

RUN ANALYZER PRO

“Aplikasi Analisis Performa Lari dengan Rekomendasi Personal”



Oleh:

Nama Kelompok : Kelompok 8

Anggota/NIM : Denielson Javier Latuparissa / 25031554065

Arya Bima Nugraha / 25031554238

Doni Wahana Putra / 25031554248

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

DAFTAR ISI

DAFTAR ISI.....	Error! Bookmark not defined.
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
BAB II ANALISIS DAN PERENCANAAN	3
2.1 Analisis Kebutuhan Aplikasi	3
2.1.1 Kebutuhan Fungsional	3
2.1.2 Kebutuhan Non-Fungsional	4
2.2 Diagram Alur	5
2.3 Sketsa Desain Antarmuka.....	6
2.3.1 Tampilan Aplikasi Utama	6
2.3.2 Tampilan Tab Input.....	6
2.3.3 Tampilan Tab Hasil.....	7
2.3.4 Tampilan Tab Gizi	7
2.3.5 Tampilan Tab Jadwal	7
2.3.6 Tampilan Tab History	8
2.3.7 Desain Visual dan Tema	8
BAB III IMPLEMENTASI.....	9
3.1 Struktur Program	9
3.2 Cuplikan Kode Penting dan Penjelasannya	10
3.2.1 Konfigurasi Program dan Tema Aplikasi	10
3.2.2 Instalasi Kelas dan Struktur Data Utama	11
3.2.3 Pembuatan Antarmuka dan Tab Aplikasi	11
3.2.4 Input Data Aktivitas Lari	11
3.2.5 Proses Analisis dan Perhitungan Data.....	12
3.2.6 Penyimpanan Riwayat Aktivitas.....	12
3.2.7 Menampilkan Hasil Analisis	12

3.2.8	Penampilan Riwayat dan Detail Aktivitas	12
3.3	Manual Penggunaan	13
3.4	Screenshot Aplikasi	14
LAMPIRAN	18
DAFTAR PUSTAKA	43

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kesadaran masyarakat terhadap pentingnya pola hidup sehat terus mengalami peningkatan, yang ditandai dengan semakin banyaknya individu yang rutin berolahraga dan memperhatikan asupan gizi. Lari menjadi salah satu aktivitas olahraga yang paling diminati karena mudah dilakukan, tidak memerlukan peralatan khusus, serta dapat disesuaikan dengan kemampuan masing-masing individu. Namun demikian, banyak pelari pemula yang masih kesulitan dalam memahami performa lari mereka, seperti menghitung pace, kecepatan, serta memperkirakan kebutuhan energi yang dibutuhkan untuk menunjang aktivitas tersebut.

Permasalahan lain yang sering ditemui adalah kurangnya pemahaman mengenai hasil lari yang telah dilakukan. Banyak pengguna hanya mencatat jarak dan waktu tanpa mengetahui apakah performa tersebut sudah sesuai dengan target kebugaran. Perhitungan secara manual dinilai kurang praktis dan berpotensi menimbulkan kesalahan. Di sisi lain, aplikasi olahraga yang tersedia saat ini umumnya memiliki fitur yang kompleks, memerlukan registrasi, serta bergantung pada koneksi internet, sehingga kurang sesuai bagi pengguna pemula yang membutuhkan kemudahan dan kecepatan analisis.

Berdasarkan kondisi tersebut, dikembangkanlah aplikasi RUN ANALYZER, yaitu aplikasi desktop berbasis Python dengan antarmuka GUI Tkinter yang dirancang untuk membantu pengguna dalam menganalisis aktivitas lari secara sederhana namun tetap informatif. Aplikasi ini memungkinkan pengguna untuk memasukkan data dasar berupa jarak tempuh lari, waktu tempuh, dan berat badan. Data tersebut kemudian diolah secara otomatis untuk menghasilkan analisis instan yang meliputi perhitungan pace, kecepatan rata-rata, serta estimasi kalori yang terbakar selama sesi lari.

Hasil analisis pace digunakan untuk mengelompokkan performa pengguna ke dalam beberapa kategori, mulai dari pemula hingga elite, yang masing-masing dilengkapi dengan rekomendasi program latihan selama satu minggu sesuai dengan tingkat kemampuan. Selain itu, aplikasi ini juga memberikan

panduan kebutuhan gizi harian berupa rekomendasi asupan protein, karbohidrat, dan cairan yang disesuaikan dengan berat badan serta aktivitas lari yang dilakukan. Dengan menggabungkan analisis performa, rekomendasi latihan, dan panduan nutrisi dalam satu aplikasi yang dapat digunakan secara offline tanpa login, RUN ANALYZER diharapkan mampu membantu pengguna menjalani aktivitas lari secara lebih terarah dan mendukung pencapaian target kebugaran secara bertahap.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah pada proyek ini adalah sebagai berikut:

1. Bagaimana merancang aplikasi sederhana yang dapat menghitung pace dan kecepatan secara otomatis, menampilkan analisis performa, serta memberikan rekomendasi gizi jelas berdasarkan hasil lari?
2. Bagaimana membuat tampilan aplikasi tetap modern namun tetap ringan dan mudah digunakan?

