LAPORAN CATATAN PENGELUARAN HARIAN

PEMROGRAMAN BERORIENTASI OBJEK PRAKTIK
KELAS VIII

T. A. Semester Ganjil 2024/2025

5230411286 Firmanti Alhilma S.

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
YOGYAKARTA
2024

DAFTAR ISI

Pendahuluan

Aplikasi Catatan Pengeluaran Harian adalah aplikasi berbasis GUI (Graphical User Interface) yang dibangun menggunakan Tkinter di Python. Aplikasi ini bertujuan untuk membantu pengguna dalam mencatat, mengelola, dan menganalisis pengeluaran harian mereka. Fitur utama aplikasi ini meliputi pencatatan pengeluaran, pengelompokan berdasarkan kategori, penambahan dan penghapusan data, serta penyajian data dalam bentuk grafik.

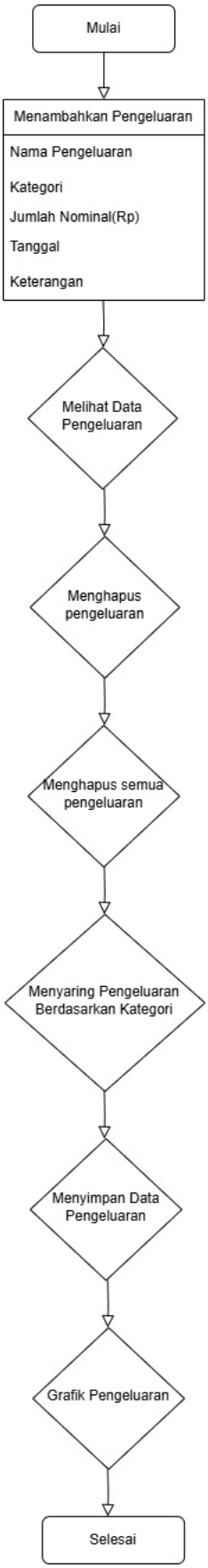
Aplikasi ini juga memungkinkan pengguna untuk menyimpan catatan pengeluaran dalam format CSV untuk keperluan pelaporan atau analisis lebih lanjut.

Struktur Program

Aplikasi ini memiliki beberapa komponen utama yang membentuk strukturnya, yaitu:

- Antarmuka Pengguna: Dibangun menggunakan modul Tkinter untuk membuat tampilan grafis.
- Pengelolaan Data: Menggunakan daftar Python untuk menyimpan data pengeluaran.
- Fungsi Tambahan: Menyediakan fungsionalitas untuk menghapus data, menyimpan ke file CSV, dan menampilkan grafik.

Aktifity diagram



Penjelasan

- Menambah Pengeluaran: Ketika tombol "Tambah" ditekan, data dari form input diproses dan ditambahkan ke daftar pengeluaran. Tabel diperbarui untuk menampilkan pengeluaran terbaru.
- Melihat Pengeluaran: Pengguna dapat melihat data apa saja yang telah diinputkan
- Menghapus Pengeluaran: Pengguna dapat memilih pengeluaran yang ingin dihapus dari tabel, dan setelah konfirmasi, pengeluaran tersebut dihapus dari daftar.
- Menyaring data: Pengguna dapat menyaring data pengeluaran berdasarkan kategori yang ditentukan.
- Menyimpan Data: Ketika tombol "Simpan" ditekan, aplikasi meminta nama file dan menyimpan data pengeluaran ke dalam file CSV.
- Menampilkan Grafik: Saat tombol "Grafik" ditekan, aplikasi memproses data pengeluaran dan menampilkan grafik batang dan pie untuk membantu pengguna menganalisis pengeluaran mereka.

Detail Code Program

1. **Import dan library**

M8_5230411286_FirmantiAlhilmaSalsabila.py > DailyExpenseApp >

```
1   import tkinter as tk
2   from tkinter import ttk, messagebox
3   from tkinter.simpledialog import askstring
4   import csv
5   from datetime import datetime
6   import matplotlib.pyplot as plt
7   import pandas as pd
8
```

- tkinter: Modul utama untuk antarmuka grafis pengguna (GUI).
- ttk: Submodul dari tkinter untuk widget yang lebih modern (seperti Combobox, Treeview).
- messagebox: Untuk menampilkan pesan kesalahan atau konfirmasi.
- askstring: Untuk meminta input berupa teks dari pengguna.
- csv: Untuk membaca dan menulis data dalam format CSV.
- datetime: Untuk memanipulasi tanggal dan waktu.
- matplotlib.pyplot: Untuk menggambar grafik pengeluaran.
  pandas: Untuk mengelola dan menganalisis data pengeluaran.

