TUGAS 8

PEMROGRAMAN BERBASIS OBJEK PRAKTIK

D I S U S U N

OLEH

Daffa Nayottama

5230411234

Prodi : Informatika – E

UNIVERSITAS TEKNOLOGI YOGYAKARTA

FAKULTAS SAINS & TEKNOLOGI

Tahun Akademik 2023

Pengembangan Aplikasi Pencatatan Kehadiran Siswa

- **Studi Kasus:**

Kehadiran siswa merupakan salah satu indikator penting dalam evaluasi kegiatan belajar-mengajar di institusi pendidikan. Namun, proses pencatatan yang dilakukan secara manual sering kali memakan waktu dan berpotensi menimbulkan kesalahan administratif. Studi kasus ini berfokus pada pengembangan aplikasi berbasis desktop untuk mempermudah proses pencatatan, pengelolaan, dan ekspor data kehadiran siswa secara digital.

- **Latar Belakang**

Institusi pendidikan memerlukan pencatatan kehadiran yang efisien untuk mendukung kelancaran administrasi dan pelaporan. Sistem manual yang selama ini digunakan rentan terhadap kesalahan pencatatan, keterlambatan pengolahan data, dan sulitnya akses data untuk keperluan evaluasi. Oleh karena itu, diperlukan aplikasi berbasis teknologi yang dapat membantu pihak sekolah dalam melakukan pencatatan kehadiran siswa secara cepat, akurat, dan mudah diekspor ke format digital seperti Excel.

Aplikasi ini dikembangkan menggunakan bahasa pemrograman Python dengan pustaka Tkinter untuk antarmuka pengguna dan openpyxl untuk pengelolaan data dalam format Excel. Dengan aplikasi ini, pencatatan kehadiran siswa dapat dilakukan dengan lebih praktis, mengurangi risiko kesalahan, dan mempermudah akses terhadap data kehadiran.

- **Penjelasan Program**

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

1. thinker
    - tk: Modul utama untuk membuat GUI.
    - ttk: Submodul dari tkinter untuk widget dengan tampilan yang lebih modern (seperti Treeview dan Combobox).
    - messagebox: Submodul untuk menampilkan dialog pesan (misalnya: konfirmasi, error, atau informasi).
2. datetime:
    - Digunakan untuk menangani tanggal dan waktu, terutama dalam format saat ini.
3. openpyxl:
    - Library untuk bekerja dengan file Excel (format .xlsx).
    - Mendukung membaca, menulis, dan memodifikasi file Excel.

```
class AppKehadiran:
    Codeium: Refactor | Explain | Generate Docstring | ×
    def __init__(self, root):
        self.root = root
        self.root.title("Aplikasi Pencatatan Kehadiran Siswa")
        self.root.geometry("900x700")
        self.siswa = []  # Daftar sisw a
        self.create_widgets()
```

- __init__: Fungsi inisialisasi yang dipanggil saat objek AppKehadiran dibuat.
- root: Parameter utama Tkinter untuk membangun aplikasi GUI.
- title dan geometry: Menentukan judul dan ukuran jendela aplikasi.
- self.siswa: List untuk menyimpan data siswa (tidak digunakan eksplisit di kode).
- create_widgets: Fungsi untuk membuat semua komponen (widgets) GUI.

1. Penjelasan
   - Pertama-tama mari kita membuat class agar bisa mengelompokan data serta fungsing-fungsi yang berhubungan di dalamnya.
   - Lalu membuat fungsi def __init__ dengan parameter self dan root
   - Kita gunakan title untuk membuat tulisan yang untuk megatur candela yang di dalamnya berisi tulisan ("Aplikasi Pencatatan Kehadiran Siswa")
   - Kemudian geometri untuk mengatur jendela Ketika baru di run selebar mana yang bagus untuk di tampilkan 900 adalah lebar dan 700 adalah Panjang.
   - Kemudian buat untuk data siswa yang di tamping ke database.

```
def create_widgets(self):
    # Label Judul
    title_label = tk.Label(self.root, text="Pencatatan Kehadiran Siswa", font=("Arial", 18, 'bold'))
    title_label.pack(pady=10)
```

1. **Buat Label**:
   - Pertama- tama buatlah fungsi create_widgets
   - Membuat teks "Pencatatan Kehadiran Siswa" sebagai judul aplikasi menggunakan widget Label.
   - Format nya diatur (font: Arial, ukuran: 18, tebal/bold).
2. **Tempatkan Label**:
   - Meletakkan label di bagian atas jendela dengan metode pack.
   - Menambahkan margin vertikal sebesar 10 piksel untuk tampilan terlihat rapi.

```
# Pilih Tanggal
self.tanggal_label = tk.Label(self.root, text="Tanggal: ")
self.tanggal_label.pack()
self.tanggal_entry = tk.Entry(self.root, width=25)
self.tanggal_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
self.tanggal_entry.pack(pady=5)
```

1. **Label Tanggal**:
   - self.tanggal_label = tk.Label(self.root, text="Tanggal: "): Membuat label dengan teks "Tanggal**"** yang ditempatkan di jendela utama.
   - self.tanggal_label.pack(): Menempatkan label di jendela yang ada.

