LAPORAN HASIL PROJEK UJIAN AKHIR SEMESTER

"Monetify – Aplikasi Finance Tracker"

MATA KULIAH PEMROGRAMAN DASAR

Oleh:

1. Nayla Rizky Amalia (25031554184)
2. Lintang Alysia Gantari (24031554097)

Dosen Pengampu Mata Kuliah:

Dr. Atik Wintarti, M. Kom

SAINS DATA

2025B 2025/2026

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengelolaan keuangan pribadi adalah hal yang krusial, terutama di tengah meningkatnya kebutuhan dan gaya hidup modern. Banyak orang kesulitan mencatat pemasukan, mengontrol pengeluaran, atau mengetahui sisa tabungan. Tanpa alat yang praktis dan mudah digunakan, pencatatan transaksi harian sering terabaikan, yang akhirnya berdampak pada ketidakstabilan kondisi finansial. Oleh karena itu, dibutuhkan sebuah solusi digital untuk membantu pengguna memahami dan memantau kondisi keuangan mereka dengan lebih efektif.

Perkembangan teknologi membuka peluang untuk menghadirkan aplikasi keuangan yang sederhana, cepat, dan mudah dipahami oleh semua kalangan. Solusi digital yang ringan dan *user-friendly* lebih dibutuhkan daripada pencatatan manual, terutama bagi mereka yang baru belajar mengatur keuangan. Aplikasi yang mampu memberikan ringkasan keuangan otomatis, menampilkan grafik pengeluaran, serta menyimpan riwayat transaksi dengan rapi, akan sangat memfasilitasi pengambilan keputusan finansial yang lebih baik.

Untuk menjawab kebutuhan tersebut, hadir *Monetify*, sebuah aplikasi pengelolaan keuangan pribadi yang fokus pada kesederhanaan, fungsionalitas, dan kenyamanan pengguna. *Monetify* menyediakan fitur lengkap seperti pencatatan pemasukan dan pengeluaran, kategori transaksi, perhitungan otomatis saldo, visualisasi data melalui diagram, serta kemampuan ekspor laporan ke format PDF untuk dokumentasi. Dengan tampilan yang bersih, intuitif, dan *user-friendly*, Monetify menjadi solusi praktis yang membantu pengguna mengatur keuangan sehari-hari dengan lebih terarah dan efisien.

1.2 Rumusan Masalah

1. Bagaimana aplikasi Monetify dapat membantu pengguna dalam memantau dan mengelola pemasukan serta pengeluaran mereka secara efektif dan efisien?
2. Apa dampak penggunaan aplikasi *Monetify* terhadap pola pengelolaan keuangan pribadi pengguna?
3. Bagaimana fitur visualisasi data, seperti grafik pemasukan dan pengeluaran pada *Monetify*, membantu pengguna memahami kondisi keuangan mereka secara lebih jelas?

   Bagaimana fitur laporan otomatis dan export PDF pada *Monetify* dapat mempermudah pengguna dalam memantau perkembangan keuangan dan melakukan pencatatan keuangan jangka panjang?

1.3 Tujuan

1. Memfasilitasi Pengelolaan Keuangan: Menyediakan aplikasi yang mudah digunakan, terorganisir, dan dapat diakses kapan saja untuk mengelola keuangan pribadi secara efektif.
2. Meningkatkan akurasi dan keamanan pencatatan transaksi melalui penyimpanan data terstruktur.
3. Meningkatkan Pemahaman Finansial: Membantu pengguna mengurangi kesalahan pencatatan dan meningkatkan pemahaman terhadap kondisi keuangan mereka melalui fitur visualisasi (grafik) pemasukan dan pengeluaran.
4. Mempermudah Pelaporan: Menyediakan kemudahan dalam pelaporan keuangan melalui fitur ekspor PDF yang rapi dan profesional.

BAB II
ANALISIS DAN PERANCANGAN

2.1 Analisis Kebutuhan Aplikasi

Aplikasi Monetify dikembangkan untuk membantu pengguna mengelola keuangan pribadi mereka dengan lebih mudah. Fungsinya mencakup pencatatan transaksi, pemantauan saldo, dan menampilkan riwayat transaksi. Untuk memastikan aplikasi ini memenuhi harapan pengguna, penting untuk mengidentifikasi kebutuhan yang harus dipenuhi. Kebutuhan ini dibagi menjadi dua kategori: kebutuhan fungsional, yang mencakup fitur dan fungsi yang harus ada, dan kebutuhan non-fungsional, yang mencakup karakteristik seperti kemudahan dan performa**.**

1. Kebutuhan Fungsional

a. Input Transaksi

Fitur **:** Pengguna dapat menambahkan transaksi pemasukan atau pengeluaran.

Kebutuhan **:**

- Form input deskripsi, jumlah, dan kategori.
- Pilihan jenis transaksi (pemasukan/pengeluaran)
- Tombal “Tambah”

b. Tampilan Riwayat Transaksi

Fitur **:** Menampilkan daftar semua transaksi yang telah dicatat.

Kebutuhan **:** Tabel (Treeview) berisi tipe, deskripsi, jumlah, serta kategori.

c. Ringkasan Keuangan

Fitur **:** Menampilkan total pemasukan, total pengeluaran, dan sisa saldo.

Kebutuhan **:** Label ringkasan yang menampilkan total pemasukan, pengeluaran dan sisa saldo dengan perhitungan otomatis setiap ada perubahan data.

d. Grafik Keuangan (Charts)

Fitur **:** Menampilkan diagram persentase pemasukan dan pengeluaran berdasarkan kategori.

Kebutuhan **:**

- Pie chart untuk pemasukan.
- Pie chart untuk pengeluaran.
- Pembaruan grafik otomatis sesuai perubahan transaksi.

e. Export Laporan PDF

Fitur : Pengguna dapat mengekspor riwayat transaksi menjadi file PDF.

Kebutuhan :

- Template laporan PDF.
- Dialog penyimpanan file.
- Konversi tabel transaksi ke format PDF.

f. Hapus Data

Fitur : Menghapus transaksi yang dipilih maupun seluruh transaksi sekaligus.

Kebutuhan :

- Fitur pilih transaksi yang ingin dihapus
- Penghapusan data transaksi dan pembaruan tampilan chart & ringkasan.
- Notifikasi konfirmasi “Yes/No” sebelum penghapusan.

