Laporan Pengubahan Kode

Menambahkan fitur status pada jadwal jika memenuhi target yang diharapkan.

Nama : DONI WAHANA PUTRA

Kelas : 2025G

Nim : 25031554248

1. kode awal

```
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
```

Kode yang diubah :

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

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

class RunningApp(tk.Tk):
   def __init__(self):
       super().__init__()
       self.title("Run Analyzer Pro")
       self.geometry("850x750")
       self.mode = "dark"

       # Variabel untuk input di tab Jadwal
       self.jadwal_vars = {
```

```
            "jarak": tk.StringVar(),
            "waktu": tk.StringVar(),
            "berat": tk.StringVar(),
            "hari": tk.StringVar(),
            "target_mingguan": tk.StringVar(value="50.0")
        }

        # Data storage
        self.history = {}
        self.daily_distances = {}
        self.daily_times = {}
        self.schedule_achievements = {}
        self.target_mingguan = 50.0

        # Setup data
        self.setup_schedule_data()
        self.make_gui()
        self.apply_theme()

    def setup_schedule_data(self):
        """Setup data jadwal latihan dengan target dan rekomendasi"""
        self.schedule_data = {
```

```
"Senin": {
    "latihan": "Lari Ringan",
    "durasi": "30",
    "target_jarak": 5.0,
    "target_waktu": "30",
    "tipe": "Pemanasan",
    "tips": "Fokus pada pernapasan dan postur tubuh"
},
"Selasa": {
    "latihan": "Interval Run",
    "durasi": "45",
    "target_jarak": 7.0,
    "target_waktu": "45",
    "tipe": "Kecepatan",
    "tips": "Gunakan pola 5 menit cepat, 2 menit recovery"
},
"Rabu": {
    "latihan": "Recovery Run",
    "durasi": "25",
    "target_jarak": 4.0,
    "target_waktu": "25",
    "tipe": "Pemulihan",
```

```
    "tips": "Lari santai, dengarkan tubuh Anda"
},
"Kamis": {
    "latihan": "Tempo Run",
    "durasi": "40",
    "target_jarak": 6.0,
    "target_waktu": "40",
    "tipe": "Ketahanan",
    "tips": "Pertahankan pace konsisten sepanjang lari"
},
"Jumat": {
    "latihan": "Cross Training",
    "durasi": "35",
    "target_jarak": 5.0,
    "target_waktu": "35",
    "tipe": "Variasi",
    "tips": "Bisa kombinasi lari, jalan, atau latihan ringan"
},
"Sabtu": {
    "latihan": "Long Run",
    "durasi": "60",
    "target_jarak": 10.0,
```

```
            "target_waktu": "60",
            "tipe": "Daya Tahan",
            "tips": "Pertahankan pace konsisten sepanjang lari"
        },
        "Minggu": {
            "latihan": "Rest Day",
            "durasi": "0",
            "target_jarak": 0.0,
            "target_waktu": "0",
            "tipe": "Pemulihan total",
            "tips": "Istirahat untuk pemulihan otot"
        }
    }

    # Inisialisasi pencapaian jadwal
     hari_list = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu",
"Minggu"]
    for hari in hari_list:
        schedule = self.schedule_data[hari]
        self.schedule_achievements[hari] = {
            "completed": False,
            "actual_distance": 0.0,
```

```
            "actual_time": 0.0,
            "target_distance": schedule["target_jarak"],
            "target_time": float(schedule["target_waktu"]),
            "persentase_jarak": 0.0,
            "persentase_waktu": 0.0,
            "kontribusi_mingguan": 0.0
        }

    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()
```

```
        self.make_hasil_tab()
        self.make_gizi_tab()
        self.make_jadwal_tab()
        self.make_history_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"])

        # Update tabs yang perlu diupdate
        for widget in self.tabs["Jadwal"].winfo_children():
            widget.destroy()
        for widget in self.tabs["History"].winfo_children():
            widget.destroy()

        self.make_jadwal_tab()
        self.make_history_tab()
```

```
        # Update display jika ada data
        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")

