PART 1

import random

import tkinter as tk

from tkinter import messagebox

class EyeTestApp:

def init (self, root):

self.root = root

self.root.title("AI Healthcare - Eye Test") self.root.geometry("400x400")

self.test\_step = 0

self.previous\_left\_power = None self.previous\_right\_power = None self.test\_left\_eye\_result = None self.test\_right\_eye\_result = None self.test\_left\_eye\_power = None self.test\_right\_eye\_power = None self.eye = None

self.attempts = 0 # Counter for attempts at the current size self.welcome\_message()

def welcome\_message(self):

welcome\_label = tk.Label(self.root, text="Healthcare\nEye-Check up", font=("Helvetica", 16)) welcome\_label.pack(pady=20)

start\_button = tk.Button(self.root, text="Start Test", command=self.start\_test) start\_button.pack(pady=10)

def start\_test(self):

self.clear\_screen()

self.display\_message("Is this your first visit? (Yes/No)") self.create\_buttons(self.process\_visit\_response)

def clear\_screen(self):

for widget in self.root.winfo\_children():

widget.destroy()

def display\_message(self, message):

message\_label = tk.Label(self.root, text=message, font=("Helvetica", 12)) message\_label.pack(pady=10)

def create\_buttons(self, callback): button\_frame = tk.Frame(self.root) button\_frame.pack(pady=10)

yes\_button = tk.Button(button\_frame, text="Yes", command=lambda: callback("yes")) yes\_button.pack(side=tk.LEFT, padx=10)

no\_button = tk.Button(button\_frame, text="No", command=lambda: callback("no")) no\_button.pack(side=tk.LEFT, padx=10)

def process\_visit\_response(self, response):

if response.lower() == 'yes':

self.clear\_screen()

self.display\_message("Let's start your eye test. Close your right eye and try to read the numbers.")

self.test\_eye("left")

elif response.lower() == 'no': self.clear\_screen()

self.ask\_previous\_power("left") else:

messagebox.showinfo("Invalid Response", "Please click 'Yes' or 'No'.") self.start\_test()

def ask\_previous\_power(self, eye):

self.display\_message(f"Do you have power in your {eye} eye? (Yes/No)")

self.create\_buttons(lambda response: self.process\_power\_response(response, eye))

def process\_power\_response(self, response, eye):

if response.lower() == 'yes':

self.display\_message(f"Please enter the previously detected power for your {eye} eye:") self.create\_input\_field(eye)

elif response.lower() == 'no':

self.display\_message(f"Great! No power detected in your {eye} eye.") if eye == "left":

self.ask\_previous\_power("right") else:

self.clear\_screen()

self.display\_message("Now, let's start testing your eyes.") self.test\_eye("left")

else:

messagebox.showinfo("Invalid Response", "Please click 'Yes' or 'No'.") self.ask\_previous\_power(eye)

def create\_input\_field(self, eye):

self.entry = tk.Entry(self.root) self.entry.pack(pady=10)

submit\_button = tk.Button(self.root, text="Submit", command=lambda: self.store\_power(eye)) submit\_button.pack(pady=5)

def store\_power(self, eye): power = self.entry.get() try:

power = float(power) if eye == "left":

self.previous\_left\_power = power self.ask\_previous\_power("right")

elif eye == "right":

self.previous\_right\_power = power self.clear\_screen()

self.display\_message("Now, let's start testing your eyes.") self.test\_eye("left")

except ValueError:

messagebox.showinfo("Invalid Input", "Please enter a valid number for the power.") self.create\_input\_field(eye)

def test\_eye(self, eye):

self.test\_step = 30 # Starting from the largest text size self.eye = eye

self.attempts = 0 # Reset attempts for the new eye self.display\_numbers()

def display\_numbers(self):

random\_numbers = random.sample(range(10), 5) random\_numbers\_str = ''.join(map(str, random\_numbers))

number\_label = tk.Label(self.root, text=random\_numbers\_str, font=("Helvetica", self.test\_step)) number\_label.pack(pady=20)

self.create\_buttons(self.process\_test\_response) def process\_test\_response(self, response):

if response.lower() == 'yes':

self.attempts = 0 # Reset attempts if the user succeeds

if self.test\_step <= 14: # If they can read the smallest size, their vision is perfect self.clear\_screen()

self.display\_message(f"Your {self.eye} eye is healthy!") self.store\_test\_result(self.eye, "healthy")

self.finalize\_test() else:

self.test\_step -= 4 # Decrease font size for the next test step self.clear\_screen()

self.display\_message("Good! Let's move to the next level.") self.display\_numbers()

elif response.lower() == 'no': self.attempts += 1

if self.attempts == 1:

# Power is higher if the user fails on the first try power = round(6.0 - (self.test\_step / 6.0), 2)

else:

# Power decreases (is less severe) as the user gets more attempts

power = round(6.0 - (self.test\_step / 6.0) + (0.1 \* self.attempts), 2)

if self.attempts >= 3: # User fails after 3 tries self.clear\_screen()

self.display\_message(f"Looks like you have a vision problem in your {self.eye} eye. Detected power: {power}")

self.store\_test\_result(self.eye, "vision problem", power) self.finalize\_test()

else:

self.clear\_screen()

self.display\_message(f"Let's try again. Current estimated power: {power}") self.display\_numbers()

else:

messagebox.showinfo("Invalid Input", "Please click 'Yes' or 'No'.") self.display\_numbers()

def store\_test\_result(self, eye, result, power=None):

if eye == "left":

self.test\_left\_eye\_result = result self.test\_left\_eye\_power = power

elif eye == "right":

self.test\_right\_eye\_result = result self.test\_right\_eye\_power = power

def finalize\_test(self):

if self.eye == "left":

self.eye = "right"

self.display\_message("Now, let's start testing your right eye.") self.test\_eye("right")

elif self.eye == "right": self.clear\_screen() self.display\_results()

def display\_results(self):

# Show the result for the left eye

if self.test\_left\_eye\_result == "healthy":

self.display\_message("Your left eye is healthy!") else:

self.display\_message(f"Your left eye has a vision problem. Detected power:

{self.test\_left\_eye\_power}")

if self.previous\_left\_power is not None:

if self.test\_left\_eye\_power > self.previous\_left\_power: self.display\_message("Left eye power has increased.")

elif self.test\_left\_eye\_power < self.previous\_left\_power: self.display\_message("Left eye power has decreased.")

else:

self.display\_message("Left eye power remains the same.")

# Show the result for the right eye

if self.test\_right\_eye\_result == "healthy":

self.display\_message("Your right eye is healthy!") else:

self.display\_message(f"Your right eye has a vision problem. Detected power:

{self.test\_right\_eye\_power}")

if self.previous\_right\_power is not None:

if self.test\_right\_eye\_power > self.previous\_right\_power: self.display\_message("Right eye power has increased.")

elif self.test\_right\_eye\_power < self.previous\_right\_power: self.display\_message("Right eye power has decreased.")

else:

self.display\_message("Right eye power remains the same.")

# Finalize

self.display\_message("Test completed!")

def finish\_test(self, response):

messagebox.showinfo("Test Completed", "Thank you for taking the test.") self.root.quit()

def run\_app():

root = tk.Tk()

app = EyeTestApp(root) root.mainloop()

if name == " main ": run\_app()

PART 2

import tkinter as tk

from tkinter import ttk, messagebox, simpledialog import random

# Diagnosis & Treatment Data irritation = {

'0001': 'Mast cell stabilizers: Alomide, Crolom, Alocril', '0010': 'Flurbiren Eye Drop',

'0011': 'Anti-inflammatory eye drops, and corticosteroid eye drops', '0100': 'Decongestant eye drops',

'0101': 'Visine Allergy Eye Relief',

'0110': 'Pheniramine maleate/naphazoline HCL',

'0111': 'Antazoline phosphate/naphazoline HCL (Vasocon-A)', '1000': 'Blink GelTears',

'1001': 'Artificial tears and antihistamine eye drops can help relieve dry eyes and itching.', '1010': 'Artificial tears, warm compresses, avoiding environmental triggers.',

'1011': 'Artificial tears, warm compresses, screen breaks, and avoiding irritants.', '1100': 'Artificial tears, eyelid hygiene, warm compresses to unclog oil glands.', '1101': 'Lubricating eye drops, environmental management, omega-3 fatty acids.', '1110': 'Artificial tears for dry eye syndrome caused by poor-quality tears.', '1111': 'Artificial tears, environmental management, and screen breaks.'