2. Kebutuhan Non-Fungsional

a. Kemudahan Penggunaan

Kebutuhan :

- Tampilan sederhana dan mudah dipahami.
- Input dan tombol mudah diakses.
- Notifikasi peringatan, error, dan konfirmasi yang jelas.

b. Keamanan Data

Kebutuhan :

- Data hanya disimpan secara lokal (offline).
- Tidak membagikan data ke pihak luar.

c. Performa

Kebutuhan :

- Aplikasi harus responsif meski data transaksi banyak.
- Pembaruan ringkasan dan grafik harus berlangsung cepat dan tepat.

d. Teknis

Kebutuhan :

- Bahasa Pemrograman: Python.
- Library :
  - Tkinter : untuk antarmuka.
  - ttk.Treeview : untuk menampilkan tabel transaksi.
  - Matplotlib : untuk menampilkan grafik pemasukan dan pengeluaran.
  - Reportlab : untuk export laporan transaksi ke format PDF.
  - Filedialog & Massagebox : untuk dialog penyimpanan dan notifikasi.
- File Tambahan :
  - File PDF (.pdf) : untuk menyimpan data transaksi.

2.2 Diagram Alur Aplikasi "Smart Finance"

Rincian Diagram Alur :

1. Mulai
   - Aplikasi dijalankan dan masuk ke halaman utama.
2. Tambah Transaksi
   - Pengguna menekan tombol Tambah Transaksi, lalu:
3. Pilih Jenis Transaksi
   - Pengguna memilih salah satu :
     - Pemasukan, atau
     - Pengeluaran
4. Isi Deskripsi
   - User mengisi deskripsi transaksi, misalnya “transfer ibu”, “belanja”, “transport”, dll.
5. Isi Jumlah
   - User memasukkan jumlah uang (dalam rupiah)
6. Pilih Kategori
   - Kategori muncul sesuai jenis transaksi :
     - Untuk pengeluaran: jajan, transport, tagihan, hiburan, dll.
     - Untuk pemasukan: gaji, uang saku, lainnya.
7. Simpan
   - Data transaksi disimpan ke tabel.

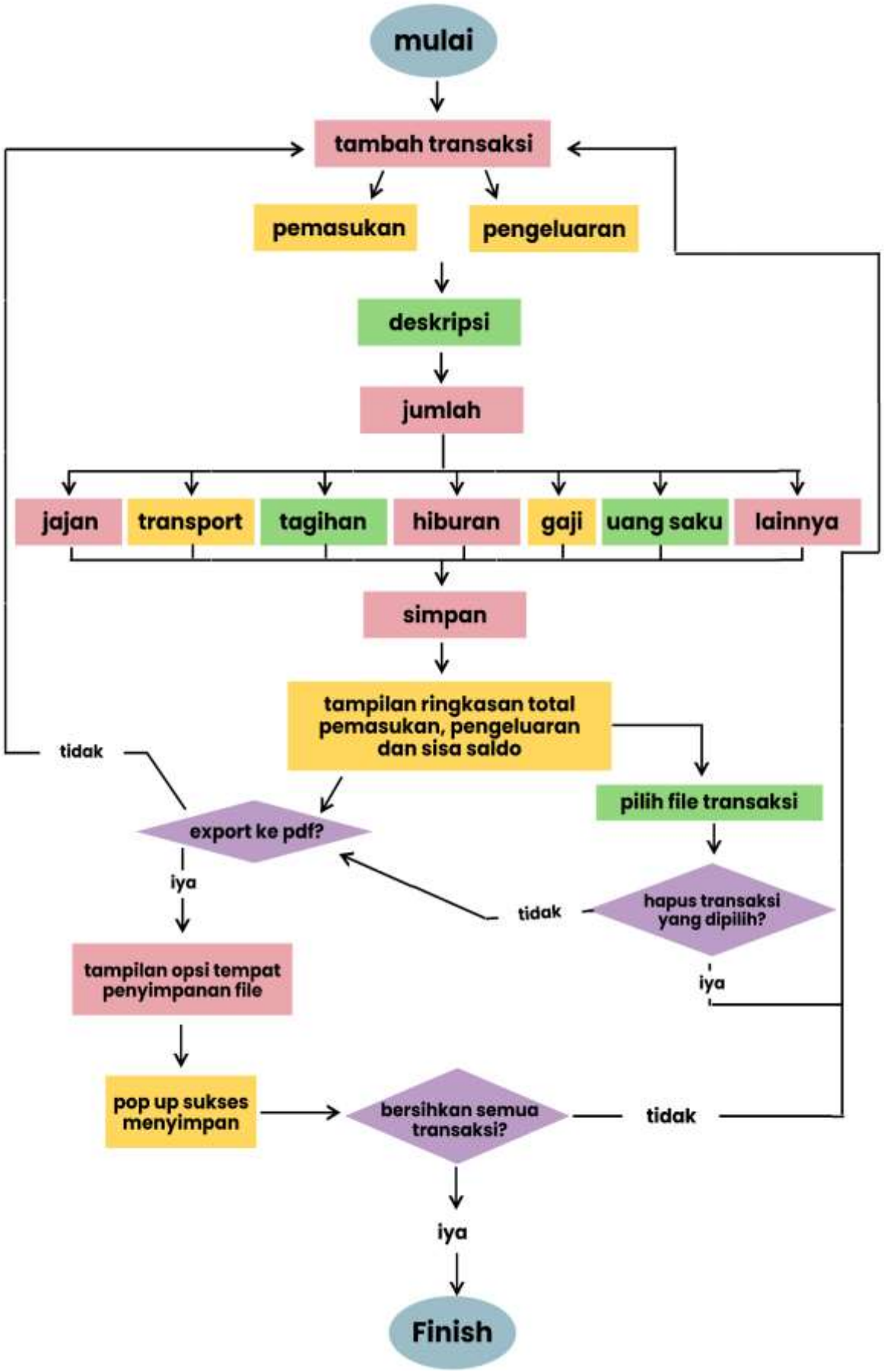
- Grafik dan ringkasan otomatis diperbarui.