2. **Input Tanggal**:
   - self.tanggal_entry = tk.Entry(self.root, width=25): Membuat field input (entry) untuk memasukkan tanggal dengan lebar 25 karakter.
   - self.tanggal_entry.insert(0, datetime.now().strftime("%Y-%m-%d")): Mengisi field input dengan tanggal saat ini, dalam format YYYY-MM-DD untuk masuk ke excelnya nanti Ketika di kirim.
   - self.tanggal_entry.pack(pady=5): Menempatkan field input di jendela dengan margin vertikal 5 piksel.

```
# Pilih Kelas
self.kelas_label = tk.Label(self.root, text="Kelas: ")
self.kelas_label.pack()
self.kelas_entry = tk.Entry(self.root, width=25)
self.kelas_entry.pack(pady=5)
```

- self.kelas_label = tk.Label(self.root, text="Kelas: "): Membuat label (teks) yang menampilkan tulisan **"Kelas: "** pada jendela GUInya.
- self.kelas_label.pack(): Menempatkan label **"Kelas: "** di dalam jendela aplikasi menggunakan metode **pack()**, yang secara otomatis mengatur posisi elemen sesuai urutan deklarasinya agar sesuai.
- **self.kelas_entry = tk.Entry(self.root, width=25)**: Membuat kolom input (entry field) untuk memungkinkan pengguna mengetikkan teks seperti nama kelas, dengan lebar kolom hingga 25 karakter ini di bisa di sesuaikan jika kamu mau dengan merubah width.
- self.kelas_entry.pack(pady=5): Menempatkan kolom input di dalam jendela aplikasi dengan margin vertikal sebesar 5 piksel untuk memberikan jarak antar elemen di GUI.

```
# Nama Siswa dan Status Kehadiran
self.nama_label = tk.Label(self.root, text="Nama Siswa: ")
self.nama_label.pack()
self.nama_entry = tk.Entry(self.root, width=25)
self.nama_entry.pack(pady=5)
```

- self.nama_label = tk.Label(self.root, text="Nama Siswa: "): Membuat label (teks) yang menampilkan tulisan **"Nama Siswa: "** pada jendela GUI.
- self.nama_label.pack(): Menempatkan label **"Nama Siswa: "** di dalam jendela aplikasi menggunakan metode **pack()**, yang secara otomatis mengatur posisi elemen sesuai urutan deklarasinya.

- self.nama_entry = tk.Entry(self.root, width=25): Membuat kolom input (entry field) untuk memungkinkan pengguna mengetikkan teks seperti nama siswa, dengan lebar kolom hingga 25 karakter.
- self.nama_entry.pack(pady=5): Menempatkan kolom input di dalam jendela aplikasi dengan margin vertikal sebesar 5 piksel untuk memberikan jarak antar elemen di GUI.

```
self.status_label = tk.Label(self.root, text="Status Kehadiran: ")
self.status_label.pack()
self.status_var = tk.StringVar(value="Hadir")
self.status_menu = ttk.Combobox(self.root, textvariable=self.status_var, values=("Hadir", "Tidak Hadir", "Izin", "Sakit"), state="readonly", width=20)
self.status_menu.pack(pady=5)
```

- self.status_label = tk.Label(self.root, text="Status Kehadiran: "): Membuat label (teks statis) yang menampilkan tulisan **"Status Kehadiran: "** pada jendela GUI.
- self.status_label.pack(): Menempatkan label **"Status Kehadiran: "** di dalam jendela aplikasi menggunakan metode **pack()**, yang secara otomatis mengatur posisi elemen sesuai urutan deklarasinya.
- self.status_var = tk.StringVar(value="Hadir"): Membuat variabel string yang digunakan untuk menyimpan nilai yang dipilih di combobox. Nilai awalnya diatur menjadi **"Hadir"**.
- self.status_menu = ttk.Combobox(self.root, textvariable=self.status_var, values=("Hadir", "Tidak Hadir", "Izin", "Sakit"), state="readonly", width=20): Membuat combobox (daftar dropdown) yang memungkinkan pengguna memilih status kehadiran siswa, dengan opsi yang tersedia: **"Hadir"**, **"Tidak Hadir"**, **"Izin"**, dan **"Sakit"**.
- textvariable=self.status_var: Menghubungkan pilihan pengguna dengan variabel status_var.
- state="readonly": Membuat combobox hanya dapat memilih opsi dari daftar (tidak dapat mengetik langsung).
- width=20: Menentukan lebar combobox sebesar 20 karakter.
- self.status_menu.pack(pady=5): Menempatkan combobox di dalam jendela aplikasi dengan margin vertikal sebesar 5 piksel untuk memberikan jarak antar elemen di GUI.

```
# Button untuk Menambahkan Kehadiran
self.add_button = tk.Button(self.root, text="Tambah Kehadiran", command=self.add_kehadiran)
self.add_button.pack(pady=10)
```