}

treatment\_plan = {

"pterygium": "Lubricating drops, anti-inflammatory drops. Surgery if vision is affected.", "orbital myositis": "Steroids, NSAIDs. Surgery rarely needed.",

"thyroid eye disease": "Steroids, radiation, immunosuppressants. Orbital decompression surgery if severe.",

"tumor": "Chemotherapy, radiation, or targeted therapy. Surgery if tumor is accessible.", "cyst": "Observation or excision if causing discomfort.",

"conjunctivitis": "Antibiotic or antihistamine eye drops, warm compress.", "orbital cellulitis": "Urgent IV antibiotics, possible surgical drainage.",

"allergic reaction": "Antihistamines, cold compress, avoid allergens.",

"strabismus": "Prism glasses, vision therapy, or eye muscle surgery if needed.", "cranial nerve palsy": "Observation, steroids, or surgery if persistent.",

"myasthenia gravis": "Acetylcholinesterase inhibitors, immunosuppressants, possible thymectomy.",

"stroke": "Emergency treatment, rehabilitation, and blood pressure management."

}

# Function to remove previous frame def show\_frame(selected\_problem):

for frame in frames.values():

frame.pack\_forget()

frames[selected\_problem].pack(pady=10)

# Function to diagnose "Irritation" def diagnose\_irritation():

p = [str(dry\_var.get()), str(water\_var.get()), str(pain\_var.get()), str(itching\_var.get())] p\_code = ''.join(p)

messagebox.showinfo("Suggestion", f"Suggestion: {irritation.get(p\_code, 'No specific suggestion found.')}")

# Function to diagnose "Red Eye" def diagnose\_red\_eye():

if itching\_var\_red.get() and not discharge\_var.get():

diagnosis, treatment = "Allergic Conjunctivitis", "Use antihistamine eye drops and avoid allergens."

elif discharge\_var.get():

diagnosis, treatment = "Infectious Conjunctivitis (Pink Eye)", "Use antibiotic eye drops if bacterial; for viral, apply warm compress and maintain hygiene."

elif pain\_var\_red.get() and vision\_var.get() and light\_var.get():

diagnosis, treatment = "Uveitis", "Urgent referral to an ophthalmologist for steroid eye drops." elif pain\_var\_red.get() and vision\_var.get():

diagnosis, treatment = "Acute Glaucoma", "Seek emergency care to lower eye pressure." elif injury\_var.get():

diagnosis, treatment = "Eye Trauma", "Rinse with clean water and seek immediate care." else:

diagnosis, treatment = "Dry Eye Syndrome", "Use artificial tears and rest the eyes."

messagebox.showinfo("Diagnosis", f"Possible Diagnosis: {diagnosis}\nSuggested Treatment:

{treatment}")

# Function to diagnose "Muscle Growth" & "Swelling" def diagnose\_condition(var):

condition = var.get()

messagebox.showinfo("Diagnosis", f"Condition: {condition}\nTreatment:

{treatment\_plan.get(condition, 'No specific treatment found.')}")

# Function to diagnose "Double Vision" def diagnose\_double\_vision():

condition = double\_vision\_condition\_var.get()

messagebox.showinfo("Diagnosis", f"Condition: {condition}\nTreatment:

{treatment\_plan.get(condition, 'No specific treatment found.')}")

# Function to diagnose "Color Blindness" def diagnose\_color\_blindness():

messagebox.showinfo("Diagnosis", "Diagnosis: Color Blindness Test In Progress\nSuggested Treatment: Refer to real-life tests provided.")

# Function for real-life color arrangement test def color\_arrangement\_test():

correct\_order = ["Red", "Green", "Blue", "Yellow"] shufled\_order = correct\_order[:]

random.shufle(shufled\_order) # Shufle the color order for clicking user\_order = []

def add\_color(color):

user\_order.append(color)

if len(user\_order) == len(correct\_order):

if user\_order == correct\_order:

messagebox.showinfo("Result", "Correct! Your color arrangement appears normal.") else:

messagebox.showinfo("Result", "Incorrect. This might indicate a color vision deficiency.")

test\_window = tk.Toplevel(root)

test\_window.title("Color Arrangement Test")

tk.Label(test\_window, text="Click the colors in the correct order: Red, Green, Blue, Yellow").pack(pady=10)

for color in shufled\_order:

ttk.Button(test\_window, text=color, command=lambda c=color: add\_color(c)).pack(pady=5)

# Function for real-life number recognition test def number\_recognition\_test():

correct\_answers = ["25", "42", "7"] # Updated: Removed "nothing" option user\_answers = []

current\_index = [0] # To track the current question

def check\_answer():

answer = answer\_entry.get().strip().lower() user\_answers.append(answer)

if len(user\_answers) == len(correct\_answers):

if all(user\_answers[i] == correct\_answers[i].lower() for i in range(len(correct\_answers))): messagebox.showinfo("Result", "Correct! Your number recognition appears normal.")