8. Tampilan Ringkasan Keuangan
    - Setelah transaksi masuk, total pemasukan, pengeluaran, dan sisa saldo akan ditampilkan secara otomatis.
9. Opsi Setelah Menyimpan

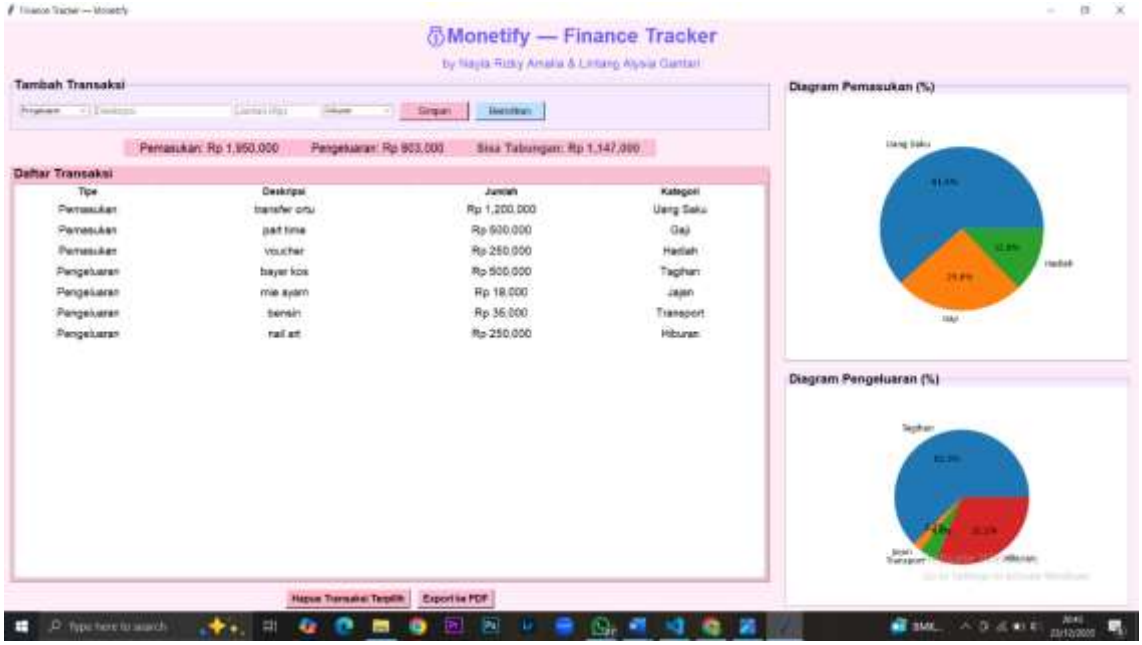
    Pengguna bisa memilih salah satu dari menu :

    - Export ke PDF
        - Muncul dialog pilihan lokasi penyimpanan.
        - Setelah file tersimpan, muncul pop-up sukses.
    - Pilih File Transaksi
        - Jika pengguna membuka tabel dan memilih baris transaksi akan muncul pertanyaan: “Hapus transaksi yang dipilih?”
        - Jika Ya, transaksi dihapus dan ringkasan serta grafik diperbarui.
        - Jika Tidak, kembali ke menu utama.
    - Bersihkan Semua Transaksi
        - Jika pengguna memilih opsi Clear All, akan muncul pertanyaan: “Bersihkan semua transaksi?”.
        - Jika Ya, semua transaksi dihapus dan grafik kosong kembali.
        - Jika Tidak, tetap pada tampilan transaksi.

Gambar Diagram Alur :

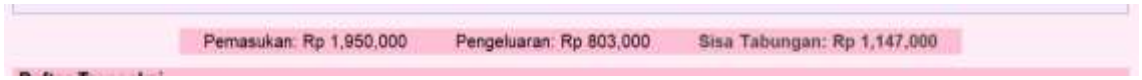


2.3 Sketsa Desain Antarmuka



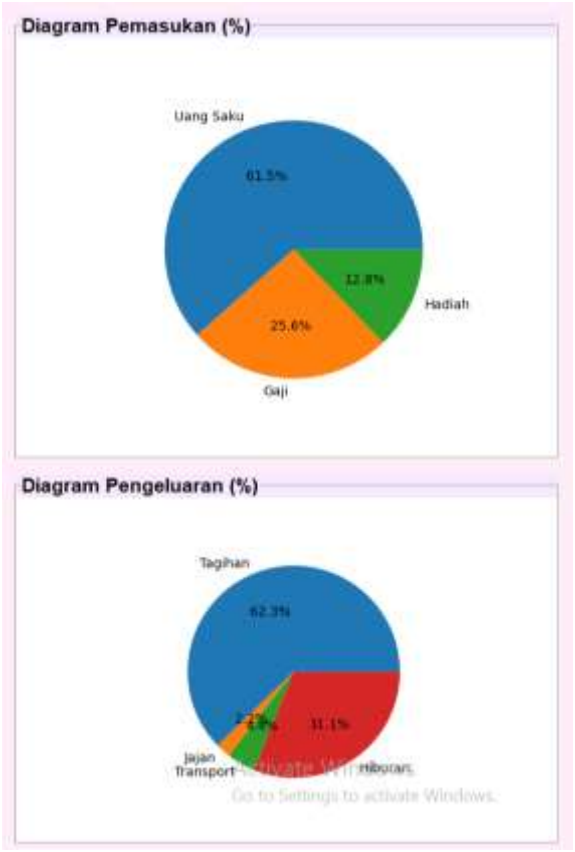
Sketsa 1 Menu Utama

Sketsa 2 Label Tambah Transaksi



Sketsa 3 Label Total Pemasukan, Pengeluaran, dan Sisa Saldo

Sketsa 4 Tabel (Treeview) Daftar Transaksi



Sketsa 5 Diagram Pemasukan dan Pengeluaran

BAB III
IMPLEMENTASI

3.1 Penjelasan Kode Program

Program yang kami buat merupakan aplikasi *Finance Tracker* bernama *Monetify* yang dibangun menggunakan Tkinter. Aplikasi ini berfungsi untuk mencatat pemasukan dan pengeluaran, menghitung dan menampilkan total pemasukan, pengeluaran, serta sisa saldo. Selain itu, aplikasi juga menampilkan diagram pie yang menggambarkan presentase tiap kategori pemasukan dan pengeluaran menggunakan matplotlib. Seluruh transaksi ditampilkan dalam table treeview, di mana pengguna dapat menambah data transaksi baru, menghapus transaksi tertentu, dan menghapus seluruh data transaksi. Aplikasi ini juga menyediakan fitur export ke PDF dengan bantuan library ReportLab, sehingga laporan lengkap berisi daftar transaksi dan ringkasan total dapat disimpan. Secara keseluruhan, program ini menggabungkan antarmuka GUI, tabel data, grafik visual, dan output PDF unuk mengelola data keuangan secara sederhana dan interaktif.

