Universitatea Națională de Știință și Tehnologie Politehnica din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Proiect Python

- Fitness App -

Nume student: Andreea Sîrbu
Grupa: 421A

2023

1. Descriere Generală

Aplicația de fitness este concepută pentru a ajuta utilizatorii să-și monitorizeze și să-și îmbunătățească starea de sănătate și forma fizică. Aceasta oferă funcționalități precum redarea și oprirea unei melodii de antrenament, calculul BMI (Indexul de Masă Corporală), aportul caloric zilnic, generarea unei rutine de antrenament personalizate, sfaturi de fitness zilnice, și provocări fitness săptămânale. De asemenea, include opțiuni pentru notificări zilnice, mesaje motivaționale și provocări săptămânale.

2. Tehnologii utilizate

În acest proiect am utilizat mai multe tehnologii si module Python: Tkinter, PIL, Random, pygame.

- **tkinter**: Bibliotecă pentru crearea interfețelor grafice.
- **ttk**: Subset de widget-uri îmbunătățite în Tkinter.
- **messagebox**: Componentă a Tkinter pentru afișarea de mesaje pop-up.
- **random**: Folosit pentru generarea de valori aleatoare.
- **PIL.Image, PIL.ImageTk**: Bibliotecă pentru manipularea imaginilor.
- **pygame**: Bibliotecă pentru redarea muzicii.

Aceste module și biblioteci sunt folosite pentru a construi o aplicație de fitness cu o interfață grafică interactivă în care utilizatorii pot introduce datele lor personale pentru a primi informații legate de fitness, rutine de antrenament personalizate, sfaturi și provocări zilnice.

3. Pentru a rula aplicația, recomand următorii pași de instalare:

1. Instalare Python

Asigură-te că Python este instalat pe sistemul tău. Poți descărca Python de pe site-ul oficial Python. În timpul instalării, bifează opțiunea "Add Python to PATH" pentru a facilita rularea scripturilor Python din terminal sau linia de comandă.

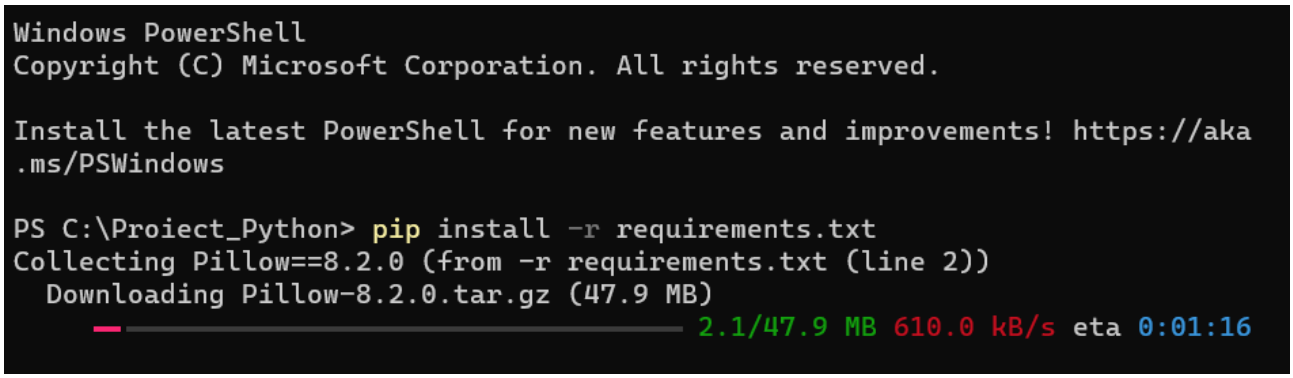
2. Descărcare Cod Sursă

3. Deschidere Terminal / Linie de Comandă

Deschide un terminal sau o linie de comandă în directorul unde ai salvat codul sursă al aplicației.

