= mouse[0] <= xcord + 140 and height/2 + ycord<= mouse[1] <= height/2 + ycord + 40:

        Dr.rect(S,(170,170,170),[xcord,height/2+ycord,140,40])

    elif pressed == True:

        Dr.rect(S,(255,0,0),[xcord,height/2+ycord,140,40])

    else:

        Dr.rect(S,(100,100,100),[xcord,height/2+ycord,140,40])

    # superimposing the text onto our button

    S.blit(txt(text, size = 35, color = "#ffffff") , (xcord + 5,height/2+ycord))

def b\_record():

    b(0,20,"RECORD",Recpress)

def b\_pause():

    b(0,80,"PAUSE",Pausepress)

def b\_play():

    b(0,140,"PLAY",Playpress)

def b\_save():

    b(0,200,"SAVE",Savepress)

def b\_songs():

    b(0,260,"SONGS",Songspress)

def b\_logout():

    b(0,-40,"LOGOUT",None)

'''class Button:

    def \_\_init\_\_(self,x = 150,y = 400,width = 100, height = 30, state = "passive",text = "LOGIN",\*\*kwargs):

        self.TL = ()

        self.TR = ()

        self.BL = ()

        self.BR = ()

        self.cols = {}

        self.w1 = width

        self.h1 = height

        self.state = state

        self.text = text

        #self.txt = txt(self.text,font = r"E:\120204 Ashish Thannickal\Project Work\Homepage\BRADHITC.TTF", color = "black")

        self.txt = txt(self.text,font = r"D:\Music Synthesizer\Homepage\BRADHITC.TTF", color = "black")

        try:

            self.cols["passive"] = kwargs["c1"]

            self.cols["active"] = kwargs["c2"]

            self.cols["hover"] = kwargs["c3"]

        except:

            self.cols["passive"] = "pink"

            self.cols["active"] = "pink"

            self.cols["hover"] = "lavender"

        self.rect\_w = width

        self.rect\_h = height

        self.rect = pygame.rect.Rect(x,y,self.rect\_w,self.rect\_h)

        self.rect\_bor = pygame.Rect(x,y,self.rect\_w,self.rect\_h)

        self.x = x

        self.y = y

    def draw(self):

        #Input Box

        self.rect = pygame.rect.Rect(self.x,self.y,self.rect\_w,self.rect\_h)

        self.rect\_bor = pygame.Rect(self.x,self.y,self.rect\_w,self.rect\_h)

        self.bl = Dr.rect(S,color=self.cols[self.state],rect=self.rect,border\_radius=10)

        self.bl\_bor = Dr.rect(S,color="pink",rect=self.rect,width=3,border\_radius=10)

        self.TL = self.bl.topleft

        self.TR = self.bl.topright

        self.BL = self.bl.bottomleft

        self.BR = self.bl.bottomright

        self.bl\_txt = S.blit(self.txt, (self.TL[0]+10,self.TL[1]+5))

recordbutton1 = Button(x=0,y=200,text = "RECORD" ,c3 = (170,170,170),c1 = (100,100,100),c2 = (255,0,0))'''

#Loop

run = True

op = "+"

ite = 0

startmsg = True

Editable = False

start\_time = time.time()

Recpress = Pausepress = Playpress = Savepress = Songspress =  False

while run:

    ite = eval(f"{ite}{op}1")

    if ite == len(colors):

        op = "-"

        ite -= 1

    if ite == 0:

        op = "+"

    S.fill((0,0,0))

    mytxt = txt("PIANO","Felix Titling", 60, colors[ite])

    mytxtw = mytxt.get\_width()

    S.blit(mytxt,(S.get\_width()//2 - mytxtw//2,S.get\_height()/6))

    if startmsg:

        msg = txt("Use 'a' to ';' for white keys top row for black",size = 40, color = "light blue")

        msgw = msg.get\_width()

        S.blit(msg,(S.get\_width()//2 - msgw//2, S.get\_height()//3))

        start\_time = time.time()

    #print(pygame.event.get())

    #print(defoctave)

    for event in E.get():

        Sh = S.get\_height()

        Sw = S.get\_width()

