LAPORAN TUBES ALPRO GROUP 1
APLIKASI PERPUSTAKAAN MINI

Anggota Kelompok :

1. Muhammad Zahir Mubasysyir 103102400073
2. Asaagama Nashrul Haq 103102400065
3. Auliya Nisa' Nur Rohmah 103102400056

PROGRAM STUDI SAINS DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY SURABAYA

2025

PENDAHULUAN

Menurut UU Perpustakaan pada Bab I pasal 1 menyatakan Perpustakaan adalah institusi yang mengumpulkan pengetahuan tercetak dan terekam, mengelolanya dengan cara khusus guna memenuhi kebutuhan intelektualitas para penggunanya melalui beragam cara interaksi pengetahuan(*Kompilasi-Public-453*, n.d.). Perpustakaan umumnya memuat koleksi buku dan majalah yang dapat dimanfaatkan oleh masyarakat dengan meminjam dalam jangka waktu tertentu. Namun sekarang fungsi perpustakaan sudah berkembang, bukan hanya sebagai tempat menyimpan dan mencari buku tapi juga bisa sebagai sumber/tempat mencari informasi. Seiring berkembangnya zaman yang serba digital, kini juga tersedia perpustakaan digital yang bisa diakses di mana saja dan kapan saja. Banyak institusi yang kini menyediakan *website* atau aplikasi agar masyarakat bisa mengakses perpustakaan lebih mudah.

Dalam pemenuhan tugas besar mata kuliah Algoritma Pemrograman akan dibuat aplikasi perpustakaan mini sederhana. Aplikasi ini nantinya akan memuat fitur-fitur dasar dari aplikasi perpustakaan digital yang diambil dari *point of view* pengguna. Fitur-fitur yang termuat antara lain penambahan buku, pencarian buku, pengurutan buku, serta penghapusan data buku. Dalam pembuatan aplikasi ini memanfaatkan bahasa pemrograman Python dengan antarmuka berbasis GUI (*Graphical User Interface*) menggunakan *CustomTkinter*. GUI merupakan laat yang memungkinkan pengguna dapat berinteraksi dengan perangkat keras pada komputer serta dapat memudahkan penggunna dalam mengoperasikan sebuah sistem operasi (Wintana et al., 2022). Sedangkan Tkinter menyediakan cara mudah bagi pengguna Python untuk membuat elemen GUI menggunakan *widget* yang ditemukan di *toolkit Tk*. *Widget Tk* ini dapat digunakan untuk membuat tombol, menu, bidang data, dan sebagainya dalam aplikasi python.

Tujuan dibuatnya aplikasi perpustakaan mini ini adalah untuk menerapkan konsep dasar algoritma pemrograman. Membangun sebuah sistem perpustakaan mini yang mampu menangani data buku secara efisien melalui fitur penambahan buku, pencarian buku, pengurutan buku, serta penghapusan buku. Mendukung pembelajaran mahasiswa dalam memahami penerapan struktur data, algoritma pencarian dan pengurutan, serta teknik pemrograman berorientasi objek dan antarmuka grafis.

DESKRIPSI APLIKASI

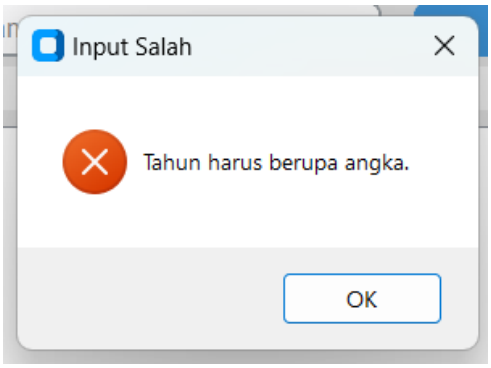
Aplikasi yang dibuat adalah aplikasi perpustakaan mini sederhana dengan mengambil *point of view* pengguna. Di dalam aplikasi akan memuat konsep dasar dari pemrograman, seperti penggunaan *array*, *searching*, *sorting*, dan *exception handling*. Dalam pembuatan aplikasi memanfaatkan bahasa pemrograman Python dan antarmuka berbasis GUI (*Graphical User Interface*) menggunakan *CustomTkinter*. Fitur-fitur yang dimuat dalam aplikasi mencakup penambahann data buku, pencarian berdasarkan judul/pengarang, pengurutan buku berdasarkan abjad/tahun terbit, *import file* csv, serta penghapusan data buku tertentu. Berikut untuk penjelasan lengkap untuk kode yang digunakan :

1. *Import customtkinter*

   Sebagai awalan pada kode akan berisi beberapa import dari modul yang berbeda. Digunakan kode sebagai berikut :

```
import customtkinter as ctk
from tkinter import messagebox
from tkinter import ttk
from dataclasses import dataclass
```

   - Kode *'import customtkinter as ctk'* digunakan untuk mengimpor *customtkinter* dengan alias ctk. *Customtkinter* ini merupakan sebuah *library* berbasis tkinter yang memungkinkan pembuatan GUI (*Graphical User Interface*) dengan tampilan lebih bersih. *Library* ini digunakan untuk membuat dan mengatur tampilan elemen GUI seperti tombol, label, input, dan lain-lain.
   - Kode *'from tkinter import messagebox'* adalah modul untuk menampilkan kotak dialog pop-up. Kotak dialog ini dimunculkan ketika suatu input salah. Berikut untuk contoh tampilannya :



   - Kode *'from tkinter import ttk'* untuk menampilkan elemen-elemen GUI, seperti *button*, *frame*, *font*, dan lain sebagainya.
   - Kode *'from dataclasses import dataclass'* digunakan untuk mendefinisikan *class* dari data yang ingin disimpan. Misal seperti *class* buku yang membutuhkan data judul, pengarang,

tahun terbit, dan kategori. Judul, pengarang, dan kategori akan disimpan dengan tipe data string, sedangkan tahun terbit akan disimpan dengan tipe data integer.

2. Struktur data dan *array* statis

```
@dataclass
class Buku:
    judul: str
    pengarang: str
    tahun: int
    kategori: str

MAX_BUKU = 115
data_buku = [None] * MAX_BUKU
jumlah_buku = 0
```

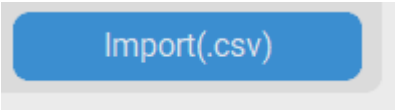
- Kode '*class Buku*' sampai pada kode '*kaegori: str'* digunakan untuk menyimpan tipe data dari data buku yang dibutuhkan. Seperti yang terlihat bahwa judul, pengarang, dan kategori disimpan dengan tipe data string, sedangkan tahun disimpan dengan tipe data integer.
- Kode '*MAX_BUKU = 115*' digunakan untuk mengidentifikasi jumlah maksimum buku yang bisa disimpan, yaitu 115 buku. Kode ini juga digunakan untuk membatasi kapasitas *array* supaya ukuran bersifat tetap.
- Kode '*data_buku = [None]*MAX_BUKU*' digunakan untuk menyediakan tempat kosong untuk menyimpan data buku nantinya.
- Kode '*jumlah_buku = 0*' merupakan variabel yang menyimpan jumlah buku yangg sudah ditambahkan ke dalam *array*. Nilainya 0 karena belum ada buku yang ditambahkan. Ketika menambah buku nanti nilainya akan berubah sesuai dengan jumlah buku yang ditambahkan.

3. Fitur *import* csv

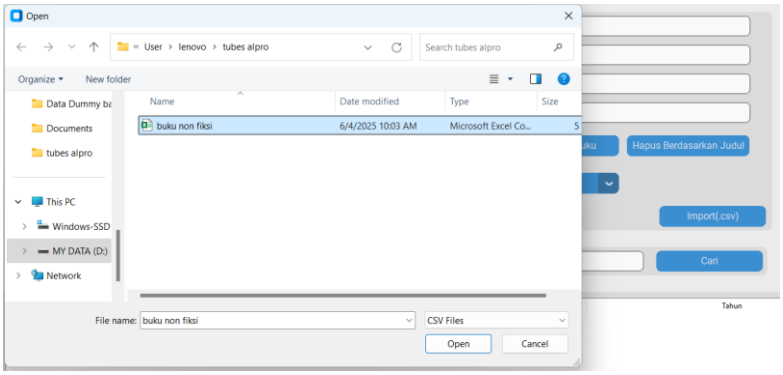
```
# Fungsi Impor
def import_csv():
    global jumlah_buku

    file_path = filedialog.askopenfilename(filetypes=[("CSV Files", "*.csv")])
    if not file_path:
        return

    try:
        with open(file_path, newline='', encoding='utf-8') as csvfile:
            reader = csv.DictReader(csvfile)
            for row in reader:
                if jumlah_buku >= MAX_BUKU:
                    messagebox.showwarning("Penuh", "Data buku sudah mencapai kapasitas maksimum.")
                    break

                try:
                    tahun = int(row["Tahun"])
                except ValueError:
                    continue  # skip baris kalau tahunnya bukan angka

                buku_baru = Buku(
                    judul=row["Judul"],
                    pengarang=row["Pengarang"],
                    tahun=tahun,
                    kategori=row["Kategori"]
                )

                data_buku[jumlah_buku] = buku_baru
                jumlah_buku += 1

        tampilkan_buku()
        messagebox.showinfo("Berhasil", "Data CSV berhasil dimuat.")
    except Exception as e:
        messagebox.showerror("Error", f"Gagal membaca file: {e}")
```

Menggunakan kode fungsi untuk mendefinisikan fitur impor data buku dari file csv ke dalam aplikasi perpustakaan.