- self.add_button = tk.Button(self.root, text="Tambah Kehadiran", command=self.add_kehadiran): Membuat tombol dengan label **"Tambah Kehadiran"** pada jendela GUI. Ketika tombol ini diklik, fungsi **self.add_kehadiran** akan dipanggil untuk menambahkan data kehadiran siswa ke dalam daftar.
- text="Tambah Kehadiran": Menentukan teks yang ditampilkan pada tombol.
- command=self.add_kehadiran: Menghubungkan tombol dengan fungsi add_kehadiran, yang akan dijalankan saat tombol diklik.
- **self.add_button.pack(pady=10)**: Menempatkan tombol **"Tambah Kehadiran"** di dalam jendela aplikasi menggunakan metode **pack()**, dengan margin vertikal sebesar **10 piksel** untuk memberikan jarak antar elemen di GUI.

```
# Daftar Kehadiran
self.list_label = tk.Label(self.root, text="Daftar Kehadiran: ")
self.list_label.pack(pady=10)
```

- self.list_label = tk.Label(self.root, text="Daftar Kehadiran: "): Membuat label (teks statis) yang menampilkan tulisan **"Daftar Kehadiran: "** pada jendela GUI. Label ini digunakan sebagai penanda untuk menampilkan daftar data kehadiran siswa di elemen berikutnya (misalnya, sebuah tabel atau daftar).
- text="Daftar Kehadiran: ": Menentukan teks yang ditampilkan pada label.
- self.list_label.pack(pady=10): Menempatkan label **"Daftar Kehadiran: "** di dalam jendela aplikasi menggunakan metode **pack()**, dengan margin vertikal sebesar **10 piksel** untuk memberikan jarak antar elemen di GUI.

```
self.treeview = ttk.Treeview(self.root, columns=("Tanggal", "Kelas", "Nama", "Status"), show="headings")
self.treeview.heading("Tanggal", text="Tanggal")
self.treeview.heading("Kelas", text="Kelas")
self.treeview.heading("Nama", text="Nama")
self.treeview.heading("Status", text="Status")
self.treeview.pack(pady=10)
```

- self.treeview = ttk.Treeview(self.root, columns=("Tanggal", "Kelas", "Nama", "Status"), show="headings"): Membuat sebuah tabel dengan menggunakan widget **Treeview** dari modul **ttk** untuk menampilkan data kehadiran siswa.
- self.root: Menentukan bahwa tabel ini akan dimasukkan ke dalam jendela utama aplikasi.
- columns=("Tanggal", "Kelas", "Nama", "Status"): Mendefinisikan empat kolom tabel, yaitu **"Tanggal"**, **"Kelas"**, **"Nama"**, dan **"Status"**.
- show="headings": Menampilkan hanya header kolom, tanpa kolom indeks default.
- self.treeview.heading("Tanggal", text="Tanggal"): Menentukan teks header untuk kolom **"Tanggal"**, yaitu **"Tanggal"**.
- self.treeview.heading("Kelas", text="Kelas"): Menentukan teks header untuk kolom **"Kelas"**, yaitu **"Kelas"**.
- self.treeview.heading("Nama", text="Nama"): Menentukan teks header untuk kolom **"Nama"**, yaitu **"Nama"**.
- self.treeview.heading("Status", text="Status"): Menentukan teks header untuk kolom **"Status"**, yaitu **"Status"**.
- self.treeview.pack(pady=10): Menempatkan tabel ke dalam jendela aplikasi dengan margin vertikal sebesar **10 piksel** untuk memberikan jarak antar elemen di GUI.

```
self.clear_button = tk.Button(self.root, text="Clear Data", command=self.clear_data)
self.clear_button.pack(pady=5)
```

- self.clear_button = tk.Button(self.root, text="Clear Data", command=self.clear_data): Membuat tombol dengan label **"Clear Data"** pada jendela GUI. Tombol ini digunakan untuk menghapus seluruh data yang ada di tabel daftar kehadiran.
- text="Clear Data": Menentukan teks yang ditampilkan pada tombol, yaitu **"Clear Data"**.
- command=self.clear_data: Menghubungkan tombol dengan fungsi **clear_data**, yang akan dijalankan saat tombol ini diklik untuk menghapus semua data dari tabel.

