CHALLENGE RUN ANALYZER PRO

“Menambahkan fitur untuk menyimpan histori analisis lari dari penggunasehingga ketika keluar dari aplikasinya, historinya tidak terhapus.**”**

Oleh:

Nama Kelompok : Kelompok 8

Nama/NIM : Denielson Javier Latuparissa / 25031554065

Kelas : 2025 G

Dosen : Hasanuddin Al-Habib, M.Si

Dr. Heri Purnawan, S.Si., M.Si

Mata Kuliah : Pemrograman Dasa

PROGRAM STUDI S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2025

Challenge yang di berikan:

Menambahkan fitur untuk menyimpan histori analisis lari dari pengguna sehingga ketika keluar dari aplikasinya, historinya tidak terhapus.

Code Lama:

```
import tkinter as tk
from tkinter import ttk, messagebox
from datetime import datetime, date


THEME = {
   "dark": {"bg":"#1e1e2e","frame":"#2d3047","card":"#3d405b","fg":"white"},
#KODE PENTING
   "light":{"bg":"#f4f4f4","frame":"#ffffff","card":"#e6e6e6","fg":"black"}
}


class RunningApp(tk.Tk):        #KODE PENTING (TAMPILAN UTAMA)
   def __init__(self):
      super().__init__()
      self.title("Run Analyzer Pro")
      self.geometry("700x750")
      self.mode = "dark"
      self.vars = {x: tk.StringVar() for x in
["jarak","waktu","berat","target_jarak"]}
      self.history = {}
```

```
        self.daily_targets = {}
        self.daily_distances = {}
        self.make_gui()
        self.apply_theme()

    def make_gui(self):
        self.title_lbl = tk.Label(self, text="RUN ANALYZER PRO",
font=("Arial",18,"bold"))
        self.title_lbl.pack(pady=20)

        self.notebook = ttk.Notebook(self)
        self.notebook.pack(expand=True, fill="both", padx=20, pady=10)

        self.tabs = {}      #KODE PENTING (TAB FITUR)
        for name in ["Input","Hasil","Gizi","Jadwal","History"]:
            self.tabs[name] = ttk.Frame(self.notebook)
            self.notebook.add(self.tabs[name], text=name)

        self.make_input_tab()
        tk.Button(self, text="Toggle Theme",
command=self.toggle_theme).pack(pady=5)

    def apply_theme(self):
        t = THEME[self.mode]
```

```
        self.configure(bg=t["bg"])
        self.title_lbl.configure(bg=t["bg"], fg=t["fg"])
        for tab in self.tabs.values():
            for widget in tab.winfo_children():
                widget.destroy()
        self.make_input_tab()
        if hasattr(self, "pace"):
            self.show_all()

    def toggle_theme(self):
        self.mode = "light" if self.mode=="dark" else "dark"
        self.apply_theme()

    def make_input_tab(self):
        t = THEME[self.mode]
        f = tk.Frame(self.tabs["Input"], bg=t["frame"], padx=25, pady=25)
        f.pack(expand=True, fill="both")

        labels = ["Jarak (km)", "Waktu (menit)", "Berat (kg)", "Target Jarak Harian
(km)"]      #KODE PENTING (INPUT DATA)
        keys = ["jarak", "waktu", "berat", "target_jarak"]

        for label, key in zip(labels, keys):
```

```
            tk.Label(f, text=label, bg=t["frame"], fg=t["fg"]).pack(anchor="w",
pady=(5,0))
            tk.Entry(f, textvariable=self.vars[key], font=("Arial",12),
                     bg=t["card"], fg=t["fg"]).pack(fill="x", pady=3)

        tk.Button(f, text="Analisis", command=self.analyze,
                  bg="#ff6b6b", fg="white", font=("Arial",11,"bold"),
                  pady=8).pack(fill="x", pady=20)

    def analyze(self):
        try:
            j = float(self.vars["jarak"].get())     #KODE PENTING (PERHITUNGAN)
            w = float(self.vars["waktu"].get())
            b = float(self.vars["berat"].get())
            t = float(self.vars["target_jarak"].get()) if
self.vars["target_jarak"].get().strip() else None
            if min(j,w,b) <= 0: raise ValueError
            if t is not None and t <= 0: raise ValueError
        except:
            return messagebox.showerror("Error","Input tidak valid!")

        today = date.today().strftime("%Y-%m-%d")
        self.pace = w/j
        self.speed = (j/w)*60
```

```
        self.kal = j*b*0.653



        if t is not None:
            self.daily_targets[today] = t
            self.target_jarak = t
        else:

            if today in self.daily_targets:
                self.target_jarak = self.daily_targets[today]
            elif hasattr(self, 'target_jarak'):
                self.daily_targets[today] = self.target_jarak
            else:
                self.target_jarak = 0

        if today not in self.daily_distances:
            self.daily_distances[today] = 0
        self.daily_distances[today] += j



        total_jarak_hari_ini = self.daily_distances[today]      #KODE PENTING
(RIWAYAT HISTORY)

        if today not in self.history:
```

```
            self.history[today] = []

        self.history[today].append({
            "time": datetime.now().strftime("%H:%M"),
            "jarak": j,
            "waktu": w,
            "pace": self.pace,
            "speed": self.speed,
            "kal": self.kal,
            "target": self.target_jarak,
            "total_jarak_harian": total_jarak_hari_ini
        })

        self.show_all()
        self.notebook.select(1)

    def show_all(self):
        self.show_hasil()
        self.show_gizi()
        self.show_jadwal()
        self.show_history()

    def show_hasil(self):
```

```
t = THEME[self.mode]
tab = self.tabs["Hasil"]
for w in tab.winfo_children(): w.destroy()

f = tk.Frame(tab, bg=t["frame"], padx=25, pady=25)
f.pack(fill="both", expand=True)

tk.Label(f, text="HASIL ANALISIS", bg=t["frame"],
        fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,20))

data = [        #KODE PENTING (HASIL)
    ("Pace", f"{self.pace:.2f} menit/km"),
    ("Kecepatan", f"{self.speed:.1f} km/jam"),
    ("Kalori Terbakar", f"{self.kal:.0f} kalori")
]

for label, value in data:
    frame = tk.Frame(f, bg=t["card"], padx=15, pady=10)
    frame.pack(fill="x", pady=5)
    tk.Label(frame, text=label, bg=t["card"], fg=t["fg"],
            font=("Arial",11)).pack(side="left")
    tk.Label(frame, text=value, bg=t["card"], fg="#4ecdc4",
            font=("Arial",11,"bold")).pack(side="right")
```

```
    def show_gizi(self):
        t = THEME[self.mode]
        tab = self.tabs["Gizi"]
        for w in tab.winfo_children(): w.destroy()

        main_frame = tk.Frame(tab, bg=t["frame"])
        main_frame.pack(fill="both", expand=True)

        canvas = tk.Canvas(main_frame, bg=t["frame"], highlightthickness=0)
        scrollbar = ttk.Scrollbar(main_frame, orient="vertical",
command=canvas.yview)
        scrollable_frame = tk.Frame(canvas, bg=t["frame"], width=650)

        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
        )

        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True, padx=(25,0), pady=25)
```

```
scrollbar.pack(side="right", fill="y", pady=25)

def _on_mousewheel(event):
    canvas.yview_scroll(int(-1*(event.delta/120)), "units")

canvas.bind_all("<MouseWheel>", _on_mousewheel)

container = tk.Frame(scrollable_frame, bg=t["frame"])
container.pack(expand=True, fill="both", padx=20, pady=10)

tk.Label(container, text="PROGRES & NUTRISI", bg=t["frame"],
         fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,25))

if hasattr(self, 'target_jarak') and self.target_jarak > 0:
    today = date.today().strftime("%Y-%m-%d")

    current_target = self.daily_targets.get(today, self.target_jarak)

    total_jarak_hari_ini = self.daily_distances.get(today, 0)


    jarak_sekarang = float(self.vars["jarak"].get())
```

```
        sisa_jarak = current_target - total_jarak_hari_ini
        persentase = (total_jarak_hari_ini / current_target) * 100 if current_target
> 0 else 0

        progress_container = tk.Frame(container, bg=t["frame"])
        progress_container.pack(fill="x", pady=(0,20))


        tk.Label(progress_container, text="PROGRES TARGET LARI
HARIAN", bg=t["frame"],
                 fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(0,15))


        data_progres = [
           ("Target Harian", f"{current_target:.1f} km"),
           ("Jarak Hari Ini", f"{total_jarak_hari_ini:.1f} km"),
           ("Sisa Jarak", f"{abs(sisa_jarak):.2f} km"),
           ("Persentase", f"{persentase:.1f}%")
        ]


        progress_grid = tk.Frame(progress_container, bg=t["frame"])
        progress_grid.pack()
```

```
        for i, (label, value) in enumerate(data_progres):
            row_frame = tk.Frame(progress_grid, bg=t["card"], padx=25, pady=12)
            row_frame.grid(row=i, column=0, sticky="ew", pady=3, padx=50)

            tk.Label(row_frame, text=label, bg=t["card"], fg=t["fg"],
                     font=("Arial",11), width=20, anchor="center").pack(side="left",
expand=True)

            tk.Label(row_frame, text=":", bg=t["card"], fg=t["fg"],
                     font=("Arial",11), padx=10).pack(side="left")

            tk.Label(row_frame, text=value, bg=t["card"], fg="#4ecdc4",
                     font=("Arial",11,"bold"), width=15,
anchor="center").pack(side="left", expand=True)

        status_container = tk.Frame(container, bg=t["card"], padx=30, pady=15)
        status_container.pack(fill="x", pady=15, padx=80)

        if sisa_jarak <= 0:
```

```
            tk.Label(status_container, text="🎯 TARGET HARIAN TERCAPAI!",
bg=t["card"], fg="#4ecdc4",
                     font=("Arial",12,"bold")).pack()
            tk.Label(status_container, text=f"Total lari hari ini:
{total_jarak_hari_ini:.1f} km (Target: {current_target:.1f} km)",
                     bg=t["card"], fg="#4ecdc4", font=("Arial",10)).pack(pady=(5,0))



            if total_jarak_hari_ini > current_target:
                excess = total_jarak_hari_ini - current_target
                tk.Label(status_container, text=f"⭐ Anda telah melewati target
harian sebesar {excess:.1f} km!",
                         bg=t["card"], fg="#ffd166", font=("Arial",10,
"bold")).pack(pady=(5,0))
            else:
                tk.Label(status_container, text="📊 BELUM TERCAPAI",
bg=t["card"], fg="#ff6b6b",
                         font=("Arial",12,"bold")).pack()
                tk.Label(status_container, text=f"Kurang {sisa_jarak:.2f} km untuk
capai target harian",
                         bg=t["card"], fg="#ff6b6b", font=("Arial",10)).pack(pady=(5,0))
            tk.Label(status_container, text=f"Progress: {persentase:.1f}% dari
{current_target:.1f} km",
                     bg=t["card"], fg="#888", font=("Arial",9)).pack(pady=(2,0))
```

```
        tk.Label(container, text="KALORI TERBAKAR", bg=t["frame"],
                 fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(20,10))

        kal_container = tk.Frame(container, bg=t["card"], padx=30, pady=15)
        kal_container.pack(fill="x", pady=5, padx=150)
        tk.Label(kal_container, text=f"{self.kal:.0f} kalori", bg=t["card"],
fg="#ff6b6b",
                 font=("Arial",14,"bold")).pack()


        info_frame = tk.Frame(container, bg=t["card"], padx=15, pady=10)
        info_frame.pack(fill="x", pady=10, padx=80)
        tk.Label(info_frame, text=f"Input terbaru: {jarak_sekarang:.1f} km pada
{datetime.now().strftime('%H:%M')}",
                 bg=t["card"], fg="#888", font=("Arial",9)).pack()


        tk.Label(container, text="REKOMENDASI NUTRISI", bg=t["frame"],
                 fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(25,15))

        makanan = [
            ("Dada Ayam (100g)", "31g protein", "Protein tinggi, rendah lemak"),
            ("Telur (2 butir)", "13g protein", "Protein lengkap, mudah dicerna"),
            ("Salmon (100g)", "25g protein", "Protein + Omega-3"),
```

```
    ("Tahu (100g)", "8g protein", "Protein nabati"),
    ("Nasi Merah (100g)", "23g karbo", "Karbohidrat kompleks"),
    ("Oatmeal (50g)", "30g karbo", "Serat tinggi, energi tahan lama"),
    ("Ubi (100g)", "20g karbo", "Vitamin A, karbo sehat"),
    ("Pisang (1 buah)", "27g karbo", "Kalium, energi cepat")
]

makanan_container = tk.Frame(container, bg=t["frame"])
makanan_container.pack(fill="x", padx=50)

for i, (nama, nutrisi, desc) in enumerate(makanan):
    frame = tk.Frame(makanan_container, bg=t["card"], padx=15, pady=10)
    frame.pack(fill="x", pady=4)

    content_frame = tk.Frame(frame, bg=t["card"])
    content_frame.pack(expand=True)

    tk.Label(content_frame, text=nama, bg=t["card"], fg=t["fg"],
             font=("Arial",10,"bold"), width=25).pack(pady=(0,5))

    nutrisi_frame = tk.Frame(content_frame, bg=t["card"])
```

```
        nutrisi_frame.pack()

        tk.Label(nutrisi_frame, text=nutrisi, bg=t["card"], fg="#ff6b6b",
                 font=("Arial",9,"bold"), width=20).pack(side="left", padx=(0,10))

        tk.Label(nutrisi_frame, text=desc, bg=t["card"], fg="#888",
                 font=("Arial",9), wraplength=200,
justify="center").pack(side="left")


    tk.Label(container, text="TIPS CEPAT", bg=t["frame"],
             fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(25,10))

    tips = [
        "Protein dalam 30 menit setelah lari",
        "Minum air 500ml setiap 30 menit lari",
        "Karbohidrat kompleks sebelum lari",
        "Hindari makanan berat 2 jam sebelum lari"
    ]

    tips_container = tk.Frame(container, bg=t["frame"])
    tips_container.pack(fill="x", padx=100)

    for tip in tips:
```

```
            frame = tk.Frame(tips_container, bg=t["card"], padx=15, pady=8)
            frame.pack(fill="x", pady=3)
            tk.Label(frame, text=f"✓ {tip}", bg=t["card"], fg="#4ecdc4",
                     font=("Arial",9)).pack(anchor="center")
        else:
            message_frame = tk.Frame(container, bg=t["frame"], padx=20, pady=40)
            message_frame.pack(expand=True, fill="both")
            tk.Label(message_frame, text="Masukkan target jarak harian di tab Input",
                     bg=t["frame"], fg=t["fg"], font=("Arial",11)).pack(expand=True)
            tk.Label(message_frame, text="Target akan digunakan untuk menghitung
progres harian",
                     bg=t["frame"], fg="#888", font=("Arial",9)).pack()

    def show_jadwal(self):
        t = THEME[self.mode]
        tab = self.tabs["Jadwal"]
        for w in tab.winfo_children(): w.destroy()

        f = tk.Frame(tab, bg=t["frame"], padx=25, pady=25)
        f.pack(fill="both", expand=True)

        tk.Label(f, text="JADWAL LATIHAN", bg=t["frame"],
                 fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,20))
```

```
hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
jadwal = [
    ("Lari Ringan", "30 menit", "Pemanasan"),
    ("Interval Run", "45 menit", "Kecepatan"),
    ("Recovery", "20 menit", "Pemulihan"),
    ("Long Run", "60 menit", "Daya tahan"),
    ("Cross Training", "40 menit", "Variasi"),
    ("Tempo Run", "50 menit", "Konsistensi"),
    ("Rest Day", "-", "Pemulihan total")
]

for i, (latihan, durasi, tipe) in enumerate(jadwal):
    frame = tk.Frame(f, bg=t["card"], padx=15, pady=10)
    frame.pack(fill="x", pady=4)

    tk.Label(frame, text=hari[i], bg=t["card"], fg=t["fg"],
             font=("Arial",11,"bold"), width=8).pack(side="left", padx=(0,10))

    tk.Label(frame, text=latihan, bg=t["card"], fg="#4ecdc4",
             font=("Arial",11,"bold"), width=15).pack(side="left", padx=(0,10))
```

```
            tk.Label(frame, text=f"{durasi} | {tipe}", bg=t["card"], fg=t["fg"],
                     font=("Arial",10)).pack(side="left")

            color = "#ff6b6b" if i % 3 == 0 else "#4ecdc4" if i % 3 == 1 else "#ffd166"
            tk.Label(frame, text="●", bg=t["card"], fg=color,
                     font=("Arial",12)).pack(side="right")

    def show_history(self):
        t = THEME[self.mode]
        tab = self.tabs["History"]
        for w in tab.winfo_children(): w.destroy()

        f = tk.Frame(tab, bg=t["frame"], padx=25, pady=25)
        f.pack(fill="both", expand=True)

        tk.Label(f, text="Riwayat Analisis", bg=t["frame"],
                 fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,20))

        if not self.history:
            tk.Label(f, text="Belum ada riwayat", fg=t["fg"], bg=t["frame"]).pack()
            return
```

```
        dates_frame = tk.Frame(f, bg=t["frame"])
        dates_frame.pack(anchor="center")

        row_frame = None
        for i, tanggal in enumerate(sorted(self.history.keys(), reverse=True)):
            if i % 3 == 0:
                row_frame = tk.Frame(dates_frame, bg=t["frame"])
                row_frame.pack(anchor="center", pady=6)

            btn = tk.Button(
                row_frame, text=tanggal,
                command=lambda tgl=tanggal: self.show_date_detail(tgl),
                bg=t["card"], fg=t["fg"], font=("Arial",10),
                relief="flat", padx=18, pady=8, cursor="hand2"
            )
            btn.pack(side="left", padx=6)

            btn.bind("<Enter>", lambda e, b=btn: b.config(bg="#444444" if
self.mode=="dark" else "#dddddd"))
            btn.bind("<Leave>", lambda e, b=btn: b.config(bg=t["card"]))

    def show_date_detail(self, tanggal):
        detail = tk.Toplevel(self)
        detail.title(f"Detail {tanggal}")
```

```
        detail.geometry("600x450")
        t = THEME[self.mode]
        detail.configure(bg=t["bg"])

        main_frame = tk.Frame(detail, bg=t["bg"])
        main_frame.pack(expand=True, fill="both")

        tk.Label(main_frame, text=f"Detail Tanggal {tanggal}", bg=t["bg"],
fg="#ffd166",
                 font=("Arial",14,"bold")).pack(pady=15)

        center_frame = tk.Frame(main_frame, bg=t["bg"])
        center_frame.pack(expand=True)

        container = tk.Frame(center_frame, bg=t["frame"], padx=20, pady=20)
        container.pack()

        target_harian = self.daily_targets.get(tanggal, "Tidak ada target")
        total_jarak = self.daily_distances.get(tanggal, 0)
```

```
        target_frame = tk.Frame(container, bg=t["card"], padx=10, pady=8)
        target_frame.pack(fill="x", pady=5)
        tk.Label(target_frame, text=f"Target Harian: {target_harian} km | Total Jarak: {total_jarak:.1f} km",
                 bg=t["card"], fg=t["fg"], font=("Arial",10, "bold")).pack()

        for item in self.history[tanggal]:
            row_frame = tk.Frame(container, bg=t["card"], padx=10, pady=8)
            row_frame.pack(fill="x", pady=4)

            info = f"{item['time']} | {item['jarak']}km | {item['waktu']}m | Pace {item['pace']:.2f} | {item['kal']:.0f} cal"

            label = tk.Label(row_frame, text=info, bg=t["card"], fg=t["fg"],
                    font=("Arial",10))
            label.pack(expand=True)

RunningApp().mainloop()     #KODE PENTING
```