Adapun penjelasan terkait kode program *Finance Tracker* - *Monetify* adalah sebagai berikut:

3.1.1 Input Frame

```
        # ------------------ INPUT FRAME ------------------
        input_frame = tk.LabelFrame(
            left_side,
            text="Tambah Transaksi",
            padx=10,
            pady=10,
            bg="#f7e9ff",
            font=("Poppins", 16, "bold")
        )
        input_frame.pack(padx=10, pady=5, fill="x")

        self.type_var = tk.StringVar(value="Pemasukan")
        ttk.Combobox(
            input_frame,
            textvariable=self.type_var,
            values=["Pemasukan", "Pengeluaran"],
            width=15, state="readonly"
        ).grid(row=0, column=0, padx=5, pady=5)

        # ---- Deskripsi Entry ----
        self.desc_entry = tk.Entry(input_frame, width=25, font=("Poppins", 12), fg="grey")
        self.desc_entry.grid(row=0, column=1, padx=5, pady=5)
        self.desc_entry.insert(0, "Deskripsi")
        self.desc_entry.bind("<FocusIn>", self.clear_placeholder_desc)
        self.desc_entry.bind("<FocusOut>", self.add_placeholder_desc)

        # ---- Jumlah Entry ----
        self.amount_entry = tk.Entry(input_frame, width=15, font=("Poppins", 12), fg="grey")
        self.amount_entry.grid(row=0, column=2, padx=5, pady=5)
        self.amount_entry.insert(0, "Jumlah (Rp)")
        self.amount_entry.bind("<FocusIn>", self.clear_placeholder_amount)
        self.amount_entry.bind("<FocusOut>", self.add_placeholder_amount)
```

```
        self.category_var = tk.StringVar(value="Lainnya")
        ttk.Combobox(
            input_frame,
            textvariable=self.category_var,
            values=["Jajan", "Transport", "Tagihan", "Hiburan", "Gaji", "Uang Saku", "Hadiah", "Lainnya"],
            width=15, state="readonly"
        ).grid(row=0, column=3, padx=5, pady=5)

        tk.Button(
            input_frame,
            text="Simpan",
            command=self.add_transaction,
            bg="#fcbad3",
            fg="black",
            width=12,
            font=("Poppins", 12)
        ).grid(row=0, column=4, padx=5, pady=5)

        tk.Button(
            input_frame,
            text="Bersihkan",
            command=self.clear_all,
            bg="#bde0fe",
            fg="black",
            width=12,
            font=("Poppins", 12)
        ).grid(row=0, column=5, padx=5, pady=5)
```

Bagian kode tersebut digunakan untuk membuat form input transaksi dalam aplikasi. Form ini memungkinkan pengguna memasukkan detail transaksi mulai dari tipe, deskripsi, jumlah uang, hingga kategori. Setiap segmen dibuat dalam satu baris agar rapi, dan tombol tambahan disediakan untuk menyimpan atau membersihkan seluruh data.

Berikut adalah bagian yang termasuk dalam input frame:

- Tipe Transaksi (Pemasukan/Pengeluaran)
  Dibuat menggunakan combobox agar pengguna dapat memilih jenis transaksi dengan mudah, yaitu pemasukan atau pengeluaran.
- Deskripsi
  Menggunakan entry dengan placeholder “Deskripsi”. Placeholder akan hilang saat kolom diklik dan muncul kembali saat kosong.
- Jumlah (Rp)
  Kolom input angka juga memiliki placeholder. Digunakan untuk memasukkan nominal transaksi.
- Kategori
  Combobox berisi pilihan seperti jajan, transport, hiburan, gaji, dan lainnya untuk mengelompokkan transaksi.
- Tombol Simpan
  Menjalankan fungsi add_transaction() untuk menambahkan data ke tabel dan daftar transaksi.
- Tombol Bersihkan
  Memanggil fungsi clear_all() untuk menghapus seluruh transaksi yang sudah ada.

3.1.2 Balance Frame

```
# ------------------ BALANCE FRAME ------------------
balance_frame = tk.Frame(left_side, bg="#ffbed3")
balance_frame.pack(pady=10)

self.income_label = tk.Label(
    balance_frame,
    text="Pemasukan: Rp 0",
    font=("Poppins", 15),
    bg="#ffbed3"
)
self.income_label.grid(row=0, column=0, padx=25)

self.expense_label = tk.Label(
    balance_frame,
    text="Pengeluaran: Rp 0",
    font=("Poppins", 15),
    bg="#ffbed3"
)
self.expense_label.grid(row=0, column=1, padx=25)

self.balance_label = tk.Label(
    balance_frame,
    text="Sisa Tabungan: Rp 0",
    font=("Poppins", 15, "bold"),
    bg="#ffbed3",
    fg="#AAA"
)
self.balance_label.grid(row=0, column=2, padx=25)
```

Bagian kode tersebut berfungsi untuk menampilkan ringkasan keuangan yang meliputi total pemasukan, total pengeluaran, dan sisa tabungan. Semua elemen ini ditempatkan dalam sebuah frame dengan latar warna merah muda agar tampilannya seragam dengan tema aplikasi. Masing-masing nilai ditampilkan menggunakan label: label pemasukan dan pengeluaran menampilkan jumlah awal “Rp 0”, sementara label sisa tabungan ditampilkan dengan font tebal agar lebih terlihat. Ketiga label ini nantinya akan diperbarui secara otomatis setiap kali pengguna menambahkan atau menghapus transaksi, sehingga informasi saldo selalu tampil terbaru.