- self.clear_button.pack(pady=5): Menempatkan tombol **"Clear Data"** di dalam jendela aplikasi menggunakan metode **pack()**, dengan margin vertikal sebesar **5 piksel** untuk memberikan jarak antar elemen di GUI.

```
self.export_button = tk.Button(self.root, text="Export to Excel", command=self.export_to_excel)
self.export_button.pack(pady=5)
```

- self.export_button = tk.Button(self.root, text="Export to Excel", command=self.export_to_excel): Membuat tombol dengan label **"Export to Excel"** pada jendela GUI. Tombol ini digunakan untuk menyimpan data yang ada di tabel kehadiran ke dalam file Excel.
- text="Export to Excel": Menentukan teks yang ditampilkan pada tombol, yaitu **"Export to Excel"**.
- command=self.export_to_excel: Menghubungkan tombol dengan fungsi **export_to_excel**, yang akan dipanggil saat tombol ini diklik untuk mengekspor data ke file Excel.
- self.export_button.pack(pady=5): Menempatkan tombol **"Export to Excel"** di dalam jendela aplikasi menggunakan metode **pack()**, dengan margin vertikal sebesar **5 piksel** untuk memberikan jarak antar elemen di GUI.

```
def add_kehadiran(self):
    # Ambil data dari form
    tanggal = self.tanggal_entry.get()
    kelas = self.kelas_entry.get()
    nama = self.nama_entry.get()
    status = self.status_var.get()
```

- def add_kehadiran(self): Mendefinisikan fungsi **add_kehadiran**, yang digunakan untuk menambahkan data kehadiran siswa dari form input ke tabel di aplikasi. Fungsi ini dipanggil ketika tombol **"Tambah Kehadiran"** diklik.
- tanggal = self.tanggal_entry.get(): Mengambil teks yang dimasukkan pengguna pada kolom input tanggal (**self.tanggal_entry**) untuk digunakan sebagai data **tanggal kehadiran**.
- kelas = self.kelas_entry.get(): Mengambil teks yang dimasukkan pengguna pada kolom input kelas (**self.kelas_entry**) untuk digunakan sebagai data **kelas siswa**.

- nama = self.nama_entry.get(): Mengambil teks yang dimasukkan pengguna pada kolom input nama siswa (**self.nama_entry**) untuk digunakan sebagai data **nama siswa**.
- status = self.status_var.get(): Mengambil nilai yang dipilih dari combobox **status kehadiran** (**self.status_var**) untuk digunakan sebagai data **status kehadiran siswa**.

```
if not nama or not kelas:
    messagebox.showerror("Error", "Nama dan Kelas harus diisi!")
    return
```

- if not nama or not kelas: Mengecek apakah input **nama** atau **kelas** kosong (tidak diisi oleh pengguna).
  - **not nama**: Mengembalikan **True** jika kolom input **nama** kosong.
  - **not kelas**: Mengembalikan **True** jika kolom input **kelas** kosong.
  - Jika salah satu atau kedua kondisi ini terpenuhi (ada yang kosong), blok kode di dalam pernyataan **if** akan dijalankan.
- messagebox.showerror("Error", "Nama dan Kelas harus diisi!"): Menampilkan kotak pesan error kepada pengguna dengan judul **"Error"** dan teks **"Nama dan Kelas harus diisi!"**. Pesan ini berfungsi sebagai peringatan agar pengguna mengisi kedua kolom input sebelum melanjutkan.
- return: Menghentikan eksekusi fungsi **add_kehadiran** jika validasi gagal (input kosong). Ini mencegah data yang tidak lengkap ditambahkan ke tabel.

```
# Tambahkan data ke daftar
self.treeview.insert("", "end", values=(tanggal, kelas, nama, status))

# Kosongkan input setelah tambah
self.nama_entry.delete(0, 'end')
```

- **self.treeview.insert("", "end", values=(tanggal, kelas, nama, status))**:
  - Menambahkan data baru ke dalam tabel **Treeview** yang menampilkan daftar kehadiran siswa.

- **""**: Menentukan bahwa data akan ditambahkan ke tingkat utama tabel, bukan sebagai child dari item lain.
- **"end"**: Menambahkan data di baris terakhir tabel.
- **values=(tanggal, kelas, nama, status)**: Menentukan data yang dimasukkan ke dalam tabel, yaitu data tanggal, kelas, nama, dan status yang telah diambil dari input pengguna.