2. **Class DailyExpenseApp**
   Kelas ini adalah inti dari aplikasi dan mengontrol semua fungsionalitas aplikasi.

```
class DailyExpenseApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Catatan Pengeluaran Harian")
        self.root.geometry("900x600")
        self.root.configure(bg="#1a237e")

        # Judul Aplikasi
        title_label = tk.Label(
            self.root,
            text="Catatan Pengeluaran Harian",
            font=("Georgia", 24, "bold"),
            fg="#ffc107",
            bg="#1a237e",
        )
        title_label.pack(pady=10)

        # Frame Utama
        main_frame = tk.Frame(self.root, bg="#1a237e")
        main_frame.pack(padx=10, pady=10, fill="both", expand=True)

        # Frame Input
        input_frame = tk.LabelFrame(main_frame, text="Input Pengeluaran", bg="#3949ab", fg="white", font=("Arial", 12, "bold"))
        input_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

        self.name_entry = self.create_entry(input_frame, "Nama Pengeluaran:", 0)
        self.category_combobox = self.create_combobox(input_frame, "Kategori:", 1)
        self.amount_entry = self.create_entry(input_frame, "Jumlah (Rp):", 2)
        self.date_entry = self.create_entry(input_frame, "Tanggal (YYYY-MM-DD):", 3, datetime.now().strftime("%Y-%m-%d"))
        self.note_entry = self.create_entry(input_frame, "Keterangan:", 4)

        add_button = tk.Button(
            input_frame, text="Tambah", font=("Arial", 12, "bold"), bg="#4caf50", fg="white",
            command=self.add_expense
        )
        add_button.grid(row=5, column=0, columnspan=2, pady=10, padx=10)

```

Metode ***__init__*** digunakan untuk inisialisasi aplikasi dengan membuat dan mengonfigurasi elemen GUI dasar seperti jendela utama (root), ukuran jendela, dan elemen-elemen seperti label, tombol, dan tabel.

```
def create_entry(self, parent, label_text, row, default_value=""):
    tk.Label(parent, text=label_text, font=("Arial", 12), bg="#3949ab", fg="white").grid(row=row, column=0, padx=10, pady=5, sticky="w")
    entry = tk.Entry(parent, font=("Arial", 12), width=25)
    entry.grid(row=row, column=1, padx=10, pady=5)
    entry.insert(0, default_value)
    return entry
```

create_entry: Membuat dan mengonfigurasi widget Entry untuk menerima input teks (seperti nama pengeluaran, jumlah, tanggal, dan keterangan).

```
def create_combobox(self, parent, label_text, row):
    tk.Label(parent, text=label_text, font=("Arial", 12), bg="#3949ab", fg="white").grid(row=row, column=0, padx=10, pady=5, sticky="w"
    combobox = ttk.Combobox(
        parent,
        values=["Makanan", "Transportasi", "Hiburan", "Lainnya"],
        font=("Arial", 12),
        state="readonly",
    )
    combobox.grid(row=row, column=1, padx=10, pady=5)
    combobox.set("Pilih Kategori")
    return combobox
```

create_combobox: Membuat Combobox untuk memilih kategori pengeluaran, seperti "Makanan", "Transportasi", "Hiburan", dll.

```python
def create_table(self, parent):
    treeview = ttk.Treeview(
        parent,
        columns=("Nama", "Kategori", "Jumlah", "Tanggal", "Keterangan"),
        show="headings",
        height=12,
    )
    treeview.heading("Nama", text="Nama Pengeluaran")
    treeview.heading("Kategori", text="Kategori")
    treeview.heading("Jumlah", text="Jumlah (Rp)")
    treeview.heading("Tanggal", text="Tanggal")
    treeview.heading("Keterangan", text="Keterangan")
    treeview.column("Nama", width=150)
    treeview.column("Kategori", width=100)
    treeview.column("Jumlah", width=100)
    treeview.column("Tanggal", width=100)
    treeview.column("Keterangan", width=150)
    treeview.pack(fill="both", expand=True, padx=10, pady=10)
    return treeview
```