Code Setelah Di Sesuaikan:

```
import tkinter as tk
from tkinter import ttk, messagebox
from datetime import datetime, date
import json
import os

THEME = {
    "dark": {"bg":"#1e1e2e","frame":"#2d3047","card":"#3d405b","fg":"white"},
    "light":{"bg":"#f4f4f4","frame":"#ffffff","card":"#e6e6e6","fg":"black"}
}

DATA_FILE = "running_data.json"

class RunningApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Run Analyzer Pro")
        self.geometry("700x750")
        self.mode = "dark"
        self.vars = {x: tk.StringVar() for x in
["jarak","waktu","berat","target_jarak"]}
```

```
        self.history = {}
        self.daily_targets = {}
        self.daily_distances = {}


        self.load_data()

        self.make_gui()
        self.apply_theme()


        self.protocol("WM_DELETE_WINDOW", self.on_closing)

    def load_data(self):
        """Memuat data dari file JSON"""
        try:
            if os.path.exists(DATA_FILE):
                with open(DATA_FILE, 'r') as f:
                    data = json.load(f)
                    self.history = data.get("history", {})
                    self.daily_targets = data.get("daily_targets", {})
                    self.daily_distances = data.get("daily_distances", {})
```

```
    except Exception as e:
        print(f"Error loading data: {e}")

        self.history = {}
        self.daily_targets = {}
        self.daily_distances = {}

def save_data(self):
    """Menyimpan data ke file JSON"""
    try:
        data = {
            "history": self.history,
            "daily_targets": self.daily_targets,
            "daily_distances": self.daily_distances
        }
        with open(DATA_FILE, 'w') as f:
            json.dump(data, f, indent=2)
    except Exception as e:
        print(f"Error saving data: {e}")

def on_closing(self):
    """Handler saat aplikasi ditutup"""
```

```
        self.save_data()
        self.destroy()

    def make_gui(self):
        self.title_lbl = tk.Label(self, text="RUN ANALYZER PRO",
font=("Arial",18,"bold"))
        self.title_lbl.pack(pady=20)

        self.notebook = ttk.Notebook(self)
        self.notebook.pack(expand=True, fill="both", padx=20, pady=10)

        self.tabs = {}
        for name in ["Input","Hasil","Gizi","Jadwal","History"]:
            self.tabs[name] = ttk.Frame(self.notebook)
            self.notebook.add(self.tabs[name], text=name)

        self.make_input_tab()


        button_frame = tk.Frame(self, bg=THEME[self.mode]["bg"])
        button_frame.pack(pady=5)

        tk.Button(button_frame, text="Toggle Theme", command=self.toggle_theme,
```

```
            bg="#4ecdc4", fg="white", font=("Arial",10),
padx=15).pack(side="left", padx=5)

        tk.Button(button_frame, text="Clear History", command=self.clear_history,
            bg="#ff6b6b", fg="white", font=("Arial",10),
padx=15).pack(side="left", padx=5)

    def clear_history(self):
        """Menghapus semua data history"""
        if messagebox.askyesno("Konfirmasi", "Apakah Anda yakin ingin
menghapus semua history?"):
            self.history = {}
            self.daily_targets = {}
            self.daily_distances = {}
            self.save_data()
            self.show_history()
            messagebox.showinfo("Sukses", "Semua history telah dihapus!")

    def apply_theme(self):
        t = THEME[self.mode]
        self.configure(bg=t["bg"])
        self.title_lbl.configure(bg=t["bg"], fg=t["fg"])
        for tab in self.tabs.values():
            for widget in tab.winfo_children():
```

```
            widget.destroy()
        self.make_input_tab()
        if hasattr(self, "pace"):
            self.show_all()

    def toggle_theme(self):
        self.mode = "light" if self.mode=="dark" else "dark"
        self.apply_theme()

    def make_input_tab(self):
        t = THEME[self.mode]
        f = tk.Frame(self.tabs["Input"], bg=t["frame"], padx=25, pady=25)
        f.pack(expand=True, fill="both")

        labels = ["Jarak (km)", "Waktu (menit)", "Berat (kg)", "Target Jarak Harian
(km)"]
        keys = ["jarak", "waktu", "berat", "target_jarak"]

        for label, key in zip(labels, keys):
            tk.Label(f, text=label, bg=t["frame"], fg=t["fg"]).pack(anchor="w",
pady=(5,0))
            tk.Entry(f, textvariable=self.vars[key], font=("Arial",12),
                     bg=t["card"], fg=t["fg"]).pack(fill="x", pady=3)
```

```
        tk.Button(f, text="Analisis", command=self.analyze,
                  bg="#ff6b6b", fg="white", font=("Arial",11,"bold"),
                  pady=8).pack(fill="x", pady=20)

    def analyze(self):
        try:
            j = float(self.vars["jarak"].get())
            w = float(self.vars["waktu"].get())
            b = float(self.vars["berat"].get())
            t = float(self.vars["target_jarak"].get()) if
self.vars["target_jarak"].get().strip() else None
            if min(j,w,b) <= 0: raise ValueError
            if t is not None and t <= 0: raise ValueError
        except:
            return messagebox.showerror("Error","Input tidak valid!")

        today = date.today().strftime("%Y-%m-%d")
        self.pace = w/j
        self.speed = (j/w)*60
        self.kal = j*b*0.653

        if t is not None:
            self.daily_targets[today] = t
            self.target_jarak = t
```

```
else:
    if today in self.daily_targets:
        self.target_jarak = self.daily_targets[today]
    elif hasattr(self, 'target_jarak'):
        self.daily_targets[today] = self.target_jarak
    else:
        self.target_jarak = 0

if today not in self.daily_distances:
    self.daily_distances[today] = 0
self.daily_distances[today] += j

total_jarak_hari_ini = self.daily_distances[today]

if today not in self.history:
    self.history[today] = []

self.history[today].append({
    "time": datetime.now().strftime("%H:%M"),
    "jarak": j,
    "waktu": w,
    "pace": self.pace,
    "speed": self.speed,
```

```
            "kal": self.kal,
            "target": self.target_jarak,
            "total_jarak_harian": total_jarak_hari_ini
        })

        self.save_data()

        self.show_all()
        self.notebook.select(1)

    def show_all(self):
        self.show_hasil()
        self.show_gizi()
        self.show_jadwal()
        self.show_history()

    def show_hasil(self):
        t = THEME[self.mode]
        tab = self.tabs["Hasil"]
        for w in tab.winfo_children(): w.destroy()

        f = tk.Frame(tab, bg=t["frame"], padx=25, pady=25)
```

```
    f.pack(fill="both", expand=True)

    tk.Label(f, text="HASIL ANALISIS", bg=t["frame"],
             fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,20))

    data = [
        ("Pace", f"{self.pace:.2f} menit/km"),
        ("Kecepatan", f"{self.speed:.1f} km/jam"),
        ("Kalori Terbakar", f"{self.kal:.0f} kalori")
    ]

    for label, value in data:
        frame = tk.Frame(f, bg=t["card"], padx=15, pady=10)
        frame.pack(fill="x", pady=5)
        tk.Label(frame, text=label, bg=t["card"], fg=t["fg"],
                 font=("Arial",11)).pack(side="left")
        tk.Label(frame, text=value, bg=t["card"], fg="#4ecdc4",
                 font=("Arial",11,"bold")).pack(side="right")

def show_gizi(self):
    t = THEME[self.mode]
    tab = self.tabs["Gizi"]
    for w in tab.winfo_children(): w.destroy()
```

```
        main_frame = tk.Frame(tab, bg=t["frame"])
        main_frame.pack(fill="both", expand=True)

        canvas = tk.Canvas(main_frame, bg=t["frame"], highlightthickness=0)
        scrollbar = ttk.Scrollbar(main_frame, orient="vertical",
command=canvas.yview)
        scrollable_frame = tk.Frame(canvas, bg=t["frame"], width=650)

        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
        )

        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True, padx=(25,0), pady=25)
        scrollbar.pack(side="right", fill="y", pady=25)

        def _on_mousewheel(event):
            canvas.yview_scroll(int(-1*(event.delta/120)), "units")

        canvas.bind_all("<MouseWheel>", _on_mousewheel)
```

```
        container = tk.Frame(scrollable_frame, bg=t["frame"])
        container.pack(expand=True, fill="both", padx=20, pady=10)

        tk.Label(container, text="PROGRES & NUTRISI", bg=t["frame"],
                 fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,25))

        if hasattr(self, 'target_jarak') and self.target_jarak > 0:
            today = date.today().strftime("%Y-%m-%d")

            current_target = self.daily_targets.get(today, self.target_jarak)

            total_jarak_hari_ini = self.daily_distances.get(today, 0)

            jarak_sekarang = float(self.vars["jarak"].get())

            sisa_jarak = current_target - total_jarak_hari_ini
            persentase = (total_jarak_hari_ini / current_target) * 100 if current_target
> 0 else 0

            progress_container = tk.Frame(container, bg=t["frame"])
            progress_container.pack(fill="x", pady=(0,20))
```

```
        tk.Label(progress_container, text="PROGRES TARGET LARI
HARIAN", bg=t["frame"],
                 fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(0,15))

        data_progres = [
            ("Target Harian", f"{current_target:.1f} km"),
            ("Jarak Hari Ini", f"{total_jarak_hari_ini:.1f} km"),
            ("Sisa Jarak", f"{abs(sisa_jarak):.2f} km"),
            ("Persentase", f"{persentase:.1f}%")
        ]

        progress_grid = tk.Frame(progress_container, bg=t["frame"])
        progress_grid.pack()

        for i, (label, value) in enumerate(data_progres):
            row_frame = tk.Frame(progress_grid, bg=t["card"], padx=25, pady=12)
            row_frame.grid(row=i, column=0, sticky="ew", pady=3, padx=50)

            tk.Label(row_frame, text=label, bg=t["card"], fg=t["fg"],
                     font=("Arial",11), width=20, anchor="center").pack(side="left",
expand=True)

            tk.Label(row_frame, text=":", bg=t["card"], fg=t["fg"],
                     font=("Arial",11), padx=10).pack(side="left")
```

```
            tk.Label(row_frame, text=value, bg=t["card"], fg="#4ecdc4",
                     font=("Arial",11,"bold"), width=15,
anchor="center").pack(side="left", expand=True)

        status_container = tk.Frame(container, bg=t["card"], padx=30, pady=15)
        status_container.pack(fill="x", pady=15, padx=80)

        if sisa_jarak <= 0:
            tk.Label(status_container, text="🎯 TARGET HARIAN TERCAPAI!",
bg=t["card"], fg="#4ecdc4",
                     font=("Arial",12,"bold")).pack()
            tk.Label(status_container, text=f"Total lari hari ini:
{total_jarak_hari_ini:.1f} km (Target: {current_target:.1f} km)",
                     bg=t["card"], fg="#4ecdc4", font=("Arial",10)).pack(pady=(5,0))

            if total_jarak_hari_ini > current_target:
                excess = total_jarak_hari_ini - current_target
                tk.Label(status_container, text=f"⭐ Anda telah melewati target
