BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Program kerja magang dilaksanakan di Universitas Multimedia Nusantara pada divisi program studi Sistem Informasi sebagai *business intelligence developer*. Mahasiswa diberikan tanggung jawab untuk membuat sebuah aplikasi visualisasi untuk staf *building management* milik Universitas Multimedia Nusantara.

Kegiatan yang dilakukan mahasiswa pada program magang ini adalah untuk membuat aplikasi visualisasi *offline* yang dapat digunakan oleh *building management* dalam membuat visualisasi untuk *sustainability dashboard* tahunan Universitas Multimedia Nusantara. Selama pembuatan aplikasi, *Visual Studio Code* digunakan sebagai aplikasi pemrograman yang digunakan karena aplikasi tersebut mudah untuk dimengerti dan dapat digunakan dengan berbagai bahasa pemrograman tanpa mengunduh aplikasi terpisah. Bahasa pemrograman yang digunakan untuk pembuatan aplikasi ini adalah *Python* karena bahasa tersebut mudah dimengerti dan dipelajari, serta banyaknya fitur yang terdapat dalam bahasa tersebut yang dapat mempercepat proses pembuatan aplikasi.

3.2 Tugas Kerja Magang

Tugas yang diberikan saat melaksanakan program kerja magang adalah pembuatan aplikasi visualisasi *offline* untuk *building management* secara mandiri selama 640 jam kerja. Aplikasi visualisasi ini merupakan sebuah program yang dirancang untuk pembuatan grafik visualisasi dengan hanya memasukkan data ke dalam kolom *input*.

3.3 Uraian Kerja Magang

Uraian kerja magang dilakukan untuk memberi penjelasan lebih detail pada kegiatan yang dilakukan selama program kerja magang. Tahap ini juga memberikan proses pembuatan aplikasi visualisasi dengan bentuk *interface* dan kodenya.

Tabel 3.1 Tugas Program Kerja Magang

No.	Tanggal	Kerja yang Dilakukan
1.	18-21 Maret 2025	Penjelasan Tugas (pembuatan aplikasi visualisasi) dan mempelajari ulang bahasa pemrograman *python*.
2.	24-28 Maret 2025	Perencanaan bentuk UI serta fungsi untuk ditambahkan pada aplikasi dan lanjut mempelajari ulang bahasa pemrograman *python*.
3.	7-30 April 2025	Pembuatan fungsi visualisasi aplikasi (grafik batang, grafik pie, dan grafik baris).
4.	2-7 Mei 2025	Menggabungkan ketiga fungsi visualisasi menjadi satu aplikasi dan membuat UI dasar.
5.	8-31 Mei 2025	Menambahkan fungsi *save, save as,* dan *load* serta tombolnya pada UI aplikasi.
6.	2-21 Juni 2025	Menambahkan fungsi penampilan data secara *real-time* pada UI aplikasi dan pembuatan aset gambar yang digunakan pada aplikasi.
7.	23-30 Juni 2025	Pembuatan fungsi penampilan tahun untuk visualisasi grafik aplikasi.
8.	1-9 Juli 2025	Pemeriksaan dan perbaikan terakhir pada aplikasi visualisasi.

3.3.1 Perencanaan Aplikasi Visualisasi

Sebelum memulai pembuatan aplikasi untuk *building management*, supervisor dan pihak dari *building management* memberikan penjelasan mengenai masalah yang sedang dialami oleh *building management* dan permintaan mereka untuk adanya aplikasi yang dapat menyelesaikan masalah tersebut. Kemudian tahap perencanaan aplikasi dimulai dengan melihat permintaan dari *building management* dan contoh hasil visualisasi yang diberikan oleh mereka.

3.3.2 Pembuatan Fungsi Visualisasi Grafik Baris

Pembuatan aplikasi dimulai dengan adanya fungsi visualisasi yang dimulai dengan *line graph* atau grafik baris. Gambar 3.1 menunjukkan kode milik grafik baris yang memiliki beberapa hal yang dibuat sesuai dengan contoh visualisasi yang diberikan oleh *building management* dalam bentuk garis merah (listrik yang digunakan) dan garis biru (listrik daur ulang).

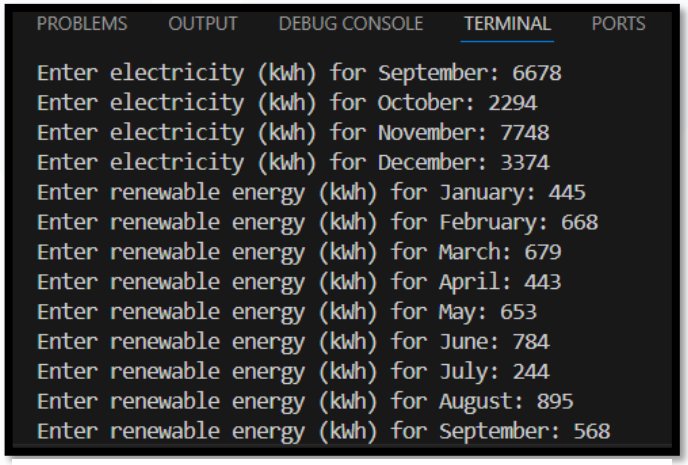
```python
import matplotlib.pyplot as plt

# Months of the year
months = ["January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"]

# Input energy values for electricity and renewable energy (in kWh)
electricity = [int(input(f"Enter electricity (kWh) for {month}: ")) for month in months]
renewable_energy = [int(input(f"Enter renewable energy (kWh) for {month}: ")) for month in months]

# Line graph
plt.figure(figsize=(10, 6))
plt.plot(months, electricity, label="Electricity", color="red", marker="o")
plt.plot(months, renewable_energy, label="Renewable Energy", color="lightblue", marker="o")

# Adding values on the nodes
for i, value in enumerate(electricity):
    plt.text(i, value, f"{value:,}", ha="center", va="bottom", fontsize=9, color="black")
for i, value in enumerate(renewable_energy):
    plt.text(i, value, f"{value:,}", ha="center", va="bottom", fontsize=9, color="black")

# Adding labels, title, and legend
plt.xlabel("Months")
plt.ylabel("Energy (kWh)")
plt.title("Energy Consumption by Month")
plt.legend()

# Rotate X-axis labels so its easier to read
plt.xticks(rotation=45)

# Graph output
plt.tight_layout()
plt.show()
```

Gambar 3.1 Kode Grafik Baris

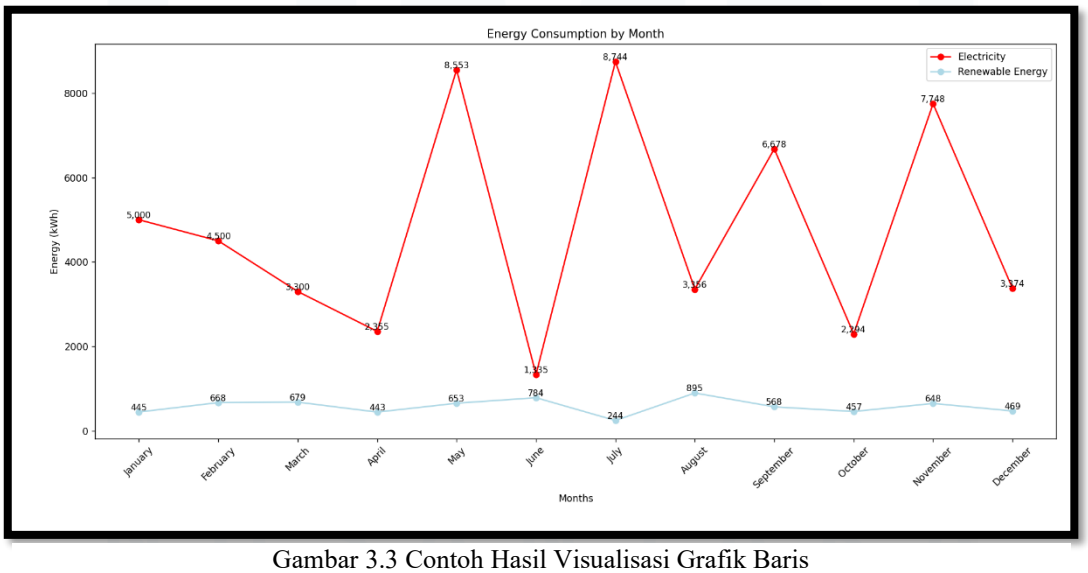
Kode yang dapat dilihat pada gambar 3.1 dimulai dengan mengimpor *matplotlib.pyplot* yang memberi akses ke fungsi pembuatan visualisasi pada bahasa pemrograman *Python.* Kemudian terdapat baris yang mengandung setiap bulan pada satu tahun yang menjadi sumbu horizontal (*X-axis*) pada grafik baris. Baris berikutnya menunjukkan kode yang menambahkan 2 kategori pemasukan dalam bentuk listrik yang digunakan dan listrik daur ulang dengan warna yang telah ditentukan. Setelah itu adalah baris yang mengandung variabel untuk membuat visualisasi grafik baris. Keempat baris terakhir adalah kode untuk membuat format visualisasi grafik lebih tertata yang dimulai dengan variabel untuk

menambahkan label teks di atas setiap titik bulan yang menampilkan nilai listrik. Kemudian adalah variabel untuk menambahkan keterangan pada sumbu horizontal dan vertikal visualisasi serta judul visualisasi. Berikutnya adalah variabel yang memutar label bulan pada sumbu horizontal agar lebih mudah untuk dibaca dan yang terakhir adalah variabel untuk menata tampilan hasil visualisasi.

Gambar 3.2 dan 3.3 menunjukkan UI pemasukan data yang masih menggunakan terminal pada *Visual Studio Code* dan contoh hasil visualisasi grafik baris menggunakan kode tersebut.



Gambar 3.2 Menu Input Grafik Baris



Gambar 3.3 Contoh Hasil Visualisasi Grafik Baris

3.3.3 Pembuatan Fungsi Visualisasi Grafik Batang

Gambar 3.4 dan 3.5 menunjukkan variabel visualisasi grafik batang yang dimulai dari mengimpor *matplotlib.pyplot* dan *math* (digunakan untuk pembulatan nilai ke atas). Kemudian adalah variabel untuk bulan yang menjadi sumbu horizontal visualisasi. Baris berikutnya menunjukkan variabel untuk pemasukan data yang menggunakan fungsi *float().* Fungsi tersebut memungkinkan pengguna untuk memasukkan angka desimal ke dalam kolom *input*. Fungsi tersebut diterapkan pada grafik baris dan grafik pie setelah semua fungsi digabungkan menjadi satu aplikasi dengan UI. Kedua baris variabel berikutnya menentukan skala sumbu vertikal secara dinamis yang membuat visualisasi grafik lebih tertata dan daftar angka pada sumbu vertikal ditampilkan dalam kelipatan 50 (0, 50, 100, 150). Fungsi ini dibenarkan agar dapat mengakomodasi nilai yang lebih besar dari 1000 pada finalisasi aplikasi. Keempat baris berikutnya adalah variabel untuk pembuatan visualisasi grafik batang dengan warna yang mengikuti contoh *building management*, penambahan judul dan label pada grafik bar, penyesuaian nilai pada sumbu vertikal agar jarak antara nilai konsisten, dan penyesuaian label bulan pada sumbu horizontal seperti yang dilakukan pada grafik baris agar terlihat rapi.

```python
import matplotlib.pyplot as plt
import math

# Months of the year
months = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
]

# Input carbon footprint values for each month
carbon_footprint = []
print("Please enter the carbon footprint values for each month (decimals allowed):")
for month in months:
    value = float(input(f"{month}: "))
    carbon_footprint.append(value)

# Calculate the maximum value and round up to the nearest 50 for the y-axis limit
max_value = max(carbon_footprint)
y_axis_limit = math.ceil(max_value / 50) * 50

# Customize y-axis ticks to be multiples of 50 up to the y-axis limit
y_ticks = list(range(0, y_axis_limit + 1, 50))

# Create the bar chart
bars = plt.bar(months, carbon_footprint, color='#32CD32')

# Set chart title and labels
plt.title('Monthly Carbon Footprint')
plt.xlabel('Months')
plt.ylabel('Carbon Footprint')

# Customize y-axis ticks
plt.yticks(y_ticks)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)
```