3.1.3 List Frame

```
# ------------------ LIST FRAME ------------------
list_frame = tk.LabelFrame(
    left_side,
    text="Daftar Transaksi",
    bg="#ffbed3",
    font=("Poppins", 16, "bold")
)
list_frame.pack(padx=10, pady=5, fill="both", expand=True)

self.tree = ttk.Treeview(
    list_frame,
    columns=("Tipe", "Deskripsi", "Jumlah", "Kategori"),
    show="headings"
)
style = ttk.Style()
style.configure("Treeview.Heading", font=("Poppins", 12, "bold"))
style.configure("Treeview", font=("Poppins", 14), rowheight=35)

for col in ("Tipe", "Deskripsi", "Jumlah", "Kategori"):
    self.tree.heading(col, text=col)

self.tree.column("Tipe", anchor="center", width=130)
self.tree.column("Deskripsi", anchor="center", width=300)
self.tree.column("Jumlah", anchor="center", width=200)
self.tree.column("Kategori", anchor="center", width=170)
self.tree.pack(fill="both", expand=True, padx=5, pady=5)
```

Bagian kode ini digunakan untuk membuat tampilan daftar transaksi, yaitu area yang menampilkan seluruh data pemasukan maupun pengeluaran yang sudah dimasukkan pengguna. Frame ini dibuat menggunakan LabelFrame agar memiliki judul dan tampak rapi. Di dalamnya terdapat komponen treeview dari ttk, yang berfungsi sebagai tabel untuk menampilkan transaksi dalam empat kolom: Tipe, Deskripsi, Jumlah, dan Kategori. Setiap kolom diatur headingnya, lebar kolom, serta posisi teks agar tampil terpusat. Setelah pengaturan selesai, tabel ditampilkan memenuhi area frame menggunakan pack() dengan opsi fill="both" dan expand=True, sehingga tabel akan menyesuaikan ukuran jendela dan tetap terlihat secara penuh.

3.1.4 Buttons Frame

```
163     # ------------------ BUTTONS ------------------
164     export_frame = tk.Frame(left_side, bg="#ffeef7")
165     export_frame.pack(pady=5)
166
167     tk.Button(
168         export_frame,
169         text="Hapus Transaksi Terpilih",
170         command=self.delete_selected,
171         bg="#ffb4c6",
172         fg="black",
173         width=20,
174         font=("Poppins", 12, "bold")
175     ).pack(side="left", padx=6)
176
177     tk.Button(
178         export_frame,
179         text="Export ke PDF",
180         command=self.export_pdf,
181         bg="#fcbad3",
182         fg="black",
183         width=12,
184         font=("Poppins", 12, "bold")
185     ).pack(side="left", padx=6)
```

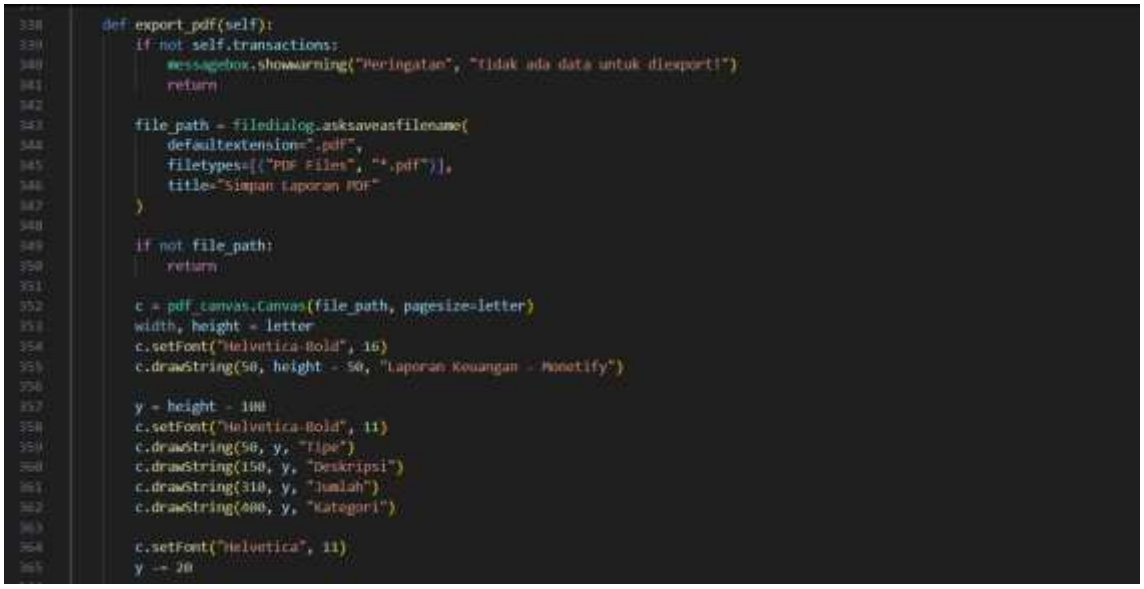
Bagian *Buttons Frame* ini berfungsi sebagai area tombol aksi yang diletakkan di bawah daftar transaksi. Tombol pertama adalah "Hapus Transaksi Terpilih"**,** yang ketika diklik akan menjalankan fungsi delete_selected untuk menghapus baris transaksi yang sedang dipilih di tabel. Tombol kedua adalah "Export ke PDF"**,** yang memanggil fungsi export_pdf untuk menyimpan seluruh daftar transaksi ke dalam file PDF.