harian sebesar {excess:.1f} km!",
                         bg=t["card"], fg="#ffd166", font=("Arial",10,
"bold")).pack(pady=(5,0))
        else:
            tk.Label(status_container, text="📊 BELUM TERCAPAI",
bg=t["card"], fg="#ff6b6b",
```

```
                font=("Arial",12,"bold")).pack()
            tk.Label(status_container, text=f"Kurang {sisa_jarak:.2f} km untuk
capai target harian",
                bg=t["card"], fg="#ff6b6b", font=("Arial",10)).pack(pady=(5,0))
            tk.Label(status_container, text=f"Progress: {persentase:.1f}% dari
{current_target:.1f} km",
                bg=t["card"], fg="#888", font=("Arial",9)).pack(pady=(2,0))

        tk.Label(container, text="KALORI TERBAKAR", bg=t["frame"],
                fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(20,10))

        kal_container = tk.Frame(container, bg=t["card"], padx=30, pady=15)
        kal_container.pack(fill="x", pady=5, padx=150)
        tk.Label(kal_container, text=f"{self.kal:.0f} kalori", bg=t["card"],
fg="#ff6b6b",
                font=("Arial",14,"bold")).pack()

        info_frame = tk.Frame(container, bg=t["card"], padx=15, pady=10)
        info_frame.pack(fill="x", pady=10, padx=80)
        tk.Label(info_frame, text=f"Input terbaru: {jarak_sekarang:.1f} km pada
{datetime.now().strftime('%H:%M')}",
                bg=t["card"], fg="#888", font=("Arial",9)).pack()

        tk.Label(container, text="REKOMENDASI NUTRISI", bg=t["frame"],
                fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(25,15))
```

```
makanan = [
    ("Dada Ayam (100g)", "31g protein", "Protein tinggi, rendah lemak"),
    ("Telur (2 butir)", "13g protein", "Protein lengkap, mudah dicerna"),
    ("Salmon (100g)", "25g protein", "Protein + Omega-3"),
    ("Tahu (100g)", "8g protein", "Protein nabati"),
    ("Nasi Merah (100g)", "23g karbo", "Karbohidrat kompleks"),
    ("Oatmeal (50g)", "30g karbo", "Serat tinggi, energi tahan lama"),
    ("Ubi (100g)", "20g karbo", "Vitamin A, karbo sehat"),
    ("Pisang (1 buah)", "27g karbo", "Kalium, energi cepat")
]

makanan_container = tk.Frame(container, bg=t["frame"])
makanan_container.pack(fill="x", padx=50)

for i, (nama, nutrisi, desc) in enumerate(makanan):
    frame = tk.Frame(makanan_container, bg=t["card"], padx=15, pady=10)
    frame.pack(fill="x", pady=4)

    content_frame = tk.Frame(frame, bg=t["card"])
    content_frame.pack(expand=True)

    tk.Label(content_frame, text=nama, bg=t["card"], fg=t["fg"],
```

```
                font=("Arial",10,"bold"), width=25).pack(pady=(0,5))

        nutrisi_frame = tk.Frame(content_frame, bg=t["card"])
        nutrisi_frame.pack()

        tk.Label(nutrisi_frame, text=nutrisi, bg=t["card"], fg="#ff6b6b",
                font=("Arial",9,"bold"), width=20).pack(side="left", padx=(0,10))

        tk.Label(nutrisi_frame, text=desc, bg=t["card"], fg="#888",
                font=("Arial",9), wraplength=200,
justify="center").pack(side="left")

    tk.Label(container, text="TIPS CEPAT", bg=t["frame"],
            fg=t["fg"], font=("Arial",12,"bold")).pack(pady=(25,10))

    tips = [
        "Protein dalam 30 menit setelah lari",
        "Minum air 500ml setiap 30 menit lari",
        "Karbohidrat kompleks sebelum lari",
        "Hindari makanan berat 2 jam sebelum lari"
    ]

    tips_container = tk.Frame(container, bg=t["frame"])
    tips_container.pack(fill="x", padx=100)
```

```
            for tip in tips:
                frame = tk.Frame(tips_container, bg=t["card"], padx=15, pady=8)
                frame.pack(fill="x", pady=3)
                tk.Label(frame, text=f"✓ {tip}", bg=t["card"], fg="#4ecdc4",
                         font=("Arial",9)).pack(anchor="center")
        else:
            message_frame = tk.Frame(container, bg=t["frame"], padx=20, pady=40)
            message_frame.pack(expand=True, fill="both")
            tk.Label(message_frame, text="Masukkan target jarak harian di tab Input",
                     bg=t["frame"], fg=t["fg"], font=("Arial",11)).pack(expand=True)
            tk.Label(message_frame, text="Target akan digunakan untuk menghitung
progres harian",
                     bg=t["frame"], fg="#888", font=("Arial",9)).pack()


    def show_jadwal(self):
        t = THEME[self.mode]
        tab = self.tabs["Jadwal"]
        for w in tab.winfo_children(): w.destroy()

        f = tk.Frame(tab, bg=t["frame"], padx=25, pady=25)
        f.pack(fill="both", expand=True)

        tk.Label(f, text="JADWAL LATIHAN", bg=t["frame"],
```

```
        fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,20))

hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
jadwal = [
    ("Lari Ringan", "30 menit", "Pemanasan"),
    ("Interval Run", "45 menit", "Kecepatan"),
    ("Recovery", "20 menit", "Pemulihan"),
    ("Long Run", "60 menit", "Daya tahan"),
    ("Cross Training", "40 menit", "Variasi"),
    ("Tempo Run", "50 menit", "Konsistensi"),
    ("Rest Day", "-", "Pemulihan total")
]

for i, (latihan, durasi, tipe) in enumerate(jadwal):
    frame = tk.Frame(f, bg=t["card"], padx=15, pady=10)
    frame.pack(fill="x", pady=4)

    tk.Label(frame, text=hari[i], bg=t["card"], fg=t["fg"],
             font=("Arial",11,"bold"), width=8).pack(side="left", padx=(0,10))

    tk.Label(frame, text=latihan, bg=t["card"], fg="#4ecdc4",
             font=("Arial",11,"bold"), width=15).pack(side="left", padx=(0,10))
```

```
        tk.Label(frame, text=f"{durasi} | {tipe}", bg=t["card"], fg=t["fg"],
                 font=("Arial",10)).pack(side="left")

        color = "#ff6b6b" if i % 3 == 0 else "#4ecdc4" if i % 3 == 1 else "#ffd166"
        tk.Label(frame, text="●", bg=t["card"], fg=color,
                 font=("Arial",12)).pack(side="right")

def show_history(self):
    t = THEME[self.mode]
    tab = self.tabs["History"]
    for w in tab.winfo_children(): w.destroy()

    f = tk.Frame(tab, bg=t["frame"], padx=25, pady=25)
    f.pack(fill="both", expand=True)


    header_frame = tk.Frame(f, bg=t["frame"])
    header_frame.pack(fill="x", pady=(0,20))

    tk.Label(header_frame, text="Riwayat Analisis", bg=t["frame"],
             fg="#ffd166", font=("Arial",14,"bold")).pack(side="left")
```

```
        total_days = len(self.history)
        if total_days > 0:
            total_runs = sum(len(runs) for runs in self.history.values())
            info_text = f"({total_days} hari, {total_runs} sesi lari)"
            tk.Label(header_frame, text=info_text, bg=t["frame"],
                     fg="#888", font=("Arial",10)).pack(side="left", padx=(10,0))

        if not self.history:
            no_data_frame = tk.Frame(f, bg=t["frame"], pady=50)
            no_data_frame.pack(expand=True)
            tk.Label(no_data_frame, text="Belum ada riwayat lari", fg=t["fg"],
                     bg=t["frame"], font=("Arial",11)).pack()
            tk.Label(no_data_frame, text="Mulai dengan menginput data di tab
'Input'",
                     fg="#888", bg=t["frame"], font=("Arial",9)).pack(pady=(10,0))
            return

        dates_frame = tk.Frame(f, bg=t["frame"])
        dates_frame.pack(anchor="center")

        row_frame = None
        dates_list = sorted(self.history.keys(), reverse=True)

        for i, tanggal in enumerate(dates_list):
```

```
            if i % 3 == 0:
                row_frame = tk.Frame(dates_frame, bg=t["frame"])
                row_frame.pack(anchor="center", pady=6)



            num_runs = len(self.history[tanggal])


            btn = tk.Button(
                row_frame, text=f"{tanggal}\n({num_runs} sesi)",
                command=lambda tgl=tanggal: self.show_date_detail(tgl),
                bg=t["card"], fg=t["fg"], font=("Arial",10),
                relief="flat", padx=20, pady=10, cursor="hand2",
                width=12, height=2
            )
            btn.pack(side="left", padx=6)


            btn.bind("<Enter>", lambda e, b=btn: b.config(bg="#444444" if
self.mode=="dark" else "#dddddd"))
            btn.bind("<Leave>", lambda e, b=btn: b.config(bg=t["card"]))


    def show_date_detail(self, tanggal):
        detail = tk.Toplevel(self)
        detail.title(f"Detail {tanggal}")
        detail.geometry("600x500")
```

```
        t = THEME[self.mode]
        detail.configure(bg=t["bg"])


        detail.protocol("WM_DELETE_WINDOW", lambda:
self.close_detail_window(detail))

        main_frame = tk.Frame(detail, bg=t["bg"])
        main_frame.pack(expand=True, fill="both")


        header_frame = tk.Frame(main_frame, bg=t["bg"])
        header_frame.pack(fill="x", pady=(10,15))

        tk.Label(header_frame, text=f"Detail Tanggal: {tanggal}", bg=t["bg"],
fg="#ffd166",
                 font=("Arial",14,"bold")).pack(side="left", padx=(20,0))

        tk.Button(header_frame, text="✕", command=lambda:
self.close_detail_window(detail),
                  bg="#ff6b6b", fg="white", font=("Arial",10),
                  width=3, relief="flat").pack(side="right", padx=20)
```

```
        container = tk.Frame(main_frame, bg=t["bg"])
        container.pack(fill="both", expand=True, padx=20, pady=10)

        summary_frame = tk.Frame(container, bg=t["frame"], padx=15, pady=15)
        summary_frame.pack(fill="x", pady=(0,15))

        target_harian = self.daily_targets.get(tanggal, "Tidak ada target")
        total_jarak = self.daily_distances.get(tanggal, 0)
        num_sessions = len(self.history[tanggal])

        tk.Label(summary_frame, text=f"📊 SUMMARY - {tanggal}",
bg=t["frame"],
                 fg=t["fg"], font=("Arial",11,"bold")).pack(anchor="w", pady=(0,8))

        tk.Label(summary_frame, text=f"• Target Harian: {target_harian} km",
                 bg=t["frame"], fg=t["fg"], font=("Arial",10)).pack(anchor="w")
        tk.Label(summary_frame, text=f"• Total Jarak: {total_jarak:.2f} km",
                 bg=t["frame"], fg="#4ecdc4", font=("Arial",10)).pack(anchor="w")
        tk.Label(summary_frame, text=f"• Jumlah Sesi: {num_sessions} sesi",
                 bg=t["frame"], fg="#888", font=("Arial",9)).pack(anchor="w")

        canvas_frame = tk.Frame(container, bg=t["bg"])
```

```
        canvas_frame.pack(fill="both", expand=True)

        canvas = tk.Canvas(canvas_frame, bg=t["bg"], highlightthickness=0,
height=300)
        scrollbar = ttk.Scrollbar(canvas_frame, orient="vertical",
command=canvas.yview)
        scrollable_content = tk.Frame(canvas, bg=t["bg"])

        scrollable_content.bind(
            "<Configure>",
            lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
        )

        canvas.create_window((0, 0), window=scrollable_content, anchor="nw",
width=550)
        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")

        tk.Label(scrollable_content, text="📝 DETAIL SESI LARI", bg=t["bg"],
                 fg=t["fg"], font=("Arial",11,"bold")).pack(anchor="w", pady=(0,10))
```