5. Instalare Dependințe:

Instalează dependințele proiectului folosind pip. În terminal, ruleaza:



6. Rulare Aplicație

Rularea aplicației se face prin intermediul terminalului sau liniei de comandă. Asigură-te că te afli în directorul aplicației și rulează:

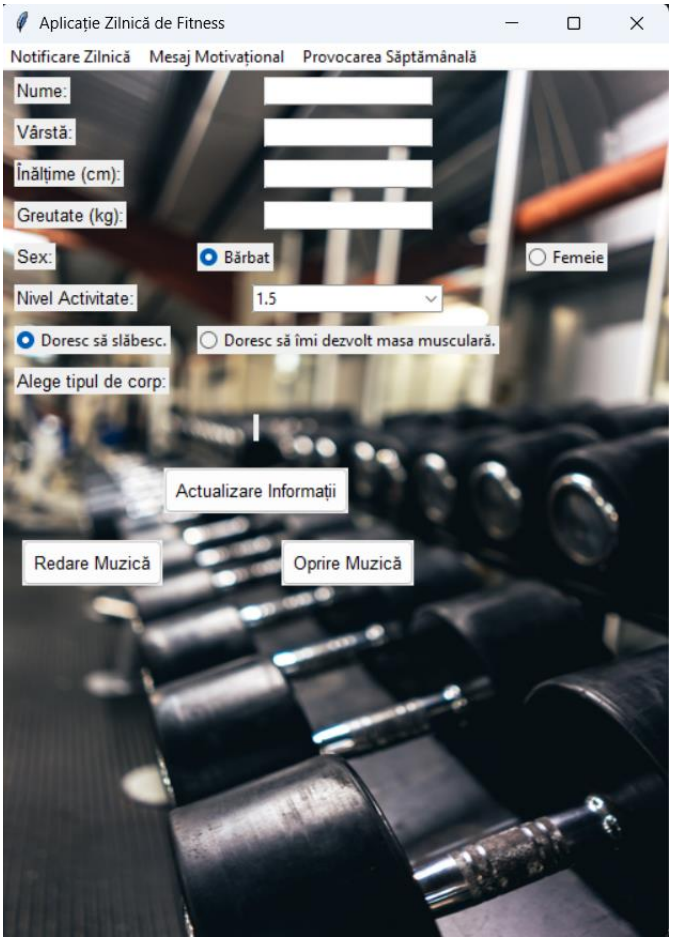
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Proiect_Python> python proiectPy.py
C:\Proiect_Python\proiectPy.py:10: SyntaxWarning: invalid escape sequence '\P'
  pygame.mixer.music.load("C:\Proiect_Python\GYM_music.mp3")
C:\Proiect_Python\proiectPy.py:167: SyntaxWarning: invalid escape sequence '\P'
  background_image_path = "C:\Proiect_Python\poza_proiectPy2.jpg"
pygame 2.5.2 (SDL 2.28.3, Python 3.12.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

4. Descrierea aplicației

3.1 Interfața



Pagina principală a interfaței oferă utilizatorilor posibilitatea de a introduce informații personale pentru a primi estimări legate de fitness, rutine de antrenament zilnice, sfaturi și provocări pentru a îi ajuta în progresul lor zilnic. Interfața grafică este creată folosind Tkinter, permițând o interacțiune ușoară și intuitivă pentru utilizatori.

Interfața grafică cuprinde următoarele elemente:

1. **Câmpuri de intrare:** Zone de text sau casete de introducere pentru nume, vârstă, înălțime, greutate etc.
2. **Etichete:** Zone de text pentru a afișa instrucțiuni sau informații.
3. **Butoane:** Pentru a activa anumite acțiuni sau pentru a actualiza informațiile.
4. **Combobox-uri și radiobutoane:** Pentru a permite utilizatorilor să selecteze din opțiuni prestabilite sau exclusive.
5. **Afisaj pentru rezultate:** O zonă de afișare pentru rezultatele calculatoarelor de fitness, sfaturi zilnice, rutina de antrenament etc.
6. **Elemente vizuale suplimentare:** Ca imagine de fundal sau orice alte imagini utilizate în aplicație.
7. **Meniu:** Oferă opțiuni suplimentare pentru notificări, mesaje motivationale etc.

Pentru a furniza o experiență mai plăcută și coerentă pentru utilizatori, elementele interfeței grafice sunt plasate într-o manieră estetică și funcțională în cadrul ferestrei principale.