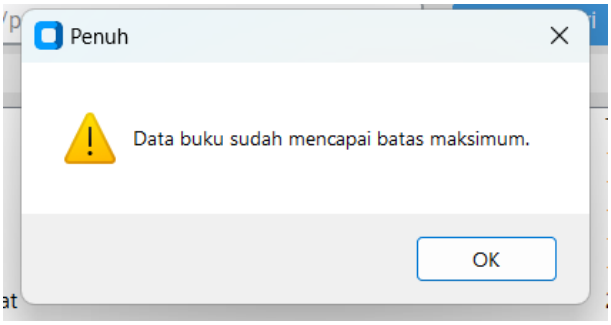
- Kode '*global jumlah_buku*' digunakan untuk mengakses variabel jumlah_buku yang berada di luar fungsi, supaya nanti bisa diubah dari dalam fungsi untuk fitur impor csv.
- Kode '*file_path = filedialog.askopenfilename(filetypes=[("CSV Files", "*.csv")]) if not file_path: return*' digunakan untuk menampilkan tombol dari impor csv pada aplikasi, apabila pengguna batal memilih file, fungsi akan langsung berhenti. Berikut untuk tampilan tombolnya :



- Kode '*with open(file_path, newline='', encoding='utf-8') as csvfile: reader = csv.DictReader(csvfile)*' digunakan agar pengguna bisa membuka file csv yang telah dipilih, dan membaca *dictionary*-nya per baris berdasarkan kolom/*header*. Berikut untuk tampilannya :



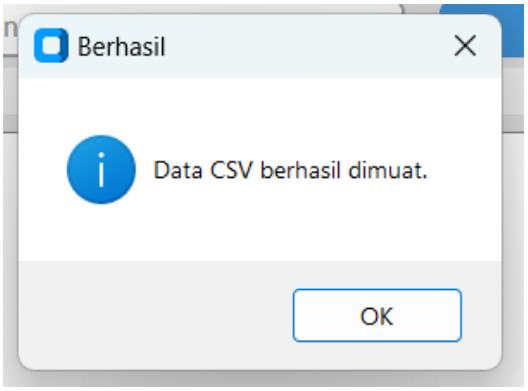
- Kode '*for row in reader: if jumlah_buku >= MAX_BUKU: messagebox.showwarning("Penuh", "Data buku sudah mencapai kapasitas maksimum.") break*' digunakan ketika *array* 'data_buku' sudah mencapai maksimum, yaitu 115. Maka nanti *output*-nya akan menampilkan semacam kotak pesan dengan tulisan yang ada pada kode. Berikut untuk tampilannya :



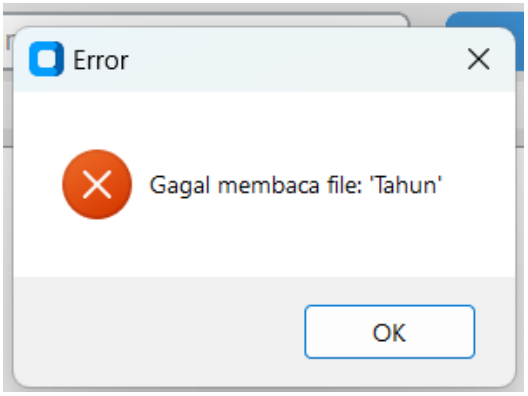
- Kode '*try: tahun = int(row["Tahun"]) except ValueError: continue*' digunakan untuk mengubah nilai pada kolom tahun menjadi angka atau integer (int). Apabila gagal seperti

kasus tahun kosong atau salah format, baris ini akan dilewati. Kode ini juga termasuk dalam *exception handling*.

- Kode '*buku_baru = Buku(judul=row["Judul"], pengarang=row["Pengarang"], tahun=tahun, kategori=row["Kategori"])*' digunakan untuk membuat objek 'Buku' dari baris csv yangg valid.
- Kode '*data_buku[jumlah_buku] = buku_baru jumlah_buku += 1*' digunakan untuk menyimpan objek 'buku_baru' ke *array* 'data_buku' pada indeks saat ini. Kemudian nilai pada variabel 'jumlah_buku' akan ditambah karena kita telah menambahkan satu buku lagi.
- Kode '*tampilkan_buku() messagebox.showinfo("Berhasil", "Data CSV berhasil dimuat.")*' digunakan ketika semua baris telah dibaca, maka akan menampilkan semacam kotak pesan dengan tulisan seperti pada kode. Setelah itu bisa dilihat bahwa dfatar buku telah ditambahkan. Berikut untuk tampilannya :



- Kode '*except Exception as e: messagebox.showerror("Error", f"Gagal membaca file: {e}")*' digunakan jika ada kesalahan ketika baca *file* (misal ketika *file* rusak atau formatnya salah) maka akan menampilkan pesan *error*. Berikut untuk tampilannya :

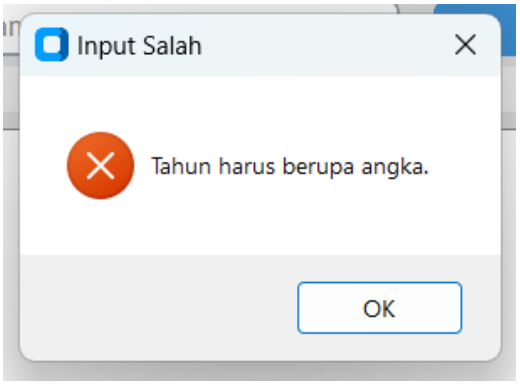


4. Fitur tambah buku

```
# --- Fungsi Tambah Buku ---
def tambah_buku():
    global jumlah_buku

    if jumlah_buku >= MAX_BUKU:
        messagebox.showwarning("Penuh", "Data buku sudah mencapai batas maksimum.")
        return

    try:
        tahun = int(entry_tahun.get())
    except ValueError:
        messagebox.showerror("Input Salah", "Tahun harus berupa angka.")
        return

    buku_baru = Buku(
        judul=entry_judul.get(),
        pengarang=entry_pengarang.get(),
        tahun=tahun,
        kategori=entry_kategori.get()
    )

    data_buku[jumlah_buku] = buku_baru
    jumlah_buku += 1
    tampilkan_buku()
    kosongkan_form()
```

Untuk kode dari *line* pertama sampai *line* ke -5 sudah dijelaskan pada poin sebelumnya.

- Kode '*try: tahun = int(entry_tahun.get()) except ValueError: messagebox.showerror("Input Salah", "Tahun harus berupa angka.") return*' berguna ketika pengguna memasukkan masukan selain tipe data yang sesuai (integer), seperti huruf atau simbol. Jika demikian maka akan mengeluarkan *output* dengan pesan error seperti pada kode. Berikut untuk tampilannya :



- Kode '*buku_baru = Buku(judul=entry_judul.get(), pengarang=entry_pengarang.get(), tahun=tahun, kategori=entry_kategori.get())*' digunakan untuk membuat objek 'Buku' yang sebelumnya sudah didefinisikan dengan '@dataclass' berdasarkan *input* dari pengguna.
- Kode '*tampilkan_buku()*' berfungsi untuk menampilkan ulang daftar semua buku agar bisa langsung terlihat.
- Kode '*kosongkan_form()*' digunakan untuk menghapus semua isi *form input*, supaya pengguna bisa langsung mengisi data baru tanpa menghapus manual.