1.3 Tujuan

Membantu pelari pemula hingga menengah untuk:

1. Menganalisis performa lari (pace, kecepatan, kalori)
2. Mendapatkan rekomendasi latihan berdasarkan level
3. Memahami kebutuhan gizi untuk user

Memantau progress secara sederhana

BAB II

ANALISIS DAN PERENCANAAN

2.1 Analisis Kebutuhan Aplikasi

RUN ANALYZER PRO merupakan aplikasi desktop berbasis Python yang dirancang untuk membantu pengguna dalam menganalisis aktivitas lari secara sederhana dan informatif. Aplikasi ini menerima input berupa jarak tempuh lari, waktu tempuh, berat badan pengguna, serta target jarak harian. Data tersebut kemudian diproses untuk menghasilkan informasi performa lari, progres target harian, estimasi kalori terbakar, jadwal latihan, serta rekomendasi nutrisi pendukung.

Aplikasi dirancang untuk dapat digunakan secara offline, tanpa memerlukan koneksi internet maupun proses login, sehingga cocok digunakan oleh pelari pemula hingga menengah.

2.1.1 Kebutuhan Fungsional

Kebutuhan fungsional aplikasi RUN ANALYZER PRO terdiri dari:

1. Aplikasi dapat menerima input:
 - Jarak lari (kilometer)
 - Waktu tempuh (menit)
 - Berat badan pengguna (kilogram)
 - Target jarak harian (kilometer)
2. Sistem dapat melakukan validasi input dan menampilkan pesan kesalahan jika data tidak valid.
3. Sistem dapat menghitung secara otomatis:
 - Pace lari (menit/km)
 - Kecepatan rata-rata (km/jam)
 - Estimasi kalori yang terbakar
4. Sistem dapat menyimpan target jarak harian berdasarkan tanggal.
5. Sistem dapat menghitung akumulasi jarak lari harian.

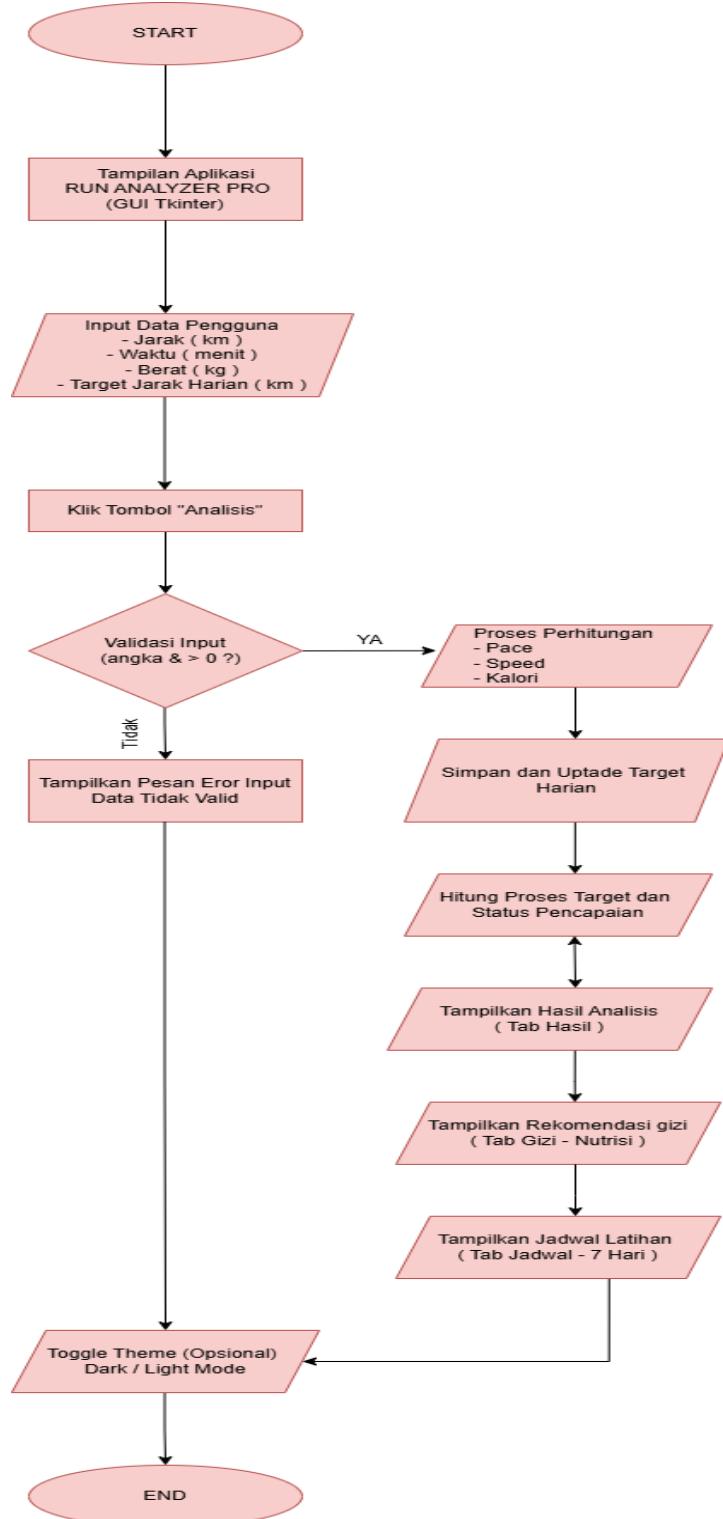
6. Sistem dapat menampilkan progres pencapaian target harian dalam bentuk:
 - Total jarak hari ini
 - Sisa jarak
 - Persentase pencapaian
7. Sistem dapat menampilkan status target harian (tercapai atau belum tercapai).
8. Aplikasi dapat menampilkan rekomendasi nutrisi dan tips pendukung aktivitas lari.
9. Aplikasi menyediakan jadwal latihan mingguan.
10. Aplikasi menyimpan riwayat aktivitas lari harian dan menampilkan detail setiap sesi lari.
11. Aplikasi menyediakan fitur penggantian tema tampilan (dark mode dan light mode).