3.2 Prezentare funcționalități

Aplicația are o funcționalitate destul de simplă, utilizatorul trebuie să urmeze doar câțiva pași:

Crearea și Actualizarea Profilului

- **Introducerea Datelor Personale:** Utilizatorii pot introduce informații precum nume, vârstă, înălțime, greutate și gen.
- **Selectarea Obiectivelor Fitness:** Utilizatorii pot alege între opțiuni precum slăbire sau creștere în masă musculară etc.
- **Ajustarea Nivelului de Activitate:** Posibilitatea de a selecta nivelul de activitate pentru a calcula necesarul zilnic de calorii.

În mod specific, în codul furnizat se utilizează biblioteca **Tkinter** pentru a crea interfața grafică. Sunt utilizate diferite widget-uri (**Entry**, **Label**, **Button**, **Radiobutton**, **Combobox**, **Menu**, etc.) pentru a construi elementele menționate mai sus și pentru a le organiza în fereastra principală a aplicației.

Eticheta **Label** Reprezintă text static sau informativ în interfața grafică și este folosită pentru a descrie sau eticheta câmpurile de intrare sau alte componente.

Exemplu: **varsta_label = ttk.Label(root, text="Vârstă:")**

Prin folosirea widget-ului **Entry**, i se permite utilizatorului să introduca un text de la tastatură.

Exemplu: **nume_intrare = ttk.Entry(root)**

Button permite utilizatorilor să interacționeze și să activeze anumite acțiuni sau funcționalități.În codul dat, sunt utilizate pentru a crea butoane pentru redarea și oprirea muzicii, actualizarea informațiilor etc.

Exemplu: **buton_redare_muzica=ttk.Button(root, text="Redare Muzică", command=reda_muzica)**

Radiobutton permite utilizatorilor să selecteze o opțiune dintr-o listă de opțiuni exclusive.În acest cod, sunt utilizate pentru a permite utilizatorilor să aleagă între diferite opțiuni, cum ar fi obiectivele de fitness (slăbire, creștere în masă musculară etc.).

Exemplu: **scadere_greutate_checkbox = ttk.Radiobutton(root, text="Doresc să slăbesc", variable=gen1_var, value='Doresc să slăbesc')**

Combobox oferă o listă derulantă de opțiuni din care utilizatorii pot selecta una .Sunt folosite pentru a permite utilizatorilor să selecteze nivelul de activitate sau alte opțiuni dintr-o listă predefinită.

Exemplu:**nivel_activitate_combobox=ttk.Combobox(root, values=optiuni_nivel_activitate)**

Menu sunt folosite pentru a crea meniuri de navigare sau pentru a oferi acces la funcționalități suplimentare.În codul dat, sunt utilizate pentru a oferi opțiuni suplimentare, cum ar fi notificări zilnice, mesaje motivaționale etc.

Exemplu: **meniu_nou = tk.Menu(root)**

Pentru a genera informatii referitoare la datele introduse de utilizator vom avea nevoie de calculul anumitor funcții:

- **Calculul BMI (Indexul de Masă Corporală):** Bazat pe înălțimea și greutatea introduse. Acest calcul este introdus prin functia **calculeaza_bmi**.

```python
# Funcție pentru calculul BMI
def calculeaza_bmi(profil_utilizator):
    inaltime_in_metri = profil_utilizator['inaltime'] / 100
    bmi = profil_utilizator['greutate'] / (inaltime_in_metri ** 2)
    return round(bmi, 2)
```

- **Calculul Aportului Caloric Zilnic:** Această funcție este esențială în determinarea cantității de calorii necesare pentru a atinge diverse obiective legate de fitness, cum ar fi pierderea în greutate sau creșterea masei musculare, luând în considerare profilul individual al utilizatorului.

```python
# Funcție pentru calculul aportului caloric zilnic
def calculeaza_calorii(profil_utilizator):
    bmi = calculeaza_bmi(profil_utilizator)
    calorii = 0

    if profil_utilizator['gen'] == 'barbat':
        calorii = 88.362 + (13.397 * profil_utilizator['greutate']) + \
                  (4.799 * profil_utilizator['inaltime']) - (5.677 * bmi)
    else:
        calorii = 447.593 + (9.247 * profil_utilizator['greutate']) + \
                  (3.098 * profil_utilizator['inaltime']) - (4.330 * bmi)

    # Ține cont de nivelul de activitate
    calorii *= profil_utilizator['nivel_activitate']

    if gen1_var.get() == 'Doresc să slăbesc.':
        calorii -= 500  # Reducerea cu 500 de calorii zilnic pentru a slăbi în jur de 0.5 kg/săptămână
    elif gen1_var.get() == 'Doresc să îmi dezvolt masa musculară.':
        calorii += 300  # Adăugarea a 300 de calorii zilnic pentru creșterea masei musculare
    return round(calorii)
```

