

# **LAPORAN APLIKASI PENGELOLAAN KOLEKSI MUSIK**

T. A. Semester Ganjil 2024/2025



**5230411302      AYU KUSUMA SETYANINGSIH**

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

**2024**

## DAFTAR ISI

DAFTAR ISI .....	2
BAB I PENDAHULUAN .....	3
A. Tujuan.....	3
B. Analisis Kebutuhan .....	3
BAB II PEMBAHASAN .....	4
A. Penjelasan Code .....	4
B. Flowchart .....	8
C. Hasil Pengujian .....	9
D. Tampilan Aplikasi .....	11
BAB III CODE .....	12
BAB IV PENUTUP .....	20
A. Kesimpulan .....	20

## **BAB I**

### **PENDAHULUAN**

Aplikasi Pengelolaan Koleksi Musik dirancang untuk membantu pengguna dalam mengelola koleksi musik pribadi. Aplikasi ini memberikan kemudahan dalam menambahkan, menghapus, dan menyaring data musik berdasarkan genre. Aplikasi ini dibangun menggunakan Python dengan pustaka Tkinter untuk menciptakan antarmuka grafis yang mudah digunakan. Laporan ini menjelaskan analisis, perancangan dan implementasi aplikasi.

#### **I. TUJUAN**

- Mempermudah pengelolaan koleksi musik.
- Memberikan fitur pencarian dan penyaringan berbasis genre.
- Mengembangkan aplikasi GUI menggunakan Python dengan pustaka Tkinter.

#### **II. ANALISIS KEBUTUHAN**

1. Pengguna dapat menambahkan data musik meliputi: judul, artis, album, dan genre.
2. Pengguna dapat melihat koleksi musik yang ditampilkan dalam tabel.
3. Pengguna dapat menyaring koleksi musik berdasarkan genre.
4. Pengguna dapat menghapus satu atau semua data musik.

## BAB II

### PEMBAHASAN

#### I. PENJELASAN CODE

##### 1. add\_music()

```
def add_music(self):
    title = self.entry_title.get()
    artist = self.entry_artist.get()
    album = self.entry_album.get()
    genre = self.genre_var.get()

    # Validasi input
    if not title or not artist or not album or genre == "Pilih Genre":
        messagebox.showwarning("Peringatan", "Semua kolom harus diisi!") # Jika ada kolom yang kosong, muncul
        return

    # Menambahkan data ke dalam list musik
    music_id = len(self.music_data) + 1
    self.music_data.append({"ID": music_id, "Judul": title, "Artis": artist, "Album": album, "Genre": genre})
    self.update_table()
    self.reset_input()
```

Fungsi `add_music()` digunakan untuk menambahkan data musik baru ke dalam koleksi. Fungsi ini pertama-tama mengambil input dari form, yaitu judul, artis, album, dan genre, lalu memvalidasi apakah semua kolom telah diisi dengan benar. Jika ada kolom yang kosong, fungsi akan menampilkan pesan peringatan menggunakan `messagebox.showwarning()` dan menghentikan proses penambahan data.

##### 2. Filter Genre

```
# Filter Genre
filter_frame = tk.Frame(self.root, bg="#DDA0DD")
filter_frame.pack(pady=10)

filter_label = tk.Label(filter_frame, text="Pilih Genre:", bg="#DDA0DD", font=("Arial", 12))
filter_label.pack(side=tk.LEFT, padx=5)

self.filter_genre_var = tk.StringVar()
self.filter_combobox = ttk.Combobox(filter_frame, textvariable=self.filter_genre_var, width=23)
self.filter_combobox['values'] = ["Semua"] + ["Klasik", "Jazz", "Pop", "R&B", "Hip-hop"]
self.filter_combobox.set("Semua")
self.filter_combobox.pack(side=tk.LEFT, padx=5)
self.filter_combobox.bind("<<ComboboxSelected>>", self.update_table)
```

Kode ini digunakan untuk membuat fitur filter berdasarkan genre musik di aplikasi. Bagian ini pertama-tama membuat `filter_frame`, sebuah frame yang berfungsi sebagai wadah untuk elemen-elemen filter, dan mengatur latar belakangnya dengan warna tertentu. Kemudian, label `filter_label` ditampilkan untuk memberikan petunjuk

kepada pengguna tentang fungsi filter tersebut. Selanjutnya, dibuatlah sebuah Combobox bernama filter\_combobox yang terhubung dengan variabel filter\_genre\_var dan berisi pilihan genre musik, termasuk opsi "Semua" untuk menampilkan semua genre.