2.1.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional aplikasi RUN ANALYZER PRO adalah sebagai berikut:

- Aplikasi berjalan pada sistem operasi desktop (Windows).
- Aplikasi dapat digunakan secara offline.
- Antarmuka sederhana dan mudah dipahami.
- Tampilan responsif dan nyaman digunakan.
- Aplikasi memiliki performa ringan dan cepat.
- Mudah digunakan oleh pengguna pemula

2.2 Diagram Alur



2.3 Sketsa Desain Antarmuka

Desain antarmuka RUN ANALYZER PRO menggunakan GUI berbasis Tkinter dengan sistem tab (Notebook) untuk memisahkan setiap fitur utama. Pendekatan ini memudahkan pengguna dalam berpindah antar menu tanpa kebingungan. Warna dan tata letak disesuaikan agar nyaman digunakan dalam jangka waktu lama.

2.3.1 Tampilan Aplikasi Utama

Aplikasi berjalan pada window desktop berukuran 700x750 pixel. Pada bagian atas terdapat judul “RUN ANALYZER PRO”. Navigasi aplikasi menggunakan sistem tab dengan lima menu utama, yaitu:

- Input
- Hasil
- Gizi dan Progres
- Jadwal
- History

Di bagian bawah terdapat tombol Toggle Theme untuk mengganti tampilan dark mode dan light mode.

2.3.2 Tampilan Tab Input

Tab Input berfungsi untuk memasukkan data lari pengguna. Komponen yang tersedia meliputi:

- Input jarak lari (km)
- Input waktu tempuh (menit)
- Input berat badan (kg)
- Input target jarak harian (km)
- Tombol **Analisis**

Jika input tidak valid (kosong atau bernilai nol), sistem akan menampilkan pesan kesalahan.

2.3.3 Tampilan Tab Hasil

Tab hasil menampilkan ringkasan analisis lari berupa:

- Pace (menit/km)
- Kecepatan (km/jam)
- Kalori terbakar

Informasi ditambahkan dalam bentuk kartu agar mudah dipahami

2.3.4 Tampilan Tab Gizi dan Progres

Tab Gizi menampilkan informasi progres dan nutrisi, antara lain:

- Target jarak harian
- Total jarak hari ini
- Sisa jarak
- Persentase pencapaian target
- Status target (tercapai atau belum)

Selain itu, tab ini juga menampilkan:

- Estimasi kalori terbakar
- Rekomendasi makanan tinggi protein dan karbohidrat
- Tips nutrisi pendukung aktivitas lari

2.3.5 Tampilan Tab Jadwal

Tab Jadwal menampilkan jadwal latihan mingguan dari Senin hingga Minggu, meliputi:

- Lari ringan
- Interval run
- Recovery
- Long run
- Cross training
- Tempo run

- Rest day

2.3.6 Tampilan Tab History

Tab History menyimpan riwayat aktivitas lari berdasarkan tanggal. Pengguna dapat memilih tanggal tertentu untuk melihat detail aktivitas lari yang dilakukan pada hari tersebut, termasuk:

- Waktu lari
- Jarak tempuh
- Durasi
- Pace
- Kalori
- Target dan total jarak harian

2.3.7 Desain Visual dan Tema

Aplikasi menyediakan dua mode tampilan:

- Dark Mode
- Light Mode

Penggunaan dapat mengganti tema kapan saja menggunakan tombol Toggle Them

BAB III

IMPLEMENTASI

3.1 Struktur Program

Program RUN ANALYZER PRO terdiri dari beberapa bagian utama yang saling terhubung untuk membantu pengguna menganalisis aktivitas lari dan menampilkan informasi secara interaktif.

1. Konfigurasi Program

Bagian konfigurasi digunakan untuk menyiapkan aplikasi sebelum dijalankan. Pada tahap ini, program memanggil library yang diperlukan seperti Tkinter dan ttk untuk antarmuka, messagebox untuk menampilkan pesan kesalahan, serta datetime untuk mengatur tanggal dan waktu. Selain itu, ditentukan pengaturan tema tampilan aplikasi melalui dictionary THEME yang menyediakan mode gelap dan terang. Konfigurasi awal juga mencakup penentuan judul aplikasi, ukuran jendela, serta inisialisasi variabel input dan struktur data yang digunakan untuk menyimpan riwayat lari, target harian, dan total jarak lari per hari.

2. Inisialisasi dan Pembuatan Antarmuka (GUI)

Antarmuka aplikasi dibuat menggunakan konsep pemrograman berorientasi objek melalui kelas RunningApp yang merupakan turunan dari tk.Tk. Tampilan aplikasi dibagi ke dalam lima tab utama menggunakan ttk.Notebook, yaitu Input, Hasil, Gizi, Jadwal, dan History. Pada bagian ini juga ditampilkan judul aplikasi di bagian atas jendela serta tombol Toggle Theme yang memungkinkan pengguna mengganti tampilan aplikasi antara mode gelap dan terang.