        # Variabel untuk tab Input
                self.input_vars = {x: tk.StringVar() for x in
["jarak","waktu","berat","target_jarak"]}

        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.input_vars[key], font=("Arial",12),
                     bg=t["card"], fg=t["fg"]).pack(fill="x", pady=3)

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

    def make_hasil_tab(self):
        """Buat tab Hasil - TIDAK DIUBAH (seperti kode asli)"""
        # Tab Hasil dibuat kosong, akan diisi saat analisis
        pass

    def make_gizi_tab(self):
        """Buat tab Gizi - TIDAK DIUBAH (seperti kode asli)"""
        # Tab Gizi dibuat kosong, akan diisi saat analisis
        pass

    def make_jadwal_tab(self):
        """Buat tab Jadwal dengan form input dan tampilan jadwal"""
        t = THEME[self.mode]
```

```
        tab = self.tabs["Jadwal"]

        # Frame utama dengan 2 bagian
        main_frame = tk.Frame(tab, bg=t["bg"])
        main_frame.pack(fill="both", expand=True, padx=10, pady=10)

        # ===== BAGIAN 1: INPUT FORM =====
          input_frame = tk.LabelFrame(main_frame, text=" INPUT DATA LARI",
                                      bg=t["frame"], fg="#ffd166",
                                      font=("Arial", 12, "bold"),
                                      padx=15, pady=15)
        input_frame.pack(fill="x", pady=(0, 15))

        # Pilihan Hari
        hari_frame = tk.Frame(input_frame, bg=t["frame"])
        hari_frame.pack(fill="x", pady=(0, 10))

        tk.Label(hari_frame, text="Pilih Hari:", bg=t["frame"], fg=t["fg"],
                 font=("Arial", 11)).pack(side="left", padx=(0, 10))

         hari_list = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
```

```
                hari_combo = ttk.Combobox(hari_frame, textvariable=self.jadwal_vars["hari"],
                                          values=hari_list, state="readonly", width=15,
                                          font=("Arial", 11))
        hari_combo.pack(side="left")
        hari_combo.set(hari_list[date.today().weekday()])

        # Info target hari yang dipilih
        self.day_info_label = tk.Label(hari_frame, text="", bg=t["frame"], fg="#4ecdc4",
                                       font=("Arial", 10))
        self.day_info_label.pack(side="left", padx=(15, 0))

        # Bind event untuk update info
        hari_combo.bind("<<ComboboxSelected>>", self.update_day_info_jadwal)

        # Update info awal
        self.update_day_info_jadwal()

        # Input Jarak
        jarak_frame = tk.Frame(input_frame, bg=t["frame"])
        jarak_frame.pack(fill="x", pady=(0, 10))
```

```
        tk.Label(jarak_frame, text="Jarak (km):", bg=t["frame"], fg=t["fg"],
                 font=("Arial", 11), width=12).pack(side="left")

        tk.Entry(jarak_frame, textvariable=self.jadwal_vars["jarak"],
                                 font=("Arial", 12), bg=t["card"], fg=t["fg"],
width=15).pack(side="left", padx=(10, 0))

        # Input Waktu
        waktu_frame = tk.Frame(input_frame, bg=t["frame"])
        waktu_frame.pack(fill="x", pady=(0, 10))

            tk.Label(waktu_frame, text="Waktu (menit):", bg=t["frame"],
fg=t["fg"],
                 font=("Arial", 11), width=12).pack(side="left")

        tk.Entry(waktu_frame, textvariable=self.jadwal_vars["waktu"],
                                 font=("Arial", 12), bg=t["card"], fg=t["fg"],
width=15).pack(side="left", padx=(10, 0))

        # Input Berat
        berat_frame = tk.Frame(input_frame, bg=t["frame"])
        berat_frame.pack(fill="x", pady=(0, 10))
```