5. Fitur tampilkan buku

```
# --- Fungsi Tampilkan Buku ---
def tampilkan_buku(data=None):
    # Bersihkan isi treeview
    for row in tree.get_children():
        tree.delete(row)

    if data is None:
        # Tampilkan semua data
        for i in range(jumlah_buku):
            b = data_buku[i]
            tree.insert("", "end", values=(b.judul, b.pengarang, b.tahun, b.kategori))
    else:
        # Tampilkan hasil pencarian dengan tag warna
        for b in data:
            tree.insert("", "end", values=(b.judul, b.pengarang, b.tahun, b.kategori), tags=('highlight',))
```

Untuk fitur tampilkan buku menggunakan kode fungsi di mana nantinya akan digunakan untuk menampilkan daftar buku di layar.

- Kode '*for row in tree.get_children(): tree.delete(row)*' digunakan untuk menghapus tampilan lama sebelum menampilkan data baru agar tidak terjadi duplikasi.
- Kode '*if data is None: for i in range(jumlah_buku): b = data_buku[i] tree.insert("", "end", values=(b.judul, b.pengarang, b.tahun, b.kategori))*' digunakan ketika kondisi data kosong dan tetap ingin menampilkan seluruh buku yang ada.
- Kode '*else: for b in data: tree.insert("", "end", values=(b.judul, b.pengarang, b.tahun, b.kategori), tags=('highlight',))*' digunakan ketika kondisi data tidak kosong dan ingin menampilkan hasil pencarian atau data tertentu saja.

6. Fitur kosongkan *form*

```
# --- Fungsi Kosongkan Form ---
def kosongkan_form():
    entry_judul.delete(0, ctk.END)
    entry_pengarang.delete(0, ctk.END)
    entry_tahun.delete(0, ctk.END)
    entry_kategori.delete(0, ctk.END)
```

Fitur kosongkan *form* menggunakan kode fungsi yang digunakan untuk menghapus atau membersihkan semua masukan yang ada pada kotak *input* setelah pengguna menambahkan buku atau ingin mulai mengisi data baru.

7. Fitur mengurutkan buku dan menampilkan buku berdasarkan pilihan pengguna

```
# --- Fungsi Urutkan dan Tampilkan Buku Berdasarkan Pilihan ---
def urutkan_buku(pilihan):
    global data_buku

    data_valid = [b for b in data_buku[:jumlah_buku] if b is not None]

    if pilihan == "A-Z":
        data_valid.sort(key=lambda b: b.judul.lower())
    elif pilihan == "Z-A":
        data_valid.sort(key=lambda b: b.judul.lower(), reverse=True)
    elif pilihan == "Terbaru":
        data_valid.sort(key=lambda b: b.tahun, reverse=True)
    elif pilihan == "Terlama":
        data_valid.sort(key=lambda b: b.tahun)

    # Tampilkan data yang sudah diurutkan
    for row in tree.get_children():
        tree.delete(row)

    for b in data_valid:
        tree.insert("", "end", values=(b.judul, b.pengarang, b.tahun, b.kategori))
```

Fitur pencarian buku menggunakan kode fungsi yang digunakan untuk mengurutkan tampilan dan menampilkan daftar buku berdasarkan pilihan dari pengguna (berdasarkan judul atau tahun). Jenis *sorting* yang digunakan adalah kombinasi dari *merge sort* dan *insertion sort*. *Merge sort* merupakan metode pengurutan dalam pemrograman di mana membagi masalah menjadi beberapa bagian kemudian mengurustkan setiap bagian tersebut, selanjutnya bagian yang telah diurutkan akan digabungkan menjadi satu. Alasan penggunaan *merge sort* dalam aplikasi perpustakaan mini karena efisien untuk data yang banyak serta tetap stabil dalam artian urutan data buku yang memiliki nilai kunci yang sama tidak akan dirubah (misalnya dua buku dengan tahun yang sama tetap urut ssesuai masukan awal). Sedangkan untuk *insertion sort* merupakan sebuah teknik pengurutan dengan carra membandingkan dan mengurutkan dua data pertama pada *array*, kemudian membandingkan data pada *array* berikutnya dan diperiksa apakah data sudah berada di tempat yang seharusnya. Alasan penggunaan *insertion sort* adalah karena waktu pengurutan sangat cepat untuk data yang kecil atau hampir terurut (efektif ketika pengguna masih menambahkan sedikit data dalam aplikasi).

- Kode '*data_valid = [b for b in data_buku[:jumlah_buku] if b is not None]*' digunakan untuk membuat *list* baru yang disimpan dalam variabel 'data_valid' yang berisi semua buku dalam *array* dari indeks 0 sampai jumlah buku yang ada dan agar hanya buku yang tidak kosong yang diambil.
- Kode '*if pilihan == "A-Z": data_valid.sort(key=lambda b: b.judul.lower())*' digunakan untuk mengurutkan buku berdasarkan judulnya dengan format A-Z. Juga terdapat kode agar pengurutan tidak peka huruf besar/kecil, jadi meskipun datanya berupa huruf kapital namun pengguna menginput data dengan huruf kecil, program tetap dapat membacanya.
- Kode '*elif pilihan == "Z-A": data_valid.sort(key=lambda b: b.judul.lower(), reverse=True)*' digunakan untuk mengurutkan buku berdasarkan judulnya dengan format Z-A. Juga terdapat kode agar pengurutan tidak peka huruf besar/kecil, jadi meskipun datanya berupa huruf kapital namuun pengguna menginput data dengan huruf kecil, program tetap dapat membacanya. Kode di samping *lower* digunakan untuk membuat urutan jadi kebalikannya.
- Kode '*elif pilihan == "Terbaru": data_valid.sort(key=lambda b: b.tahun, reverse=True)*' digunakan untuk mengurutkan buku berdasarkan tahun terbitnya dari yang paling baru sampai ke yang terlama.
- Kode '*elif pilihan == "Terlama": data_valid.sort(key=lambda b: b.tahun)*' digunakan untuk mengurutkan buku berdasarkan tahun terbitnya dari yang paling lama sampai ke yang terbaru.
- Kode '*for row in tree.get_children(): tree.delete(row)*' digunakan untuk mengosongkan tabel sebelum diisi ulang, supaya tidak terjadi duplikat data ketika menampilkan data baru.

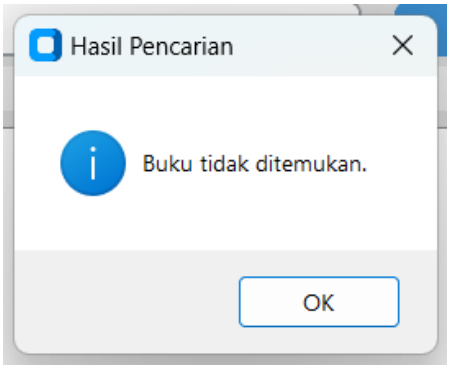
- Kode '*for b in data_valid: tree.insert("", "end", values=(b.judul, b.pengarang, b.tahun, b.kategori))*' digunakan untuk menampilkan data buku yang baru sesudah diurutkan ke dalam tabel.

8. Fitur hapus buku

```
# --- Fungsi Hapus Buku ---
def hapus_buku():
    global jumlah_buku
    judul_hapus = entry_judul.get().lower()

    index = -1
    for i in range(jumlah_buku):
        if data_buku[i] and data_buku[i].judul.lower() == judul_hapus:
            index = i
            break

    if index == -1:
        messagebox.showerror("Tidak Ditemukan", "Judul buku tidak ditemukan.")
        return

    for i in range(index, jumlah_buku - 1):
        data_buku[i] = data_buku[i + 1]
    data_buku[jumlah_buku - 1] = None
    jumlah_buku -= 1

    tampilkan_buku()
    kosongkan_form()
    messagebox.showinfo("Berhasil", "Buku berhasil dihapus.")
```

Fitur hapus buku menggunakan kode fungsi yang berguna untuk menghapus data buku dari *array* berdasarkan judul yang di-*input* oleh pengguna.

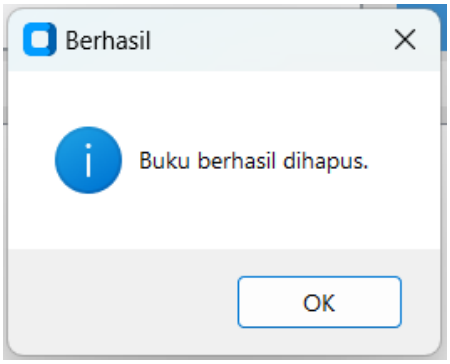
- Kode '*judul_hapus = entry_judul.get().lower()*' digunakan untuk mengambil masukan dari kolom '*entry_judul*'.
- Kode '*index = -1 for i in range(jumlah_buku): if data_buku[i] and data_buku[i].judul.lower() == judul_hapus: index = I break*' digunakan untuk mencari indeks buku yang cocok sesuai dengan masukan dari pengguna ke kolom.
- Kode '*if index == -1: messagebox.showerror("Tidak Ditemukan", "Judul buku tidak ditemukan.") return*' digunakan ketika judul buku yang dimasukkan pengguna tidak ditemukan. Akan muncul sebuah kotak pesandengan pesan seperti pada kode. Berikut tampilannya:



- Kode '*for i in range(index, jumlah_buku - 1): data_buku[i] = data_buku[i + 1] data_buku[jumlah_buku - 1] = None jumlah_buku -= 1*' digunakan untuk memindahkan

semua data buku setelah ada buku yang terhapus ke satu posisi lebih awal supaya celah yang kosong tertutupi, dan mengosongkan slot terakhir.

