PORTABLE SELF ASSESSMENT AUDIOMETER USING RASPBERRY PI

SOURCE CODE

PYTHON CODE

```
import argparse
from datetime import datetime, timedelta
from time import sleep
import numpy as np
import pyaudio
import threading
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from pynput.mouse import Listener, Button
import tkinter as tk
# Use interactive backend for displaying plots in a separate window
plt.switch_backend('TkAgg') # You may need to install TkAgg backend if not already installed
class HearingTest:
def __init__(self):
self.signal = None
self.right_data = []
self.detected = False
self.start_time = None
def display_instructions(self):
"""Display instructions in a new window"""
instructions_window = tk.Tk()
instructions_window.title("Instructions")
instructions_label = tk.Label(instructions_window, text="Portable Self Assessment Audiometer", font=("Arial", 16, "bold"))
```

```
instructions_label.pack(padx=10, pady=10)
instructions_text = "1. Put on your headphones.\n\n2. Click the left mouse button when you hear pulsing sounds.\n\n3. The test will run for the right ear, playing sounds at different frequencies and volumes.
instructions_text_label = tk.Label(instructions_window, text=instructions_text, font=("Arial", 12), justify=tk.LEFT)
instructions_text_label.pack(padx=10, pady=10)
start_button = tk.Button(instructions_window, text="Start Test", command=self.start_test)
start_button.pack(padx=10, pady=10)
instructions_window.mainloop()
def start_test(self):
"""Start the hearing test"""
self.display_instructions()
self.run_test()
def player(self, p, repeat=1, ear='right'):
"""Plays sounds with different frequencies and volume levels"""
volumes = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90] # Adjusted volume levels in dB
frequencies = [125, 250, 500, 1000, 2000, 4000, 8000] # Adjusted frequencies in Hz
# Repeat each frequency based on the provided argument
frequencies = np.repeat(frequencies, repeat)
stream = p.open(format=pyaudio.paFloat32,
channels=1,
rate=44100,
output=True)
sleep(0.1)
for freq in frequencies:
self.detected = False
for vol in volumes:
print(f"Playing frequency: {freq} Hz at volume: {vol} dB for {ear} ear")
self.signal = [freq, vol, datetime.now()]
audio_data = (np.sin(2 * np.pi * np.arange(44100 * 0.5) * freq / 44100)).astype(np.float32)
audio_data = audio_data * 10**(vol / 20)
stream.write(audio_data.tobytes())
sleep(2) # Adding 2-second pause after playing each volume level
```

```
if self.detected:
break
sleep(2) # Adding 2-second pause after playing each frequency
stream.stop_stream()
stream.close()
def on_click(self, x, y, button, pressed):
"""Callback function for mouse clicks"""
if button == Button.left and pressed:
if self.signal:
d = self.signal + [datetime.now()]
print(f'Recording event: {d}')
self.right_data.append(d)
self.detected = True
def listener(self):
"""Listens to mouse clicks"""
with Listener(on_click=self.on_click) as listener:
listener.join()
def analyse_results(self, data, ear):
"""Stores and visualizes results"""
now = datetime.now()
# Load data to DataFrame
df = pd.DataFrame(data, columns=['frequency', 'volume', 'played', 'heard'])
df['reaction_time'] = (df['heard'] - df['played']).dt.microseconds // 1000
# Create audiogram chart
audiogram_fig = plt.figure()
ax1 = audiogram_fig.add_subplot(111)
ax1.plot(df['frequency'], df['volume'], marker='x', linestyle='-', color='black')
ax1.set(title=f"Audiogram for {ear} ear", ylim=[90, -10], yticks=[90, 80, 70, 60, 50, 40, 30,
20, 10, 0, -10])
ax1.grid(True)
ax1.set_ylabel('Hearing Level in decibels (volume in dB)')
# Add x-axis ticks and labels at the top of the chart
ax2 = ax1.twiny()
ax2.set_xlim(ax1.get_xlim())
```

```
ax2.set_xticks(df['frequency'])
ax2.set_xticklabels(df['frequency'])
ax2.set_xlabel('Pitch (frequency in Hz)')
ax2.xaxis.tick_top()
# Add colored rows for different hearing loss stages
ax1.axhspan(-10, 15, facecolor='green', alpha=0.3, label='Normal Hearing (0-15 dB)')
ax1.axhspan(16, 25, facecolor='yellow', alpha=0.3, label='Slight Hearing Loss (16-25 dB)')
ax1.axhspan(26, 40, facecolor='orange', alpha=0.3, label='Mild Hearing Loss (26-40 dB)')
ax1.axhspan(41, 55, facecolor='red', alpha=0.3, label='Moderate Hearing Loss (41-55 dB)')