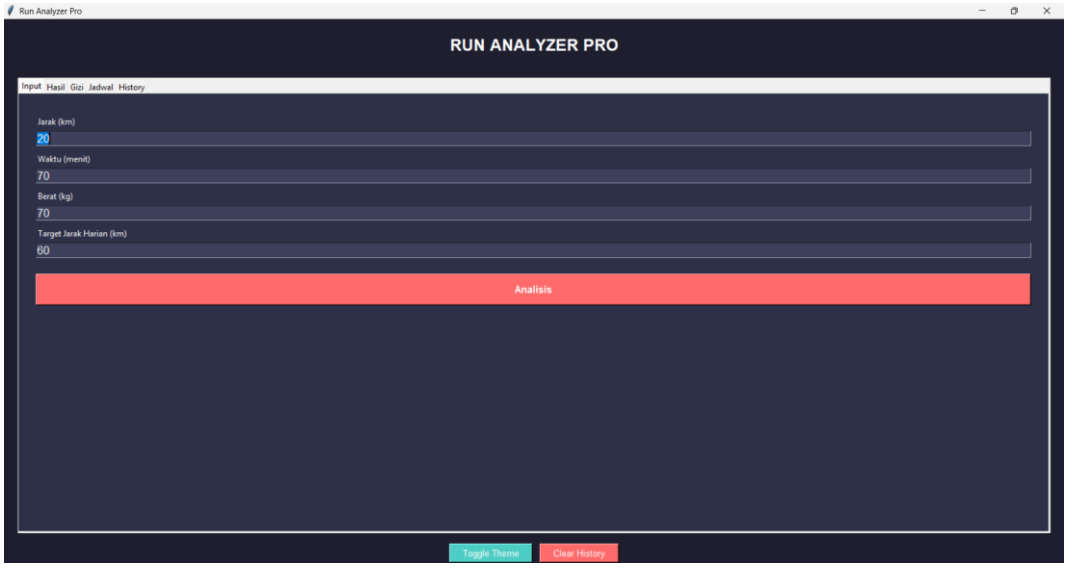
```
        for idx, item in enumerate(self.history[tanggal], 1):
            session_frame = tk.Frame(scrollable_content, bg=t["card"], padx=15,
pady=12)
            session_frame.pack(fill="x", pady=4)



            header_sesi = tk.Frame(session_frame, bg=t["card"])
            header_sesi.pack(fill="x", pady=(0,8))

            tk.Label(header_sesi, text=f"Sesi #{idx} - {item['time']}",
                     bg=t["card"], fg=t["fg"], font=("Arial",10,"bold")).pack(side="left")



            detail_frame = tk.Frame(session_frame, bg=t["card"])
            detail_frame.pack()



            left_frame = tk.Frame(detail_frame, bg=t["card"])
            left_frame.grid(row=0, column=0, padx=(0,20))

            tk.Label(left_frame, text=f"Jarak: {item['jarak']} km",
                     bg=t["card"], fg="#4ecdc4", font=("Arial",9)).pack(anchor="w")
            tk.Label(left_frame, text=f"Waktu: {item['waktu']} menit",
                     bg=t["card"], fg="#4ecdc4", font=("Arial",9)).pack(anchor="w")
```