- Kode '*tampilkan_buku() kosongkan_form() messagebox.showinfo("Berhasil", "Buku berhasil dihapus.")*' digunakan untuk menghapus isi masukan supaya kotak *input* dapat digunakan lagi. Setelah buku berhasil dihapus, akan ada kotak pesan dengan tulisan seperti di kode. Berikut untuk tampilannya :



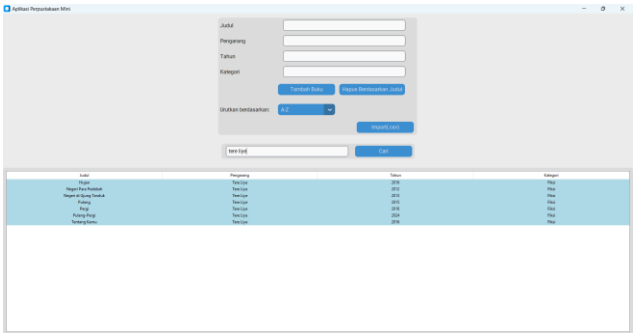
9. Fitur cari buku

```
# --- Fungsi Cari Buku ---
def cari_buku():
    keyword = entry_cari.get().lower()
    hasil = []

    for i in range(jumlah_buku):
        b = data_buku[i]
        if keyword in b.judul.lower() or keyword in b.pengarang.lower():
            hasil.append(b)

    if hasil:
        tampilkan_buku(hasil)
    else:
        messagebox.showinfo("Hasil Pencarian", "Buku tidak ditemukan.")
```

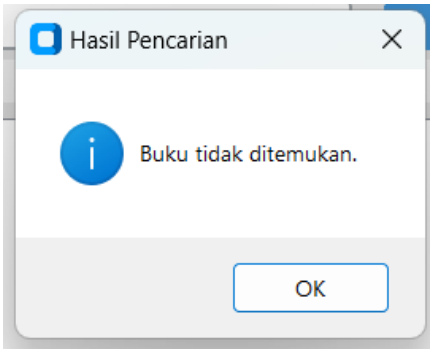
Fitur cari buku menggunakan kode fungsi yang berguna untuk mencari buku berdasarkan judul atau pengarang yang dimasukkan pengguna. Dalam fitur ini digunakan *sequential search*, karena data yang disimpan merupakan data yang tidak terurut. Metode pencarian pada *sequential search* dilakukan dengan membandingkan sebuah elemen data yang dicari dengan elemen pertama sampai dengan terakhir, jika data yang dicari ditemukan maka pencarian akan dihentikan (Cahyono et al., n.d.).

- Kode '*keyword = entry_cari.get().lower() hasil = []*' digunakan untuk mengambil teks pencarian dari input '*entry_cari*' lalu diubah ke huruf kecil supaya pencarian tidak peka terhadap huruf besar atau kecil. Kemudian data-data buku yang cocok dengan pencarian akan ditampung pada *list* kosong dengan variabel 'hasil'.
- Kode '*for i in range(jumlah_buku): b = data_buku[i] if keyword in b.judul.lower() or keyword in b.pengarang.lower(): hasil.append(b)*' digunakan untuk mencari buku yang sesuai dengan *keyword* dari masukan pengguna. Dalam kode ini dimanfaatkan kode perulangan dan ketika ditemukan buku yang cocok, buku itu akan ditambahkan dalam *list* 'hasil'.

- Kode '*if hasil: tampilkan_buku(hasil)*' digunakan untuk menampilkan buku-buku hasill pencarian. Berikut untuk tampilannya :



- Kode '*else: messagebox.showinfo("Hasil Pencarian", "Buku tidak ditemukan.")*' digunakan ketika tidak ada data buku yang cocok dengan masukan pengguna. Akan ada kotak pesan dengan tulisan seperti pada kode. Berikut untuk tampilannya :



10. GUI

```
# --- GUI ---
ctk.set_appearance_mode("System")  # bisa diganti 'Light' atau 'Dark'
ctk.set_default_color_theme("blue")  # bisa diganti tema lain
```

GUI merupakan tampilan antarmuka grafis

- Kode *'ctk.set_appearance_mode("System") # bisa diganti 'Light' atau 'Dark''* berguna untuk mengatur mode tampilan aplikasi GUI. Dengan nilai *"System"*, tampilan aplikasi akan mengikuti pengaturan sistem operasi pengguna (otomatis menggunakan mode terang atau gelap).
- Kode *'ctk.set_default_color_theme("blue") # bisa diganti tema lain'* berguna untuk menetapkan tema warna utama dari antarmuka GUI. Dalam kasus ini, tema yang digunakan adalah *"blue"*. Tema ini menentukan warna dasar dari komponen GUI seperti tombol, kotak input, dan lain-lain. Nilai ini bisa diganti dengan tema lain seperti *"green"*, *"dark-blue"*, atau tema kustom sesuai keinginan.

```
root = ctk.CTk()
root.title("Aplikasi Perpustakaan Mini")
root.geometry("700x500")
```

- Kode *'root = ctk.CTk()'* membuat objek jendela utama dari aplikasi GUI dengan menggunakan *class CTk* dari pustaka *customtkinter* yang disimpan dalam variabel *root*, yang nantinya akan digunakan untuk menambahkan komponen GUI lainnya.
- Kode *'root.title("Aplikasi Perpustakaan Mini")'* merupakan judul aplikasi menjadi "Aplikasi Perpustakaan Mini". Judul ini akan ditampilkan pada bagian atas jendela (*title bar*).
- Kode *'root.geometry("700x500")'* mengatur ukuran awal jendela aplikasi menjadi 700 piksel lebar dan 500 piksel tinggi. Nilai bisa disesuaikan dengan kebutuhan desain antarmuka.

11. *Frame input*

```
# Frame Input
frame_input = ctk.CTkFrame(root)
frame_input.pack(pady=10)

ctk.CTkLabel(frame_input, text="Judul").grid(row=0, column=0, sticky="w", padx=5, pady=5)
ctk.CTkLabel(frame_input, text="Pengarang").grid(row=1, column=0, sticky="w", padx=5, pady=5)
ctk.CTkLabel(frame_input, text="Tahun").grid(row=2, column=0, sticky="w", padx=5, pady=5)
ctk.CTkLabel(frame_input, text="Kategori").grid(row=3, column=0, sticky="w", padx=5, pady=5)

entry_judul = ctk.CTkEntry(frame_input, width=300)
entry_pengarang = ctk.CTkEntry(frame_input, width=300)
entry_tahun = ctk.CTkEntry(frame_input, width=300)
entry_kategori = ctk.CTkEntry(frame_input, width=300)

entry_judul.grid(row=0, column=1, padx=10, pady=5)
entry_pengarang.grid(row=1, column=1, padx=10, pady=5)
entry_tahun.grid(row=2, column=1, padx=10, pady=5)
entry_kategori.grid(row=3, column=1, padx=10, pady=5)

btn_tambah = ctk.CTkButton(frame_input, text="Tambah Buku", command=tambah_buku)
btn_tambah.grid(row=4, column=1, pady=10,padx=(15,200))

btn_hapus = ctk.CTkButton(master=frame_input, text="Hapus Berdasarkan Judul", command=hapus_buku)
btn_hapus.grid(row=4, column=1, padx=(150,15), pady=5)

btn_import = ctk.CTkButton(master=frame_input, text="Import(.csv)", command=import_csv)
btn_import.grid(row=6, column=1, pady=5,padx=(200,0))
```

- Kode *'frame_input = ctk.CTkFrame(root)'* dan *'frame_input.pack(pady=10)'* membuat *frame* yang menjadi wadah untuk komponen *input* seperti label, *entry*, dan tombol. *Frame* ini ditempatkan di dalam jendela utama *root* dan diberi jarak 10 piksel menggunakan *'pack(pady=10)'*.
- Label *input* dengan kode:

  '*ctk.CTkLabel(frame_input, text="Judul").grid(row=0, column=0, sticky="w", padx=5, pady=5)*'

  '*ctk.CTkLabel(frame_input, text="Pengarang").grid(row=1, column=0, sticky="w", padx=5, pady=5)*'

  '*ctk.CTkLabel(frame_input, text="Tahun").grid(row=2, column=0, sticky="w", padx=5, pady=5)*'

  '*ctk.CTkLabel(frame_input, text="Kategori").grid(row=3, column=0, sticky="w", padx=5, pady=5)*'

Berfungsi untuk menampilkan label teks “Judul”, “Pengarang”, “Tahun”, dan “Kategori” secara vertikal. Masing-masing label ditempatkan di kolom ko-0 (sebelah kiri) dari *frame*, dengan pengaturan *sticky=”w”* agar rata kiri, serta diberi jarak antar komponen menggunakan *padx* dan *pady*.