```
        tk.Label(berat_frame, text="Berat (kg):", bg=t["frame"], fg=t["fg"],
                 font=("Arial", 11), width=12).pack(side="left")

        tk.Entry(berat_frame, textvariable=self.jadwal_vars["berat"],
                                 font=("Arial", 12), bg=t["card"], fg=t["fg"],
width=15).pack(side="left", padx=(10, 0))

        # Target Mingguan - BISA DIUBAH
        target_frame = tk.Frame(input_frame, bg=t["frame"])
        target_frame.pack(fill="x", pady=(0, 15))

       tk.Label(target_frame, text="Target Mingguan (km):", bg=t["frame"],
fg=t["fg"],
                 font=("Arial", 11), width=18).pack(side="left")

        # Entry untuk target mingguan yang bisa diubah
                                 self.target_entry = tk.Entry(target_frame,
textvariable=self.jadwal_vars["target_mingguan"],
                font=("Arial", 12), bg=t["card"], fg=t["fg"], width=15)
        self.target_entry.pack(side="left", padx=(10, 0))
```

```
        # Button untuk update target mingguan saja (tanpa harus input data lari)
        update_target_btn = tk.Button(target_frame, text="Update Target",
                                      command=self.update_target_mingguan_saja,
                                     bg="#4ecdc4", fg="white", font=("Arial", 9, "bold"),
                                      padx=10, cursor="hand2")
        update_target_btn.pack(side="left", padx=(10, 0))

        # Button Analisis
        btn_frame = tk.Frame(input_frame, bg=t["frame"])
        btn_frame.pack(fill="x")

            tk.Button(btn_frame, text="📊 SIMPAN DATA LARI", command=self.analyze_from_jadwal,
            bg="#ff6b6b", fg="white", font=("Arial", 11, "bold"),
            pady=8, cursor="hand2").pack(fill="x")

        # ===== BAGIAN 2: JADWAL MINGGUAN =====
            jadwal_frame = tk.LabelFrame(main_frame, text="📅 JADWAL LATIHAN MINGGUAN",
                                         bg=t["frame"], fg="#ffd166",
                                         font=("Arial", 12, "bold"),
                                         padx=15, pady=15)
```

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

    # Canvas untuk scroll
    canvas = tk.Canvas(jadwal_frame, bg=t["frame"], highlightthickness=0)
    scrollbar = ttk.Scrollbar(jadwal_frame, orient="vertical", command=canvas.yview)
    scrollable_frame = tk.Frame(canvas, bg=t["frame"])

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

    canvas.create_window((0, 0), window=scrollable_frame, anchor="nw", width=750)
    canvas.configure(yscrollcommand=scrollbar.set)

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

    # Container untuk konten jadwal
```

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

        # Bind mouse wheel untuk scroll
        def _on_mousewheel_jadwal(event):
            canvas.yview_scroll(int(-1*(event.delta/120)), "units")
        canvas.bind_all("<MouseWheel>", _on_mousewheel_jadwal)

    def make_history_tab(self):
        """Buat tab History - TIDAK DIUBAH (seperti kode asli)"""
        # Tab History dibuat kosong, akan diisi saat analisis
        pass

    def update_day_info_jadwal(self, event=None):
        """Update informasi target untuk hari yang dipilih di tab Jadwal"""
        hari = self.jadwal_vars["hari"].get()
        if hari and hari in self.schedule_data:
            schedule = self.schedule_data[hari]
            if schedule["latihan"] != "Rest Day":
                info = f"🎯 Target: {schedule['target_jarak']} km | ⏱️ Waktu: {schedule['durasi']} menit"
                self.day_info_label.config(text=info, fg="#4ecdc4")
```