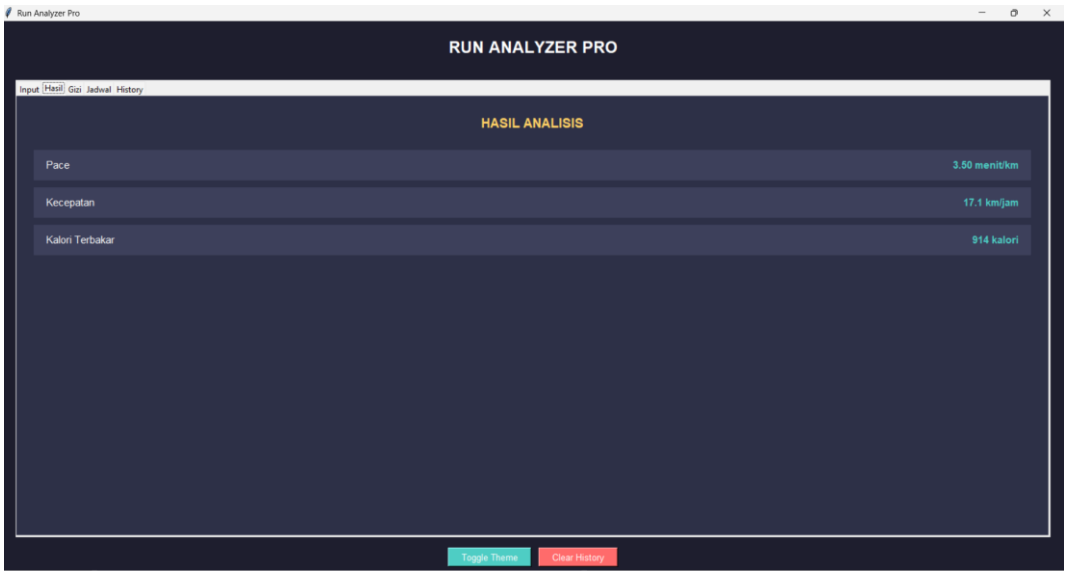
```
            tk.Label(left_frame, text=f"Pace: {item['pace']:.2f} menit/km",
                     bg=t["card"], fg="#4ecdc4", font=("Arial",9)).pack(anchor="w")

            right_frame = tk.Frame(detail_frame, bg=t["card"])
            right_frame.grid(row=0, column=1)

            tk.Label(right_frame, text=f"Speed: {item['speed']:.1f} km/jam",
                     bg=t["card"], fg="#ff6b6b", font=("Arial",9)).pack(anchor="w")
            tk.Label(right_frame, text=f"Kalori: {item['kal']:.0f} kal",
                     bg=t["card"], fg="#ff6b6b", font=("Arial",9)).pack(anchor="w")
            tk.Label(right_frame, text=f"Total Harian: {item['total_jarak_harian']:.1f}
km",
                     bg=t["card"], fg="#888", font=("Arial",9)).pack(anchor="w")


        def _on_mousewheel(event):
            canvas.yview_scroll(int(-1*(event.delta/120)), "units")

        canvas.bind_all("<MouseWheel>", _on_mousewheel)


        def delete_date_history():
```