- **Generarea Rutinei de Antrenament:** Această funcție, **genereaza_rutina_antrenament**, este responsabilă de generarea unei rutine zilnice de antrenament în funcție de tipul de corp ales și de obiectivul de fitness al utilizatorului (dacă acesta dorește să slăbească sau nu).

```
def genereaza_rutina_antrenament(tip_corp_ales, vreau_sa_slabesc):
    rutina_antrenament = []

    if vreau_sa_slabesc:
        rutina_antrenament.extend([
            'Cardio - 30 de minute de alergare',
            'Skipping - 3 seturi de 50 de repetări',
            'Plank - 3 seturi de 45 de secunde',
            'Yoga sau Pilates - 1 oră',
        ])
    else:
        if "athlete" in tip_corp_ales:
            rutina_antrenament.extend([
                'Flotări - 3 seturi de 12 repetări',
                'Squat-uri - 3 seturi de 12 repetări',
                'Jumping Jacks - 3 seturi de 30 de secunde',
                'Plank - 3 seturi de 60 de secunde',
            ])
        if "hero" in tip_corp_ales:
            rutina_antrenament.extend([
                'Deadlifts - 3 seturi de 10 repetări',
                'Bench Press - 3 seturi de 10 repetări',
                'Pull-ups - 3 seturi de 8 repetări',
                'Leg Press - 3 seturi de 12 repetări',
            ])
        if "bodybuilder" in tip_corp_ales:
            rutina_antrenament.extend([
                'Barbell Curls - 4 seturi de 8 repetări',
                'Triceps Dips - 4 seturi de 10 repetări',
                'Lateral Raises - 4 seturi de 12 repetări',
                'Hammer Curls - 4 seturi de 10 repetări',
            ])

    return rutina_antrenament
```

Determinarea rutinei în funcție de obiectivul utilizatorului: Dacă utilizatorul dorește să slăbească (**vreau_sa_slabesc == True**), se generează o rutină axată pe activități cardio și antrenament de bază, care ar putea ajuta la arderea caloriilor și îmbunătățirea tonusului muscular.

Generarea rutinei în funcție de tipul de corp: În cazul în care utilizatorul nu dorește să slăbească, rutina se personalizează în funcție de tipul de corp selectat (**tip_corp_ales**).

Pentru fiecare tip de corp (**"athlete"**, **"hero"**, **"bodybuilder"**), rutina include exerciții specifice care vizează dezvoltarea specifică a grupelor musculare asociate acelui tip de corp. De exemplu, un tip de corp **"athlete"** ar putea avea exerciții de bază și cardio, în timp ce un tip de corp **"hero"** ar include exerciții de ridicare a greutății și exerciții complexe pentru forță și rezistență.

Returnarea rutinei de antrenament generată: Funcția returnează o listă de exerciții formate în funcție de obiectivul utilizatorului și tipul de corp selectat.

Sfaturi și Motivare

- **Sfaturi Fitness Zilnice:** Sfaturi personalizate în funcție de tipul de corp și obiectivele utilizatorului.

```python
# Funcție pentru obținerea unui sfat de fitness zilnic
def obtine_sfat_fitness(tip_corp_ales, vreau_sa_slabesc):
    sfaturi = {
        "athlete": [
            'Bea multă apă pentru a rămâne hidratat.',
            'Găsește un partener de antrenament pentru a rămâne responsabil.',
            # Other advice for the "athlete" body type...
        ],
        "hero": [
            'Folosește greutăți mai mari pentru progres.',
            'Odihnește-te suficient pentru recuperare.',
            # Other advice for the "hero" body type...
        ],
        "bodybuilder": [
            'Concentrează-te pe izolare și contracția musculară.',
            'Mănâncă suficient pentru a susține creșterea musculară.',
            # Other advice for the "bodybuilder" body type...
        ]
    }

    if vreau_sa_slabesc:
        return "Fă exerciții cardio regulate și urmează un regim alimentar echilibrat, de deficit caloric."
    elif "Doresc să îmi dezvolt masa musculară." in tip_corp_ales:
        return "Folosește greutăți progresiv mai mari pentru antrenamentele de forță și mănâncă în surplus caloric pentru creșterea musculară."
    else:
        return random.choice(sfaturi.get(tip_corp_ales, ["Fă exerciții regulate și mănâncă sănătos."]))
```