3. Input Data Aktivitas Lari

Bagian input data berfungsi untuk menerima informasi aktivitas lari dari pengguna. Pada tab Input, pengguna dapat memasukkan jarak lari, waktu tempuh, berat badan, serta target jarak harian. Data dimasukkan melalui kolom isian dan disimpan dalam variabel StringVar, sehingga data dapat diolah dan diperbarui secara langsung oleh sistem.

4. Proses Analisis dan Pengolahan Data

Pada tahap ini, sistem memeriksa data yang dimasukkan untuk memastikan nilainya valid dan lebih dari nol. Jika data tidak sesuai, sistem akan menampilkan pesan kesalahan. Setelah data dinyatakan valid, sistem menghitung nilai pace, kecepatan rata-rata, dan perkiraan kalori yang terbakar. Selain itu, sistem juga mencatat target jarak harian, menghitung total jarak lari dalam satu hari, serta menyimpan setiap aktivitas lari ke dalam riwayat berdasarkan tanggal dan waktu.

5. Penyimpanan Data dan Manajemen Riwayat

Setiap hasil analisis disimpan dalam bentuk dictionary yang dikelompokkan berdasarkan tanggal. Dalam satu hari, pengguna dapat memiliki lebih dari satu sesi lari. Data target harian dan total jarak lari yang tersimpan digunakan untuk menampilkan perkembangan aktivitas lari serta detail riwayat pada menu History.

6. Penampilan Hasil dan Informasi Pendukung

Hasil analisis ditampilkan pada tab yang sesuai. Tab Hasil menampilkan informasi berupa pace, kecepatan, dan kalori terbakar. Tab Gizi menampilkan progres pencapaian target jarak harian, status target tercapai atau belum, estimasi kalori, serta rekomendasi nutrisi dan tips singkat. Tab Jadwal berisi susunan program latihan mingguan, sedangkan tab History menampilkan riwayat aktivitas lari dan detail setiap sesi berdasarkan tanggal.

3.2 Cuplikan Kode Penting dan Penjelasannya

Subbab ini membahas beberapa bagian kode utama yang berperan penting dalam implementasi aplikasi *Run Analyzer Pro*

3.2.1 Konfigurasi Program dan Tema Aplikasi

```
1 import tkinter as tk
2 from tkinter import ttk, messagebox
3 from datetime import datetime, date
4
5 THEME = {
6     "dark": {"bg": "#1e1e2e", "frame": "#2d3047", "card": "#3d405b", "fg": "white"},
7     "light": {"bg": "#f4f4f4", "frame": "#ffffff", "card": "#e6e6e6", "fg": "black"}
8 }
```

Kode ini digunakan untuk memanggil library yang diperlukan serta

mendefinisikan pengaturan tema aplikasi yang mendukung mode gelap dan terang agar tampilan lebih nyaman bagi pengguna

3.2.2 Instalasi Kelas dan Struktur Data Utama

```
10  class RunningApp(tk.Tk):
11      def __init__(self):
12          super().__init__()
13          self.title("Run Analyzer Pro")
14          self.geometry("700x750")
15          self.mode = "dark"
16          self.vars = {x: tk.StringVar() for x in ["jarak", "waktu", "berat", "target_jarak"]}
17          self.history = {}
18          self.daily_targets = {} # Menyimpan target harian per tanggal
19          self.daily_distances = {} # Menyimpan total jarak harian per tanggal
20          self.make_gui()
21          self.apply_theme()
```

Bagian ini berfungsi untuk membuat jendela utama aplikasi, mengatur ukuran tampilan, serta menyiapkan variabel input dan struktur data untuk menyimpan riwayat aktivitas lari, target harian, dan total jarak per hari.

3.2.3 Pembuatan Antarmuka dan Tab Aplikasi

```
27      self.notebook = ttk.Notebook(self)
28      self.notebook.pack(expand=True, fill="both", padx=20, pady=10)
29
30      self.tabs = {}
31      for name in ["Input", "Hasil", "Gizi", "Jadwal", "History"]:
32          self.tabs[name] = ttk.Frame(self.notebook)
33          self.notebook.add(self.tabs[name], text=name)
```

Kode ini digunakan untuk membagi aplikasi ke dalam lima tab utama sehingga fitur-fitur dapat diakses dengan lebih terstruktur dan mudah dipahami oleh pengguna.

3.2.4 Input Data Aktivitas Lari

```
58      labels = ["Jarak (km)", "Waktu (menit)", "Berat (kg)", "Target Jarak Harian (km)"]
59      keys = ["jarak", "waktu", "berat", "target_jarak"]
60
61      for label, key in zip(labels, keys):
62          tk.Label(f, text=label, bg=t["frame"], fg=t["fg"]).pack(anchor="w", pady=(5,0))
63          tk.Entry(f, textvariable=self.vars[key], font=("Arial",12),
64                    bg=t["card"], fg=t["fg"]).pack(fill="x", pady=3)
65
66          tk.Button(f, text="Analisis", command=self.analyze,
67                     bg="#ff6b6b", fg="white", font=("Arial",11,"bold"),
68                     pady=8).pack(fill="x", pady=20)
```

Bagian ini menyediakan form input untuk memasukkan jarak tempuh, waktu lari, dan berat badan yang nantinya akan diproses oleh sistem.

3.2.5 Proses Analisis dan Perhitungan Data

```
81     today = date.today().strftime("%Y-%m-%d")
82     self.pace = w/j
83     self.speed = (j/w)*60
84     self.kal = j*b*1.036
```

Kode ini digunakan untuk menghitung pace (menit per kilometer), kecepatan rata-rata (km/jam), serta estimasi kalori yang terbakar berdasarkan jarak dan berat badan pengguna.