- **self.nama_entry.delete(0, 'end')**:
  - Membersihkan kolom input **Nama Siswa** (**self.nama_entry**) setelah data berhasil ditambahkan ke tabel.
  - **delete(0, 'end')**: Menghapus semua teks di dalam kolom input dari posisi awal (**0**) hingga akhir (**'end'**).

```
def clear_data(self):
    # Menghapus semua data dari treeview
    for item in self.treeview.get_children():
        self.treeview.delete(item)
```

- **def clear_data(self):**:
  Mendefinisikan fungsi **clear_data**, yang digunakan untuk menghapus semua data yang ada di tabel **Treeview** (daftar kehadiran). Fungsi ini dipanggil ketika tombol **"Clear Data"** diklik.
- **for item in self.treeview.get_children():**:
  - Mendapatkan semua item (baris) yang ada di tabel **Treeview** menggunakan metode **get_children()**.
  - Melakukan iterasi (perulangan) untuk setiap item yang ditemukan di tabel.
- **self.treeview.delete(item)**:
  - Menghapus masing-masing item (baris) dari tabel **Treeview** berdasarkan **ID item** yang diperoleh dari perulangan.

```
def export_to_excel(self):
    # Membuat workbook baru
    workbook = openpyxl.Workbook()
    sheet = workbook.active
    sheet.title = "Kehadiran Siswa"
```

- **def export_to_excel(self):**:
  Mendefinisikan fungsi **export_to_excel**, yang digunakan untuk mengekspor data dari tabel **Treeview** ke dalam file Excel. Fungsi ini dipanggil saat tombol **"Export to Excel"** diklik.
- **workbook = openpyxl.Workbook()**:
  - Membuat workbook Excel baru menggunakan modul **openpyxl**. Workbook ini merupakan file Excel kosong yang akan diisi dengan data dari tabel.
  - Sebelum menggunakan pastikan nyalakan venv terlebih dahulu dengan cara ketik di terminal “python -m venv venv” dan setelah itu aktifkan dengan mengetik “venv\Scripts\activate”
- **sheet = workbook.active**:
  - Mengambil lembar kerja (worksheet) aktif dari workbook yang baru dibuat. Lembar kerja ini adalah tempat data akan ditulis.
- **sheet.title = "Kehadiran Siswa"**:
  - Memberikan nama pada lembar kerja aktif, yaitu **"Kehadiran Siswa"**, untuk menunjukkan bahwa data yang disimpan berkaitan dengan kehadiran siswa.

```
# Header
headers = ["Tanggal", "Kelas", "Nama", "Status"]
sheet.append(headers)
```

- **headers = ["Tanggal", "Kelas", "Nama", "Status"]**:
  - Mendefinisikan sebuah **list** yang berisi nama-nama kolom untuk data yang akan diekspor ke dalam file Excel. Kolom-kolom ini mewakili informasi yang ingin disimpan dari tabel kehadiran, yaitu:
    - **Tanggal**: Tanggal kehadiran siswa.

    - **Kelas**: Kelas siswa.
    - **Nama**: Nama siswa.
    - **Status**: Status kehadiran siswa (Hadir, Tidak Hadir, Izin, Sakit).
- **sheet.append(headers)**:
    - Menambahkan **list** yang berisi nama kolom **headers** ke dalam lembar kerja (**sheet**) pada file Excel.
    - **append(headers)**: Metode ini menambahkan **headers** sebagai baris pertama pada lembar kerja, sehingga kolom-kolom tersebut akan menjadi judul kolom di Excel.

```
for item in self.treeview.get_children():
    row_data = self.treeview.item(item, "values")
    sheet.append(row_data)
```

 for item in self.treeview.get_children():

- Mengambil semua item (baris) yang ada dalam tabel **Treeview** dan melakukan iterasi (perulangan) untuk setiap item.
- **self.treeview.get_children()** mengembalikan semua item di dalam **Treeview**, yang berupa ID dari setiap baris dalam tabel.

 row_data = self.treeview.item(item, "values")

- Mengambil data dari setiap baris yang ditemukan selama iterasi.
- **self.treeview.item(item, "values")** digunakan untuk mendapatkan **nilai** atau **data** dari setiap baris. Nilai ini berupa tuple yang berisi data **Tanggal**, **Kelas**, **Nama**, dan **Status** yang ada di baris tersebut.

 sheet.append(row_data)

- Menambahkan data **row_data** yang telah diambil dari **Treeview** ke dalam lembar kerja (**sheet**) di file Excel.
- **append(row_data)** menambahkan **row_data** sebagai baris baru pada lembar kerja, sehingga setiap baris dari tabel **Treeview** akan disalin ke dalam Excel.

```
# Simpan file
filename = f"Kehadiran_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"
try:
    workbook.save(filename)
    messagebox.showinfo("Sukses", f"Data berhasil disimpan ke file: {filename}")
except Exception as e:
    messagebox.showerror("Error", f"Gagal menyimpan file: {str(e)}")
```