- *Entry input* dengan kode:

  ‘*entry_judul = ctk.CTkEntry(frame_input, width=300)*’

  ‘*entry_pengarang = ctk.CTkEntry(frame_input, width=300)*’

  ‘*entry_tahun = ctk.CTkEntry(frame_input, width=300*’

  ‘*entry_kategori = ctk.CTkEntry(frame_input, width=300*’

  Merupakan *input field* (CTkEntry) dengan lebar 300 piksel untuk menerima masukan dari pengguna, yaitu judul buku, nama pengarang, tahun terbit, dan kategori buku.
- Penempatan *entry input* dengan kode:

  ‘*entry_judul.grid(row=0, column=1, padx=10, pady=5)*’

  ‘*entry_pengarang.grid(row=1, column=1, padx=10, pady=5)*’

  ‘*entry_tahun.grid(row=2, column=1, padx=10, pady=5)*’

  ‘*entry_kategori.grid(row=3, column=1, padx=10, pady=5)*’

  Berfungsi untuk mengatur posisi *entry* yang telah dibuat sebelumnya. Masing-masing *entry* ditempatkan sejajar dengan labelnya. Diberikan jarak horizontal dan vertikal agar tampilan antar *input* nyaman dilihat.
- Tombol hapus dengan kode:

  ‘btn_hapus = ctk.CTkButton(master=frame_input, text="Hapus Berdasarkan Judul", command=hapus_buku)’

  ‘btn_hapus.grid(row=4, column=1, padx=(150,15), pady=5)’

  Merupakan kode untuk tombol menghapus data berdasarkan judul buku yang diinputkan.
- Tombol *import* dengan kode:

  ‘btn_import = ctk.CTkButton(master=frame_input, text="Import(.csv)", command=import_csv)’

  ‘btn_import.grid(row=6, column=1, pady=5, padx=(200,0))’

  Tombol untuk mengimpor data buku dari *file* eksternal berformat csv. Kode juga berguna untuk menjalankan fungsi *import_csv* saat tombol diklik.

12. *Frame* cari dan urutkan

```
# --- Frame Cari dan Urutkan ---
frame_cari = ctk.CTkFrame(master=root)
frame_cari.pack(pady=10)

entry_cari = ctk.CTkEntry(master=frame_cari, placeholder_text="Cari judul/pengarang...", width=300)
entry_cari.grid(row=0, column=0, padx=10, pady=(5,5))

btn_cari = ctk.CTkButton(master=frame_cari, text="Cari", command=cari_buku)
btn_cari.grid(row=0, column=1, padx=5,pady=(5,5))
```

- *Frame* dengan kode:

'frame_cari = ctk.CTkFrame(master=root)'

'frame_cari.pack(pady=10)'

Memuat sebuah *frame* baru bernama '*frame_cari*' yang berfungsi sebagai wadah untuk komponen pencarian data buku. Ditempatkan di dalam root utama dengan jarak vertikal sebesar 10 piksel kebawah.

- *Entry* pencarian dengan kode:

  'entry_cari = ctk.CTkEntry(master=frame_cari, placeholder_text="Cari judul/pengarang...", width=300)'

  'entry_cari.grid(row=0, column=0, padx=10, pady=(5,5))'

  Merupakan kode yang membuat kolom input untuk memberikan petunjuk kepada pengguna.

- Tombol cari dengan kode:

  'btn_cari = ctk.CTkButton(master=frame_cari, text="Cari", command=cari_buku)'

  'btn_cari.grid(row=0, column=1, padx=5, pady=(5,5))'

  Memuat tombol dengan label "Cari" yang berfungsi untuk mengeksekusi pencarian berdasarkan *input* dari kolom *entry* sebelumnya.

13. *Dropdown* urutan:

```
# Dropdown Urutan
label_urut = ctk.CTkLabel(frame_input, text="Urutkan berdasarkan:")
label_urut.grid(row=5, column=0, padx=5, pady=10, sticky="w")

opsi_urutan = ["A-Z", "Z-A", "Terbaru", "Terlama"]
combo_urutan = ctk.CTkOptionMenu(frame_input, values=opsi_urutan, command=urutkan_buku)
combo_urutan.grid(row=5, column=1, padx=(15,10), pady=5, sticky="w")
```

- Label *dropdown* urutan:

  'label_urut = ctk.CTkLabel(frame_input, text="Urutkan berdasarkan:")'

  'label_urut.grid(row=5, column=0, padx=5, pady=10, sticky="w")'

  Baris ini memuat label teks yang bertuliskan "Urutankan Berdasarkan:"

- Daftar opsi urutan:

  'opsi_urutan = ["A-Z", "Z-A", "Terbaru", "Terlama"]'

  Variabel berisi daftar urutan yang dapat dipilih *user*, seperti mengurutkan dari A ke Z, Z ke A, berdasarkan waktu terbaru atau terlama.

- *Dropdown* (*option menu*):

  'combo_urutan = ctk.CTkOptionMenu(frame_input, values=opsi_urutan, command=urutkan_buku)'

  'combo_urutan.grid(row=5, column=1, padx=(15,10), pady=5, sticky="w")'

  Bagian ini membuat elemen *dropdown* dengan isi yang diambil dari '*opsi_urutan*'. Parameter *'command=urutkan_buku'* berarti setiap kali pengguna memilih salah satu opsi, fungsi urutkan_buku akan dijalankan untuk memperbarui urutan data sesuai pilihan.

Penggunaan ‘*padx=(15,10)’* memberikan jarak kanan kiri, dan ‘*sticky=”w”’* memastikan *dropdown* rata kiri.

14. *Frame* tabel:

```
# Frame Tabel
frame_tabel = ctk.CTkFrame(root)
frame_tabel.pack(pady=10, fill="both", expand=True)

kolom = ("Judul", "Pengarang", "Tahun", "Kategori")
tree = ttk.Treeview(frame_tabel, columns=kolom, show="headings", height=10)

for kol in kolom:
    tree.heading(kol, text=kol)
    tree.column(kol, width=150, anchor="center")
```

- *Frame*:

  ‘frame_tabel = ctk.CTkFrame(root)’

  ‘frame_tabel.pack(pady=10, fill="both", expand=True)’

  Membuat *frame* pada jendela utama dengan tampilan *frame* horizontal dan vertikal *'fill=”both”*, dan membesar ketika jendela diperluas *“expand=True”*
- Kolom:

  ‘kolom = ("Judul", "Pengarang", "Tahun", "Kategori")’

  ‘tree = ttk.Treeview(frame_tabel, columns=kolom, show="headings", height=10)’

  Didefinisikan sebagai *tuple* yang berisi nama-nama kolom yang akan ditampilkan. Tabel itu dibuat dengan *widget ‘ttk.Treeview’*, menampilkan kolom-kolom sesuai variabel kolom, hanya menampilkan *heading*-nya saja *‘show=”heading”’*.
- *‘for kol in kolom:’*

  ‘tree.heading(kol, text=kol)’

  ‘tree.column(kol, width=150, anchor="center")’

  Melakukan iterasi *‘for kol in kolom:’* untuk mengatur masing-masing *heading* dan properti kolom tabel. Dilakukan *‘tree.heading(kol, text=kol’* untuk menetapkan nama *heading* kolom sesuai label yang didefinisikan. Lalu *‘tree.column(kol, width=150, anchor="center")’* digunakan untuk mengatur lebar kolom dan rata tengah.

15. *Widget tree*:

```
# tree.pack(padx=10, pady=10, fill="both", expand=True)
tree.pack(padx=10, pady=10, fill="both", expand=True)
```

- Kode *‘tree.pack(padx=10, pady=10, fill="both", expand=True)’* berfungsi untuk menampilkan tabel *Treeview* dengan jarak luar, mengisi seluruh area *frame* secara horizontal dan vertikal, serta mengikuti perubahan ukuran jendela agar tampilan lebih responsif.