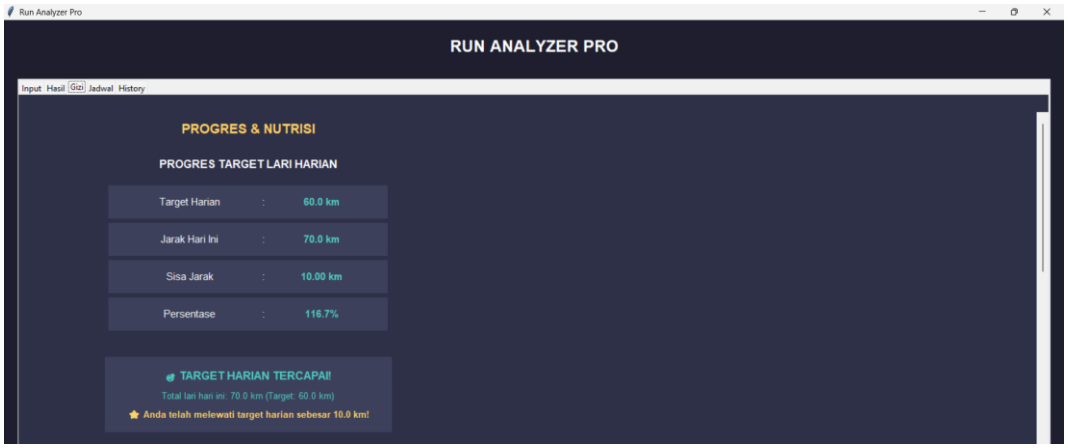
```
            if messagebox.askyesno("Konfirmasi", f"Hapus semua data untuk tanggal {tanggal}?"):
                if tanggal in self.history:
                    del self.history[tanggal]
                if tanggal in self.daily_targets:
                    del self.daily_targets[tanggal]
                if tanggal in self.daily_distances:
                    del self.daily_distances[tanggal]
                self.save_data()
                detail.destroy()
                self.show_history()
                messagebox.showinfo("Sukses", f"Data untuk {tanggal} telah dihapus!")

        button_frame = tk.Frame(container, bg=t["bg"])
        button_frame.pack(fill="x", pady=(15,0))

        tk.Button(button_frame, text="Hapus Data Tanggal Ini", command=delete_date_history,
                  bg="#ff6b6b", fg="white", font=("Arial",10), padx=15).pack()

    def close_detail_window(self, window):
        """Menutup window detail"""
        window.destroy()
```