3.2.6 Penyimpanan Riwayat Aktivitas

```
105     total_jarak_hari_ini = self.daily_distances[today]
106
107     if today not in self.history:
108         self.history[today] = []
109
110     self.history[today].append({
111         "time": datetime.now().strftime("%H:%M"),
112         "jarak": j,
113         "waktu": w,
114         "pace": self.pace,
115         "speed": self.speed,
116         "kal": self.kal,
117         "target": self.target_jarak,
118         "total_jarak_harian": total_jarak_hari_ini
119     })
```

Data hasil analisis disimpan ke dalam dictionary berdasarkan tanggal, sehingga pengguna dapat melihat riwayat aktivitas lari setiap hari.

3.2.7 Menampilkan Hasil Analisis

```
141     data = [
142         ("Pace", f"{self.pace:.2f} menit/km"),
143         ("Kecepatan", f"{self.speed:.1f} km/jam"),
144         ("Kalori Terbakar", f"{self.kal:.0f} kalori")
145     ]
146
147     for label, value in data:
148         frame = tk.Frame(f, bg=t["card"], padx=15, pady=10)
149         frame.pack(fill="x", pady=5)
150         tk.Label(frame, text=label, bg=t["card"], fg=t["fg"],
151                  font=("Arial",11)).pack(side="left")
152         tk.Label(frame, text=value, bg=t["card"], fg="#4ecdc4",
153                  font=("Arial",11,"bold")).pack(side="right")
```

Hasil perhitungan ditampilkan secara langsung kepada pengguna dalam bentuk informasi yang mudah dipahami.

3.2.8 Penampilan Riwayat dan Detail Aktivitas

474 RunningApp().mainloop()

Kode ini digunakan untuk menjalankan aplikasi dan menampilkan antarmuka grafis kepada pengguna.

3.3 Manual Penggunaan

Manual penggunaan ini dibuat untuk membantu pengguna dalam menjalankan dan menggunakan aplikasi RUN ANALYZER PRO. Dengan adanya panduan ini, diharapkan pengguna dapat memahami cara menggunakan fitur-fitur yang tersedia sehingga aplikasi dapat dioperasikan dengan mudah dan sesuai dengan kebutuhan.

1. Menjalankan Aplikasi

Pastikan Python telah terpasang di komputer. Jalankan file program RUN ANALYZER PRO dengan menjalankan file Python melalui terminal atau dengan menjalankannya langsung dari editor kode. Setelah dijalankan, jendela utama aplikasi akan muncul di layar.

2. Tampilan Utama

Pada tampilan utama, pengguna akan melihat judul aplikasi serta beberapa tab, yaitu Input, Hasil, Gizi, Jadwal, dan History. Di bagian bawah aplikasi tersedia tombol Toggle Theme yang digunakan untuk mengganti tampilan aplikasi.

3. Menginput Data Aktivitas Lari

Pilih tab **Input**, kemudian masukkan data jarak lari, waktu tempuh, berat badan, serta target jarak harian jika diperlukan. Pastikan semua data yang dimasukkan berupa angka dan bernilai lebih dari nol.

4. Melakukan Analisis Data

Setelah semua data diisi, tekan tombol **Analisis**. Sistem akan memproses data yang dimasukkan. Jika terdapat kesalahan input, aplikasi akan menampilkan pesan peringatan.

5. Melihat Hasil Analisis

Hasil perhitungan akan ditampilkan pada tab **Hasil**, yang berisi informasi pace lari, kecepatan rata-rata, dan estimasi kalori terbakar.

6. Melihat Progres dan Informasi Gizi

Pilih tab Gizi untuk melihat progres pencapaian target jarak harian, persentase progres, status target, serta rekomendasi nutrisi dan tips singkat yang mendukung aktivitas lari.

7. Melihat Jadwal Latihan

Pada tab Jadwal, pengguna dapat melihat program latihan mingguan yang berisi jenis latihan, durasi, dan tujuan latihan sebagai panduan berolahraga.

8. Melihat Riwayat Aktivitas Lari

Pilih tab History untuk melihat riwayat aktivitas lari berdasarkan tanggal. Klik salah satu tanggal untuk melihat detail aktivitas lari pada hari tersebut.

9. Mengganti Tema Tampilan

Pengguna dapat menekan tombol Toggle Theme untuk mengganti tampilan aplikasi antara mode gelap dan mode terang sesuai kenyamanan.

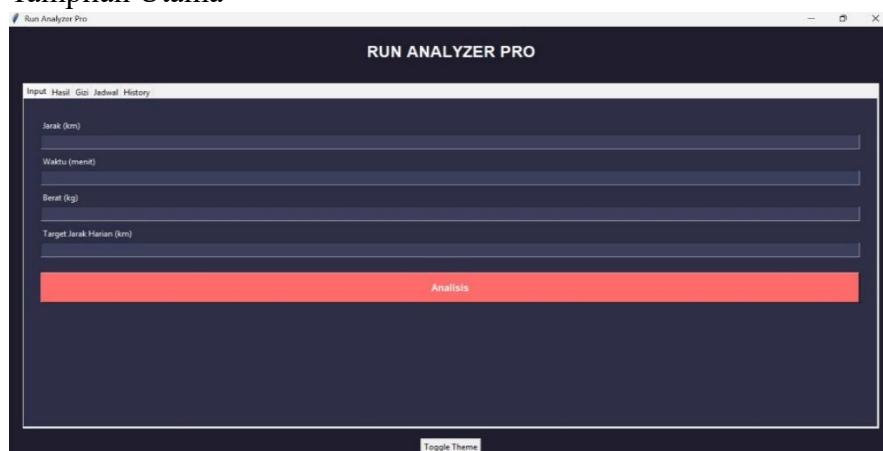
10. Menutup Aplikasi

Setelah selesai menggunakan aplikasi, pengguna dapat menutup aplikasi dengan menekan tombol tutup pada jendela aplikasi.