create_table: Membuat tabel Treeview untuk menampilkan data pengeluaran. Tabel ini menampilkan kolom seperti nama, kategori, jumlah, tanggal, dan keterangan pengeluaran.

```python
def create_action_buttons(self, parent):
    tk.Button(parent, text="Filter", font=("Arial", 12, "bold"), bg="#4caf50", fg="white", command=self.filter_data).pack(side="left", padx=10, pady=5)
    tk.Button(parent, text="Hapus", font=("Arial", 12, "bold"), bg="#e53935", fg="white", command=self.delete_expense).pack(side="left", padx=10, pady=5)
    tk.Button(parent, text="Hapus Semua", font=("Arial", 12, "bold"), bg="#d32f2f", fg="white", command=self.clear_all_data).pack(side="left", padx=10, pady=5)
    tk.Button(parent, text="Simpan", font=("Arial", 12, "bold"), bg="#1976d2", fg="white", command=self.save_expenses).pack(side="left", padx=10, pady=5)
    tk.Button(parent, text="Grafik", font=("Arial", 12, "bold"), bg="#ff8f00", fg="white", command=self.show_graph).pack(side="left", padx=10, pady=5)
```

create_action_buttons digunakan untuk membuat tombol-tombol di antarmuka grafis (GUI) menggunakan Tkinter. Setiap tombol memiliki fungsi tertentu, seperti:

1. **Filter**: Memanggil metode filter_data.
2. **Hapus**: Memanggil metode delete_expense.
3. **Hapus Semua**: Memanggil metode clear_all_data.
4. **Simpan**: Memanggil metode save_expenses.
5. **Grafik**: Memanggil metode show_graph.

Tombol-tombol ini ditempatkan secara horizontal dengan padding di sekitarnya untuk memberi jarak antar tombol.

```python
def add_expense(self):
    name = self.name_entry.get()
    category = self.category_combobox.get()
    amount = self.amount_entry.get()
    date = self.date_entry.get()
    note = self.note_entry.get()

    if not name or category == "Pilih Kategori" or not amount or not date:
        messagebox.showerror("Error", "Semua kolom wajib diisi!")
        return

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

    try:
        datetime.strptime(date, "%Y-%m-%d")
    except ValueError:
        messagebox.showerror("Error", "Tanggal harus dalam format YYYY-MM-DD!")
        return

    self.expenses.append({"name": name, "category": category, "amount": amount, "date": date, "note": note})
    self.update_table()
    self.clear_entries()
```

add_expense: Fungsi untuk menambahkan pengeluaran baru setelah memvalidasi input dari pengguna.

```
def update_table(self):
    for row in self.expense_table.get_children():
        self.expense_table.delete(row)

    for expense in self.expenses:
        self.expense_table.insert(
            "", "end",
            values=(
                expense["name"],
                expense["category"],
                f"Rp {expense['amount']:,}",
                expense["date"],
                expense["note"],
            ),
        )

    total = sum(expense["amount"] for expense in self.expenses)
    self.total_label.config(text=f"Total Pengeluaran: Rp {total:,}")
```

update_table: Fungsi untuk memperbarui tabel dengan data pengeluaran terbaru dan menghitung total pengeluaran.

```
def clear_entries(self):
    self.name_entry.delete(0, tk.END)
    self.category_combobox.set("Pilih Kategori")
    self.amount_entry.delete(0, tk.END)
    self.date_entry.delete(0, tk.END)
    self.note_entry.delete(0, tk.END)
```

clear_entries digunakan untuk menghapus atau mereset nilai-nilai yang ada pada field input (seperti entry dan combobox) dalam antarmuka pengguna, sehingga kembali kosong atau ke nilai default.

```
def delete_expense(self):
    selected_item = self.expense_table.selection()
    if not selected_item:
        messagebox.showerror("Error", "Pilih data yang ingin dihapus!")
        return

    for item in selected_item:
        values = self.expense_table.item(item, "values")
        self.expenses = [expense for expense in self.expenses if not (
            expense["name"] == values[0] and
            expense["category"] == values[1] and
            expense["amount"] == int(values[2]) and
            expense["date"] == values[3] and
            expense["note"] == values[4]
        )]
        self.expense_table.delete(item)

    self.update_table()
```