- **filename = f"Kehadiran_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"**:
    - Membuat nama file Excel secara dinamis menggunakan format waktu saat ini.
    - **datetime.now()**: Mengambil waktu saat ini.
    - **strftime('%Y%m%d_%H%M%S')**: Memformat waktu tersebut menjadi string dengan format **tahun-bulan-hari_jam-menit-detik** (misalnya, 20241128_143012), memastikan nama file unik setiap kali diekspor.
    - **f"Kehadiran_{...}.xlsx"**: Nama file yang dihasilkan akan berbentuk **"Kehadiran_YYYYMMDD_HHMMSS.xlsx"**, di mana **YYYYMMDD_HHMMSS** adalah timestamp yang unik.
- **try:**
    - Blok kode ini digunakan untuk menangani potensi **error** (kesalahan) saat menyimpan file. Jika ada kesalahan, program akan menangani dengan cara yang lebih baik, seperti menampilkan pesan error kepada pengguna.
- **workbook.save(filename)**:
    - Menyimpan **workbook** (file Excel) dengan nama yang telah ditentukan oleh **filename**. File akan disimpan di direktori tempat program dijalankan.
    - Jika file disimpan dengan sukses, eksekusi akan melanjutkan ke blok **messagebox.showinfo** untuk memberi tahu pengguna bahwa ekspor berhasil.
- **messagebox.showinfo("Sukses", f"Data berhasil disimpan ke file: {filename}")**:
    - Menampilkan kotak pesan **info** kepada pengguna yang memberi tahu bahwa data berhasil disimpan, termasuk nama file yang disimpan.

- **except Exception as e:**
    - Menangkap kesalahan (error) yang terjadi selama eksekusi blok **try**.
    - **Exception as e** menangkap pesan kesalahan dan menyimpannya dalam variabel **e** untuk diakses lebih lanjut.
- **messagebox.showerror("Error", f"Gagal menyimpan file: {str(e)}")**:
    - Menampilkan kotak pesan **error** yang memberi tahu pengguna bahwa ada masalah saat menyimpan file.
    - **str(e)**: Mengubah objek kesalahan **e** menjadi string untuk menjelaskan jenis kesalahan yang terjadi.

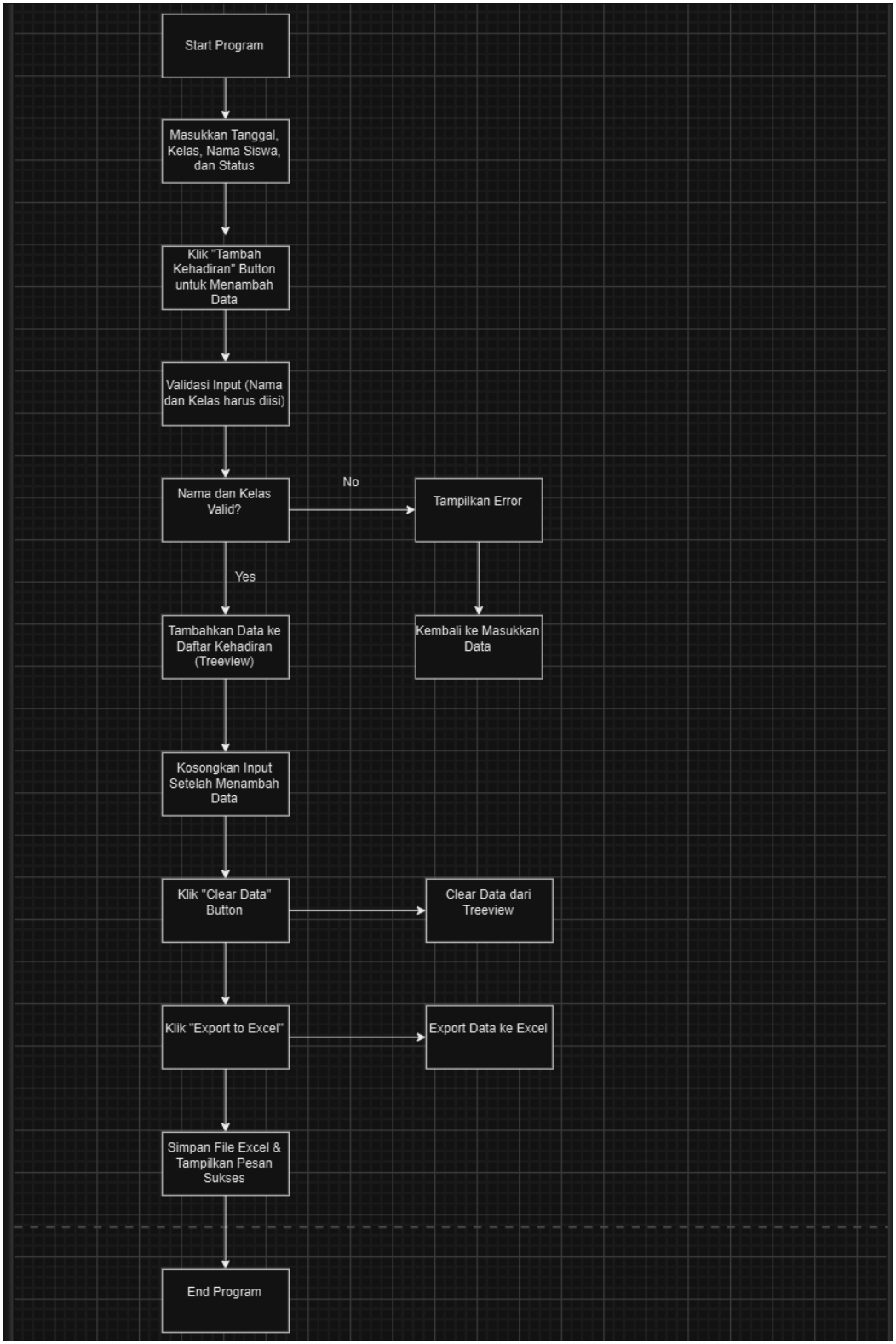
```
if __name__ == "__main__":
    root = tk.Tk()
    app = AppKehadiran(root)
    root.mainloop()
```