Pentru fiecare tip de corp (**"athlete"**, **"hero"**, **"bodybuilder"**), sunt disponibile sfaturi specifice care se concentrează pe nevoile și obiectivele asociate acelui tip de corp. De exemplu, un "athlete" ar avea sfaturi legate de hidratare și responsabilitate, în timp ce un "hero" ar fi sfătuit să se odihnească suficient pentru recuperare.

Dacă utilizatorul dorește să slăbească (**vreau_sa_slabesc == True**), se furnizează sfaturi generale legate de exercițiile cardio și regimul alimentar echilibrat.În cazul în care utilizatorul dorește să își dezvolte masa musculară (**"Depunere masă musculară"** în **tip_corp_ales**), se oferă sfaturi specifice pentru antrenamentele de forță și nutriție.

Funcția returnează un sfat aleator din lista specifică tipului de corp ales sau sfaturi generale pentru obiectivele de fitness selectate. Dacă nu există sfaturi pentru tipul de corp ales, se returnează un sfat general pentru a face exerciții regulate și a mânca sănătos.

- **Provocări Fitness Zilnice/Săptămânale:** Sugestii pentru activități sau obiective fitness de atins.

```
def obtine_provocare_fitness():
    provocari = [
        'Completează 10 burpees în fiecare dimineață.',
        'Mergi 10.000 de pași în fiecare zi.',
        'Încearcă o nouă clasă de fitness în această săptămână.',
        'Înoată timp de 30 de minute în fiecare zi.',
    ]
    return random.choice(provocari)
```

Afișarea rutinei zilnice:

```
# Funcție pentru actualizarea afișajului cu informații de fitness zilnice
def actualizeaza_afisaj():
    profil_utilizator['nume'] = nume_intrare.get()

    profil_utilizator['varsta'] = int(varsta_intrare.get())
    profil_utilizator['inaltime'] = int(inaltime_intrare.get())
    profil_utilizator['greutate'] = int(greutate_intrare.get())
    profil_utilizator['gen'] = gen_var.get()
    profil_utilizator['nivel_activitate'] = float(nivel_activitate_var.get())
    tip_corp_ales = tip_corp_var.get()
    vreau_sa_slabesc = gen1_var.get() == 'Doresc să slăbesc'
    sfat_fitness = obtine_sfat_fitness(tip_corp_ales, vreau_sa_slabesc)
    # Obține tipul de corp ales și dacă utilizatorul dorește să slăbească
    tip_corp_ales = tip_corp_var.get()
    vreau_sa_slabesc = gen1_var.get() == 'Doresc să slăbesc'
    # Obține rutina de antrenament în funcție de tipul de corp și obiectivul utilizatorului
    rutina_antrenament = genereaza_rutina_antrenament(tip_corp_ales, vreau_sa_slabesc)
    rezultat = (
        f"Salut, {profil_utilizator['nume']}!\n\n"
        "🍽️ Aport Caloric Zilnic: {}\n\n"
        f"🏋️ BMI: {calculeaza_bmi(profil_utilizator)}\n\n"
        "💪 Rutina de Antrenament:\n{rutina}\n\n"
        "💡 Sfat de Fitness Zilnic: {sfat}\n\n"
        "⏳ Provocare de Fitness Zilnică: {provocare}"
    ).format(
        calculeaza_calorii(profil_utilizator),
        rutina='\n'.join(rutina_antrenament),
        sfat=sfat_fitness,
        provocare=obtine_provocare_fitness()
    )
    rezultat_text.set(rezultat)
```

Se obțin sfaturile de fitness (sfat_fitness) și rutina de antrenament (rutina_antrenament) în funcție de tipul de corp ales și de obiectivul utilizatorului. Sfaturile sunt obținute apelând funcția obtine_sfat_fitness și genereaza_rutina_antrenament. Se construiește un rezumat complet care va fi afișat în interfața grafică. Acest rezumat include informații cum ar fi numele utilizatorului, aportul caloric zilnic, BMI (Indexul de Masă Corporală), rutina de antrenament, sfatul zilnic de fitness și provocarea zilnică.