        #print(event)

        #print("Semicolon",pygame.key.get\_pressed()[pygame.K\_SEMICOLON],end = "\r")

        if event.type == pygame.QUIT:

            dead = True

            run = False

            os.\_exit(0)

            pygame.quit()

            sys.exit()

        elif event.type == pygame.KEYDOWN:

            startmsg = False

            wk = list(white\_keys)

            bk = list(black\_keys)

            if event.key == pygame.K\_a:

                press\_action(0,wk,time.time()-start\_time,pygame.K\_a)

            if event.key == pygame.K\_s:

                press\_action(1,wk,time.time()-start\_time,pygame.K\_s)

            if event.key == pygame.K\_d:

                press\_action(2,wk,time.time()-start\_time,pygame.K\_d)

            if event.key == pygame.K\_f:

                press\_action(3,wk,time.time()-start\_time,pygame.K\_f)

            if event.key == pygame.K\_g:

                press\_action(4,wk,time.time()-start\_time,pygame.K\_g)

            if event.key == pygame.K\_h:

                if len(wk)>5:

                    press\_action(5,wk,time.time()-start\_time,pygame.K\_h)

            if event.key == pygame.K\_j:

                if len(wk)>6:

                    press\_action(6,wk,time.time()-start\_time,pygame.K\_j)

            if event.key == pygame.K\_k:

                if len(wk)>7:

                    press\_action(7,wk,time.time()-start\_time,pygame.K\_k)

            if event.key == pygame.K\_l:

                if len(wk)>8:

                    press\_action(8,wk,time.time()-start\_time,pygame.K\_l)

            if event.key == pygame.K\_SEMICOLON:

                if len(wk)>9:

                    press\_action(9,wk,time.time()-start\_time,pygame.K\_SEMICOLON)

            if event.key == pygame.K\_w:

                press\_action(0,bk,time.time()-start\_time,pygame.K\_w)

            if event.key == pygame.K\_e:

                press\_action(1,bk,time.time()-start\_time,pygame.K\_e)

            if event.key == pygame.K\_t:

                press\_action(2,bk,time.time()-start\_time,pygame.K\_t)

            if event.key == pygame.K\_y:

                press\_action(3,bk,time.time()-start\_time,pygame.K\_y)

            if event.key == pygame.K\_u:

                press\_action(4,bk,time.time()-start\_time,pygame.K\_u)

            if event.key == pygame.K\_o:

                press\_action(5,bk,time.time()-start\_time,pygame.K\_o)

            if event.key == pygame.K\_p:

                press\_action(6,bk,time.time()-start\_time,pygame.K\_p)

            if event.key == pygame.K\_DOWN:

                if defoctave >0:

                    defoctave -= 1

            if event.key == pygame.K\_UP:

                if defoctave <7:

                    defoctave += 1

        elif event.type == pygame.KEYUP:

            if event.key == pygame.K\_a:

                lift\_action(0,wk,pygame.K\_a,time.time()-start\_time)

            if event.key == pygame.K\_s:

                lift\_action(1,wk,pygame.K\_s,time.time()-start\_time)

            if event.key == pygame.K\_d:

                lift\_action(2,wk,pygame.K\_d,time.time()-start\_time)

            if event.key == pygame.K\_f:

                lift\_action(3,wk,pygame.K\_f,time.time()-start\_time)

            if event.key == pygame.K\_g:

                lift\_action(4,wk,pygame.K\_g,time.time()-start\_time)

            if event.key == pygame.K\_h:

                if len(wk)>5:

                    lift\_action(5,wk,pygame.K\_h,time.time()-start\_time)

            if event.key == pygame.K\_j:

                if len(wk)>6:

                    lift\_action(6,wk,pygame.K\_j,time.time()-start\_time)

            if event.key == pygame.K\_k:

                if len(wk)>7:

                    lift\_action(7,wk,pygame.K\_k,time.time()-start\_time)

            if event.key == pygame.K\_l:

                if len(wk)>8:

                    lift\_action(8,wk,pygame.K\_l,time.time()-start\_time)

            if event.key == pygame.K\_SEMICOLON:

                if len(wk)>9:

                    lift\_action(9,wk,pygame.K\_SEMICOLON,time.time()-start\_time)

            if event.key == pygame.K\_w:

                lift\_action(0,bk,pygame.K\_w,time.time()-start\_time)

            if event.key == pygame.K\_e:

                lift\_action(1,bk,pygame.K\_e,time.time()-start\_time)

            if event.key == pygame.K\_t:

                lift\_action(2,bk,pygame.K\_t,time.time()-start\_time)

            if event.key == pygame.K\_y:

                lift\_action(3,bk,pygame.K\_y,time.time()-start\_time)

            if event.key == pygame.K\_u:

                lift\_action(4,bk,pygame.K\_u,time.time()-start\_time)

            if event.key == pygame.K\_o:

                lift\_action(5,bk,pygame.K\_o,time.time()-start\_time)

            if event.key == pygame.K\_p:

                lift\_action(6,bk,pygame.K\_p,time.time()-start\_time)

        elif event.type == pygame.MOUSEBUTTONDOWN:

            if 0 <= mouse[0] <= 140 and Sh/2+20 <= mouse[1] <= Sh/2+60:

                Editable = True

                time\_record = {}

                if Recpress == False:

                    Recpress = True

                else:

                    Recpress = False

                    Editable = False

                Playpress = False

                Pausepress = False

                #time\_record = {}

                startmsg = True

            if 0 <= mouse[0] <= 140 and Sh/2+80 <= mouse[1] <= Sh/2+120:

                Recpress = False

                Pausepress = True

                Editable = False

                Pausepress = False

            if 0 <= mouse[0] <= 140 and Sh/2+140 <= mouse[1] <= Sh/2+160:

                Recpress = False

                Pausepress = False

                Playpress = True

                Editable = False

                d = time\_record

                startmsg = True

                t = Thread(target=multiwork,args = (d,))

                t.start()

            if 0 <= mouse[0] <= 140 and Sh/2+180 <= mouse[1] <= Sh/2+220:

                Recpress=False

                Pausepress=False

                #Tk().wm\_withdraw()

                win = Tk()

                win.wm\_withdraw()

                answer = askyesno("Save recording","Do you wish to save the recording?")

                if answer == True:

                    from Constants import getuser

                    UN = getuser()

                    songname = askstring("Songname", "Enter Song Name:")

                    #to check for repetition

                    try:

                        #print(time\_record)

                        Exe(f"insert into {UN}songs values (\"{songname}\",\"{time\_record}\")")

                        MyDB.commit()

                    except:

                        answer1 = askyesno("Save recording","Song already exists. Replace the song?")

                        if answer1 == True:

                            Exe(f"update {UN}songs set recording = \"{time\_record}\" where songname = \"{songname}\"")

                            MyDB.commit()

                    showinfo("Message",f"{songname} saved")

                win.destroy()

                Recpress = Pausepress = Playpress = Savepress = Songspress =  False

            if 0 <= mouse[0] <= 140 and Sh/2+240 <= mouse[1] <= Sh/2+300:

                from Constants import playsong,getuser

                Recpress = False

                Pausepress = False

                Playpress = False

                Editable = False

                try:

                    d = eval(playsong(getuser())[0])

                    #print(d)

                    startmsg = True

                    t = Thread(target=multiwork,args = (d,))

                    t.start()

                    time\_record = {}

                except:

                    pass

                    #print("No Song Chosen")

            if 0 <= mouse[0] <= 140 and Sh/2-40 <= mouse[1] <= Sh/2:

                run=False

    mouse = pygame.mouse.get\_pos()

    b\_play()

    b\_record()

    b\_pause()

    b\_save()

    b\_songs()

    b\_logout()

    #recordbutton1.draw()