ax1.axhspan(56, 70, facecolor='purple', alpha=0.3, label='Moderately Severe Hearing Loss (56-70 dB)')
ax1.axhspan(71, 90, facecolor='brown', alpha=0.3, label='Severe Hearing Loss (71-90 dB)')
ax1.axhspan(91, 120, facecolor='black', alpha=0.3, label='Profound Hearing Loss (91 dB or greater)')ax1.legend()
# Save audiogram chart as image
audiogram_fig.savefig(f'./results_{ear}_{now:%Y%m%d%H%M%S}_audiogram.png')
# Display audiogram chart in a new window
audiogram_window = tk.Tk()
audiogram_window.title(f"Audiogram for {ear} ear")
canvas = FigureCanvasTkAgg(audiogram_fig, master=audiogram_window)
canvas.draw()
canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
# Display Excel table in a new window
excel_window = tk.Tk()
excel_window.title(f"Portable Self Assessment Audiometer - {ear} ear")
excel_table = tk.Frame(excel_window)
excel_table.grid(row=0, column=0, padx=10, pady=10)
table_label = tk.Label(excel_table, text="Portable Self Assessment Audiometer", font=("Arial", 16, "bold"))
table_label.grid(row=0, columnspan=5, sticky="w")
headers = ['Sl. No.', 'Pitch (Frequency Hz)', 'Hearing Level (Volume dB)', 'Hearing Loss Range']
for i, header in enumerate(headers):
col_label = tk.Label(excel_table, text=header, font=("Arial", 12, "bold"))
```

```
col_label.grid(row=1, column=i, padx=5, pady=5)
for i, row in df.iterrows():
sl_no = tk.Label(excel_table, text=i+1, font=("Arial", 12))
sl_no.grid(row=i + 2, column=0, padx=5, pady=5, sticky="w")
freq_label = tk.Label(excel_table, text=row['frequency'], font=("Arial", 12))
freq_label.grid(row=i + 2, column=1, padx=5, pady=5)
vol_label = tk.Label(excel_table, text=row['volume'], font=("Arial", 12))
vol_label.grid(row=i + 2, column=2, padx=5, pady=5)
range_label = tk.Label(excel_table, text=self.get_hearing_loss_range(row['volume']), font=("Arial", 12))
range_label.grid(row=i + 2, column=3, padx=5, pady=5)
excel_window.mainloop()
# Create CSV file
df.to_csv(f'./results_{ear}_{now:%Y%m%d%H%M%S}.csv', index=None)
# Create Excel sheet
df_excel = pd.DataFrame(data, columns=['Sl. No.', 'Pitch (Frequency Hz)', 'Hearing Level (Volume dB)', 'Hearing Loss Range'])
df_excel['Hearing Loss Range'] = df_excel['Hearing Level (Volume dB)'].apply(self.get_hearing_loss_range)
df_excel.to_excel(f'./results_{ear}_{now:%Y%m%d%H%M%S}.xlsx', index=None)
print("Audiogram chart, CSV file, and Excel sheet created successfully.")
return df
def get_hearing_loss_range(self, volume):
"""Determines the hearing loss range based on volume level"""
if volume <= 15:
return 'Normal Hearing (0-15 dB)'
elif volume <= 25:
return 'Slight Hearing Loss (16-25 dB)'
elif volume <= 40:
return 'Mild Hearing Loss (26-40 dB)'
elif volume <= 55:
return 'Moderate Hearing Loss (41-55 dB)'
elif volume <= 70:
return 'Moderately Severe Hearing Loss (56-70 dB)'
```

```
elif volume <= 90:
return 'Severe Hearing Loss (71-90 dB)'
else:
return 'Profound Hearing Loss (91 dB or greater)'
def run_test(self):
parser = argparse.ArgumentParser()
parser.add_argument('-r', '--repeat', help='Number of times each frequency is repeated', type=int, default=1) # Change default value to 1
args = parser.parse_args()
self.start_time = datetime.now()
p = pyaudio.PyAudio()
# Start listener
p2 = threading.Thread(target=self.listener, daemon=True)
p2.start()
# Run test for the right ear
print('Testing right ear...')
self.player(p, repeat=args.repeat, ear='right')
# Analyse and visualize results for the right ear
right_df = self.analyse_results(self.right_data, 'right')
self.right_data = []
print('Test is finished. Please check visualizations and files.')
self.display_date_time_duration()
def display_date_time_duration(self):
now = datetime.now()
duration = now - self.start_time
# Display test information in a new window
info_window = tk.Tk()
info_window.title("Test Information")
date_label = tk.Label(info_window, text=f"Date: {self.start_time.strftime('%Y-%m-%d')}", font=("Arial", 12))
date_label.pack()
start_time_label = tk.Label(info_window, text=f"Start Time: {self.start_time.strftime('%H:%M:%S')}", font=("Arial", 12))
start_time_label.pack()
```

```
duration_label = tk.Label(info_window, text=f"Duration: {duration}", font=("Arial", 12))
duration_label.pack()
info_window.mainloop()
if __name__ == '__main__':
test = HearingTest()
test.start_test()
```