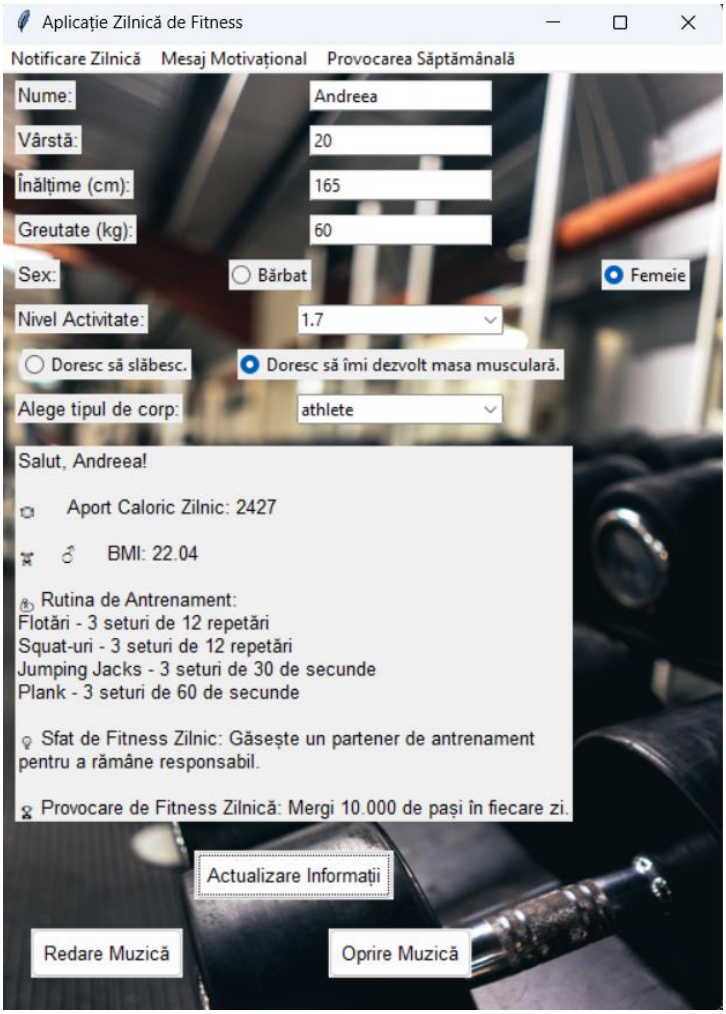
Utilizând f-string-uri (**f"..."**), valorile sunt integrate în șirul formatat.

Variabila rezultat_text este actualizată cu șirul formatat care conține toate informațiile colectate anterior.

Aceasta este o variabilă de legătură cu un widget de afișaj din interfața grafică, astfel încât atunci când se modifică această variabilă, widget-ul se actualizează automat.

rezultat_text.set(rezultat) actualizează conținutul widget-ului de afișaj pentru a reflecta noul rezumat construit anterior.

Această funcție este crucială pentru actualizarea afișajului în aplicație, reflectând informațiile și sfaturile personalizate pentru fiecare utilizator în funcție de datele introduse și obiectivele fitness ale acestuia.



Interacțiune Multimedia:

- **Redarea Muzicii:** Opțiunea de a reda muzică direct din aplicație pentru a motiva antrenamentele.

```
def reda_muzica():
    pygame.mixer.init()
    pygame.mixer.music.load("C:/Users/cucoa/Downloads/Best GYM Music  Best Workout Music  Best Trainings Music  NEFFEX  #neffex.mp3")
    pygame.mixer.music.set_volume(0.5)  # Ajustează volumul
    pygame.mixer.music.play(-1)  # -1 înseamnă că melodia va fi redată în mod repetat

def opreste_muzica():
    pygame.mixer.music.stop()
```