```
            else:
                self.day_info_label.config(text="😴 Hari Istirahat", fg="#888")
        else:
            self.day_info_label.config(text="")

    def update_target_mingguan_saja(self):
        """Hanya update target mingguan tanpa harus input data lari"""
        try:
            target_mingguan = float(self.jadwal_vars["target_mingguan"].get())

            if target_mingguan <= 0:
                raise ValueError("Target mingguan harus lebih dari 0")

        except ValueError as e:
            return messagebox.showerror("Error", f"Input tidak valid! {str(e)}")

        # Update target mingguan
        self.target_mingguan = target_mingguan

        # Update tampilan jadwal
        self.update_jadwal_display()
```

```
        messagebox.showinfo("Berhasil", f"Target mingguan berhasil diubah menjadi {target_mingguan:.1f} km!")


    def analyze_from_input(self):
        """Analisis dari tab Input"""
        try:
            j = float(self.input_vars["jarak"].get())
            w = float(self.input_vars["waktu"].get())
            b = float(self.input_vars["berat"].get())
                        t = float(self.input_vars["target_jarak"].get()) if self.input_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 ValueError:
            return messagebox.showerror("Error", "Input tidak valid! Pastikan semua angka positif.")

        # Update target mingguan jika ada input target harian
```

```
        if t is not None:
            # Konversi target harian ke target mingguan (asumsi 7 hari)
            self.target_mingguan = t * 7
            # Update juga di variabel tab Jadwal
            self.jadwal_vars["target_mingguan"].set(str(self.target_mingguan))

        # Simpan data menggunakan hari ini
        today = date.today()
        hari_list = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
        hari_nama = hari_list[today.weekday()]

        # Panggil fungsi analyze dengan parameter
        self._process_analysis(j, w, b, hari_nama)
        self.notebook.select(1)

    def analyze_from_jadwal(self):
        """Analisis dari tab Jadwal"""
        try:
            # Cek apakah ada input jarak, waktu, dan berat
            jarak_text = self.jadwal_vars["jarak"].get().strip()
            waktu_text = self.jadwal_vars["waktu"].get().strip()
```

```
        berat_text = self.jadwal_vars["berat"].get().strip()

        # Jika semua field kosong, hanya update target mingguan
        if not jarak_text and not waktu_text and not berat_text:
            return self.update_target_mingguan_saja()

        # Jika ada input, validasi semua field harus terisi
        if not jarak_text or not waktu_text or not berat_text:
            raise ValueError("Semua field (jarak, waktu, berat) harus diisi untuk menyimpan data lari")

        j = float(jarak_text)
        w = float(waktu_text)
        b = float(berat_text)
        hari_nama = self.jadwal_vars["hari"].get()

        # Cek target mingguan (opsional, default pakai yang ada)
        target_mingguan_text = self.jadwal_vars["target_mingguan"].get().strip()
        if target_mingguan_text:
            target_mingguan = float(target_mingguan_text)
            if target_mingguan <= 0:
                raise ValueError("Target mingguan harus lebih dari 0")
```

```
            self.target_mingguan = target_mingguan


        if min(j, w, b) <= 0:
            raise ValueError("Jarak, waktu, dan berat harus lebih dari 0")
        if not hari_nama:
            raise ValueError("Pilih hari terlebih dahulu!")


    except ValueError as e:
            return messagebox.showerror("Error", f"Input tidak valid! {str(e)}")


    # Proses analisis
    self._process_analysis(j, w, b, hari_nama)
    self.notebook.select(1)


    # Clear input
    self.jadwal_vars["jarak"].set("")
    self.jadwal_vars["waktu"].set("")
    self.jadwal_vars["berat"].set("")


    messagebox.showinfo("Berhasil", f"Data untuk {hari_nama} berhasil disimpan!")
```

```
def _process_analysis(self, jarak, waktu, berat, hari_nama):
    """Proses analisis data lari"""
    # Hitung metrics
    self.pace = waktu / jarak
    self.speed = (jarak / waktu) * 60
    self.kal = jarak * berat * 0.653