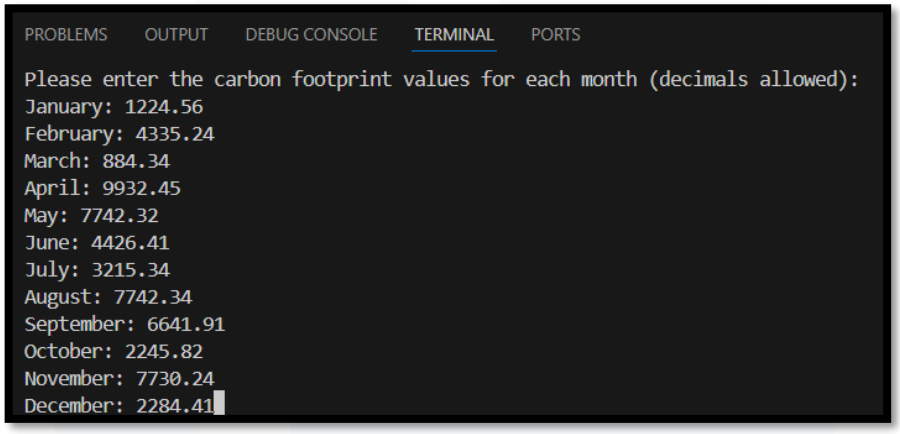
Gambar 3.4 Bagian Pertama Kode Grafik Bar

Dua baris variabel terakhir untuk grafik bar dimulai dengan kode untuk menambahkan nilai *input* di atas setiap batang yang menentukan besar serta posisi teks dan kemudian variabel untuk menampilkan grafik bar setelah semua data telah dimasukkan ke kolom *input*.

```python
# Add values on top of the bars and adjust font size dynamically
for bar, value in zip(bars, carbon_footprint):
    bar_width = bar.get_width()
    font_size = max(10, bar_width * 8)
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height() + 1,
        f"{value:.2f}",
        ha='center', va='bottom', fontsize=int(font_size)
    )

# Display the chart
plt.tight_layout()
plt.show()
```

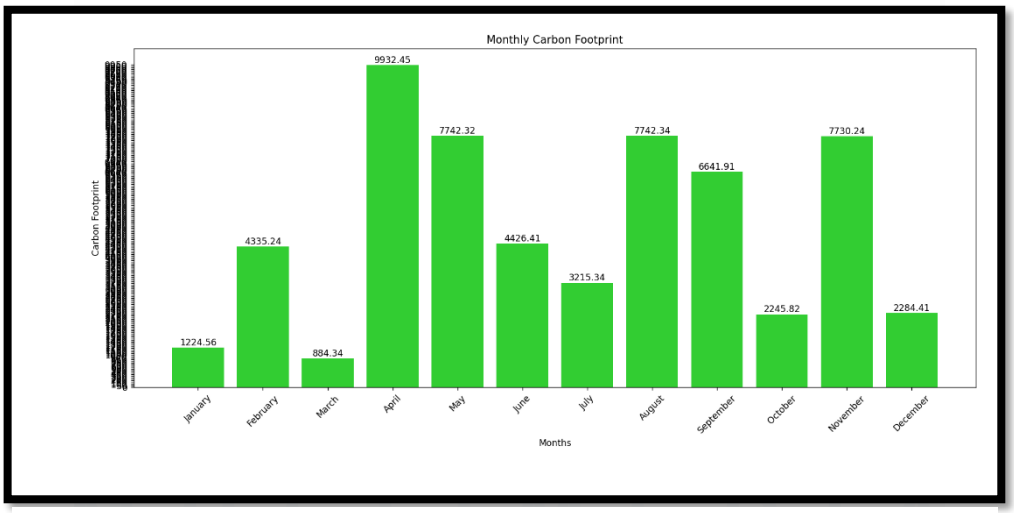
Gambar 3.5 Bagian Kedua Kode Grafik Bar

Gambar 3.6 menunjukkan menu pemasukan data sebelum pembuatan UI aplikasi dengan menggunakan terminal pada *Visual Studio Code.*



Gambar 3.6 Menu Input Grafik Batang

Gambar 3.7 menunjukkan contoh hasil visualisasi grafik batang pada tahap awal pembuatan aplikasi yang masih mengandung masalah perhitungan nilai pada sumbu vertikal yang dibenarkan pada pemeriksaan terakhir aplikasi.



Gambar 3.7 Contoh Hasil Visualisasi Grafik Batang (Sebelum Diperbaiki)

3.3.4 Pembuatan Fungsi Visualisasi Grafik Pie

Gambar 3.8 dan 3.9 menunjukkan seluruh kode untuk fungsi visualisasi grafik pie. Mengikuti grafik baris dan batang, grafik pie juga menggunakan *matplotlib.pyplot* untuk mengakses fungsi pembuatan grafik. Barisan kode berikutnya menunjukkan daftar bulan dengan kamus bernama data yang digunakan untuk menyimpan masukan data air setiap bulan termasuk volume air dan persentase distribusinya. Setelah itu adalah fungsi pemasukan data dalam bentuk air bersih dan air daur ulang di mana nilai kedua kategori tersebut akan dihitung sebagai persentase yang kemudian dimasukkan ke dalam kamus data.

```
import matplotlib.pyplot as plt

# Data input for each month
months = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
]

data = {}
for month in months:
    print(f"Enter water data for {month} (in liters):")
    clean_water = float(input("  Clean water (liters): "))
    recycled_water = float(input("  Recycled water (liters): "))
    total_water = clean_water + recycled_water
    data[month] = {
        "clean_water": clean_water,
        "recycled_water": recycled_water,
        "percentages": [recycled_water / total_water * 100, clean_water / total_water * 100]
    }
```

Gambar 3.8 Bagian Pertama Kode Grafik Pie

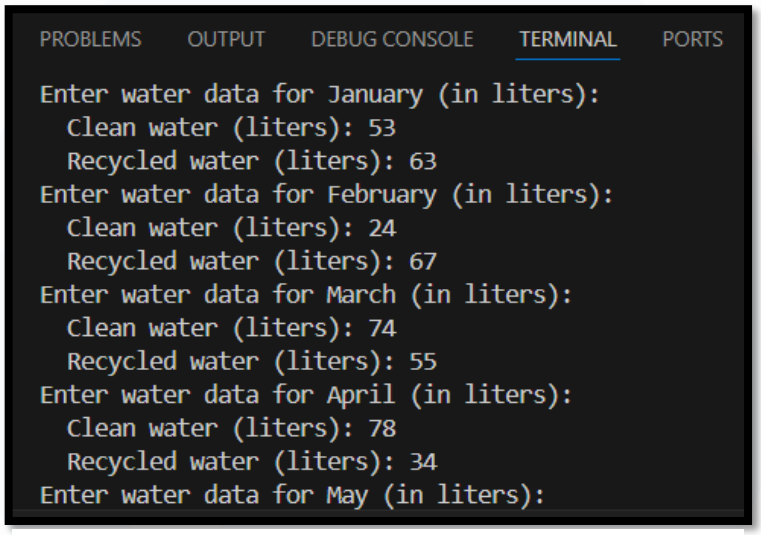
Bagian berikut adalah variabel untuk pemetaan grafik pie yang ditampilkan untuk setiap bulan. Dimulai dengan *plt.subplots* yang digunakan untuk menyusun grafik pie dalam format 3 baris dan 4 kolom. Kemudian adalah *figsize* dan *tight_layout* yang digunakan untuk menyesuaikan ukuran grafik pie agar hasil visualisasinya terlihat rapi. Setelah itu adalah fungsi yang membuat grafik pie untuk setiap bulan dengan format dari contoh *building management* dalam bentuk warna biru muda untuk air bersih dan abu-abu untuk air daur ulang serta grafik pie yang

memiliki bagian tengah yang kosong. Setelah itu adalah kode untuk menambahkan judul dan legenda pada hasil visualisasi. Variabel *plt.show* digunakan untuk mencetak grafik yang telah diberikan *input* oleh pengguna aplikasi.

```
# Plotting all months in a grid
fig, axes = plt.subplots(3, 4, figsize=(15, 10))
fig.tight_layout(pad=5.0)

for i, (month, values) in enumerate(data.items()):
    row, col = divmod(i, 4)  #
    ax = axes[row, col]
    ax.pie(
        values["percentages"],
        labels=["", ""],
        colors=["#0077be", "#d3d3d3"],
        startangle=90,
        radius=1,
        wedgeprops=dict(width=0.5)
    )
    ax.set_title(month, fontsize=14, weight="bold")
    legend_labels = [
        f"Clean Water: {values['clean_water']} L ({values['percentages'][1]:.1f}%)",
        f"Recycled Water: {values['recycled_water']} L ({values['percentages'][0]:.1f}%)"
    ]
    ax.legend(
        legend_labels,
        loc="lower center",
        bbox_to_anchor=(0.5, -0.2),
        fontsize=10
    )

plt.show()
```

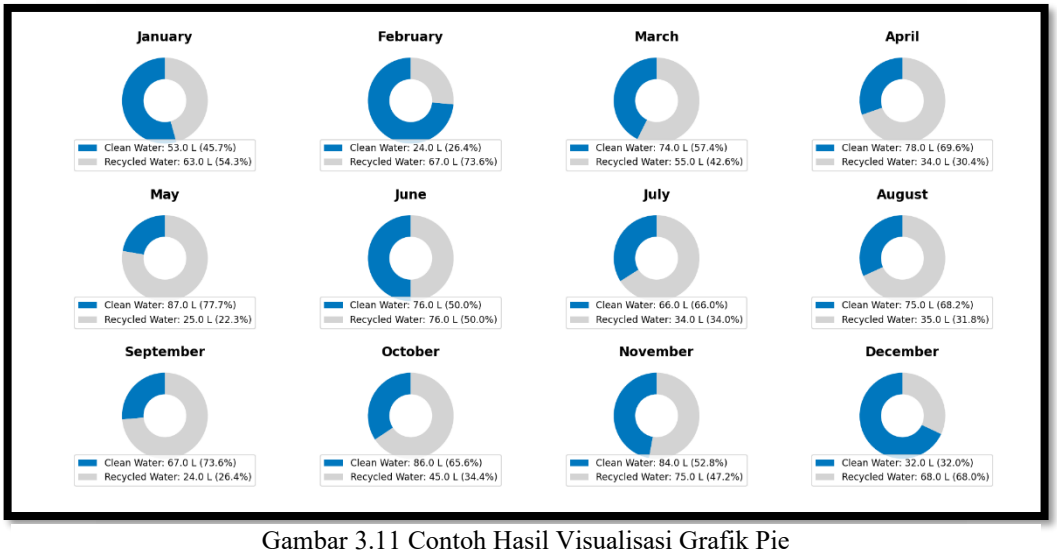
Gambar 3.9 Bagian Kedua Kode Grafik Pie

Gambar 3.10 menunjukkan menu pemasukan data menggunakan terminal pada *Visual Studio Code*. Menu pemasukan data memiliki 2 kategori *input* pada setiap bulan.