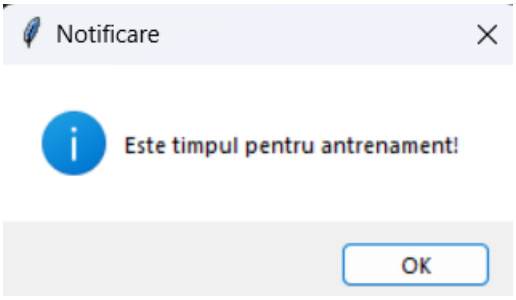
- **Mesaje Motivaționale sau Notificări Zilnice:** Afișarea de mesaje sau notificări pentru a menține utilizatorii motivați.

```
# Creare meniu pentru funcționalitățile noi
meniu_nou = tk.Menu(root)
root.config(menu=meniu_nou)

meniu_nou.add_command(label="Notificare Zilnică", command=lambda: afiseaza_notificare("Este timpul pentru antrenament!"))
meniu_nou.add_command(label="Mesaj Motivațional", command=afiseaza_mesaj_motivational)
meniu_nou.add_command(label="Provocare Săptămânală", command=afiseaza_provocare_saptamanala)
```

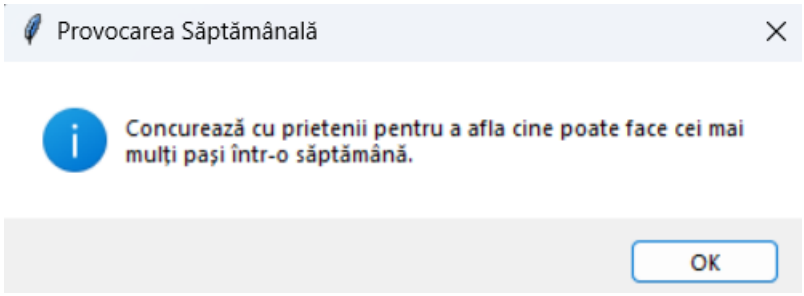
Pentru opțiunile adăugate:

- **"Notificare Zilnică"**: Atunci când această opțiune este selectată, se va apela funcția **afiseaza_notificare("Este timpul pentru antrenament!")**.



- **"Mesaj Motivațional"**: La selectarea acestei opțiuni, se va apela funcția **afiseaza_mesaj_motivational**.

- **"Provocare Săptămânală"**: La selectarea acestei opțiuni, se va apela funcția **afiseaza_provocare_saptamanala**.



Aceste opțiuni de meniu permit utilizatorilor să acceseze diverse funcționalități suplimentare ale aplicației, cum ar fi primirea de notificări zilnice, afișarea de mesaje motivaționale sau accesarea unei provocări săptămânale, în funcție de opțiunea aleasă.

5. Concluzie:

Aplicația de fitness reprezintă o soluție comprehensivă și personalizată pentru gestionarea și îmbunătățirea stării de sănătate a utilizatorilor.

<u>Puncte Cheie:</u>

1. Interfața Grafică Atractivă

- Utilizarea modulelor Tkinter și ttk pentru crearea unei interfețe intuitive și estetice.
- Adăugarea de imagini, inclusiv a imaginii de fundal, pentru personalizarea aspectului.

2. Gestionarea Profilului Utilizator:
   - Inițializarea și actualizarea profilului utilizatorului în funcție de informațiile introduse.

3. Funcționalități Diverse

   - Redare și oprire controlată a muzicii de antrenament.
   - Calculul BMI și aportului caloric zilnic.
   - Generarea de rutine de antrenament personalizate.

4. Sfaturi și Provocări:

   - Furnizarea de sfaturi de fitness zilnice personalizate.
   - Oferirea de provocări săptămânale pentru a menține utilizatorii motivați.

5. Notificări și Personalizare:

   - Implementarea notificărilor zilnice și a meniului adițional pentru funcționalități suplimentare.
   - Posibilitatea de a personaliza aspectul aplicației prin încărcarea imaginilor.

6. Referințe:

tkinter — Python interface to Tcl/Tk — Python 3.12.1 documentation

Python Tkinter Tutorial - GeeksforGeeks

tkinter.ttk — Tk themed widgets — Python 3.12.1 documentation

Pillow (PIL Fork) 10.2.0 documentation

GitHub - python-pillow/Pillow: Python Imaging Library (Fork)

Pygame Front Page — pygame v2.6.0 documentation

3.12.1 Documentation (python.org)

pygame · GitHub