delete_expense digunakan untuk menghapus data pengeluaran yang dipilih dari tabel. Jika tidak ada data yang dipilih, akan muncul pesan kesalahan. Setelah memilih data, fungsi ini akan menghapus data dari daftar pengeluaran dan tabel yang ada. Setelah itu, tabel diperbarui dengan update_table.

```python
def clear_all_data(self):
    if messagebox.askyesno("Konfirmasi", "Apakah Anda yakin ingin menghapus semua data?"):
        self.expenses.clear()
        self.update_table()
```

clear_all_data digunakan untuk menghapus semua data pengeluaran. Sebelum menghapus, pengguna akan diminta konfirmasi melalui dialog. Jika pengguna setuju, semua data dihapus dari daftar pengeluaran dan tabel diperbarui.

```python
def save_expenses(self):
    file_name = askstring("Simpan Data", "Masukkan nama file (tanpa ekstensi):")
    if not file_name:
        return

    file_name += ".csv"
    with open(file_name, mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["Nama Pengeluaran", "Kategori", "Jumlah (Rp)", "Tanggal", "Keterangan"])
        for expense in self.expenses:
            writer.writerow([expense["name"], expense["category"], expense["amount"], expense["date"], expense["note"]])

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

save_expenses untuk menyimpan data pengeluaran ke file CSV.

```python
def show_graph(self):
    if not self.expenses:
        messagebox.showerror("Error", "Tidak ada data untuk ditampilkan dalam grafik!")
        return

    df = pd.DataFrame(self.expenses)
    summary = df.groupby("category")["amount"].sum()

    # Diagram Batang
    plt.figure(figsize=(10, 6))
    summary.plot(kind="bar", color=["#4caf50", "#2196f3", "#ff5722", "#ff9800"])
    plt.title("Pengeluaran Berdasarkan Kategori")
    plt.xlabel("Kategori")
    plt.ylabel("Total Pengeluaran (Rp)")
    plt.xticks(rotation=45)
    plt.tight_layout()

    # Diagram Lingkaran
    plt.figure(figsize=(8, 8))
    summary.plot(kind="pie", autopct='%1.1f%%', colors=["#4caf50", "#2196f3", "#ff5722", "#ff9800"])
    plt.title("Proporsi Pengeluaran Berdasarkan Kategori")
    plt.ylabel("")
    plt.tight_layout()

    plt.show()
```

show_graph digunakan untuk menampilkan grafik pengeluaran berdasarkan kategori. Jika tidak ada data pengeluaran, akan muncul pesan kesalahan. Fungsi ini membuat dua jenis grafik menggunakan data pengeluaran:

1. Diagram Batang: Menampilkan total pengeluaran per kategori.
2. Diagram Lingkaran: Menampilkan proporsi pengeluaran berdasarkan kategori.

Kedua grafik ini dibuat dengan menggunakan pustaka matplotlib dan pandas, kemudian ditampilkan menggunakan plt.show().

```python
def filter_data(self):
    category = askstring("Filter Data", "Masukkan kategori yang ingin difilter:")
    if not category:
        return

    filtered_expenses = [expense for expense in self.expenses if expense["category"].lower() == category.lower()]
    if not filtered_expenses:
        messagebox.showinfo("Hasil Filter", f"Tidak ada data dengan kategori '{category}'.")
        return

    total = sum(expense["amount"] for expense in filtered_expenses)
    messagebox.showinfo("Hasil Filter", f"Total Pengeluaran untuk kategori '{category}': Rp {total:,}")
```

filter_data digunakan untuk memfilter data pengeluaran berdasarkan kategori yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan kategori, dan jika kategori tersebut ditemukan, total pengeluaran untuk kategori tersebut akan dihitung dan ditampilkan. Jika tidak ada data yang sesuai dengan kategori yang dimasukkan, akan muncul pesan bahwa tidak ada data yang ditemukan.

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

bagian ini menjalankan aplikasi ketika file Python dieksekusi secara langsung.

Kesimpulan

Aplikasi Catatan Pengeluaran Harian ini memberikan solusi yang sederhana namun fungsional untuk mengelola pengeluaran sehari-hari. Dengan antarmuka yang intuitif dan fitur-fitur seperti filter, penghapusan data, dan visualisasi grafik, aplikasi ini membantu pengguna untuk mencatat, menganalisis, dan mengelola pengeluaran mereka dengan lebih mudah dan efektif.