3.1.5 Chart Frame

```
        # ------------------ CHART FRAME ------------------
        charts_container = tk.Frame(right_side, bg="#ffeef7")
        charts_container.pack(fill="both", expand=True, padx=10)

        top_chart = tk.LabelFrame(
            charts_container,
            text="Diagram Pemasukan (%)",
            bg="#f7e9ff",
            font=("Poppins", 16, "bold")
        )
        top_chart.pack(fill="both", expand=True, padx=5, pady=8)

        self.fig_income, self.ax_income = plt.subplots(figsize=(5.8, 4.5), dpi=100)
        self.canvas_income = FigureCanvasTkAgg(self.fig_income, master=top_chart)
        self.canvas_income.get_tk_widget().pack(fill="both", expand=True)

        bottom_chart = tk.LabelFrame(
            charts_container,
            text="Diagram Pengeluaran (%)",
            bg="#f7e9ff",
            font=("Poppins", 16, "bold")
        )
        bottom_chart.pack(fill="both", expand=True, padx=5, pady=8)

        self.fig_expense, self.ax_expense = plt.subplots(figsize=(5.8, 4.5), dpi=100)
        self.canvas_expense = FigureCanvasTkAgg(self.fig_expense, master=bottom_chart)
        self.canvas_expense.get_tk_widget().pack(fill="both", expand=True)

        self.draw_empty_charts()
```

Bagian kode ini digunakan untuk menampilkan grafik visual transaksi dalam bentuk diagram pie. Kode diawali dengan pembuatan charts_container sebagai wadah utama grafik. Di dalamnya terdapat dua label frame, yaitu Diagram Pemasukan (%) di sisi atas dan Diagram Pengeluaran (%) di sisi bawah, yang masing-masing menampilkan persentase berdasarkan kategori transaksi. Grafik dibuat menggunakan matplotlib dengan ukuran yang diperbesar agar lebih jelas, lalu ditampilkan ke dalam aplikasi Tkinter melalui FigureCanvasTkAgg. Pada bagian akhir, fungsi draw_empty_charts() dipanggil untuk menampilkan grafik awal dengan keterangan “Belum ada data” ketika belum ada transaksi yang dimasukkan.

3.1.6 Export PDF



```
    def export_pdf(self):
        if not self.transactions:
            messagebox.showwarning("Peringatan", "Tidak ada data untuk diexport!")
            return

        file_path = filedialog.asksaveasfilename(
            defaultextension=".pdf",
            filetypes=[("PDF Files", "*.pdf")],
            title="Simpan Laporan PDF"
        )

        if not file_path:
            return

        c = pdf_canvas.Canvas(file_path, pagesize=letter)
        width, height = letter
        c.setFont("Helvetica-Bold", 16)
        c.drawString(50, height - 50, "Laporan Keuangan - Moonify")

        y = height - 100
        c.setFont("Helvetica-Bold", 11)
        c.drawString(50, y, "Tipe")
        c.drawString(150, y, "Deskripsi")
        c.drawString(310, y, "Jumlah")
        c.drawString(400, y, "Kategori")

        c.setFont("Helvetica", 11)
        y -= 20
```

```
        for t in self.transactions:
            if y < 60:
                c.showPage()
                c.setFont("Helvetica", 11)
                y = height - 60
            c.drawString(50, y, t["type"])
            c.drawString(150, y, t["desc"][:25])
            c.drawString(310, y, f"Rp {t['amount']:,.0f}")
            c.drawString(400, y, t["cat"])
            y -= 18

        income = sum(t["amount"] for t in self.transactions if t["type"] == "Pemasukan")
        expense = sum(t["amount"] for t in self.transactions if t["type"] == "Pengeluaran")
        balance = income - expense

        y -= 20
        c.setFont("Helvetica-Bold", 11)
        c.drawString(50, y, f"Total Pemasukan  : Rp {income:,.0f}")
        y -= 15
        c.drawString(50, y, f"Total Pengeluaran: Rp {expense:,.0f}")
        y -= 15
        c.drawString(50, y, f"Sisa Tabungan    : Rp {balance:,.0f}")

        c.save()
        messagebox.showinfo("Berhasil", "File PDF berhasil disimpan!")
```

Fungsi export_pdf ini berfungsi untuk menyimpan laporan keuangan ke dalam file PDF. Pertama, program mengecek apakah terdapat data transaksi; jika tidak ada, akan muncul peringatan dan proses dihentikan. Selanjutnya, pengguna diminta memilih lokasi penyimpanan file melalui dialog penyimpanan. Setelah itu, program membuat dokumen PDF menggunakan library ReportLab, menuliskan judul laporan, lalu menampilkan header tabel berisi tipe, deskripsi, jumlah, dan kategori transaksi. Seluruh data transaksi kemudian dituliskan satu per satu ke dalam PDF, dengan pengaturan agar otomatis pindah halaman jika ruang halaman sudah penuh. Di bagian akhir, program menghitung total pemasukan, total pengeluaran, serta sisa tabungan, lalu menampilkannya sebagai ringkasan. Setelah file selesai dibuat, PDF disimpan dan pengguna akan mendapat notifikasi bahwa proses ekspor berhasil.

BAB IV
LAMPIRAN

4.1 Kode Program

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from reportlab.pdfgen import canvas as pdf_canvas
from reportlab.lib.pagesizes import letter


class FinanceTracker:
    def __init__(self, master):
        self.master = master
        self.master.title("Finance Tracker — Monetify")
        self.master.configure(bg="#ffeef7")

        self.transactions = []

        # ------------------- HEADER -------------------
        title = tk.Label(
            master,
            text="💰Monetify — Finance Tracker",
            font=("Poppins", 26, "bold"),
            bg="#ffeef7",
            fg="#6d5dfc"
        )
        title.pack(pady=5)

        subtitle = tk.Label(
            master,
            text="by Nayla Rizky Amalia & Lintang Alysia Gantari",
            font=("Arial", 16),
            bg="#ffeef7",
            fg="#6d5dfc"
        )
        subtitle.pack(pady=3)

        # ------------------- MAIN CONTAINER -------------------
        main_layout = tk.Frame(master, bg="#ffeef7")
        main_layout.pack(fill="both", expand=True)

        left_side = tk.Frame(main_layout, bg="#ffeef7")
        left_side.pack(side="left", fill="both", expand=True)
```

```
        right_side = tk.Frame(main_layout, bg="#ffeef7")
        right_side.pack(side="right", fill="y")

        # ------------------- INPUT FRAME -------------------
        input_frame = tk.LabelFrame(
            left_side,
            text="Tambah Transaksi",
            padx=10,
            pady=10,
            bg="#f7e9ff",
            font=("Poppins", 16, "bold")
        )
        input_frame.pack(padx=10, pady=5, fill="x")

        self.type_var = tk.StringVar(value="Pemasukan")
        ttk.Combobox(
            input_frame,
            textvariable=self.type_var,
            values=["Pemasukan", "Pengeluaran"],
            width=15, state="readonly"
        ).grid(row=0, column=0, padx=5, pady=5)