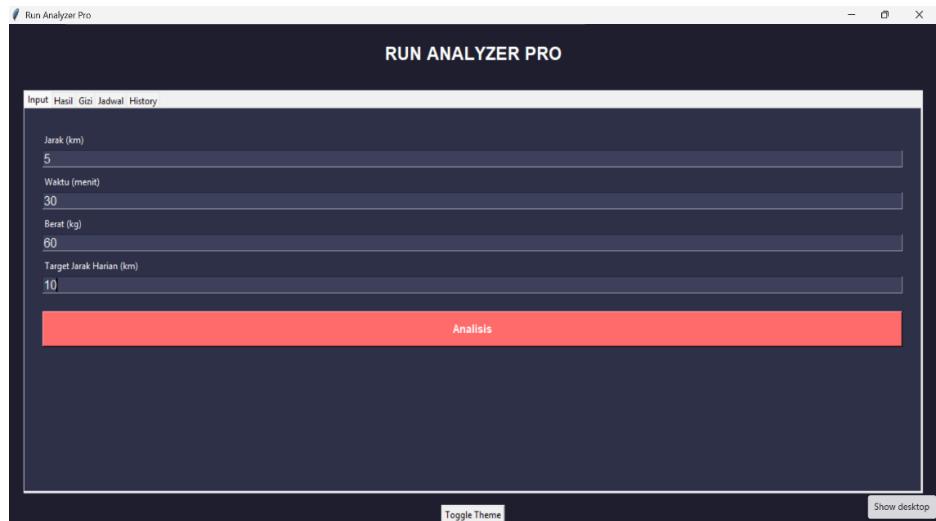
3.4 Screenshot Aplikasi

Berikut merupakan tampilan aplikasi RUN ANALYZER PRO:

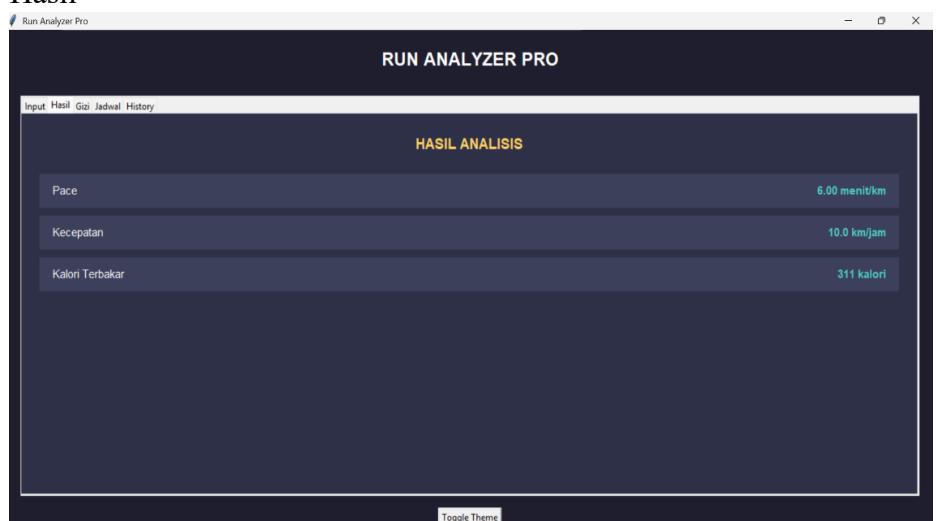
1. Tampilan Utama



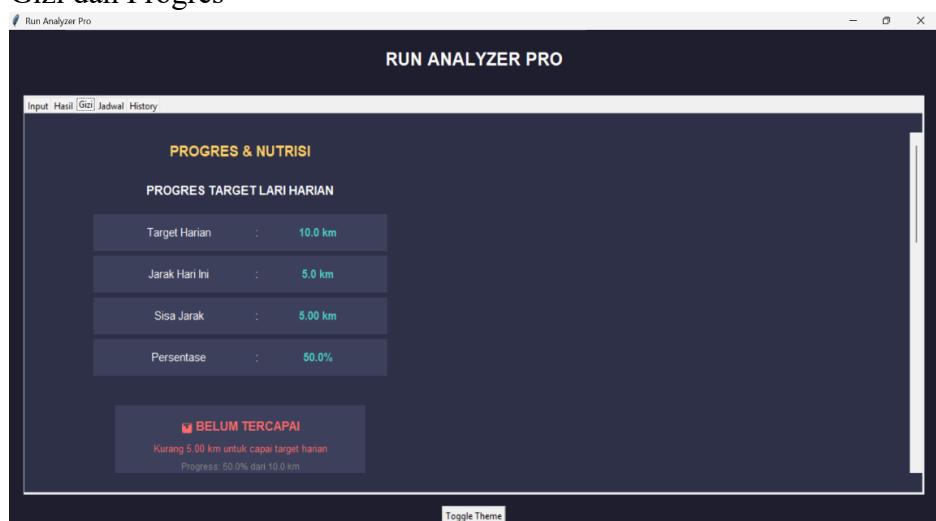
2. Input

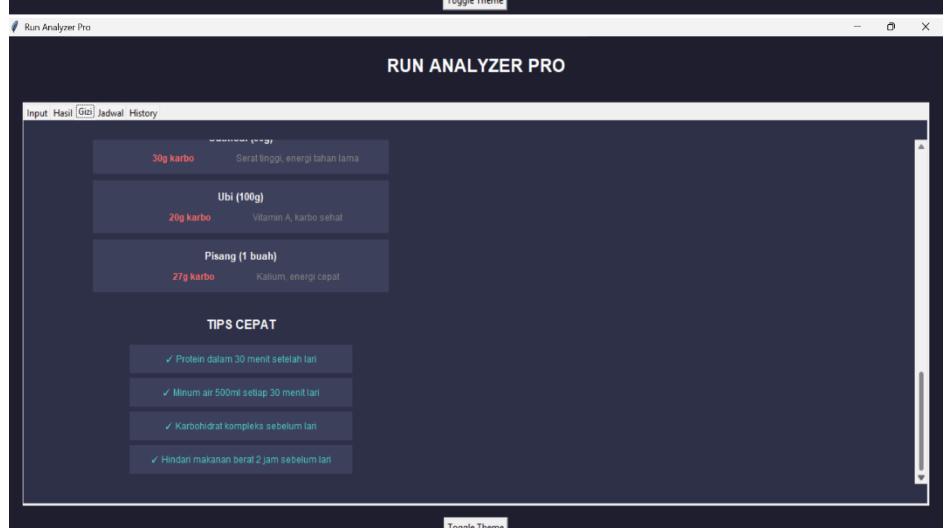
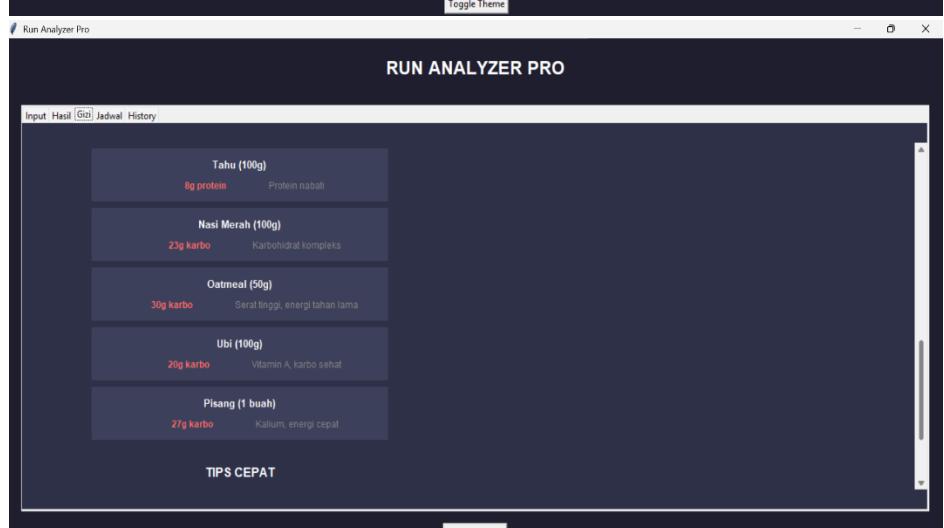
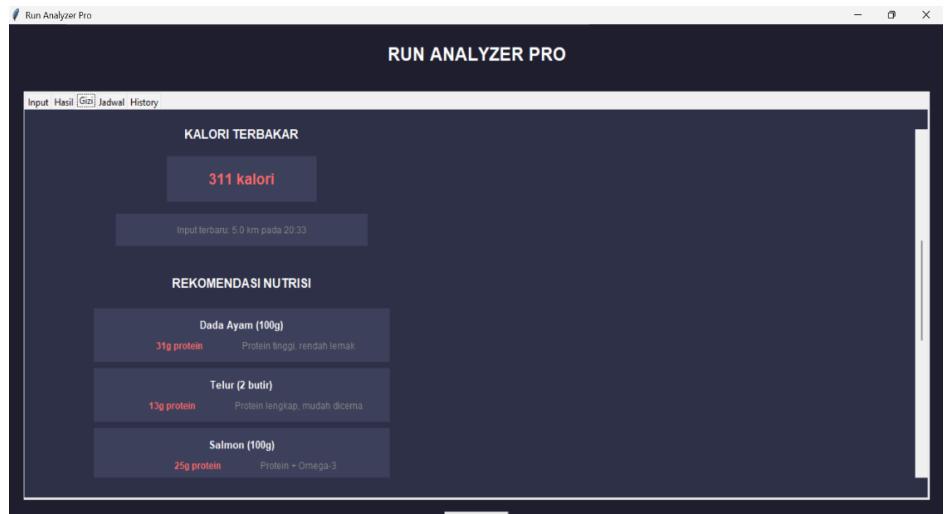


3. Hasil



4. Gizi dan Progres





5. Jadwal

The screenshot shows a dark-themed application window titled "RUN ANALYZER PRO". At the top, there is a navigation bar with tabs: "Input", "Hasil", "Gizi", "Jadwal" (which is highlighted in blue), and "History". Below the navigation bar, the title "JADWAL LATIHAN" is centered. A table lists the weekly training schedule:

Hari	Kegiatan	durasi / deskripsi
Senin	Lari Ringan	30 menit Pemanasan
Selasa	Interval Run	45 menit Kecepatan
Rabu	Recovery	20 menit Penulihan
Kamis	Long Run	60 menit Daya tahan
Jumat	Cross Training	40 menit Variasi
Sabtu	Tempo Run	50 menit Konsistensi
Minggu	Rest Day	- Penulihan total

A "Toggle Theme" button is located at the bottom right of the window.