- **if __name__ == "__main__":**
    - Kode ini memastikan bahwa bagian program yang ada di dalamnya hanya akan dieksekusi jika file Python dijalankan langsung, bukan diimpor sebagai modul ke dalam file Python lain.
    - **__name__** adalah variabel khusus di Python yang berisi nama modul yang sedang dijalankan. Jika modul dijalankan langsung (bukan diimpor), maka nilai **__name__** adalah **"__main__"**, yang memungkinkan program di bawahnya untuk dieksekusi.
- **root = tk.Tk()**
    - Membuat instance dari **Tk()**, yang merupakan objek utama dari aplikasi **Tkinter**. Objek **root** ini mewakili jendela utama dari aplikasi GUI (Graphical User Interface).
    - Setelah objek **root** dibuat, aplikasi GUI siap untuk dibangun dan ditampilkan.
- **app = AppKehadiran(root)**
    - Membuat instance dari kelas **AppKehadiran** dan melewatkan objek **root** sebagai parameter.

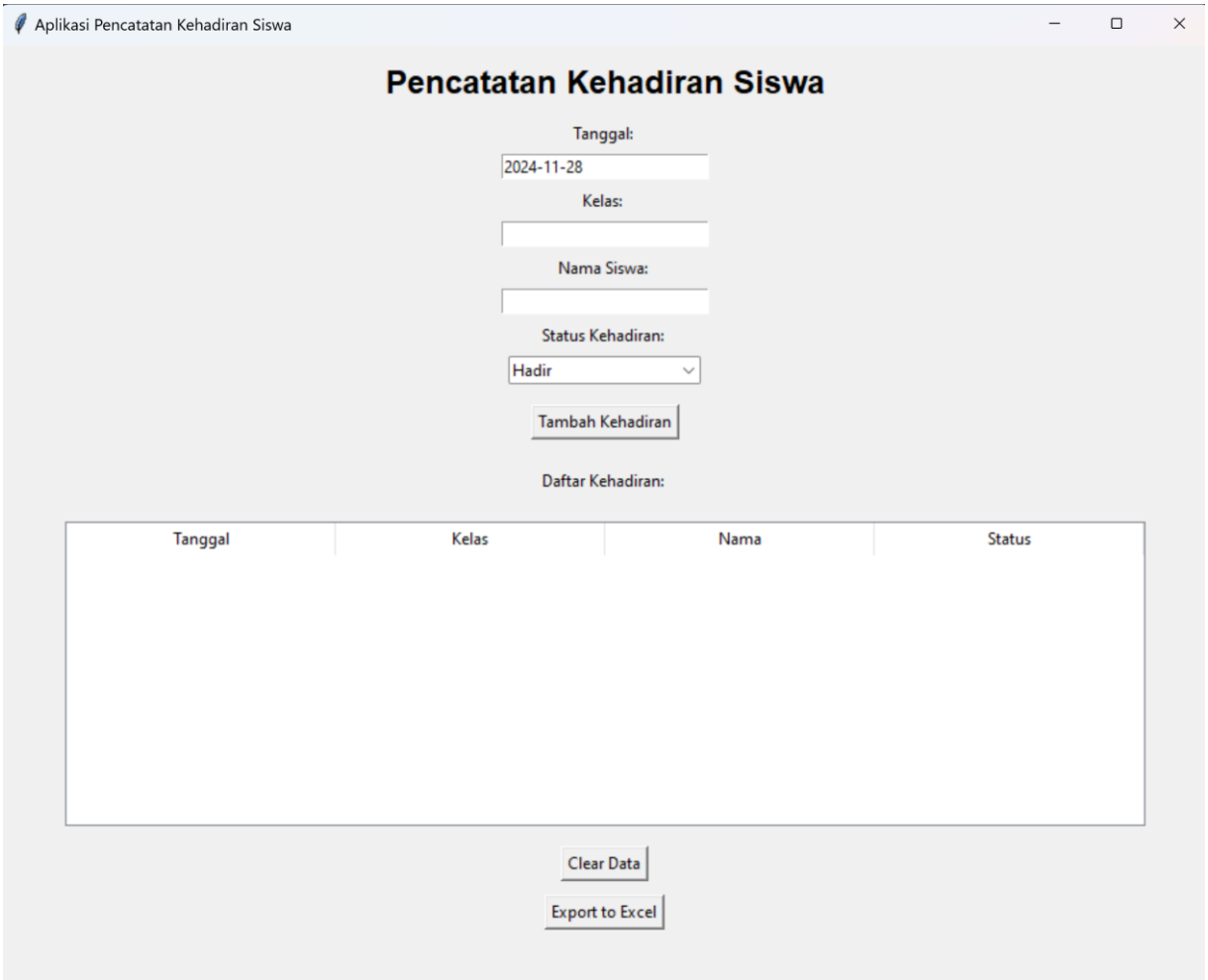
    - **AppKehadiran** adalah kelas yang mendefinisikan aplikasi pencatatan kehadiran siswa. Ketika objek **app** dibuat, konstruktor **__init__** dari kelas **AppKehadiran** akan dipanggil dan aplikasi GUI akan mulai dibangun di dalam jendela **root**.