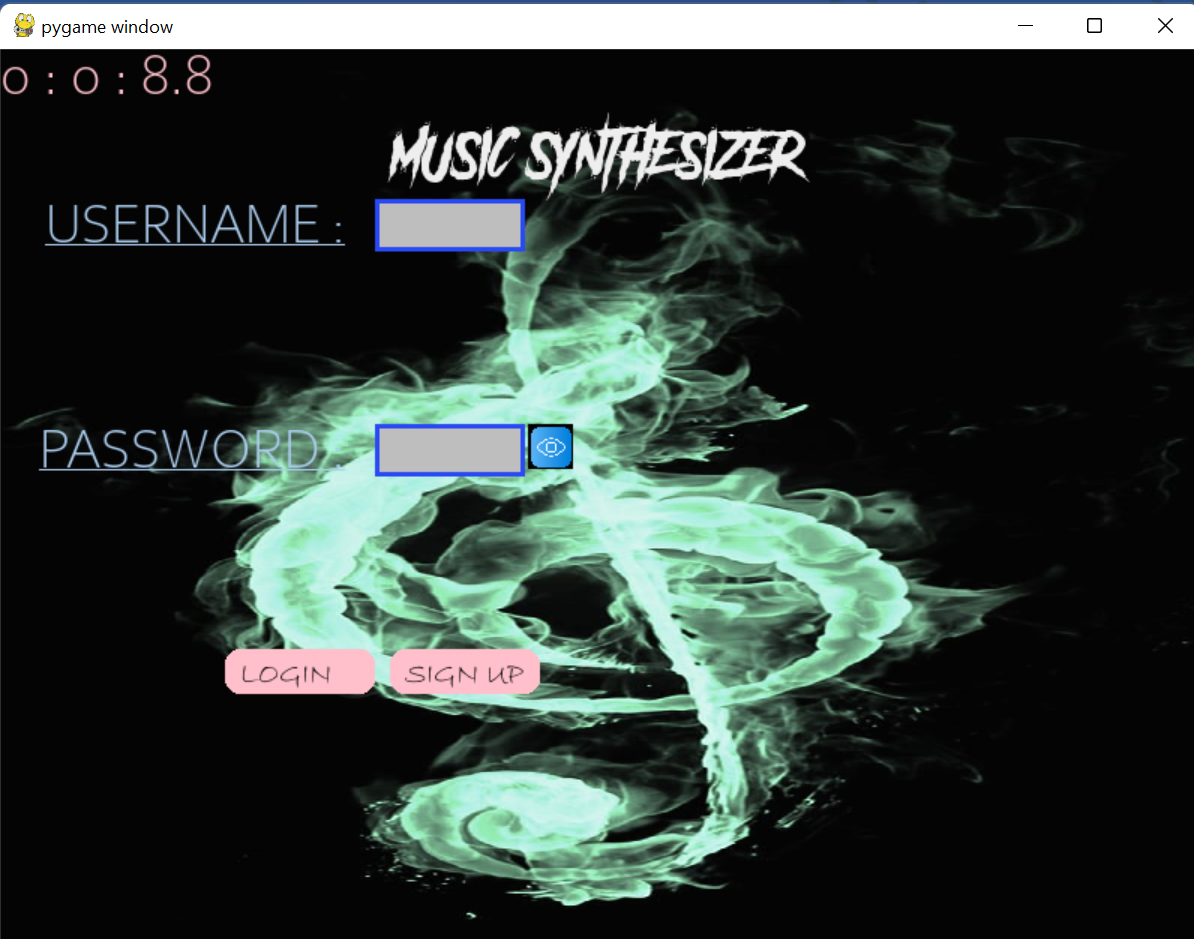
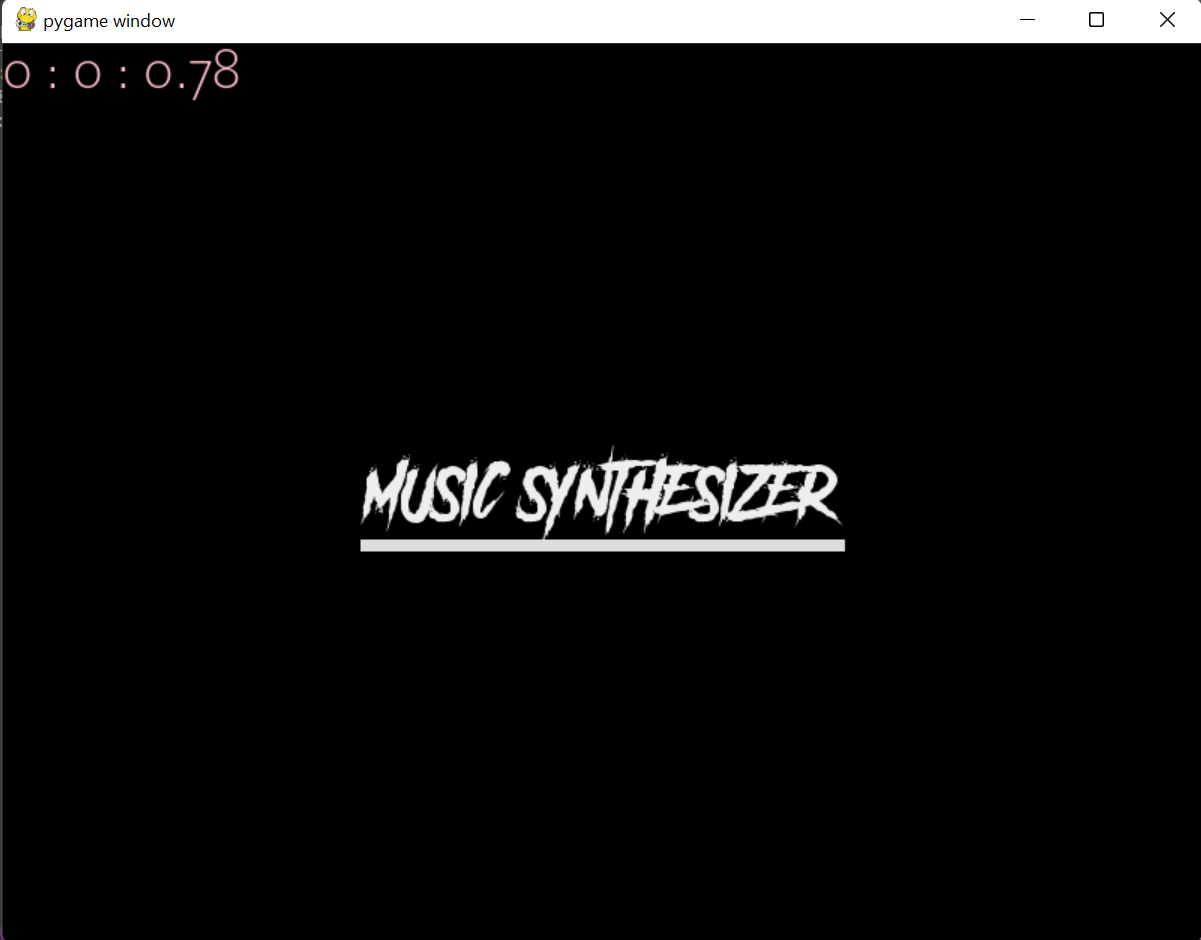
    keys()