    # Buat key unik berdasarkan hari
    today_str = date.today().strftime("%Y-%m-%d")
    key = f"{today_str}-{hari_nama}"

    # Simpan data harian
    if key not in self.daily_distances:
        self.daily_distances[key] = 0
        self.daily_times[key] = 0

    self.daily_distances[key] += jarak
    self.daily_times[key] += waktu

    # Update pencapaian jadwal
    self.update_schedule_achievement(hari_nama, key)
```

```
    # Simpan ke history
    if today_str not in self.history:
        self.history[today_str] = []

    self.history[today_str].append({
        "time": datetime.now().strftime("%H:%M"),
        "jarak": jarak,
        "waktu": waktu,
        "pace": self.pace,
        "speed": self.speed,
        "kal": self.kal,
        "hari": hari_nama,
        "total_jarak_harian": self.daily_distances[key],
        "total_waktu_harian": self.daily_times[key]
    })

    # Update semua tab
    self.show_all()

def update_schedule_achievement(self, hari_nama, key):
    """Update pencapaian jadwal berdasarkan input"""
    schedule = self.schedule_data.get(hari_nama, {})
```

```
        if schedule["latihan"] == "Rest Day":
            return

        # Target dari jadwal
        target_jarak = schedule["target_jarak"]
        target_waktu = float(schedule["target_waktu"])

        # Akumulasi data
        total_jarak = self.daily_distances.get(key, 0)
        total_waktu = self.daily_times.get(key, 0)

        # Hitung persentase
        persentase_jarak = (total_jarak / target_jarak) * 100 if target_jarak > 0 else 0
        persentase_waktu = (total_waktu / target_waktu) * 100 if target_waktu > 0 else 0

        # Hitung kontribusi terhadap target mingguan
        kontribusi_mingguan = (total_jarak / self.target_mingguan) * 100 if self.target_mingguan > 0 else 0

        # Tentukan status
```

```
        completed = total_jarak >= target_jarak

        # Update pencapaian
        self.schedule_achievements[hari_nama] = {
            "completed": completed,
            "actual_distance": total_jarak,
            "actual_time": total_waktu,
            "target_distance": target_jarak,
            "target_time": target_waktu,
            "persentase_jarak": persentase_jarak,
            "persentase_waktu": persentase_waktu,
            "kontribusi_mingguan": kontribusi_mingguan
        }

    def update_jadwal_display(self):
        """Update tampilan jadwal mingguan"""
        if not hasattr(self, 'jadwal_container'):
            return

        # Clear container
        for widget in self.jadwal_container.winfo_children():
            widget.destroy()
```

```
t = THEME[self.mode]

# Progress mingguan
total_jarak_mingguan = sum(self.daily_distances.values())
progress_persen = (total_jarak_mingguan / self.target_mingguan) * 100 if self.target_mingguan > 0 else 0

# Info progress
progress_frame = tk.Frame(self.jadwal_container, bg=t["card"], padx=15, pady=12)
progress_frame.pack(fill="x", pady=(0, 15))

tk.Label(progress_frame, text="📊 PROGRESS MINGGUAN",
                         bg=t["card"], fg=t["fg"], font=("Arial", 11, "bold")).pack(anchor="w", pady=(0, 5))

tk.Label(progress_frame, text=f"🎯 Target: {self.target_mingguan:.1f} km | 📈 Total: {total_jarak_mingguan:.1f} km | 📊 {progress_persen:.1f}%",
                         bg=t["card"], fg="#4ecdc4", font=("Arial", 10)).pack(anchor="w")

# Jadwal per hari
```

```
        hari_list = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu",
"Minggu"]

        for hari_nama in hari_list:
            schedule = self.schedule_data[hari_nama]
            achievement = self.schedule_achievements[hari_nama]

            # Frame untuk setiap hari
                day_frame = tk.Frame(self.jadwal_container, bg=t["card"],
padx=15, pady=12)
            day_frame.pack(fill="x", pady=5)