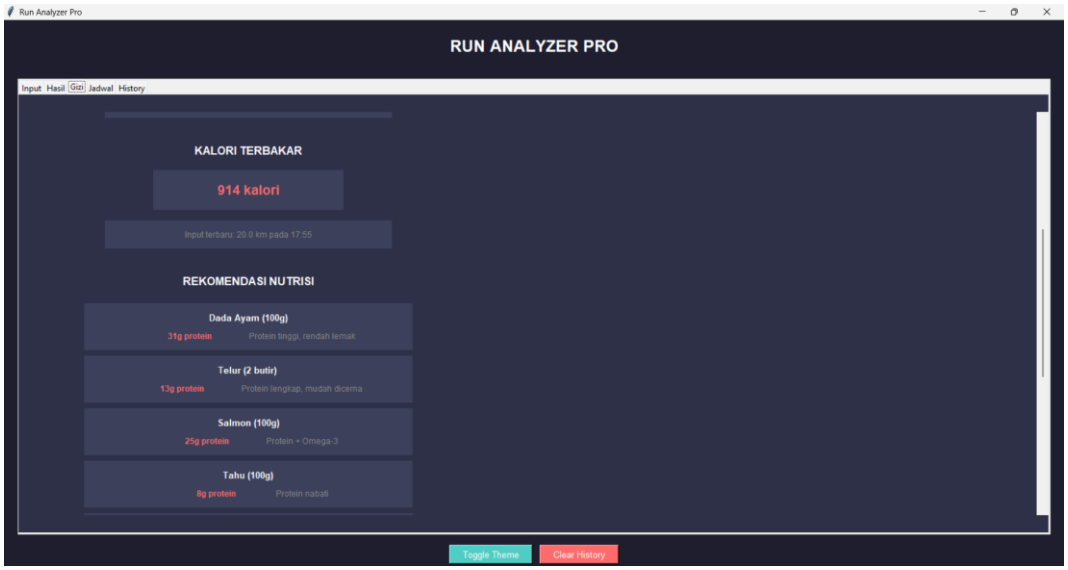
```
        # Refresh history tab
        self.show_history()

if __name__ == "__main__":
    app = RunningApp()
    app.mainloop()
```

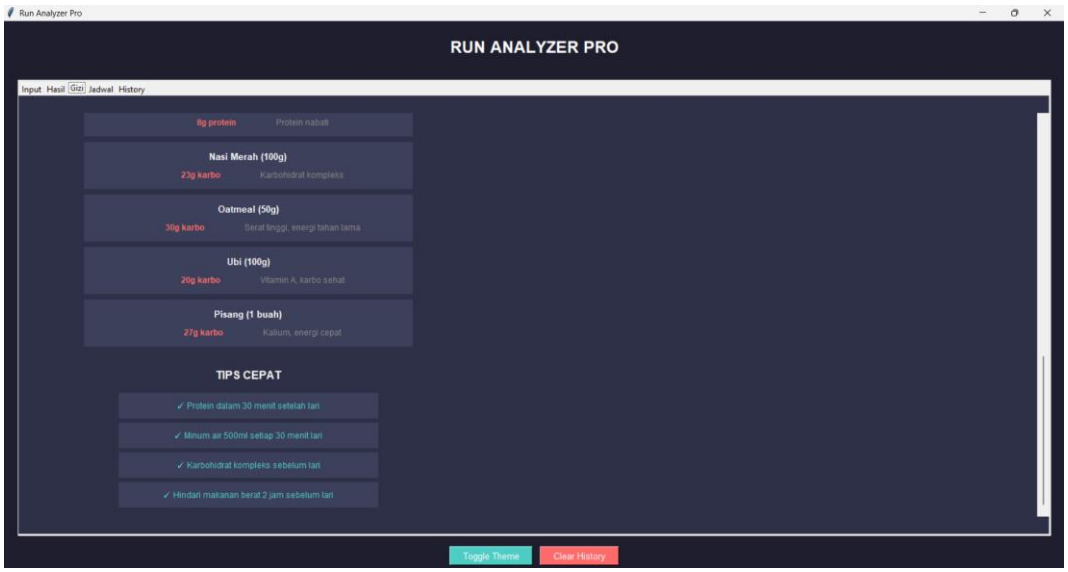
Tampillan:

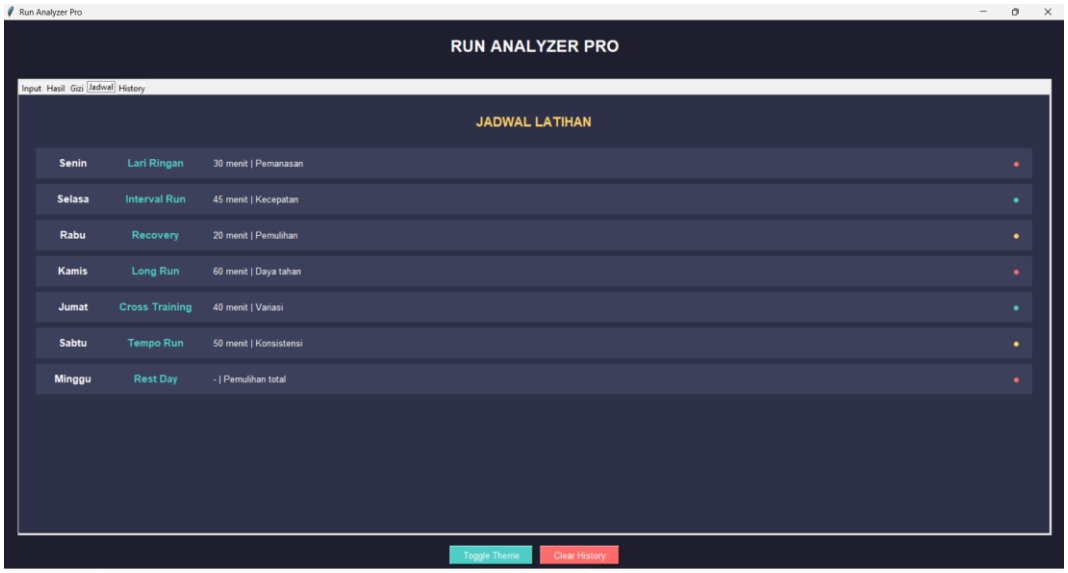


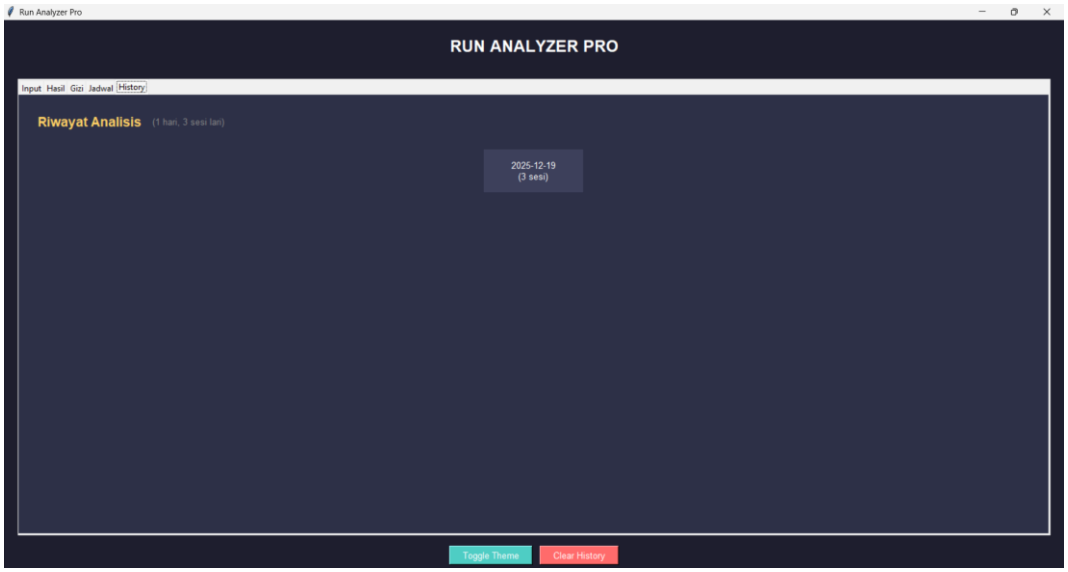


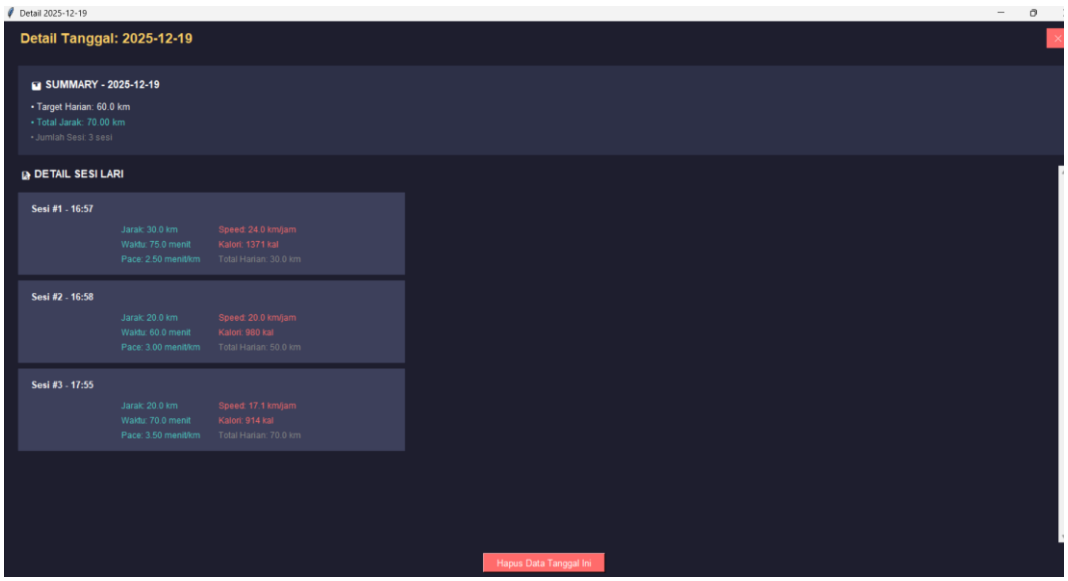

Run Analyzer Pro
RUN ANALYZER PRO
Input Hasil Gizi Jadwal History
PROGRES & NUTRISI
PROGRES TARGET LARI HARIAN
Target Harian : 60.0 km
Jarak Hari Ini : 70.0 km
Sisa Jarak : 10.00 km
Persentase : 116.7%
TARGET HARIAN TERCAPAI!
Total lari hari ini: 70.0 km (Target: 60.0 km)
Anda telah melewati target harian sebesar 10.0 km!


Run Analyzer Pro
RUN ANALYZER PRO
Input Hasil Gizi Jadwal History
KALORI TERBAKAR
914 kalori
Input terbaru: 20.0 km pada 17:55
REKOMENDASI NUTRISI
Dada Ayam (100g)
31g protein Protein tinggi, rendah lemak
Telur (2 butir)
13g protein Protein lengkap, mudah dicerna
Salmon (100g)
25g protein Protein + Omega-3
Tahu (100g)
8g protein Protein nabati
Toggle Theme
Clear History


Run Analyzer Pro
RUN ANALYZER PRO
Input Hasil Gizi Jadwal History
8g protein Protein nabati
Nasi Merah (100g)
23g karbo Karbohidrat kompleks
Oatmeal (50g)
30g karbo Serat tinggi, energi tahan lama
Ubi (100g)
20g karbo Vitamin A, karbo sehat
Pisang (1 buah)
27g karbo Kalium, energi cepat
TIPS CEPAT
✓ Protein dalam 30 menit setelah lari
✓ Minum air 500ml setiap 30 menit lari
✓ Karbohidrat kompleks sebelum lari
✓ Hindari makanan berat 2 jam sebelum lari
Toggle Theme
Clear History


Run Analyzer Pro
RUN ANALYZER PRO
Input Hasil Gizi Jadwal History
JADWAL LATIHAN
Senin Lari Ringan 30 menit | Pemanasan
Selasa Interval Run 45 menit | Kecepatan
Rabu Recovery 20 menit | Pemulihan
Kamis Long Run 60 menit | Daya tahan
Jumat Cross Training 40 menit | Variasi
Sabtu Tempo Run 50 menit | Konsistensi
Minggu Rest Day - | Pemulihan total
Toggle Theme
Clear History


Run Analyzer Pro
RUN ANALYZER PRO
Input Hasil Gizi Jadwal History
Riwayat Analisis (1 hari, 3 sesi lari)
2025-12-19
(3 sesi)
Toggle Theme
Clear History


Detail 2025-12-19
Detail Tanggal: 2025-12-19
SUMMARY - 2025-12-19
• Target Harian: 60.0 km
• Total Jarak: 70.00 km
• Jumlah Sesi: 3 sesi
DETAIL SESI LARI
Sesi #1 - 16:57
Jarak: 30.0 km
Speed: 24.0 km/jam
Waktu: 75.0 menit
Kalori: 1371 kal
Pace: 2.50 menit/km
Total Harian: 30.0 km
Sesi #2 - 16:58
Jarak: 20.0 km
Speed: 20.0 km/jam
Waktu: 60.0 menit
Kalori: 980 kal
Pace: 3.00 menit/km
Total Harian: 50.0 km
Sesi #3 - 17:55
Jarak: 20.0 km
Speed: 17.1 km/jam
Waktu: 70.0 menit
Kalori: 914 kal
Pace: 3.50 menit/km
Total Harian: 70.0 km
Hapus Data Tanggal Ini