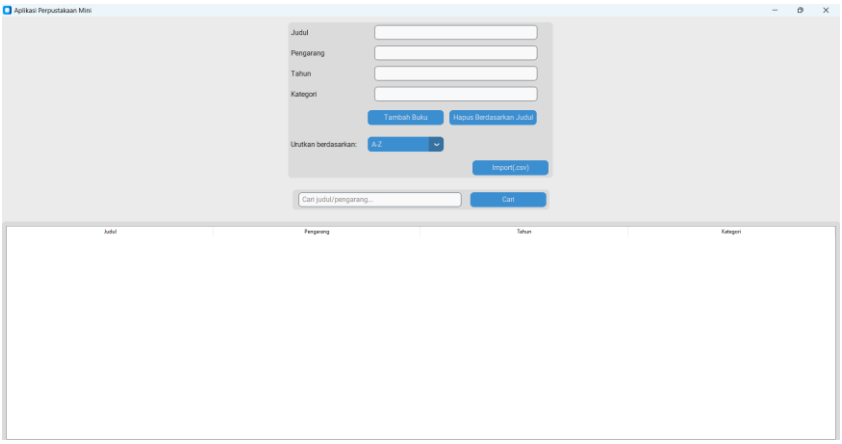
16. *Treeview*:

```
# Konfigurasi tag untuk highlight biru
tree.tag_configure('highlight', background='lightblue')

root.mainloop()
```

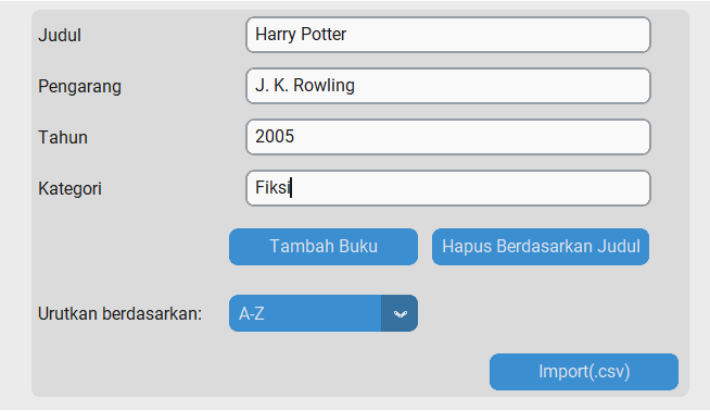
- Baris *'tree.tag_configure('highlight',background='lightblue')'* digunakan untuk mengatur tampilan baris dalam *Treeview* dengan tag *'highlight'* agar memiliki latar belakang berwarna biru muda.
- Sedangkan *'root.mainloop()'* menjalankan *loop* utama aplikasi, menjaga antarmuka tatap aktif dan responsif terhadap interaksi pengguna.

Untuk tampilan dari aplikasi perpustakana mini akan dijelaskan sebagai berikut :

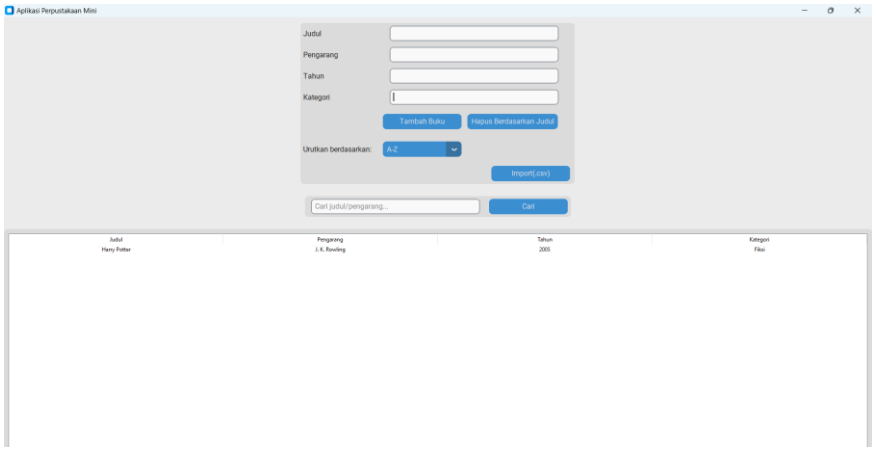
1. Berikut untuk tampilan awal dari aplikasi yang telah jadi :



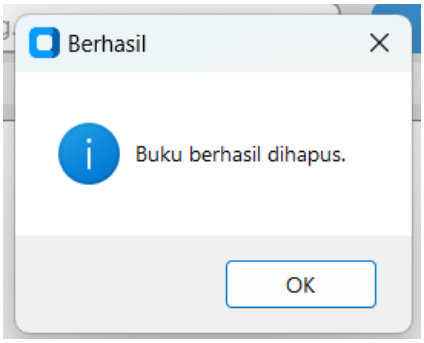
2. Ketika akan menambahkan buku secara manual dapat diketik melalui kotak seperti ini :



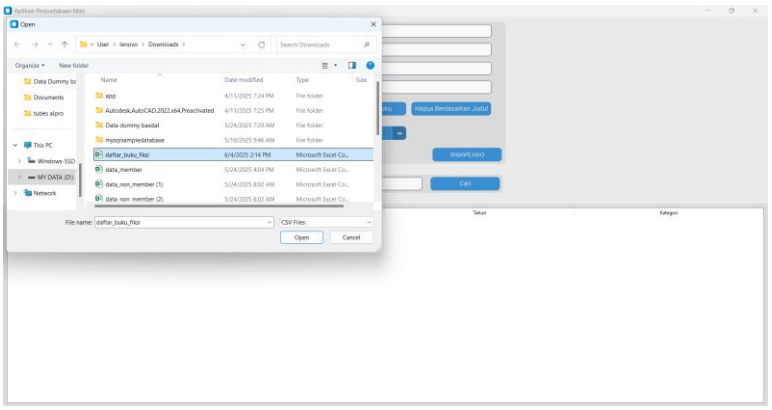
3. Setelahnya bisa diklik tombol ‘Tambah Buku’ dan otomatis buku akan ditambahkan, berikut tampilannya :



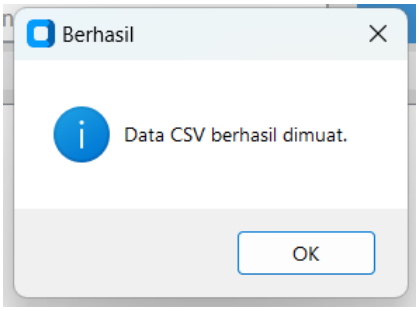
4. Ketika ingin menghapus buku, ketikkan judul pada kotak dan klik tombol ‘Hapus Berdasarkan Judul’. Kemudian data buku akan terhapus dan akan muncul notifikasi seperti ini :



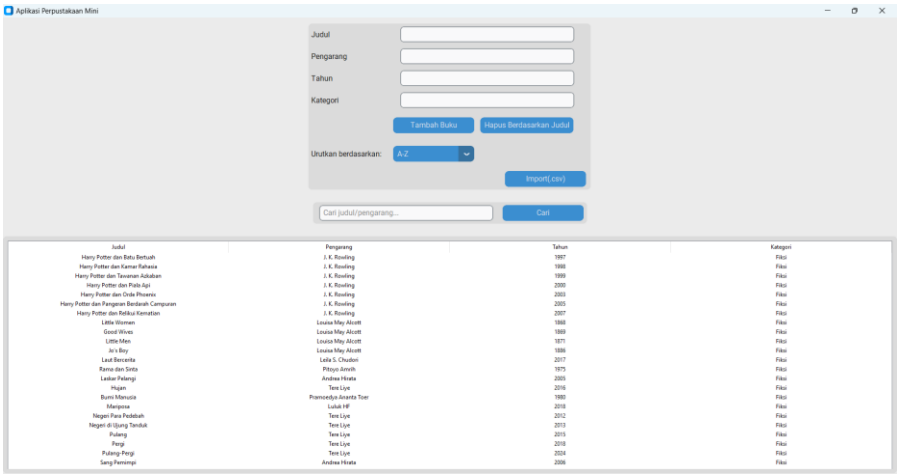
5. Ketika ingin menambahkan data melalui impor csv bisa klik tombol ‘Impor(.csv)’ dan pilih *file* yang ingin diimpor (pastikan *file* dalam bentuk csv dan dengan memuat ‘Judul, Pengarang, Tahun, Kategori’). Berikut untuk tampilannya :