else:

messagebox.showinfo("Result", "Incorrect. This might indicate a color vision deficiency.") test\_window.destroy() # Close the test window after completing the test

else:

current\_index[0] += 1 update\_question()

def update\_question():

number\_var.set(f"Question {current\_index[0] + 1}: Identify the number in the pattern") number\_label.config(text=f"Pattern: {correct\_answers[current\_index[0]]}") # Display the correct

pattern text

answer\_entry.delete(0, tk.END)

test\_window = tk.Toplevel(root)

test\_window.title("Number Recognition Test")

number\_var = tk.StringVar(value="Question 1: Identify the number in the pattern") tk.Label(test\_window, textvariable=number\_var).pack(pady=10)

number\_label = tk.Label(test\_window, text="Pattern: 25") # Initialize the first pattern display number\_label.pack(pady=10)

answer\_entry = ttk.Entry(test\_window, width=20) answer\_entry.pack(pady=5)

ttk.Button(test\_window, text="Submit", command=check\_answer).pack(pady=10)

# GUI Setup root = tk.Tk()

root.title("Eye Health Diagnosis") root.geometry("400x700")

tk.Label(root, text="Suggestions for Eye problems", font=("Arial", 14, "bold")).pack(pady=10) problems = ["Irritation", "Red Eye", "Muscle Growth", "Swelling of Eye", "Double Vision", "Color Blindness"]

problem\_var = tk.StringVar(value="Select a problem")

ttk.Combobox(root, textvariable=problem\_var, values=problems, state="readonly", width=30).pack(pady=10)

ttk.Button(root, text="Diagnose", command=lambda:

show\_frame(problem\_var.get())).pack(pady=10) frames = {}

# Irritation Frame

symptom\_frame = tk.Frame(root)

dry\_var, water\_var, pain\_var, itching\_var = tk.IntVar(), tk.IntVar(), tk.IntVar(), tk.IntVar()

for text, var in [("Dry Eyes", dry\_var), ("Frequent Watering", water\_var), ("Eye Pain", pain\_var), ("Itching", itching\_var)]:

tk.Checkbutton(symptom\_frame, text=text, variable=var).pack(anchor="w") ttk.Button(symptom\_frame, text="Get Suggestion", command=diagnose\_irritation).pack(pady=10) frames["Irritation"] = symptom\_frame

# Red Eye Frame

red\_eye\_frame = tk.Frame(root)

pain\_var\_red, discharge\_var, itching\_var\_red, vision\_var, light\_var, injury\_var = tk.BooleanVar(), tk.BooleanVar(), tk.BooleanVar(), tk.BooleanVar(), tk.BooleanVar(), tk.BooleanVar()

for text, var in [("Pain?", pain\_var\_red), ("Discharge?", discharge\_var), ("Itching?", itching\_var\_red), ("Blurry Vision?", vision\_var), ("Light Sensitivity?", light\_var), ("Injury?", injury\_var)]:

tk.Checkbutton(red\_eye\_frame, text=text, variable=var).pack(anchor="w")

ttk.Button(red\_eye\_frame, text="Get Diagnosis", command=diagnose\_red\_eye).pack(pady=10) frames["Red Eye"] = red\_eye\_frame

# Muscle Growth & Swelling Frames

for condition in ["Muscle Growth", "Swelling of Eye"]: frame = tk.Frame(root)

var = tk.StringVar()

ttk.Combobox(frame, textvariable=var, values=list(treatment\_plan.keys()), state="readonly", width=30).pack()

ttk.Button(frame, text="Get Diagnosis", command=lambda v=var: diagnose\_condition(v)).pack(pady=10)

frames[condition] = frame

# Double Vision Frame

double\_vision\_frame = tk.Frame(root)

double\_vision\_condition\_var = tk.StringVar()

ttk.Combobox(double\_vision\_frame, textvariable=double\_vision\_condition\_var, values=list(treatment\_plan.keys()), state="readonly", width=30).pack()

ttk.Button(double\_vision\_frame, text="Get Diagnosis", command=diagnose\_double\_vision).pack(pady=10)

frames["Double Vision"] = double\_vision\_frame

# Color Blindness Frame

color\_blindness\_frame = tk.Frame(root)

ttk.Button(color\_blindness\_frame, text="Real-life Color Arrangement Test", command=color\_arrangement\_test).pack(pady=10)

ttk.Button(color\_blindness\_frame, text="Real-life Number Recognition Test", command=number\_recognition\_test).pack(pady=10)

ttk.Button(color\_blindness\_frame, text="Get Diagnosis", command=diagnose\_color\_blindness).pack(pady=10)

frames["Color Blindness"] = color\_blindness\_frame

# Run the main GUI loop root.mainloop()