Gambar 3.10 Menu Input Grafik Pie

Gambar 3.11 menunjukkan contoh hasil visualisasi grafik pie yang menampilkan data air setiap bulan dengan adanya label kategori air serta persentase kategori pada setiap bulan.



Gambar 3.11 Contoh Hasil Visualisasi Grafik Pie

3.3.5 Pembuatan Aplikasi Visualisasi Final

Tahap ini mengandung seluruh kode dan bentuk UI aplikasi visualisasi setelah ketiga fungsi visualisasi telah digabungkan serta mengandung perbaikan berdasarkan *feedback* yang diberikan oleh *supervisor* dan pihak *building management*. Gambar 3.12 adalah bagian pertama dari kode aplikasi visualisasi. Variabel ini dimulai dengan mengimpor *os, sys* agar dapat menggunakan fungsi *sys._MEIPASS.* Fungsi tersebut digunakan untuk memastikan aplikasi dapat mencari aset gambar ketika diekspor dan digunakan pada sistem berbeda.

```
import os, sys

if hasattr(sys, '_MEIPASS'):
    base_path = sys._MEIPASS
else:
    base_path = os.path.abspath(".")
imageassets_path = os.path.join(base_path, "imageassets")
os.chdir(imageassets_path)
print("Current working directory:", os.getcwd())
```

Gambar 3.12 Variabel Yang Berfungsi Untuk Membuat Aplikasi Menjadi Portable

Gambar 3.13 adalah variabel yang berfungsi untuk memuat GUI aplikasi yang dimulai dengan mengimpor *tkinter*. Fungsi *tk.Tk()* adalah untuk menolong proses pemuatan aset gambar pada tahap awal. Kemudian *withdraw()* digunakan untuk memastikan tidak ada kendala visual ketika aplikasi sedang menyiapkan diri untuk membuka. Jika GUI aplikasi dimuat lebih dahulu, maka aplikasi akan mengalami kendala pada GUI.

```
import tkinter as tk

root = tk.Tk()
root.withdraw()
```

Gambar 3.13 Variabel Yang Berfungsi Untuk Memastikan Aplikasi Dapat Terbuka

Gambar 3.14 adalah variabel untuk mengimpor seluruh fungsi yang digunakan pada aplikasi yang dimulai dengan *matplotlib.pyplot* untuk semua grafik yang dibuat untuk aplikasi, *math* untuk pembulatan dan menyesuaikan skala visualisasi, *JSON* untuk menyimpan dan memuat data pengguna, *pillow* untuk memuat serta menyesuaikan besar aset gambar, dan *NumPy* yang digunakan untuk pemuatan gambar dan memastikan ada kompatibilitas dengan *matplotlib.pyplot*.

```
import matplotlib.pyplot as plt
import math
from tkinter import messagebox, filedialog
import json
from PIL import Image, ImageTk
import numpy as np
```

Gambar 3.14 Variabel Yang Mengimpor Fungsi Yang Digunakan Pada Aplikasi

Gambar 3.15 adalah variabel untuk menyesuaikan besar aset gambar menggunakan fungsi *pillow*. Fungsi *Image.open()* digunakan untuk memuat aset gambar dari lokasi asal aset tersebut. *Image* membuat sebuah objek pada aplikasi yang mengandung data aset gambar. *Ratio* digunakan untuk menghitung skala yang dibutuhkan untuk mengganti besarnya aset gambar dan fungsi *float* adalah untuk konversi nilai besarnya agar dapat dibagi dengan lebih akurat. *Image.size[1]* mengambil tinggi asli aset gambar. Kemudian *new_width* digunakan untuk menghitung lebar barunya aset gambar. *Image.size[0]* mengambil lebar asli aset gambar dan kemudian dibulatkan karena tipe ukuran gambar *pixels* (px) harus dalam bentuk angka bulat. Selanjutnya *image.resize* berfungsi untuk menggantikan besar aset gambar serta penggunaan *Image.LANCZOS* untuk memastikan kualitas aset gambar tetap tinggi walaupun besarnya diganti menjadi lebih kecil. Setelah

itu *ImageTk.Photoimage* melakukan konversi pada format aset gambar yang menggunakan *pillow* menjadi gambar yang kompatibel dengan *Tkinter*.

```
# Function to load and resize an image from file (for icons in the live totals display)
def load_and_resize_image(path, new_height):
    image = Image.open(path)
    ratio = new_height / float(image.size[1])
    new_width = int(image.size[0] * ratio)
    return ImageTk.PhotoImage(image.resize((new_width, new_height), Image.LANCZOS))
```

Gambar 3.15 Variabel Untuk Menyesuaikan Besar Aset Gambar

Gambar 3.16 adalah lanjutan dari 3.15 dan merupakan format penempatan aset gambar agar lebih rapi. Baris variabel dimulai dengan *ICON_HEIGHT* digunakan untuk mendefinisikan besar aset gambar pada fungsi penampilan data total. Berikutnya, fungsi *icon_name* digunakan untuk memastikan aset gambar dimuat terlebih dahulu sebelum seluruh aplikasi terbuka. Ini memastikan aset gambar dapat dimuat tanpa mengalami kendala ketika menu utama sedang dimuat.

```
# Set a target height for the live totals icons.
ICON_HEIGHT = 20

# Load icons for each total
icon_carbon = load_and_resize_image("carbon.png", ICON_HEIGHT)
icon_electricity = load_and_resize_image("electricity.png", ICON_HEIGHT)
icon_renewable = load_and_resize_image("leaf.png", ICON_HEIGHT)
icon_recwater = load_and_resize_image("recwater.png", ICON_HEIGHT)
icon_water = load_and_resize_image("water.png", ICON_HEIGHT)
```

Gambar 3.16 Lanjutan Variabel Pada Gambar 3.15 Untuk Format Penempatan Dan Memuat Aset Gambar

Gambar 3.17 menunjukkan variabel untuk penempatan logo Universitas Multimedia Nusantara pada UI aplikasi pada atas kiri layar dan selalu dapat dilihat jika pengguna navigasi ke menu berbeda. Baris *Image.open()* memuat gambar logo Universitas Multimedia Nusantara ke

dalam aplikasi. Fungsi *.size* pada baris kedua merekam besar asli gambar logo dalam ukuran *pixels*. Rumus perhitungan besar logo dikali dengan **0.2* yang menjadikan besar logo menjadi 20% dari besar aslinya dan *int()* digunakan untuk memastikan besar dimensi logo tetap akurat. Seperti aset gambar kategori data, *ImageTk.PhotoImage()* digunakan untuk konversi format logo menjadi yang kompatibel dengan *Tkinter*. Kemudian *top_logo_frame* digunakan untuk memastikan posisi logo Universitas Multimedia Nusantara dalam layar aplikasi. *Side=”top”* memaksa logo ke bagian atas layar aplikasi dan *anchor”w”* memastikan aplikasi tetap pada sisi kiri layar aplikasi.

```
# Uni logo formatting
logo_image = Image.open("uni_logo.png")
orig_width, orig_height = logo_image.size
new_size = (int(orig_width * 0.2), int(orig_height * 0.2))
resized_logo = logo_image.resize(new_size, Image.LANCZOS)
uni_logo = ImageTk.PhotoImage(resized_logo)
top_logo_frame = tk.Frame(root)
top_logo_frame.pack(side="top", anchor="w")
tk.Label(top_logo_frame, image=uni_logo).pack(side="left", anchor="w")
```

Gambar 3.17 Variabel Untuk Format Logo UMN

Gambar 3.18 dan 3.19 menunjukkan lokasi logo Universitas Multimedia Nusantara pada menu utama dan menu visualisasi. Logo akan selalu ditampilkan di layar aplikasi.

Gambar 3.18 Tampilan UI Utama Dengan Logo UMN Pada Atas Kiri Layar

Gambar 3.19 Tampilan Menu Visualisasi Grafik Batang Dengan Logo UMN Pada Atas Kiri Layar

Gambar 3.20 menampilkan barisan variabel untuk konfigurasi terakhir layar aplikasi sebelum pengguna dapat melihat menu utama. Variabel ini dimulai dengan *root.deiconify()* yang berfungsi untuk menampilkan layar yang disembunyikan pada awal. Kemudian *root.title* berfungsi untuk menampilkan nama aplikasi. Terakhir, *root.geometry()* menyetel besarnya layar aplikasi setiap kali dibuka.

```
# Configure the already-created root window
root.deiconify()
root.title("Visualization App")
root.geometry("1000x500")
```

Gambar 3.20 Variabel Konfigurasi Terakhir Sebelum Menu Utama Ditampilkan

Gambar 3.21 adalah kode untuk daftar bulan yang digunakan oleh grafik baris, grafik batang, grafik pie pada hasil visualisasi dan kolom pemasukan data pada menu ketiga visualisasi.

```
# Months of the year
months = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
]
```

Gambar 3.21 Variabel Dengan Daftar Bulan

Gambar 3.22 adalah variabel yang mengandung fungsi untuk menyimpan lokasi *file* terakhir yang dimuat pada aplikasi. Jika pengguna belum menyimpan *file*, maka nilai pada variabel ini akan tetap sebagai *none*.

```
# Store the currently loaded file path
current_file = None
```

Gambar 3.22 Variabel Untuk Menyimpan Lokasi File Yang Digunakan Pada Aplikasi

Gambar 3.23 adalah variabel untuk fungsi *Save Data*. Variabel dimulai dengan pemeriksaan *file* untuk mengetahui jika pengguna telah memuat *file json* pada aplikasi. Fungsi utama variabel ini adalah untuk menyimpan progres pemasukan data oleh pengguna agar mereka dapat melanjutkan progres tersebut pada bulan berikutnya. Jika tidak ada *file* yang dimuat ketika pengguna menekan tombol *save*, aplikasi akan menggunakan fungsi *save as* agar pengguna dapat menyimpan *file* mereka. Baris *data* = adalah barisan data yang disimpan oleh fungsi penyimpanan aplikasi dalam bentuk *file json*. Bagian bawah variabel menunjukkan pesan untuk memberi tahun pengguna jika *file* mereka telah disimpan.