        # ---- Deskripsi Entry ----
        self.desc_entry = tk.Entry(input_frame, width=25, font=("Poppins", 12),
fg="grey")
        self.desc_entry.grid(row=0, column=1, padx=5, pady=5)
        self.desc_entry.insert(0, "Deskripsi")
        self.desc_entry.bind("<FocusIn>", self.clear_placeholder_desc)
        self.desc_entry.bind("<FocusOut>", self.add_placeholder_desc)

        # ---- Jumlah Entry ----
        self.amount_entry = tk.Entry(input_frame, width=15, font=("Poppins", 12),
fg="grey")
        self.amount_entry.grid(row=0, column=2, padx=5, pady=5)
        self.amount_entry.insert(0, "Jumlah (Rp)")
        self.amount_entry.bind("<FocusIn>", self.clear_placeholder_amount)
        self.amount_entry.bind("<FocusOut>", self.add_placeholder_amount)

        self.category_var = tk.StringVar(value="Lainnya")
        ttk.Combobox(
            input_frame,
            textvariable=self.category_var,
            values=["Jajan", "Transport", "Tagihan", "Hiburan", "Gaji", "Uang Saku",
"Hadiah", "Lainnya"],
            width=15, state="readonly"
        ).grid(row=0, column=3, padx=5, pady=5)
```

```
tk.Button(
    input_frame,
    text="Simpan",
    command=self.add_transaction,
    bg="#fcbad3",
    fg="black",
    width=12,
    font=("Poppins", 12)
).grid(row=0, column=4, padx=5, pady=5)

tk.Button(
    input_frame,
    text="Bersihkan",
    command=self.clear_all,
    bg="#bde0fe",
    fg="black",
    width=12,
    font=("Poppins", 12)
).grid(row=0, column=5, padx=5, pady=5)

# ------------------- BALANCE FRAME -------------------
balance_frame = tk.Frame(left_side, bg="#ffbed3")
balance_frame.pack(pady=10)

self.income_label = tk.Label(
    balance_frame,
    text="Pemasukan: Rp 0",
    font=("Poppins", 15),
    bg="#ffbed3"
)
self.income_label.grid(row=0, column=0, padx=25)

self.expense_label = tk.Label(
    balance_frame,
    text="Pengeluaran: Rp 0",
    font=("Poppins", 15),
    bg="#ffbed3"
)
self.expense_label.grid(row=0, column=1, padx=25)

self.balance_label = tk.Label(
    balance_frame,
    text="Sisa Tabungan: Rp 0",
    font=("Poppins", 15, "bold"),
    bg="#ffbed3",
    fg="#444"
```

```
)
self.balance_label.grid(row=0, column=2, padx=25)

# ------------------- LIST FRAME -------------------
list_frame = tk.LabelFrame(
    left_side,
    text="Daftar Transaksi",
    bg="#ffbed3",
    font=("Poppins", 16, "bold")
)
list_frame.pack(padx=10, pady=5, fill="both", expand=True)

self.tree = ttk.Treeview(
    list_frame,
    columns=("Tipe", "Deskripsi", "Jumlah", "Kategori"),
    show="headings"
)
style = ttk.Style()
style.configure("Treeview.Heading", font=("Poppins", 12, "bold"))
style.configure("Treeview", font=("Poppins", 14), rowheight=35)

for col in ("Tipe", "Deskripsi", "Jumlah", "Kategori"):
    self.tree.heading(col, text=col)

self.tree.column("Tipe", anchor="center", width=130)
self.tree.column("Deskripsi", anchor="center", width=300)
self.tree.column("Jumlah", anchor="center", width=200)
self.tree.column("Kategori", anchor="center", width=170)
self.tree.pack(fill="both", expand=True, padx=5, pady=5)

# ------------------- BUTTONS -------------------
export_frame = tk.Frame(left_side, bg="#ffeef7")
export_frame.pack(pady=5)

tk.Button(
    export_frame,
    text="Hapus Transaksi Terpilih",
    command=self.delete_selected,
    bg="#ffb4c6",
    fg="black",
    width=20,
    font=("Poppins", 12, "bold")
).pack(side="left", padx=6)

tk.Button(
    export_frame,
    text="Export ke PDF",
```

```
            command=self.export_pdf,
            bg="#fcbad3",
            fg="black",
            width=12,
            font=("Poppins", 12, "bold")
        ).pack(side="left", padx=6)

        # ------------------- CHART FRAME -------------------
        charts_container = tk.Frame(right_side, bg="#ffeef7")
        charts_container.pack(fill="both", expand=True, padx=10)

        top_chart = tk.LabelFrame(
            charts_container,
            text="Diagram Pemasukan (%)",
            bg="#f7e9ff",
            font=("Poppins", 16, "bold")
        )
        top_chart.pack(fill="both", expand=True, padx=5, pady=8)

        self.fig_income, self.ax_income = plt.subplots(figsize=(5.8, 4.5), dpi=100)
        self.canvas_income = FigureCanvasTkAgg(self.fig_income, master=top_chart)
        self.canvas_income.get_tk_widget().pack(fill="both", expand=True)

        bottom_chart = tk.LabelFrame(
            charts_container,
            text="Diagram Pengeluaran (%)",
            bg="#f7e9ff",
            font=("Poppins", 16, "bold")
        )
        bottom_chart.pack(fill="both", expand=True, padx=5, pady=8)

        self.fig_expense, self.ax_expense = plt.subplots(figsize=(5.8, 4.5), dpi=100)
        self.canvas_expense = FigureCanvasTkAgg(self.fig_expense,
master=bottom_chart)
        self.canvas_expense.get_tk_widget().pack(fill="both", expand=True)

        self.draw_empty_charts()

    # -------- PLACEHOLDER --------
    def clear_placeholder_desc(self, event):
        if self.desc_entry.get() == "Deskripsi":
            self.desc_entry.delete(0, "end")
            self.desc_entry.config(fg="black")

    def clear_placeholder_amount(self, event):
        if self.amount_entry.get() == "Jumlah (Rp)":
            self.amount_entry.delete(0, "end")
```