- **root.mainloop()**
    - Memulai **main event loop** dari aplikasi **Tkinter**.
    - Fungsi **mainloop()** menjaga aplikasi GUI tetap berjalan dan mendengarkan aksi pengguna, seperti klik tombol, masukan teks, dan lainnya.
    - Tanpa **mainloop()**, aplikasi GUI tidak akan muncul atau berfungsi.

- **Actifity Diagram**



- Hasil Di Thinker



- Penjelasan Pemakaian:
  - Sebelum kalian melakukan run install terlebih dahulu melalui terminal yaitu “pip install openpyxl” mengapa karena openpyxl adalah pustaka Python yang digunakan untuk membaca dan menulis file Excel (.xlsx).
  - Langkah selanjutnya aktifkan venv pada terminal dengan menuliskan “python -m venv venv” dan kemudian perintahkan “venv\Scripts\activate”.
  - Mengapa perlu mengaktifkan venv agar Menghindari Konflik Versi, Isolasi Paket, dan Manajemen yang Lebih Mudah
  - Baru kemudian kalian bisa run dengan pertintah di terminal “python nama_file.py. Karena disini nama file saya adalah Program.py maka perintahnya menjadi “python Program.py”

- Maka akan muncul tampilannya berisi:
  - Tanggal sudah terisi otomatis sesuai dengan hari ini tanggal berapa.
  - Kalian bisa mengisikan kelas sesuai dengan yang kalian ingin kan dan jika kalian tidak mengisi maka akan error karena kolom kosong.
  - Sama hal nya dengan tanggal kalian bisa mengisi Nama Siswa sebebas kalian dan pastikan jangan kosong
  - Terdapat 4 pilihan yang bisa kalian pilih dengan Membuat combobox (daftar dropdown) sesuai apa yang mau kamu pilih di antara ke 4 pilihan itu.
  - Kemudia table akan terisi sendiri jika kalian menekan tombol Tambah Kehadiran.
  - Fungsi tombol Clear Data adalah menghapus table jika kalian ingin membuangnya
  - Tombol Export To Excel adalah tombol dimana kalian bisa mengubah data yang sudah kalian input menjadi data excel dan tersimpan di folder yang sama.

- **Kesimpulan**

Aplikasi Pencatatan Kehadiran Siswa berbasis Python dan Tkinter ini menawarkan solusi praktis untuk pengelolaan data kehadiran siswa secara digital. Dengan antarmuka yang intuitif dan fitur ekspor data, aplikasi ini dapat meningkatkan efisiensi proses pencatatan di institusi pendidikan. Di masa depan, aplikasi ini dapat ditingkatkan dengan fitur tambahan seperti pencarian data, filter berdasarkan tanggal atau kelas, serta integrasi dengan sistem administrasi sekolah.