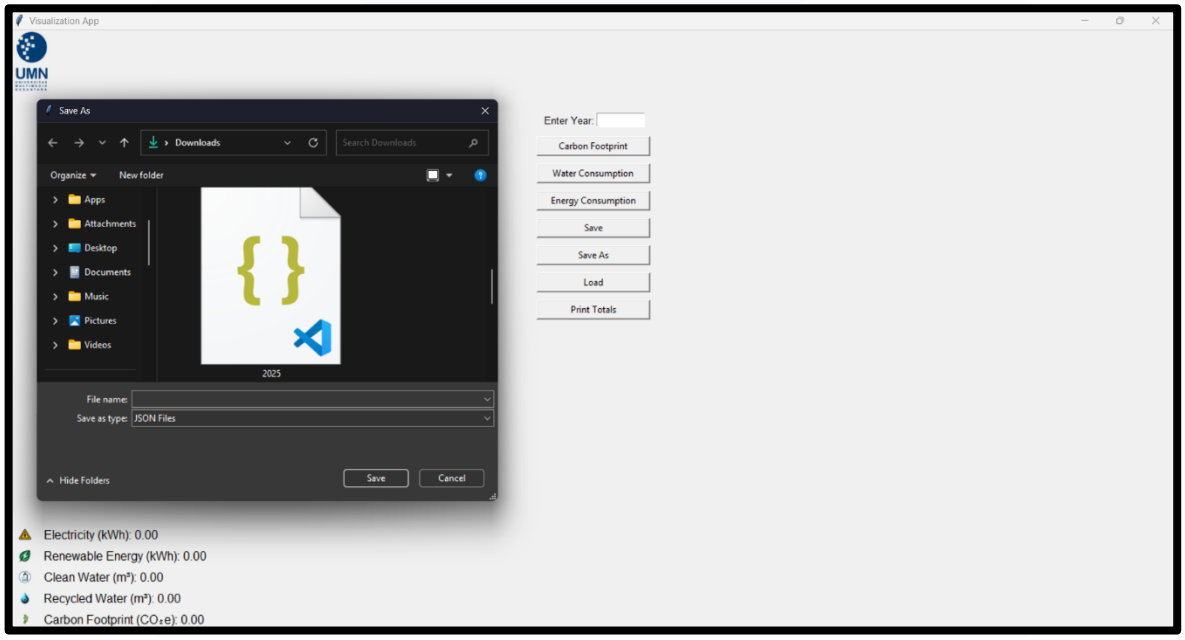
```
# --------------------- Data Save/Load Functions ---------------------
def save_data():
    global current_file
    if not current_file:
        save_data_as()
        return
    data = {
        "bar": [entry.get() for entry in entry_bar],
        "pie_clean": [entry.get() for entry in entry_pie_clean],
        "pie_recycled": [entry.get() for entry in entry_pie_recycled],
        "line_electric": [entry.get() for entry in entry_line_electric],
        "line_renewable": [entry.get() for entry in entry_line_renewable]
    }
    with open(current_file, "w") as file:
        json.dump(data, file)
    messagebox.showinfo("Save Data", f"Data saved successfully to {current_file}")
```

Gambar 3.23 Variabel Fungsi Save

Gambar 3.24 adalah variabel untuk fungsi *save* as yang berhubungan dengan gambar 3.25. Fungsi *save* as membuka menu penyimpanan *file* untuk pengguna di mana mereka dapat memilih lokasi penyimpanan pada sistem mereka yang ditampilkan pada gambar 3.26. Variabel ini melakukan pemeriksaan pada aplikasi sama seperti variabel *save* untuk mengetahui jika ada *file* yang telah dimuat pada aplikasi. Jika pengguna menekan tombol *save* as ketika ada *file* yang dimuat, maka aplikasi akan menggunakan fungsi *save* dan tidak akan membuka menu pemilihan lokasi penyimpanan.

```
def save_data_as():
    global current_file
    filename = filedialog.asksaveasfilename(defaultextension=".json",
                                            filetypes=[("JSON Files", "*.json")])
    if filename:
        current_file = filename
        save_data()
```

Gambar 3.24 Variabel Fungsi Save As



Gambar 3.25 Tampilan Menu Save As Ketika Menekan Tombol Save atau Save As

Gambar 3.26 adalah variabel dengan fungsi untuk memuat *file* yang pengguna telah simpan. Fungsi dimulai dengan membuka menu sistem untuk memilih *file* untuk dimuat ke aplikasi. Ketika pengguna memuat *file* dari sistem mereka, data yang dimasukkan pada kolom *input* aplikasi akan dihapus menggunakan fungsi *entry.delete* dan diganti oleh data pada *file* yang dimuat menggunakan *entry.insert*. Jika *file* sukses untuk dimuat, maka pengguna akan diberikan sebuah pesan untuk memberi tahu mereka dan ada pesan eror jika *file* yang pengguna memuat bukan dalam bentuk *json*.

```
def load_data():
    global current_file
    filename = filedialog.askopenfilename(filetypes=[("JSON Files", "*.json")])
    if filename:
        try:
            with open(filename, "r") as file:
                data = json.load(file)
                for i, entry in enumerate(entry_bar):
                    entry.delete(0, tk.END)
                    entry.insert(0, data["bar"][i])
                for i, entry in enumerate(entry_pie_clean):
                    entry.delete(0, tk.END)
                    entry.insert(0, data["pie_clean"][i])
                for i, entry in enumerate(entry_pie_recycled):
                    entry.delete(0, tk.END)
                    entry.insert(0, data["pie_recycled"][i])
                for i, entry in enumerate(entry_line_electric):
                    entry.delete(0, tk.END)
                    entry.insert(0, data["line_electric"][i])
                for i, entry in enumerate(entry_line_renewable):
                    entry.delete(0, tk.END)
                    entry.insert(0, data["line_renewable"][i])
            current_file = filename
            messagebox.showinfo("Load Data", f"Data loaded successfully from {filename}")
        except (FileNotFoundError, KeyError, json.JSONDecodeError):
            messagebox.showerror("Error", "Failed to load data. Please select a valid JSON file.")
```

Gambar 3.26 Variabel Fungsi Load Data

Gambar 3.27 adalah isi *file json* ketika ada pemasukan data pada aplikasi yang telah disimpan menggunakan fungsi *save*.


```
{"bar": ["319.72", "343.20", "348.94", "331.56", "384.07", "308.27", "343.33", "389.40", "427.37", "419.06", "405.68", "325.98"], "pie_clean": ["100", "200", "300", "400", "250", "530",
"400", "300", "340", "120", "200", "210"], "pie_recycled": ["45", "75", "45", "50", "65", "45", "67", "", "", "35", "68", "26"], "line_electric": ["378120", "402000", "409860", "388500",
"449952", "364692", "406764", "459104", "499104", "491304", "473856", "379710"], "line_renewable": ["0", "0", "0", "0", "11010", "15780", "17270", "19180", "18850", "20640", "17600",
"16940"]}
```


Gambar 3.27 Isi File JSON Ketika Terdapat Input Yang Disimpan Dari Aplikasi

Gambar 3.28 adalah variabel untuk visualisasi grafik batang dengan perbaikan yang didapatkan dari pihak *building management*. Perbaikan pertama adalah pada sistem *input* di mana kolom yang tidak ada isi atau diberikan *input* 0 tidak akan ditampilkan pada hasil visualisasi grafik. Visualisasi grafik batang tidak dapat dibuat jika kolom *input* tidak ada isi dan pengguna akan diberikan pesan yang memberitahukan mereka. Perbaikan kedua adalah pada kesalahan perhitungan pada sumbu vertikal yang ada pada gambar 3.7. Sekarang kalkulasi nilai yang ditampilkan pada sumbu vertikal dilakukan dengan menghitung besar *input* yang dimulai dari per 50 untuk *input* yang sama dengan atau kurang dari 500, per 500 untuk *input* yang kurang dari atau sama dengan 5.000, per 1.000 untuk *input* yang kurang dari atau sama dengan 10.000, per 10.000 untuk *input* yang kurang dari atau sama dengan 100.000, dan per 50.000 untuk *input* yang lebih dari 100.000.

```
def generate_bar_chart():
    values = [float(entry.get() or 0) for entry in entry_bar]
    filtered_months = []
    filtered_values = []
    for m, v in zip(months, values):
        if v != 0:
            filtered_months.append(m)
            filtered_values.append(v)
    if len(filtered_values) == 0:
        messagebox.showinfo("No Data", "No valid input for Bar Chart found.")
        return

    max_value = max(filtered_values)

    if max_value <= 500:
        step = 50
    elif max_value <= 5000:
        step = 500
    elif max_value <= 10000:
        step = 1000
    elif max_value <= 100000:
        step = 10000
    else:
        step = 50000
    y_axis_limit = math.ceil(max_value / step) * step
    y_ticks = list(range(0, y_axis_limit + 1, step))
    bars = plt.bar(filtered_months, filtered_values, color='#32CD32')
```

Gambar 3.28 Bagian Pertama Variabel Grafik Batang Dengan Perbaikan

Gambar 3.29 adalah variabel tahun yang ditampilkan pada judul visualisasi grafik batang jika pengguna mengisi tahun pada menu utama sebelum membuat visualisasi grafik dan format visualisasi grafik batang dari. Gambar 3.30 menunjukkan judul grafik batang jika pengguna mengisi kotak *input* tahun pada menu utama.

```
year_text = entry_year.get().strip()
if year_text:
    plt.title("Monthly Carbon Footprint (" + year_text + ")")
else:
    plt.title("Monthly Carbon Footprint")

plt.xlabel("Months")
plt.ylabel("Carbon Footprint")
plt.yticks(y_ticks)
plt.xticks(rotation=45)
for bar, value in zip(bars, filtered_values):
    font_size = max(10, bar.get_width() * 8)
    plt.text(bar.get_x() + bar.get_width() / 2,
             bar.get_height() + 1,
             f"{value:,.2f}",
             ha='center', va='bottom', fontsize=int(font_size))
plt.tight_layout()
plt.show()
```

Gambar 3.29 Bagian Kedua Variabel Grafik Batang Dengan Penampilan Tahun Pada Judul Visualisasi

Gambar 3.30 Penampilan Tahun Pada Judul Visualisasi Grafik Baris

Gambar 3.31 adalah bagian pertama variabel untuk grafik pie dengan fungsi untuk membuat grafik pie dengan kedua kategori yang ditampilkan pada setiap grafik pie. Bagian pertama variabel juga mengandung format lokasi dan besar setiap grafik pie.

```
def generate_pie_chart():
    clean_values = [float(entry.get() or 0) for entry in entry_pie_clean]
    recycled_values = [float(entry.get() or 0) for entry in entry_pie_recycled]
    fig, axes = plt.subplots(3, 4, figsize=(15, 10))
```

Gambar 3.31 Bagian Pertama Variabel Grafik Pie

Gambar 3.32 adalah bagian kedua variabel grafik pie dengan fungsi penampilan tahun pada judul hasil visualisasi dan format spasi antara setiap grafik pie yang ditampilkan agar label dapat dibaca.

```
year_text = entry_year.get().strip()
if year_text:
    fig.suptitle("Monthly Water Consumption (" + year_text + ")", fontsize=16)
else:
    fig.suptitle("Monthly Water Consumption", fontsize=16)

# Adjust tight layout to make room for subtitle
fig.tight_layout(rect=[0, 0, 1, 0.95])
# Adjust the vertical spacing
fig.subplots_adjust(hspace=0.317)
```

Gambar 3.32 Bagian Kedua Variabel Grafik Pie

Gambar 3.33 adalah bagian ketika variabel grafik pie yang mengandung fungsi untuk mengumpat satu bulan jika bulan tersebut tidak memiliki *input* pada kedua kategori air, fungsi perhitungan persentase air bersih dan air daur ulang pada setiap grafik pie, warna untuk kedua kategori air, pengecilan besar setiap grafik pie sebesar 10%, label pada visualisasi, dan fungsi mencetak hasil visualisasi.

```
for i, month in enumerate(months):
    row, col = divmod(i, 4)
    ax = axes[row, col]
    total_water = clean_values[i] + recycled_values[i]
    if total_water == 0:
        ax.axis('off')
        continue
    percentages = [recycled_values[i] / total_water * 100, clean_values[i] / total_water * 100]

    # Reduce the radius by 10%
    ax.pie(percentages,
           labels=["", ""],
           colors=["#0077be", "#d3d3d3"],
           startangle=90,
           radius=0.9,
           wedgeprops=dict(width=0.5))
    ax.set_title(month, fontsize=14, weight="bold")
    legend_labels = [
        f"Clean Water: {clean_values[i]:,.2f} L ({percentages[1]:.1f}%)",
        f"Recycled Water: {recycled_values[i]:,.2f} L ({percentages[0]:.1f}%)"
    ]
    ax.legend(legend_labels,
              loc="lower center",
              bbox_to_anchor=(0.5, -0.2),
              fontsize=10)
plt.show()
```