            # Header dengan hari
            header_frame = tk.Frame(day_frame, bg=t["card"])
            header_frame.pack(fill="x", pady=(0, 8))

            # Nama hari
            tk.Label(header_frame, text=hari_nama, bg=t["card"], fg=t["fg"],
                    font=("Arial", 11, "bold")).pack(side="left")

            # Status pencapaian
            if achievement["completed"]:
                status_text = "✅ TERCAPAI"
```

```
            status_color = "#4ecdc4"
        elif achievement["actual_distance"] > 0:
            status_text = f"📊 {achievement['persentase_jarak']:.0f}%"
            status_color = "#ffd166"
        else:
            status_text = "⏳ BELUM"
            status_color = "#888"

                tk.Label(header_frame, text=status_text, bg=t["card"],
fg=status_color,
                 font=("Arial", 10, "bold")).pack(side="right")

        # Jenis latihan
        tk.Label(day_frame, text=f"🏃 {schedule['latihan']}",
                 bg=t["card"], fg="#4ecdc4",
                 font=("Arial", 10, "bold")).pack(anchor="w", pady=(0, 5))

        # Target
        if schedule["latihan"] != "Rest Day":
            target_frame = tk.Frame(day_frame, bg=t["card"])
            target_frame.pack(fill="x", pady=(0, 5))
```

```
                tk.Label(target_frame, text="🎯 Target:", bg=t["card"],
fg=t["fg"],
                font=("Arial", 9, "bold")).pack(side="left", padx=(0, 5))

            tk.Label(target_frame, text=f"Jarak: {schedule['target_jarak']}
km | Waktu: {schedule['durasi']} menit",
                                bg=t["card"], fg="#888", font=("Arial",
9)).pack(side="left", padx=(0, 10))

        # Hasil aktual jika ada
        if achievement["actual_distance"] > 0 and schedule["latihan"] !=
"Rest Day":
            # Data aktual
            actual_frame = tk.Frame(day_frame, bg=t["card"])
            actual_frame.pack(fill="x", pady=(5, 0))

                tk.Label(actual_frame, text="📊 Hasil:", bg=t["card"],
fg=t["fg"],
                font=("Arial", 9, "bold")).pack(side="left", padx=(0, 5))

                                tk.Label(actual_frame,      text=f"Jarak:
{achievement['actual_distance']:.1f} km",
                                bg=t["card"], fg="#4ecdc4", font=("Arial", 9,
"bold")).pack(side="left", padx=(0, 10))
```

```
            # Status detail
            status_frame = tk.Frame(day_frame, bg=t["card"])
            status_frame.pack(fill="x", pady=(5, 0))

            if achievement["completed"]:
                tk.Label(status_frame, text="✅ TARGET TERPENUHI",
bg=t["card"], fg="#4ecdc4",
                         font=("Arial", 9, "bold")).pack(side="left")

            # Tambahkan kontribusi mingguan
            kontribusi_frame = tk.Frame(day_frame, bg=t["card"])
            kontribusi_frame.pack(fill="x", pady=(3, 0))

                        tk.Label(kontribusi_frame, text=f"➕ Kontribusi:
{achievement['kontribusi_mingguan']:.1f}% dari target mingguan",
                                 bg=t["card"], fg="#ffd166", font=("Arial", 8,
"bold")).pack(side="left")

            # Jika melebihi target
                                    if achievement["actual_distance"] >
achievement["target_distance"]:
                                kelebihan = achievement["actual_distance"] -
achievement["target_distance"]
```