6. History

The screenshot shows a dark-themed application window titled "RUN ANALYZER PRO". At the top, there is a navigation bar with tabs: "Input", "Hasil", "Gizi", "Jadwal" (which is highlighted in blue), and "History". Below the navigation bar, the title "Riwayat Analisis" is centered. A date box shows "2025-12-18".

Below the date box, the title "Detail Tanggal 2025-12-18" is centered. A summary box displays:

Target Harian: 10.0 km | Total Jarak: 5.0 km
20.33 | 5.0km | 30.0m | Pace 6.00 | 311 cal

LAMPIRAN

Berikut merupakan kode sumber lengkap aplikasi "Run Analyzer Pro" yang dibangun menggunakan Python dengan library Tkinter untuk antarmuka grafisnya. Aplikasi ini berfungsi sebagai alat analisis performa lari yang menghitung *pace*, kecepatan, kalori terbakar, serta menyediakan fitur pelacakan target harian, rekomendasi nutrisi, dan riwayat aktivitas.

Kode di bawah ini telah siap dijalankan dan mencakup semua fitur utama yang dibahas dalam dokumentasi sebelumnya:

```
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("700x750")

        self.mode = "dark"

        self.vars = {x: tk.StringVar() for x in
                    ["jarak","waktu","berat","target_jarak"]}

        self.history = {}

        self.daily_targets = {} # Menyimpan target harian per tanggal

        self.daily_distances = {} # Menyimpan total jarak harian per tanggal

        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 = {}

    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)"]

    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*1.036

Simpan target harian jika ada input target

if t is not None:

self.daily_targets[today] = t

self.target_jarak = t

else:

Gunakan target terakhir jika ada, atau gunakan target saat ini

```
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  
  
  
  
# Akumulasi jarak harian  
  
if today not in self.daily_distances:  
    self.daily_distances[today] = 0  
  
    self.daily_distances[today] += j  
  
  
  
# Hitung total jarak hari ini  
  
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.show_all()  
  
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()

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

    # Buat canvas untuk scroll dengan width yang cukup
    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)

# Pack canvas dan scrollbar

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

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

# Bind mouse wheel untuk scroll

def _on_mousewheel(event):

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

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

# Kontainer utama untuk konten di tengah

container = tk.Frame(scrollable_frame, bg=t["frame"])

container.pack(expand=True, fill="both", padx=20, pady=10)

# Judul di tengah

```

```

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")

    # Gunakan target terbaru untuk hari ini

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

    # Hitung total jarak hari ini

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

    # Ambil data dari input terbaru

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

    # Hitung progress berdasarkan akumulasi

    sisa_jarak = current_target - total_jarak_hari_ini

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

    # Container untuk progres target (di tengah)

```

```

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

progress_container.pack(fill="x", pady=(0,20))

# Judul progres di tengah

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 dengan layout terpusat

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"),
    ("Percentase", f"{percentase:.1f}%")
]

# Frame untuk grid data progres

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)

# Label

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

# Separator

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

# Value

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 target di tengah

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))

# Pesan khusus jika melebihi target

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))

# Kalori terbakar di tengah

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()

# Informasi input terbaru

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()

```

```
# Rekomendasi makanan fokus protein & karbohidrat

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")  
]
```

```
# Frame untuk makanan yang terpusat

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)

# Container untuk konten makanan

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

content_frame.pack(expand=True)

# Nama makanan (di tengah)

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

# Nutrisi dan deskripsi

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,"bold"), width=20).pack(side="right", padx=(10,0))

```

```
        font=("Arial",9), wraplength=200,  
justify="center").pack(side="left")  
  
# Tips nutrisi singkat  
  
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:

    # Pesan jika tidak ada target

    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)
```

```
# Hari
```

```
    tk.Label(frame, text=hari[i], bg=t["card"], fg=t["fg"],
```

```
font=("Arial",11,"bold"), width=8).pack(side="left", padx=(0,10))
```

```
# Latihan
```

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

```
# Durasi & Tipe
```

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

```
# Indikator
```

```
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"])
```

```
# Frame utama untuk center
```

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

```

# Title di tengah

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

# Frame untuk konten history yang di-center

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

center_frame.pack(expand=True)

# Container untuk data history

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

container.pack()

# Tampilkan target harian untuk tanggal tersebut

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()
```

DAFTAR PUSTAKA

- World Health Organization. (2020). *WHO guidelines on physical activity and sedentary behaviour*. World Health Organization.
- Wilmore, J. H., Costill, D. L., & Kenney, W. L. (2015). *Physiology of sport and exercise* (6th ed.). Human Kinetics.
- Python Software Foundation. (2024). *Python documentation*.
<https://docs.python.org/3/>
- Shipman, J. W. (2013). *Tkinter 8.5 reference: A GUI for Python*. New Mexico Tech Computer Center