Gambar 3.33 Bagian Ketiga Variabel Grafik Pie

Gambar 3.34 adalah bagian pertama variabel grafik baris dengan 2 kategori listrik yaitu listrik digunakan dan listrik didaur ulang. Grafik baris telah diperbaiki dengan *input* dari supervisor mengenai tipe baris yang digunakan. Baris telah diganti menjadi baris terpotong dari garis utuh yang digunakan dalam pembuatan variabel grafik baris pertama kali. Perbaikan kedua pada grafik baris adalah penampilan pesan eror jika pengguna mencoba untuk mencetak grafik tanpa melakukan *input* pada kolom pemasukan data.

```
def generate_line_graph():
    electricity = [float(entry.get() or 0) for entry in entry_line_electric]
    renewable_energy = [float(entry.get() or 0) for entry in entry_line_renewable]
    filtered_months = []
    filtered_electricity = []
    filtered_renewable = []
    for m, e, r in zip(months, electricity, renewable_energy):
        if e == 0 and r == 0:
            continue
        filtered_months.append(m)
        filtered_electricity.append(e)
        filtered_renewable.append(r)
    if len(filtered_months) == 0:
        messagebox.showinfo("No Data", "No valid input for Line Graph found.")
        return
    plt.figure(figsize=(10, 6))
    plt.plot(filtered_months, filtered_electricity, label="Electricity", color="red", marker="o", linestyle="--")
    plt.plot(filtered_months, filtered_renewable, label="Renewable Energy", color="lightblue", marker="o", linestyle="--")
```

Gambar 3.34 Bagian Pertama Variabel Grafik Baris

Gambar 3.35 adalah bagian kedua variabel grafik baris yang mengandung fungsi penampilan tahun pada judul visualisasi grafik baris dan format hasil visualisasi yang telah diterapkan pada pembuatan variabel grafik baris untuk pertama kali.

```
year_text = entry_year.get().strip()
if year_text:
    plt.title("Energy Consumption by Month (" + year_text + ")")
else:
    plt.title("Energy Consumption by Month")
for i, value in enumerate(filtered_electricity):
    plt.text(i, value, f"{value:,.2f}", ha="center", va="bottom", fontsize=9, color="black")
for i, value in enumerate(filtered_renewable):
    plt.text(i, value, f"{value:,.2f}", ha="center", va="bottom", fontsize=9, color="black")
plt.xlabel("Months")
plt.ylabel("Energy (kWh)")
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Gambar 3.35 Bagian Kedu Variabel Grafik Baris

Gambar 3.36 adalah bagian pertama variabel dengan fungsi untuk mencetak *input* data secara seluruh melalui tombol *print totals* pada menu utama. Fungsi ini mengambil *input* dari setiap kategori data (listrik, listrik daur ulang, air bersih, air daur ulang, dan jejak karbon) dan mencetak dalam bentuk gambar dengan aset gambar agar pengguna lebih mudah dalam mengerti datanya. Besar aset gambar dijadikan 750px agar mudah dilihat.

```
# ---------------------- Print Totals Function with Icons ----------------------
def generate_totals_image():
    total_electricity = sum([float(e.get() or 0) for e in entry_line_electric])
    total_renewable = sum([float(e.get() or 0) for e in entry_line_renewable])
    total_clean_water = sum([float(e.get() or 0) for e in entry_pie_clean])
    total_recycled_water = sum([float(e.get() or 0) for e in entry_pie_recycled])
    total_carbon = sum([float(e.get() or 0) for e in entry_bar])

    # Icon size when printing
    PRINT_ICON_HEIGHT = 750
```

Gambar 3.36 Bagian Pertama Variabel Pencetakan Data Total

Gambar 3.37 adalah bagian kedua dari variabel *print totals* yang merupakan fungsi pemuatan dan menggantikan besar aset gambar dengan *pillow*.

```
# Local helper function to load and resize images
def pil_resize(path, new_height):
    img = Image.open(path)
    ratio = new_height / float(img.size[1])
    new_width = int(img.size[0] * ratio)
    return img.resize((new_width, new_height), Image.LANCZOS)
```

Gambar 3.37 Bagian Kedua Variabel Pencetakan Data Total

Gambar 3.38 adalah bagian ketiga variabel *print totals* yang merupakan fungsi konversi aset gambar menjadi format yang dapat ditampilkan dengan *matplotlib* yang digunakan pada *print totals*.

```
img_electricity = np.array(pil_resize("electricity.png", PRINT_ICON_HEIGHT))
img_renewable = np.array(pil_resize("leaf.png", PRINT_ICON_HEIGHT))
img_water = np.array(pil_resize("water.png", PRINT_ICON_HEIGHT))
img_recwater = np.array(pil_resize("recwater.png", PRINT_ICON_HEIGHT))
img_carbon = np.array(pil_resize("carbon.png", PRINT_ICON_HEIGHT))
```

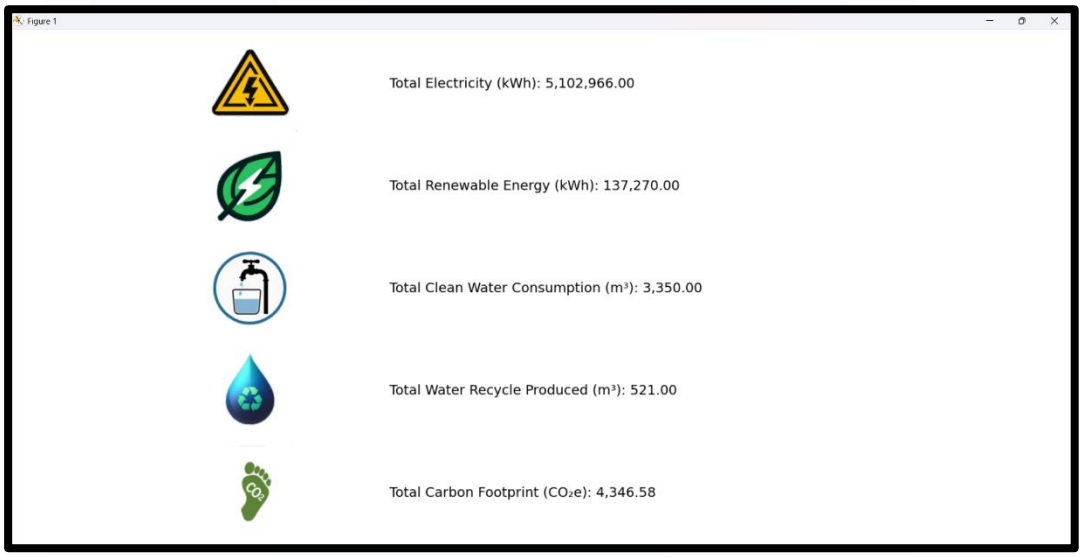
Gambar 3.38 Bagian Ketiga Variabel Pencetakan Data Total

Gambar 3.39 adalah bagian terakhir variabel pencetakan data total dengan fungsi untuk mencetak data total dalam format gambar yang pengguna dapat menyimpan pada sistem mereka.

```
fig, axs = plt.subplots(5, 2, figsize=(8, 10))
for ax in axs.flat:
    ax.axis('off')

axs[0, 0].imshow(img_electricity)
axs[0, 1].text(0, 0.5, f"Total Electricity (kWh): {total_electricity:,.2f}",
               fontsize=14, verticalalignment='center')

axs[1, 0].imshow(img_renewable)
axs[1, 1].text(0, 0.5, f"Total Renewable Energy (kWh): {total_renewable:,.2f}",
               fontsize=14, verticalalignment='center')

axs[2, 0].imshow(img_water)
axs[2, 1].text(0, 0.5, f"Total Clean Water Consumption (m³): {total_clean_water:,.2f}",
               fontsize=14, verticalalignment='center')

axs[3, 0].imshow(img_recwater)
axs[3, 1].text(0, 0.5, f"Total Water Recycle Produced (m³): {total_recycled_water:,.2f}",
               fontsize=14, verticalalignment='center')

axs[4, 0].imshow(img_carbon)
axs[4, 1].text(0, 0.5, f"Total Carbon Footprint (CO₂e): {total_carbon:,.2f}",
               fontsize=14, verticalalignment='center')

fig.tight_layout()
file_name = filedialog.asksaveasfilename(defaultextension=".png",
                                         filetypes=[("PNG Files", "*.png")])
if file_name:
    plt.savefig(file_name)
    messagebox.showinfo("Image Saved", f"Totals image saved as {file_name}")
plt.show()
```

Gambar 3.39 Bagian Terakhir Fungsi Pencetakan Data Total

Gambar 3.40 menunjukkan contoh pencetakan data total menggunakan fungsi *print totals*. Gambar tersebut menampilkan semua kategori data yang digunakan pada aplikasi visualisasi, data total pada semua kategori, tipe ukuran pada setiap kategori data, dan aset gambar yang digunakan untuk memudahkan identifikasi kategori data.



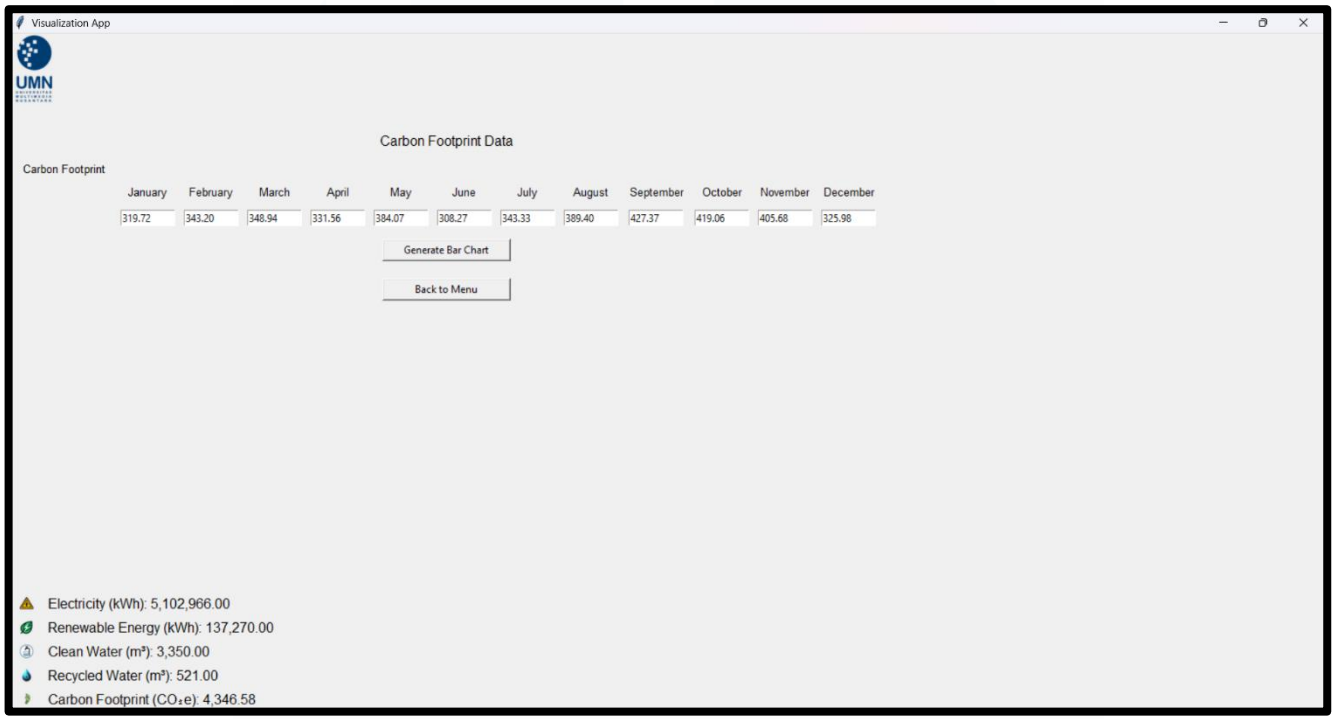
Gambar 3.40 Contoh Pencetakan Data Total

Gambar 3.41 adalah variabel untuk fungsi *live totals display* yang menampilkan total *input* data pada bawah kiri layar aplikasi. Fungsi ini mengambil hasil *input* pada setiap kolom *input* visualisasi dan menambahkannya untuk mendapatkan jumlah total. Data tersebut diperbarui setiap kali pengguna menekan tombol ketika menggunakan aplikasi.