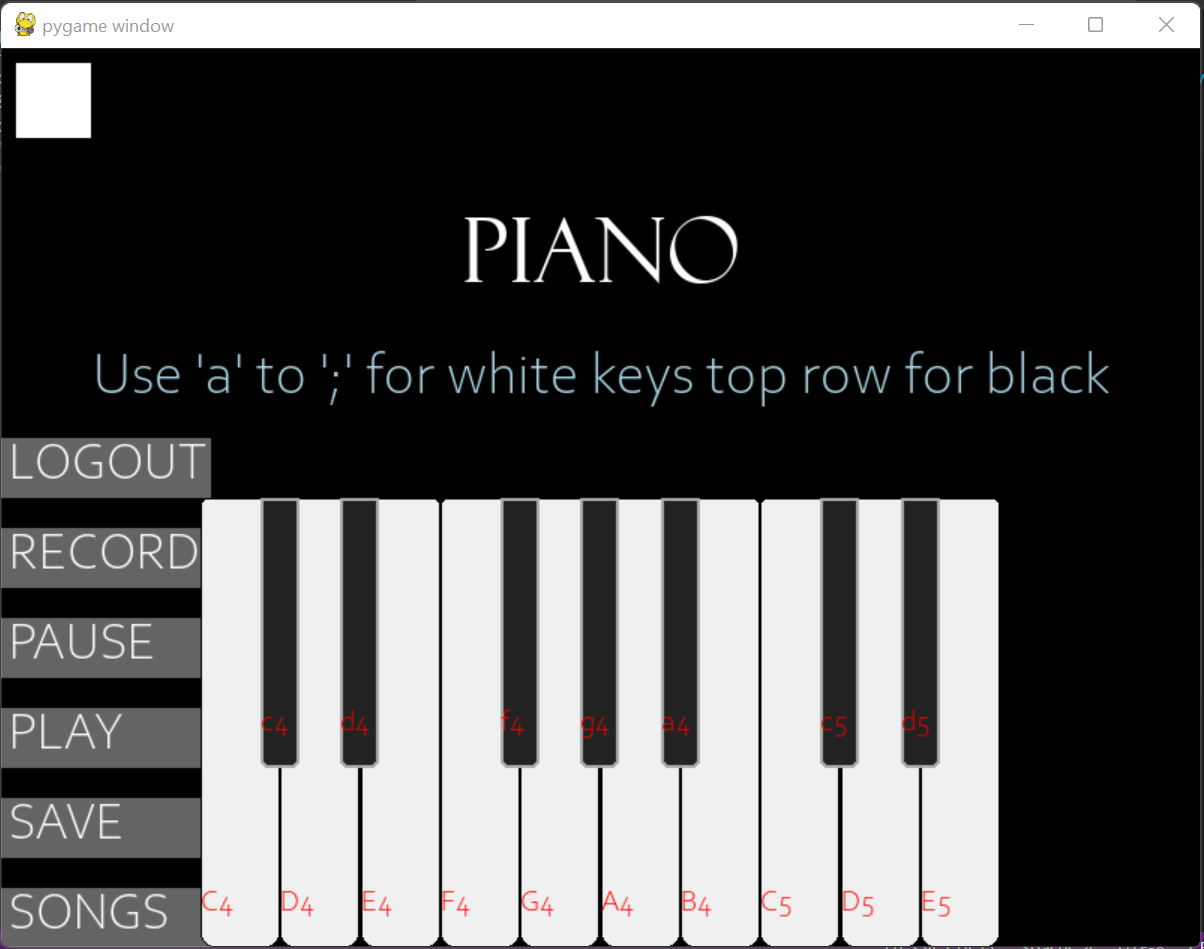
    button()