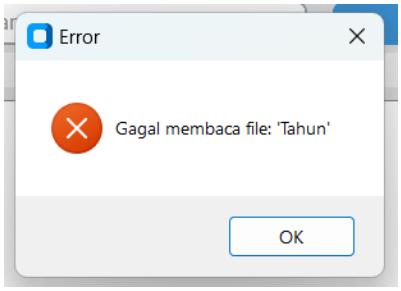
6. Klik ‘Open’ dan otomatis data pada *file* akan tersedia di tampilan aplikasi. Ketika *file* berhasil diimpor aka nada notifikasi seperti berikut :



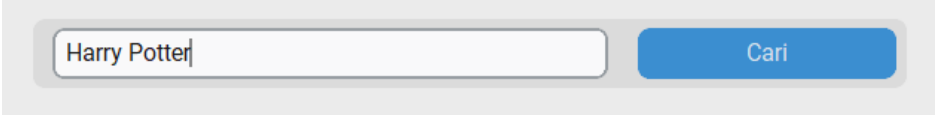
7. Berikut tampilannya ketika aplikasi sudah memuat data :



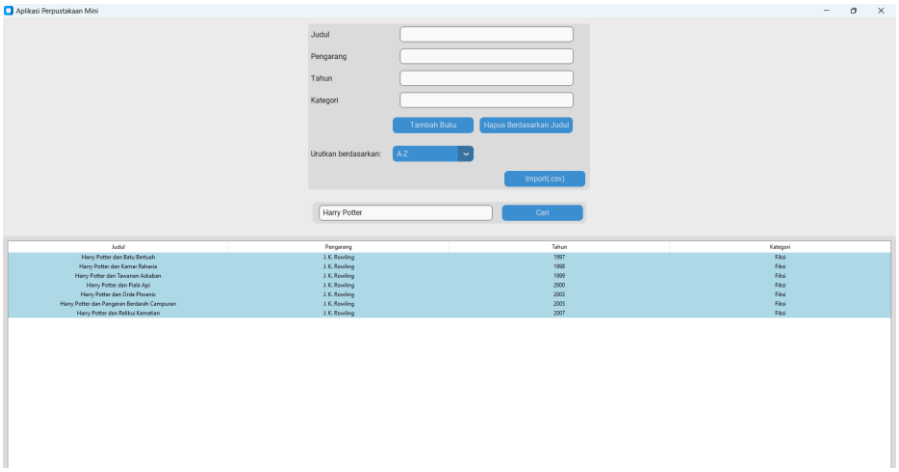
8. Apabila *file* tidak berhasil diimpor akan muncul notifikasi sebagai berikut :



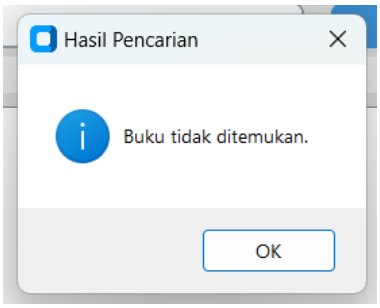
9. Ketika ingin mencari judul buku tertentu, bisa diketik berdasarkan judul atau pengarang. Kemudian klik ‘Cari’. Berikut untuk tampilannya :



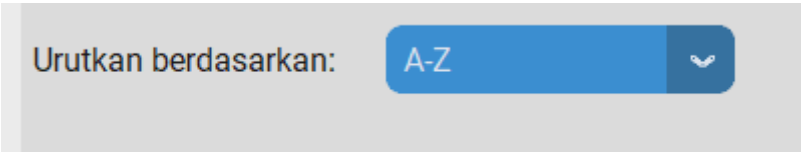
10. Berikut untuk tampilannya ketika menampilkan judul buku yang dicari :



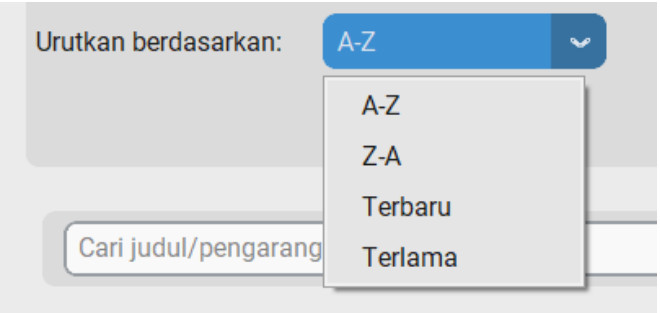
11. Apabila buku yang dicari tidak ada, maka akan muncul notifikasi seperti demikian :



12. Ketika ingin mengurutkan buku berdasarkan judul atau tahun terbit, bisa klik tombol berikut :

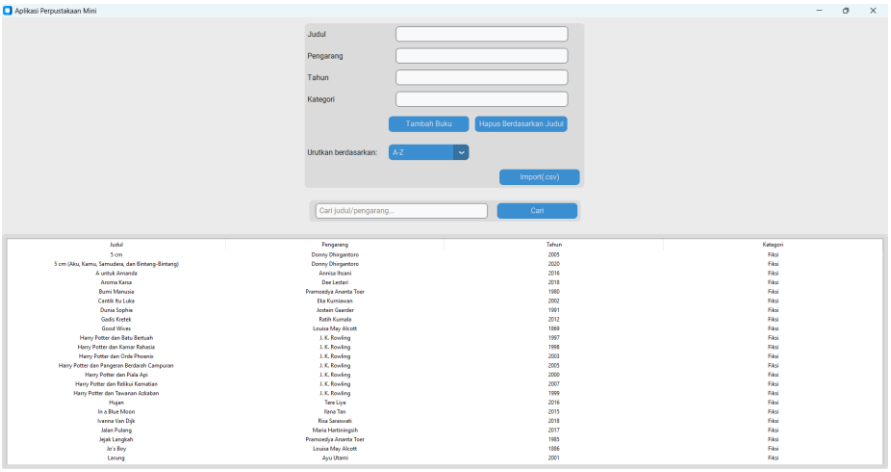


13. Klik tanda panah ke bawah maka akan muncul pilihan seperti berikut :

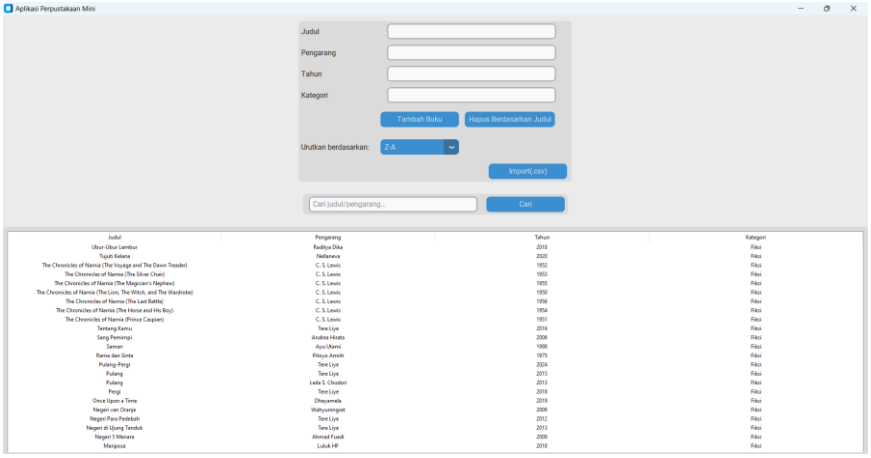


Pilihan ‘A-Z’ dan ‘Z-A’ untuk mengurutkan berdasarkan judul, sedangkan pilihan ‘Terbaru’ dan ‘Terlama’ untuk mengurutkan berdasarkan tahun terbit.