```
# ---------------------- Live Totals Display Function ----------------------
def update_totals_display():
    total_electricity = sum([float(e.get() or 0) for e in entry_line_electric])
    total_renewable = sum([float(e.get() or 0) for e in entry_line_renewable])
    total_clean_water = sum([float(e.get() or 0) for e in entry_pie_clean])
    total_recycled_water = sum([float(e.get() or 0) for e in entry_pie_recycled])
    total_carbon = sum([float(e.get() or 0) for e in entry_bar])
    label_text_electricity.config(text=f"Electricity (kWh): {total_electricity:,.2f}")
    label_text_renewable.config(text=f"Renewable Energy (kWh): {total_renewable:,.2f}")
    label_text_water.config(text=f"Clean Water (m³): {total_clean_water:,.2f}")
    label_text_recwater.config(text=f"Recycled Water (m³): {total_recycled_water:,.2f}")
    label_text_carbon.config(text=f"Carbon Footprint (CO₂e): {total_carbon:,.2f}")

def bind_totals_update(entries):
    for e in entries:
        e.bind("<KeyRelease>", lambda event: update_totals_display())
```

Gambar 3.41 Variabel Penampilan Data Total Real Time

Gambar 3.42 adalah bentuk fungsi penampilan data total secara *real time* pada menu *input* grafik batang. Fungsi ini akan ditampilkan pada layar secara permanen agar pengguna dapat melihat total data yang telah dimasukkan.



Gambar 3.42 Fungsi Penampilan Total Data Real Time Pada Layar Aplikasi

Gambar 3.43 adalah variabel dengan fungsi navigasi antar menu pada aplikasi. Perbaikan telah dibuat untuk memastikan hanya menu yang pengguna memilih akan ditampilkan.

```
def show_section(section):
    menu_frame.pack_forget()
    bar_frame.pack_forget()
    pie_frame.pack_forget()
    line_frame.pack_forget()
    section.pack(fill="both", expand=True, pady=20)
```

Gambar 3.43 Variabel Navigasi Menu

Gambar 3.44 adalah variabel untuk menu *input* grafik batang. Menu ini mengandung 12 kotak *input* untuk setiap bulan, sebuah tombol untuk mencetak visualisasi grafik batang, dan tombol untuk kembali ke menu utama.

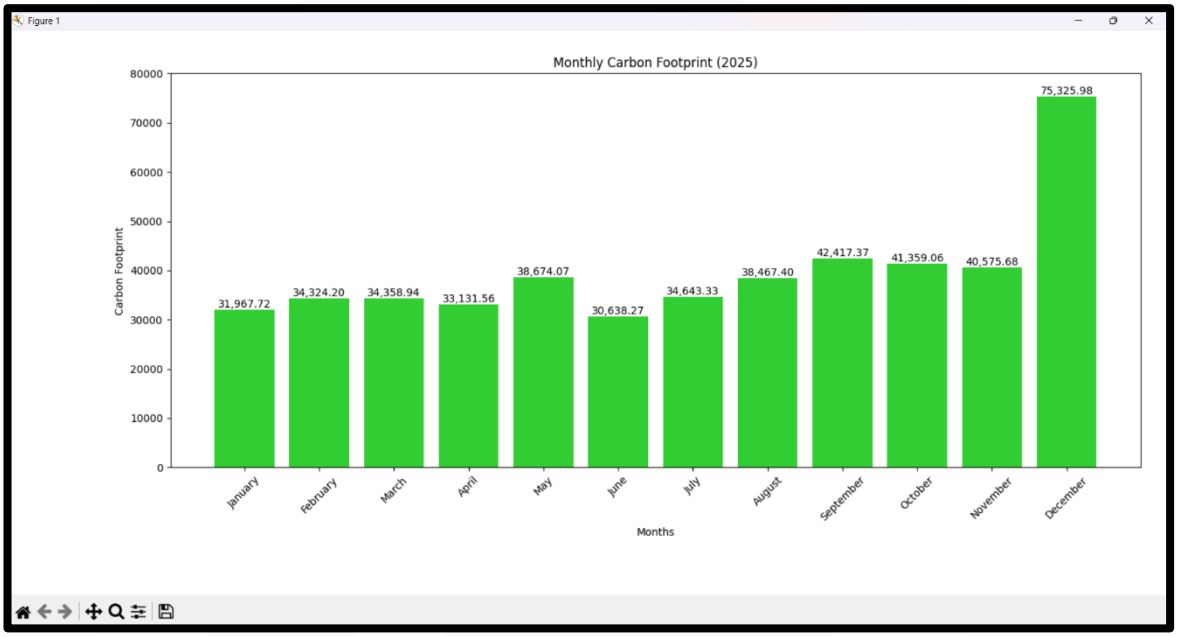
```
# --- Bar Chart Input Menu ---
bar_frame = tk.Frame(root)
tk.Label(bar_frame, text="Carbon Footprint Data", font=("Arial", 12)).grid(row=0, columnspan=13, pady=10)
tk.Label(bar_frame, text="Carbon Footprint", font=("Arial", 10)).grid(row=1, column=0, padx=10, sticky="w")
entry_bar = [tk.Entry(bar_frame, width=10) for _ in range(12)]
for i in range(12):
    tk.Label(bar_frame, text=months[i], font=("Arial", 10)).grid(row=2, column=i+1, padx=5, pady=5)
    entry_bar[i].grid(row=3, column=i+1, padx=5, pady=5)
tk.Button(bar_frame, text="Generate Bar Chart", command=generate_bar_chart, width=20).grid(row=4, columnspan=13, pady=10)
tk.Button(bar_frame, text="Back to Menu", command=lambda: show_section(menu_frame), width=20).grid(row=5, columnspan=13, pady=10)
bind_totals_update(entry_bar)
```

Gambar 3.44 Variabel Menu Input Grafik Batang

Gambar 3.45 adalah bentuk UI untuk menu input grafik batang.

Gambar 3.45 Bentuk UI Menu Input Grafik Batang

Gambar 3.46 adalah contoh visualisasi grafik batang dengan perbaikan yang telah diterapkan pada variabel.



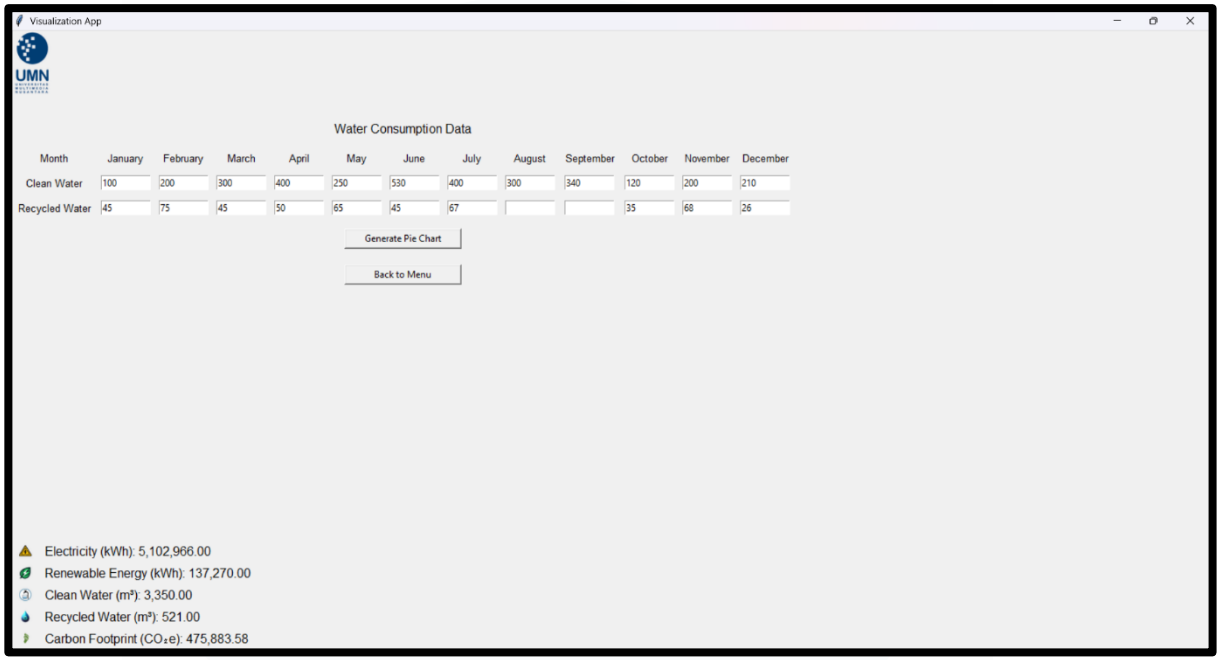
Gambar 3.46 Contoh Visualisasi Grafik Baris Dengan Perbaikan

Gambar 3.47 adalah variabel menu grafik pie yang memiliki 2 baris kotak *input* untuk kategori air bersih dan air daur ulang dengan 24 kotak *input* secara total, tombol untuk mencetak grafik pie, dan tombol kembali ke menu utama. Baris paling bawah variabel menunjukkan hubungan dengan fungsi penampilan data *real time* yang akan memperbarui penampilan data total.