```
            self.amount_entry.config(fg="black")

    def add_placeholder_desc(self, event):
        if self.desc_entry.get().strip() == "":
            self.desc_entry.insert(0, "Deskripsi")
            self.desc_entry.config(fg="grey")

    def add_placeholder_amount(self, event):
        if self.amount_entry.get().strip() == "":
            self.amount_entry.insert(0, "Jumlah (Rp)")
            self.amount_entry.config(fg="grey")

    def draw_empty_charts(self):
        self.ax_income.clear()
        self.ax_expense.clear()
        self.ax_income.pie([1], labels=["Belum ada data"])
        self.ax_expense.pie([1], labels=["Belum ada data"])
        self.canvas_income.draw()
        self.canvas_expense.draw()

    def add_transaction(self):
        tipe = self.type_var.get()
        desc = self.desc_entry.get().strip()
        kategori = self.category_var.get().strip()

        try:
            amount = float(self.amount_entry.get())
        except ValueError:
            messagebox.showerror("Error", "Jumlah harus berupa angka!")
            return

        if desc == "Deskripsi" or not desc or amount <= 0:
            messagebox.showwarning("Peringatan", "Isi deskripsi dan jumlah dengan
benar!")
            return

        self.transactions.append({"type": tipe, "desc": desc, "amount": amount, "cat":
kategori})
        self.tree.insert("", "end", values=(tipe, desc, f"Rp {amount:,.0f}", kategori))

        self.desc_entry.delete(0, "end")
        self.desc_entry.insert(0, "Deskripsi")
        self.desc_entry.config(fg="grey")

        self.amount_entry.delete(0, "end")
        self.amount_entry.insert(0, "Jumlah (Rp)")
        self.amount_entry.config(fg="grey")
```

```
        self.update_summary()
        self.plot_charts()

    def update_summary(self):
        income = sum(t["amount"] for t in self.transactions if t["type"] == "Pemasukan")
        expense = sum(t["amount"] for t in self.transactions if t["type"] == "Pengeluaran")
        balance = income - expense
        self.income_label.config(text=f"Pemasukan: Rp {income:,.0f}")
        self.expense_label.config(text=f"Pengeluaran: Rp {expense:,.0f}")
        self.balance_label.config(text=f"Sisa Tabungan: Rp {balance:,.0f}")

    def plot_charts(self):
        self.ax_income.clear()
        pemasukan = {}
        for t in self.transactions:
            if t["type"] == "Pemasukan":
                pemasukan[t["cat"]] = pemasukan.get(t["cat"], 0) + t["amount"]

        if pemasukan:
            self.ax_income.pie(pemasukan.values(), labels=pemasukan.keys(),
autopct="%1.1f%%")
        else:
            self.ax_income.pie([1], labels=["Belum ada pemasukan"])
        self.canvas_income.draw()

        self.ax_expense.clear()
        pengeluaran = {}
        for t in self.transactions:
            if t["type"] == "Pengeluaran":
                pengeluaran[t["cat"]] = pengeluaran.get(t["cat"], 0) + t["amount"]

        if pengeluaran:
            self.ax_expense.pie(pengeluaran.values(), labels=pengeluaran.keys(),
autopct="%1.1f%%")
        else:
            self.ax_expense.pie([1], labels=["Belum ada pengeluaran"])
        self.canvas_expense.draw()

    def clear_all(self):
        if messagebox.askyesno("Konfirmasi", "Hapus semua data transaksi?"):
            self.transactions.clear()
            for i in self.tree.get_children():
                self.tree.delete(i)
            self.update_summary()
            self.draw_empty_charts()
```

```
    def delete_selected(self):
        selected_item = self.tree.selection()
        if not selected_item:
            messagebox.showwarning("Peringatan", "Pilih transaksi yang mau dihapus!")
            return

        if not messagebox.askyesno("Konfirmasi", "Hapus transaksi yang dipilih?"):
            return

        item = self.tree.item(selected_item)["values"]
        tipe, desc, jumlahStr, kategori = item
        jumlah = float(jumlahStr.replace("Rp ", "").replace(",", ""))

        for i, t in enumerate(self.transactions):
            if t["type"] == tipe and t["desc"] == desc and t["amount"] == jumlah and t["cat"]
== kategori:
                del self.transactions[i]
                break

        self.tree.delete(selected_item)
        self.update_summary()
        self.plot_charts()

    def export_pdf(self):
        if not self.transactions:
            messagebox.showwarning("Peringatan", "Tidak ada data untuk diexport!")
            return

        file_path = filedialog.asksaveasfilename(
            defaultextension=".pdf",
            filetypes=[("PDF Files", "*.pdf")],
            title="Simpan Laporan PDF"
        )

        if not file_path:
            return

        c = pdf_canvas.Canvas(file_path, pagesize=letter)
        width, height = letter
        c.setFont("Helvetica-Bold", 16)
        c.drawString(50, height - 50, "Laporan Keuangan - Monetify")

        y = height - 100
        c.setFont("Helvetica-Bold", 11)
        c.drawString(50, y, "Tipe")
        c.drawString(150, y, "Deskripsi")
        c.drawString(310, y, "Jumlah")
```

```
        c.drawString(400, y, "Kategori")

        c.setFont("Helvetica", 11)
        y -= 20

        for t in self.transactions:
            if y < 60:
                c.showPage()
                c.setFont("Helvetica", 11)
                y = height - 60
            c.drawString(50, y, t["type"])
            c.drawString(150, y, t["desc"][:25])
            c.drawString(310, y, f"Rp {t['amount']:,.0f}")
            c.drawString(400, y, t["cat"])
            y -= 18

        income = sum(t["amount"] for t in self.transactions if t["type"] == "Pemasukan")
        expense = sum(t["amount"] for t in self.transactions if t["type"] == "Pengeluaran")
        balance = income - expense

        y -= 20
        c.setFont("Helvetica-Bold", 11)
        c.drawString(50, y, f"Total Pemasukan  : Rp {income:,.0f}")
        y -= 15
        c.drawString(50, y, f"Total Pengeluaran: Rp {expense:,.0f}")
        y -= 15
        c.drawString(50, y, f"Sisa Tabungan    : Rp {balance:,.0f}")

        c.save()
        messagebox.showinfo("Berhasil", "File PDF berhasil disimpan!")


if __name__ == "__main__":
    root = tk.Tk()
    app = FinanceTracker(root)
    root.mainloop()
```

DAFTAR PUSTAKA