```
                bonus_frame = tk.Frame(day_frame, bg=t["card"])
                bonus_frame.pack(fill="x", pady=(3, 0))
                    tk.Label(bonus_frame, text=f"⭐ Melebihi target: +{kelebihan:.1f} km",
                             bg=t["card"], fg="#ffd166", font=("Arial", 8, "bold")).pack(side="left")
            else:
                sisa = achievement["target_distance"] - achievement["actual_distance"]
                tk.Label(status_frame, text=f"⚠️ Kurang {sisa:.1f} km", bg=t["card"], fg="#ff6b6b",
                         font=("Arial", 9, "bold")).pack(side="left")

        elif schedule["latihan"] == "Rest Day":
            # Hari istirahat
            rest_frame = tk.Frame(day_frame, bg=t["card"])
            rest_frame.pack(fill="x", pady=(5, 0))
            tk.Label(rest_frame, text="😴 Hari istirahat - Fokus pemulihan",
                     bg=t["card"], fg="#888", font=("Arial", 9, "italic")).pack()

        # Tips
        tips_frame = tk.Frame(day_frame, bg=t["card"])
        tips_frame.pack(fill="x", pady=(5, 0))
```

```
            tk.Label(tips_frame, text=f"💡 {schedule['tips']}", bg=t["card"],
fg="#888",
                                    font=("Arial", 8), wraplength=680,
justify="left").pack(anchor="w")

            # Separator
            if hari_nama != "Minggu":
                separator = tk.Frame(day_frame, height=1,
                                     bg="#444444" if self.mode=="dark" else "#cccccc")
                separator.pack(fill="x", pady=(10, 0))

    def show_all(self):
        """Update semua tab"""
        self.show_hasil()
        self.show_gizi()
        self.update_jadwal_display()
        self.show_history()

    def show_hasil(self):
        """Tampilkan hasil analisis - TIDAK DIUBAH (seperti kode asli)"""
        if not hasattr(self, "pace"):
            return
```

```
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):
        """Tampilkan informasi gizi - TIDAK DIUBAH (seperti kode asli)"""
        if not hasattr(self, "pace"):
            return

        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))

        # Info sederhana
        info_frame = tk.Frame(container, bg=t["card"], padx=20, pady=15)
        info_frame.pack(fill="x", pady=(0, 20))
```

```
            tk.Label(info_frame, text=f"🔥 Kalori Terbakar: {self.kal:.0f} kalori",
                                bg=t["card"], fg="#ff6b6b", font=("Arial", 12, "bold")).pack(anchor="w", pady=(0, 5))

        tk.Label(info_frame, text=f"⏱️ Pace: {self.pace:.2f} menit/km | ⚡ Kecepatan: {self.speed:.1f} km/jam",
                                  bg=t["card"], fg="#4ecdc4", font=("Arial", 10)).pack(anchor="w")

    def show_history(self):
        """Tampilkan riwayat - TIDAK DIUBAH (seperti kode asli)"""
        if not hasattr(self, "pace"):
            return

        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)
```

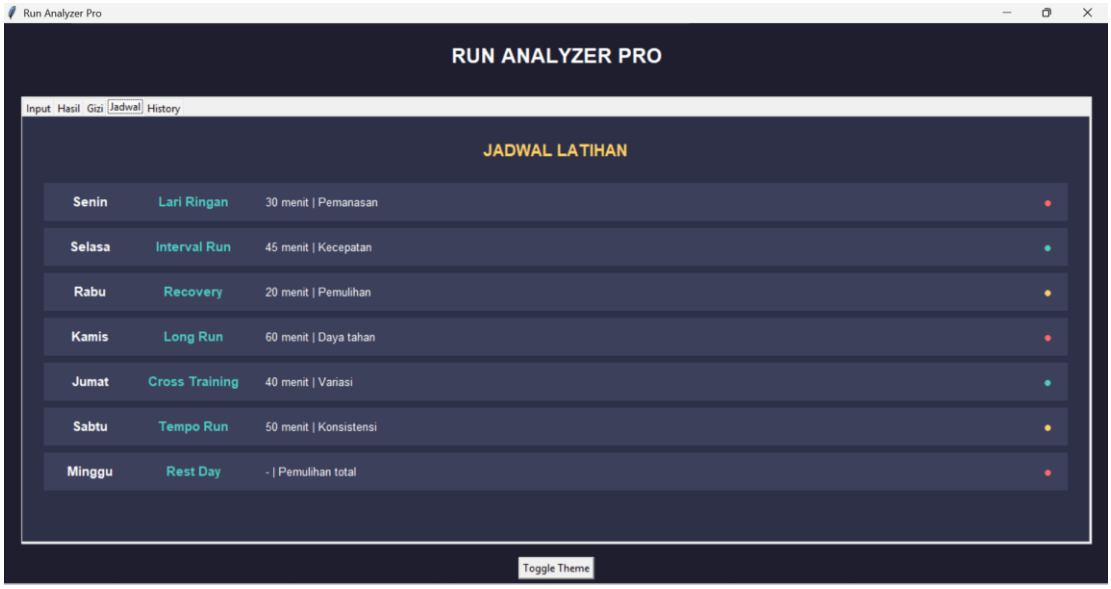
```
        tk.Label(f, text="Riwayat Analisis", bg=t["frame"],
                 fg="#ffd166", font=("Arial",14,"bold")).pack(pady=(0,20))