```
pie_frame = tk.Frame(root)
tk.Label(pie_frame, text="Water Consumption Data", font=("Arial", 12)).grid(row=0, columnspan=13, pady=10)
tk.Label(pie_frame, text="Month", font=("Arial", 10)).grid(row=1, column=0, padx=5, pady=5)
tk.Label(pie_frame, text="Clean Water", font=("Arial", 10)).grid(row=2, column=0, padx=5, pady=5)
tk.Label(pie_frame, text="Recycled Water", font=("Arial", 10)).grid(row=3, column=0, padx=5, pady=5)
entry_pie_clean = [tk.Entry(pie_frame, width=10) for _ in range(12)]
entry_pie_recycled = [tk.Entry(pie_frame, width=10) for _ in range(12)]
for i in range(12):
    tk.Label(pie_frame, text=months[i], font=("Arial", 10)).grid(row=1, column=i+1, padx=5, pady=5)
    entry_pie_clean[i].grid(row=2, column=i+1, padx=5, pady=5)
    entry_pie_recycled[i].grid(row=3, column=i+1, padx=5, pady=5)
tk.Button(pie_frame, text="Generate Pie Chart", command=generate_pie_chart, width=20).grid(row=4, columnspan=13, pady=10)
tk.Button(pie_frame, text="Back to Menu", command=lambda: show_section(menu_frame), width=20).grid(row=5, columnspan=13, pady=10)
bind_totals_update(entry_pie_clean)
bind_totals_update(entry_pie_recycled)
```

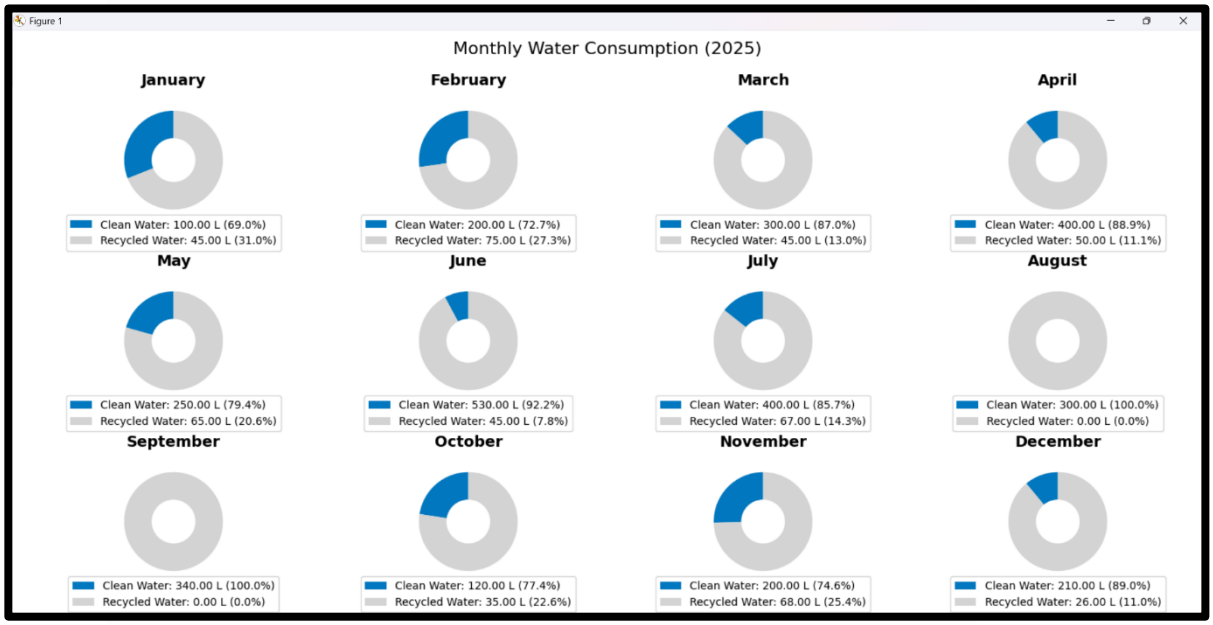
Gambar 3.47 Variabel Menu Input Grafik Pie

Gambar 3.48 menunjukkan bentuk UI menu input grafik pie.



Gambar 3.48 Bentuk UI Menu Input Grafik Pie

Gambar 3.49 adalah contoh visualisasi grafik pie setelah implementasi perbaikan format penampilan visualisasi.



Gambar 3.49 Contoh Visualisasi Grafik Pie Dengan Perbaikan

Gambar 3.50 adalah variabel untuk menu grafik baris dengan menu yang berbentuk sama dengan menu grafik batang, 12 kotak *input*, tombol pencetakan grafik baris, dan tombol untuk kembali ke menu utama.

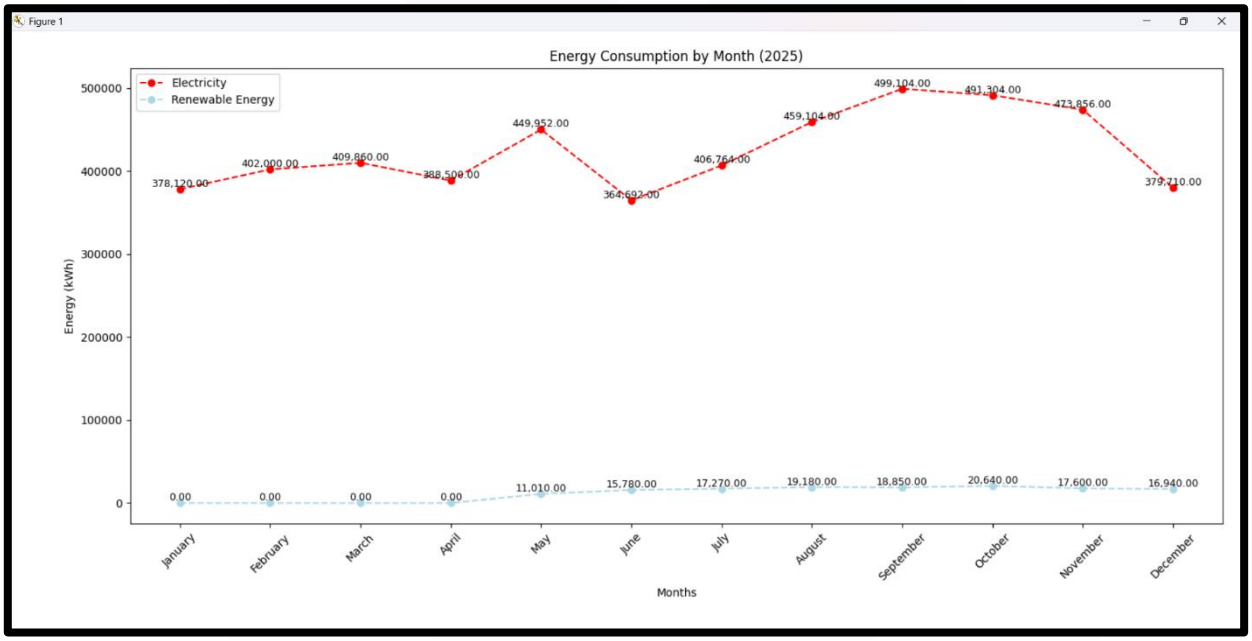
```
line_frame = tk.Frame(root)
tk.Label(line_frame, text="Energy Consumption Data", font=("Arial", 12)).grid(row=0, columnspan=13, pady=10)
tk.Label(line_frame, text="Month", font=("Arial", 10)).grid(row=1, column=0, padx=5, pady=5)
tk.Label(line_frame, text="Electricity", font=("Arial", 10)).grid(row=2, column=0, padx=5, pady=5)
tk.Label(line_frame, text="Renewable Energy", font=("Arial", 10)).grid(row=3, column=0, padx=5, pady=5)
entry_line_electric = [tk.Entry(line_frame, width=10) for _ in range(12)]
entry_line_renewable = [tk.Entry(line_frame, width=10) for _ in range(12)]
for i in range(12):
    tk.Label(line_frame, text=months[i], font=("Arial", 10)).grid(row=1, column=i+1, padx=5, pady=5)
    entry_line_electric[i].grid(row=2, column=i+1, padx=5, pady=5)
    entry_line_renewable[i].grid(row=3, column=i+1, padx=5, pady=5)
tk.Button(line_frame, text="Generate Line Graph", command=generate_line_graph, width=20).grid(row=4, columnspan=13, pady=10)
tk.Button(line_frame, text="Back to Menu", command=lambda: show_section(menu_frame), width=20).grid(row=5, columnspan=13, pady=10)
bind_totals_update(entry_line_electric)
bind_totals_update(entry_line_renewable)
```

Gambar 3.50 Variabel Menu Input Grafik Baris

Gambar 3.51 merupakan penampilan menu *input* grafik baris.

Gambar 3.51 Tampilan Menu Input Grafik Baris

Gambar 3.52 adalah contoh visualisasi grafik baris dengan perbaikan yang telah diterapkan pada variabelnya.



Gambar 3.52 Contoh Visualisasi Grafik Baris Dengan Perbaikan

Gambar 3.53 adalah variabel untuk menentukan menu utama aplikasi.

```
menu_frame = tk.Frame(root)
menu_frame.pack(fill="both", expand=True)
```

Gambar 3.53 Variabel Yang Menentukan Menu Utama

Gambar 3.54 adalah variabel yang menambah kotak *input* tahun pada menu utama. Tahun akan ditampilkan pada judul visualisasi grafik jika pengguna mengisi kotak *input*. Tahun tidak akan ditampilkan pada judul jika pengguna tidak mengisi kotak *input*.

```
year_frame = tk.Frame(menu_frame)
year_frame.pack(pady=5)
tk.Label(year_frame, text="Enter Year:", font=("Arial", 10)).pack(side="left")
entry_year = tk.Entry(year_frame, width=10)
entry_year.pack(side="left")
```

Gambar 3.54 Variabel Kotak Input Tahun

Gambar 3.55 adalah variabel tombol pada menu utama yang dimulai dengan *carbon footprint* (grafik batang), *water consumption* (grafik pie), *energi consumption* (grafik baris), *save, save as, load,* dan *print totals.*

```
tk.Button(menu_frame, text="Carbon Footprint", command=lambda: show_section(bar_frame), width=20).pack(pady=5)
tk.Button(menu_frame, text="Water Consumption", command=lambda: show_section(pie_frame), width=20).pack(pady=5)
tk.Button(menu_frame, text="Energy Consumption", command=lambda: show_section(line_frame), width=20).pack(pady=5)
tk.Button(menu_frame, text="Save", command=save_data, width=20).pack(pady=5)
tk.Button(menu_frame, text="Save As", command=save_data_as, width=20).pack(pady=5)
tk.Button(menu_frame, text="Load", command=load_data, width=20).pack(pady=5)
tk.Button(menu_frame, text="Print Totals", command=generate_totals_image, width=20).pack(pady=5)
```

Gambar 3.55 Variabel Tombol Pada Menu Utama

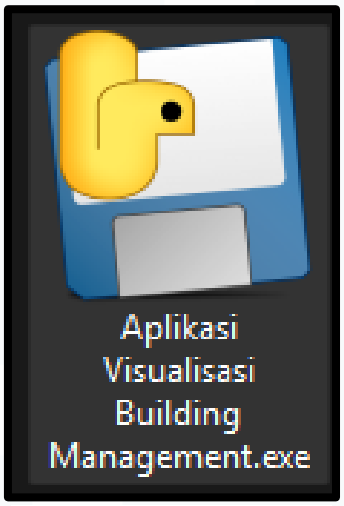
Gambar 3.56 adalah variabel terakhir pada aplikasi visualisasi dalam bentuk visualisasi data total *real time.*