    D.update()

import homepagescl

SCREENSHOTS





LIMITATIONS

* There is a time delay between login page and application as it takes time to load all the objects to be displayed on the screen. If the user presses any keys while this is taking place, the app will experience an error and crash
* When playing a recorded song, it is not able to cope with the memory demand and there is a lag when the colours of the pressed keys are displayed.
* Keys from multiple octaves cannot be played simultaneously, hence you cannot play a high note and a low note together. This is due to limit on number of keys on keyboard and space to display keys on the screen.
* Also if the recording involves keys from multiple octaves, it will not play the keys in the different octave unless the user manually switches the octaves at the correct moments

SCOPE FOR IMPROVEMENT

* Can integrate more sounds (eg -violin, flute etc)
* Can be made more user friendly in certain areas
* Can possibly minimise time delays and glitches but not remove them completely

BIBLIOGRAPHY

* <https://plainenglish.io/blog/making-a-synth-with-python-oscillators-2cb8e68e9c3b>
* <https://www.projectrhea.org/rhea/index.php/Fourier_analysis_in_Music>
* <https://data-flair.training/blogs/deep-surveillance-with-deep-learning-intelligent-video-surveillance-project/>
* <https://soundcloud.com/ashamaluevmusic/sets/instrumental-background-music-for-videos>
* <https://www.ashamaluevmusic.com/summer-music>
* <https://www.youtube.com/watch?v=spUNpyF58BY>
* <https://en.wikipedia.org/wiki/Fourier_transform>
* <https://www.geeksforgeeks.org/>
* <https://www.w3schools.com/>
* <https://bsouthga.dev/posts/color-gradients-with-python>