        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):
        """Tampilkan detail tanggal - TIDAK DIUBAH (seperti kode asli)"""
        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"])
```

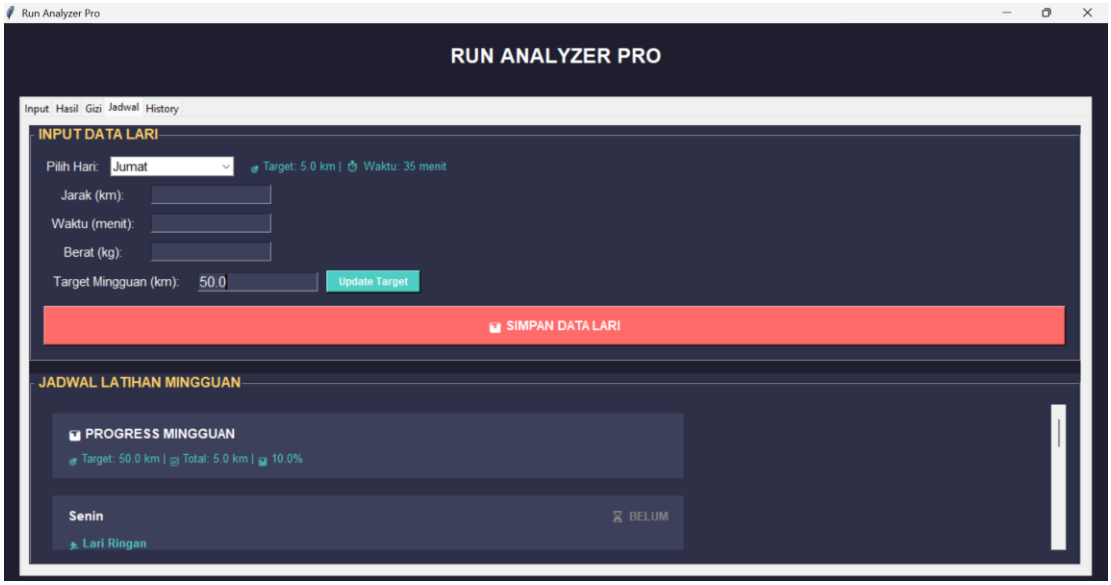
```
        center_frame.pack(expand=True)

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

        total_jarak = 0
        for key, value in self.daily_distances.items():
            if key.startswith(tanggal):
                total_jarak += value

        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"Total Jarak: {total_jarak:.1f} km",
                 bg=t["card"], fg=t["fg"], font=("Arial",10, "bold")).pack()

        if tanggal in self.history:
            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()
```

Tampilan Kode Awal :



Tampilan Setelah Di ubah



Dibagian target mingguan bisa diubah jika dibelakang setelah jarak yang diiginkan ditambahkan .0, contoh jarak 50km maka 50.0