```
totals_frame = tk.Frame(root)
totals_frame.pack(side="bottom", fill="x")
label_icon_electricity = tk.Label(totals_frame, image=icon_electricity)
label_icon_electricity.grid(row=0, column=0, padx=5, pady=2, sticky="w")
label_text_electricity = tk.Label(totals_frame, text="", font=("Arial", 12))
label_text_electricity.grid(row=0, column=1, padx=5, pady=2, sticky="w")
label_icon_renewable = tk.Label(totals_frame, image=icon_renewable)
label_icon_renewable.grid(row=1, column=0, padx=5, pady=2, sticky="w")
label_text_renewable = tk.Label(totals_frame, text="", font=("Arial", 12))
label_text_renewable.grid(row=1, column=1, padx=5, pady=2, sticky="w")
label_icon_water = tk.Label(totals_frame, image=icon_water)
label_icon_water.grid(row=2, column=0, padx=5, pady=2, sticky="w")
label_text_water = tk.Label(totals_frame, text="", font=("Arial", 12))
label_text_water.grid(row=2, column=1, padx=5, pady=2, sticky="w")
label_icon_recwater = tk.Label(totals_frame, image=icon_recwater)
label_icon_recwater.grid(row=3, column=0, padx=5, pady=2, sticky="w")
label_text_recwater = tk.Label(totals_frame, text="", font=("Arial", 12))
label_text_recwater.grid(row=3, column=1, padx=5, pady=2, sticky="w")
label_icon_carbon = tk.Label(totals_frame, image=icon_carbon)
label_icon_carbon.grid(row=4, column=0, padx=5, pady=2, sticky="w")
label_text_carbon = tk.Label(totals_frame, text="", font=("Arial", 12))
label_text_carbon.grid(row=4, column=1, padx=5, pady=2, sticky="w")
update_totals_display()

root.mainloop()
```

Gambar 3.56 Variabel Format Penempatan Fungsi Penampilan Data Real Time

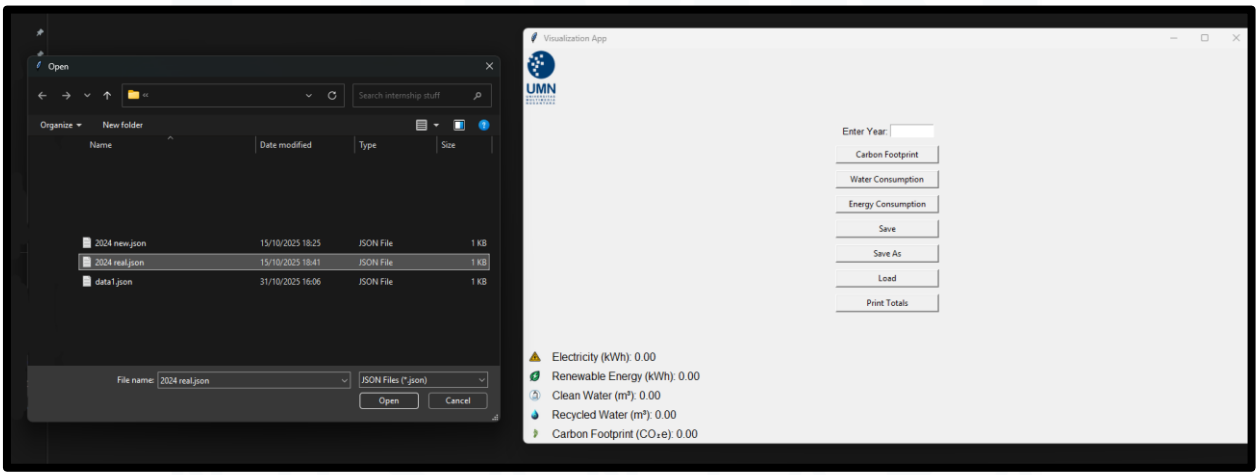
3.4 Pengalaman Pengguna

Aplikasi terlihat dalam *file explorer* pada komputer dalam bentuk aplikasi dengan nama "Aplikasi Visualisasi Building Management.exe" dan dapat dibuka dengan menekan tombol kiri pada mouse dua kali.



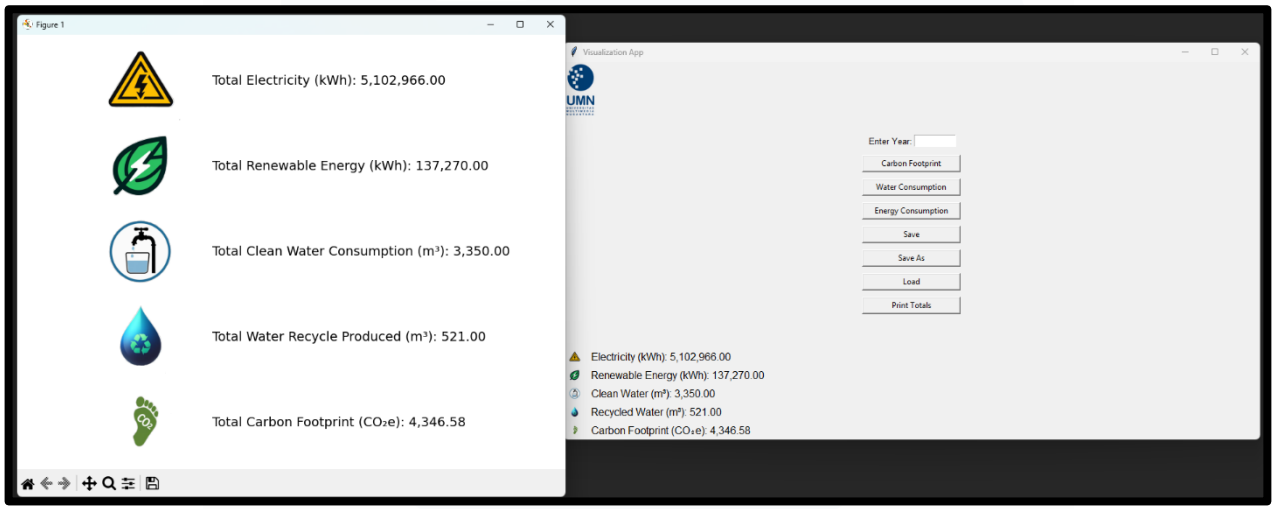
Gambar 3.57 Bentuk Aplikasi
Dalam File Explorer

Menu utama aplikasi visualisasi memiliki tombol untuk membuka menu *input* untuk jejak karbon, konsumsi air, dan konsumsi listrik. Selain itu, terdapat tombol untuk menyimpan data yang dimasukkan ke dalam menu *input*. Data tersebut dapat dibuka ulang dengan tombol "*Load*", yang membuka menu *file explorer* untuk memilih data yang disimpan.



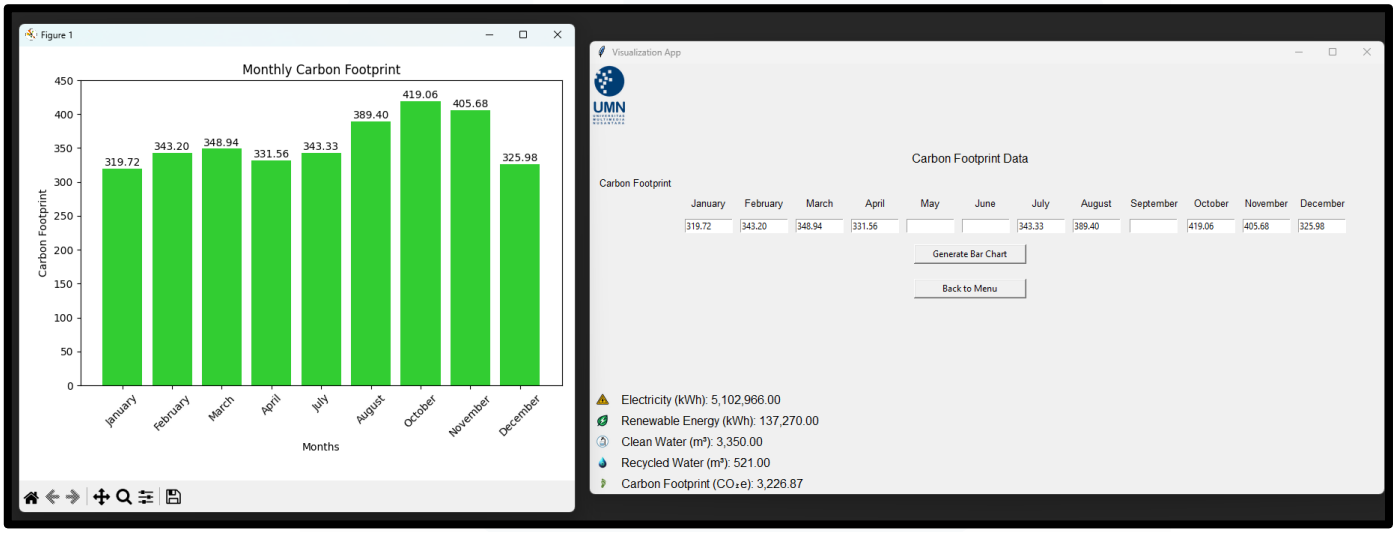
Gambar 3.58 Bentuk Menu File Explorer Yang Dibuka Oleh Fungsi Load

Tombol terakhir adalah tombol untuk mencetak jumlah total data yang telah dimasukkan ke dalam menu *input* jejak karbon, konsumsi air, dan konsumsi listrik. Setelah menekan tombol "*Print Totals*", menu *file explorer* akan terbuka dan tempat penyimpanan gambar data total dapat dipilih. Setelah gambar disimpan, menu yang menampilkan hanya data total akan terbuka.



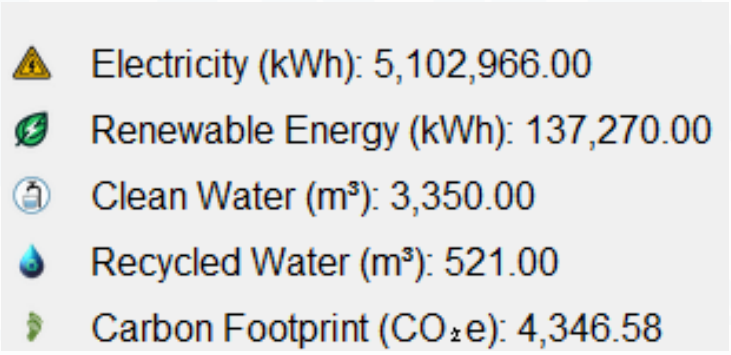
Gambar 3.59 Bentuk Output Print Totals

Fungsi *input* data memiliki fitur di mana bulan yang tidak memiliki *input* tidak akan ditampilkan pada hasil visualisasi. Setelah memasukkan data ke dalam kolom *input*, tombol "*generate*" ditekan untuk membuat visualisasi. Jejak karbon menggunakan grafik batang, konsumsi air menggunakan grafik pie, dan konsumsi listrik menggunakan grafik baris.



Gambar 3.60 Bentuk Output Grafik Baris Dengan Kolom Data Mei, Juni, dan September Yang Konsong

Fitur *live data display* menampilkan data pada bawah kiri aplikasi. Data yang ditampilkan pada fitur tersebut diperbarui setiap adanya masukan data pada kolom *input*.



Gambar 3.61 Tampilan Fitur Live Data Display

3.5 Kendala yang Ditemukan

Selama kegiatan program kerja magang berlangsung, terdapat beberapa yang dialami ketika melakukan pembuatan aplikasi visualisasi, yaitu:

1. Bahasa pemrograman yang digunakan untuk pembuatan aplikasi tidak ditentukan dengan jelas.
2. Kesulitan dalam menyesuaikan aplikasi berdasarkan permintaan pihak *building management* karena kurang spesifik dalam spesifikasi aplikasi yang dibutuhkan.
3. Mendapatkan aset gambar untuk digunakan dalam aplikasi visualisasi sebagai identifikasi visual kategori data.

3.6 Solusi atas Kendala yang Ditemukan

Berdasarkan kendala yang dialami selama kegiatan program kerja magang, terdapat solusi atas kendala yang ditemukan untuk mengatasi permasalahan, yaitu:

1. Memanfaatkan forum *online* dan dokumentasi bahasa pemrograman yang digunakan untuk menentukan bahasa pemrograman yang akan digunakan dalam pembuatan aplikasi.
2. Konsultasi dengan pihak *building management* untuk mendapatkan informasi yang lebih detail untuk aplikasi serta memberitahukan progres aplikasi agar pihak dapat melihat langsung hal yang dapat disesuaikan pada kebutuhan mereka.
3. Menggunakan situs AI untuk generasi aset gambar yang kemudian di edit hingga mirip dengan contoh logo.