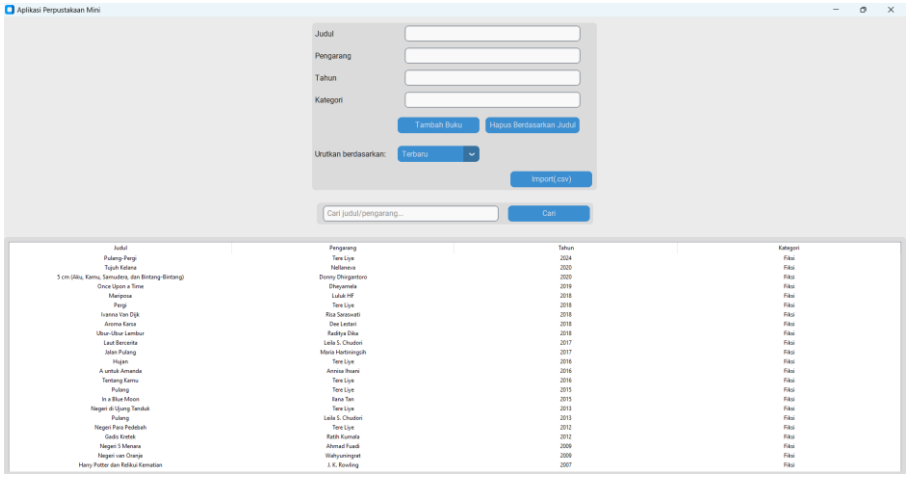
14. Berikut contoh tampilan apabila diurutkan berdasarkan judul :
    - Untuk A-Z



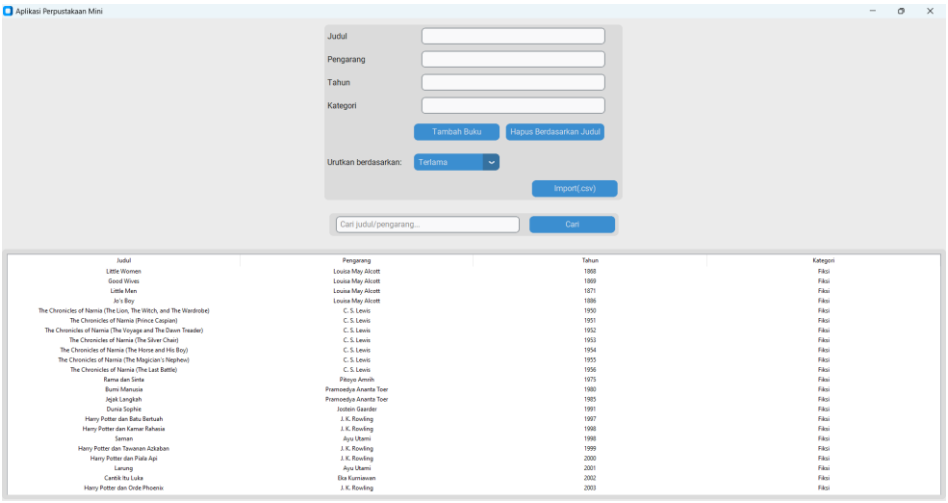
    - Untuk Z-A



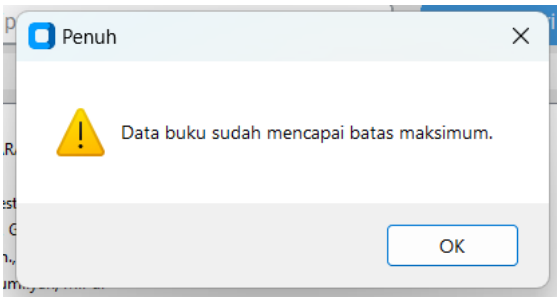
15. Berikut contoh tampilan apabila diurutkan berdasarkan tahun terbit :
    - Terbaru



- Terlama

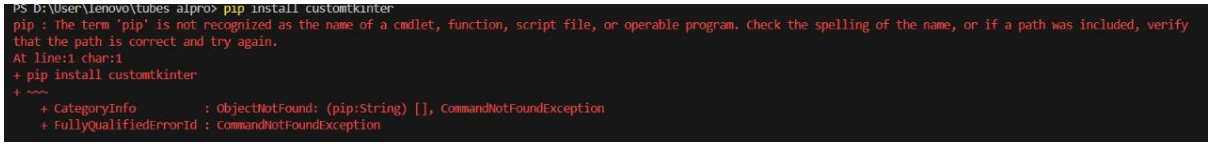


16. Apabila data buku yang dimasukkan sudah mencapai batas maksimum, maka pengguna sudah tidak dapat menambahkan data buku lagi dan akan muncul notifikasi seperti ini :



TANTANGAN & SOLUSI

Tantangan yang sempat dihadapi adalah ketika akan running kode yang sudah jadi, terdapat masalah pada tkinter. Masalah tersebut ditandai dengan munculnya error seperti berikut :



Setelah diidentifikasi, ternyata masalahnya adalah python atau pip belum dikenali oleh sistem. Sehingga solusinya adalah *install* python melalui *website*. Kemudian *file* python dijalankan dari aplikasi VSCode melalui '*Select Folder'*. Dengan cara ini, masalah sudah teratasi dan kode dapat dijalankan.

KESIMPULAN & REKOMENDASI

Kesimpulan

Aplikasi perpustakaan mini yang dibuat sebagai pemenuhan tugas besar mata kuliah Algoritma Pemrograman berhasil mengimplementasikan berbagai konsep dasar pemrograman. Fitur-fitur utama yang diminta dalam pemenuhan tugas besar juga dapat dijalankan dengan baik, seperti penambahan buku, pencarian buku berdasarkkan judul atau pengarang, pengurutan berrdasarkan judul atau tahun terbit, validasi *input* (*exception handling*), dan penerapan GUI dengan *tkinter*.

Aplikasi perpustakaan mini menggunakan fitur tambahan yaitu penambahan data menggunakan impor *file* dengan format csv. Hal ini akan mempermudah pengguna supaya bisa menambahkan data dalam jumlah banyak sekaligus tanpa harus mengetik satu persatu. Fitur pengurutandata menggunakan kombinasi dari *merge sort* dan *insertion sort*. Sedangkan untuk fitur pencarian menggunakan *sequential search*. Keduanya telah diterapkan secara efektif sesuai dengan karakteristik datanya.

Rekomendasi

Karena keterbatasan waktu membuat aplikasi belum bisa dikembangkan secara maksimal. Berikut beberapa rekomendasi fitur yang mungkin bisa dikembangkan :

1. Desain tampilan yang lebih responsif dan menarik untuk pengguna
   Meskipun *tkinter* sudah memberikan tampilan yang modern, *layout* dari aplikasi masih bisa dikembangkan lagi supaya lebih responsif untuk kenyamanan pengguna
2. Penambahan fitur penyimpanan permanen
   Aplikasi yang tersedia belum menyediakan fitur simpan permanen, sehingga ketika aplikasi ditutup data buku yang sudah ditambahkan hilang. Disarankan untuk menambah fitur penyimpanan lokal yangg bisa disimpan secara permanen
3. Fitur pengeditan buku
   Fitur edit data buku diperlukan supaya pengguna tidak perlu *input* ulang ketika ada data yang salah.
4. Fitur menghapus data dalam jumlah banyak sekaligus
   Fitur penghapusan data yang disediakan dalam aplikasi hanya bisa dilakukan satu-persatu. Fitur penghapusan data sekaligus akan memberikan efektifitas waktu bagi pengguna yang menggunakan aplikasi.
5. Fitur penambahan *cover*, sinopsis, dan isi buku
   Dalam aplikasi belum tersedia fitur yang disebutkan, sehingga pengguna hanya bisa menyimpan judul, pengarang, tahun terbit, dan kategori saja. Disarankan untuk menambahkan ketiga fitur tadi supaya fungsionalitas dari aplikasi juga bisa bertambah.
6. Validasi *input* lebih ketat
   Disarankan untuk menambahkan validasi yang lebih lengkap, misalnya seperti memastikkan semua kolom wajib diisi, atau bisa berupa hal lain untuk menghindari data tidak valid.

TAUTAN GITHUB

REFERENSI

Cahyono, T., Yulianti, L., & Yupianti,) ; (n.d.). Comparison of Sequential Searching Method and Turbo Boyer Method in Student Data Search at School Perbandingan Metode Sequential Searching Dan Metode Turbo Boyer Dalam Pencarian Data Siswa Di Sekolah. In *Jurnal Komputer* (Vol. 1, Issue 2).

kompilasi-public-453. (n.d.).

Wintana, D., Pribadi, D., & Nurhadi, M. Y. (2022). *Analisis Perbandingan Efektifitas White-Box Testing dan Black-Box Testing*. http://jurnal.bsi.ac.id/index.php/larik

Politeknik Yakpermas Banyumas. (2025). Tentang Perpustakaan. Diakses 12 Juni 2025, dari https://perpustakaan.politeknikyakpermas.ac.id/sample-page/tentang-perpustakaan/

ActiveState. (tanpa tahun). *What is Tkinter used for and how to install it?* Diakses 12 Juni 2025, dari https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-installit/#:~:text=Kerangka%20kerja%20ini%20menyediakan%20cara,dalam%20aplikasi%20Python

Nurhadi, A. (2023, Agustus 20). *Contoh merge sort: pengertian beserta cara kerjanya*. Kumparan. Diakses 12 Juni 2025, dari https://kumparan.com/how-to-tekno/contoh-merge-sort-pengertian-beserta-cara-kerjanya-20Qu12Ui8J7/4

School of Computer Science, BINUS University. (2019, 30 Desember). *Insertion Sort*. Diakses 12 Juni 2025, dari https://socs.binus.ac.id/2019/12/30/insertion-sort/