### 3. Table Musik

```
# Table musik
music_table_label = tk.Label(frame_table, text="Koleksi Musikmu", bg="#F0F8FF", font=("Arial", 14, "bold"))
music_table_label.pack(pady=10)

self.music_table = ttk.Treeview(frame_table, columns=("ID", "Judul", "Artis", "Album", "Genre"), show="headings")
self.music_table.heading("ID", text="ID")
self.music_table.heading("Judul", text="Judul")
self.music_table.heading("Artis", text="Artis")
self.music_table.heading("Album", text="Album")
self.music_table.heading("Genre", text="Genre")
self.music_table.column("ID", width=50, anchor="center")
self.music_table.column("Judul", width=150)
self.music_table.column("Artis", width=150)
self.music_table.column("Album", width=150)
self.music_table.column("Genre", width=100)
self.music_table.pack(pady=10)
```

Kode ini digunakan untuk membuat tabel yang menampilkan koleksi musik di aplikasi. Label music\_table\_label ditampilkan di atas tabel untuk memberikan informasi kepada pengguna bahwa bagian ini berisi koleksi musikmu. Kemudian, sebuah Treeview bernama music\_table dibuat dalam frame\_table untuk menampilkan data musik dalam bentuk tabel. Kolom-kolom tabel ini terdiri dari "ID", "Judul", "Artis", "Album", dan "Genre".

### 4. Update Table

```
def update_table(self, event=None):
    # Hapus semua data di tabel
    for item in self.music_table.get_children():
        self.music_table.delete(item)

    # Dapatkan genre yang dipilih
    selected_genre = self.filter_genre_var.get()

    # Filter dan tambahkan data ke tabel
    for music in self.music_data:
        if selected_genre == "Semua" or music["Genre"] == selected_genre:
            self.music_table.insert("", "end",
                                     values=(music["ID"], music["Judul"], music["Artis"], music["Album"], music["Genre"]))
```

Kode ini berfungsi untuk memperbarui tampilan tabel musik berdasarkan filter genre yang dipilih oleh pengguna. Jika genre yang dipilih adalah "Semua", maka semua data akan ditampilkan; jika tidak, hanya data dengan genre yang sesuai yang akan

ditampilkan. Data musik yang memenuhi kriteria tersebut ditambahkan ke tabel menggunakan insert() dengan menampilkan ID, judul, artis, album, dan genre pada kolom yang sesuai.

## 5. Delete

```
def delete_selected(self):
    selected_item = self.music_table.selection()
    if not selected_item:
        messagebox.showwarning("Error", "Silakan pilih baris untuk dihapus!")
        return
    confirm = messagebox.askquestion("Konfirmasi", "Yakin ingin menghapus lagu yang dipilih?")
    if confirm == "yes":
        for item in selected_item:
            values = self.music_table.item(item, "values")
            music_id = int(values[0])
            self.music_table.delete(item)
            self.music_data = [music for music in self.music_data if music["ID"] != music_id]

def delete_all(self):
    confirm = messagebox.askquestion("Konfirmasi", "Yakin ingin menghapus semua lagu?")
    if confirm == "yes":
        self.music_data.clear()
        self.update_table()
```

Kode ini berfungsi untuk menghapus data musik yang ada di tabel, baik yang dipilih secara individual maupun seluruhnya. Fungsi **delete\_selected** memungkinkan pengguna untuk memilih baris tertentu di tabel dan menghapusnya setelah konfirmasi. Sementara itu, fungsi **delete\_all** memungkinkan pengguna untuk menghapus semua data musik yang ada.

## 6. Simpan

```
def save_to_file(self):
    # Menyimpan data koleksi musik ke dalam file CSV
    file_path = filedialog.asksaveasfilename(defaultextension=".csv",
                                             filetypes=[("CSV files", "*.csv"), ("All files", "*.*)])
    if file_path:
        try:
            with open(file_path, "w", newline="", encoding="utf-8") as file:
                writer = csv.writer(file)
                writer.writerow(["ID", "Judul", "Artis", "Album", "Genre"])
                for music in self.music_data:
                    writer.writerow([music["ID"], music["Judul"], music["Artis"], music["Album"], music["Genre"]])
            messagebox.showinfo("Berhasil", f"Data berhasil disimpan ke file '{file_path}'")
        except Exception as e:
            messagebox.showerror("Error", f"Gagal menyimpan data: {e}")
```

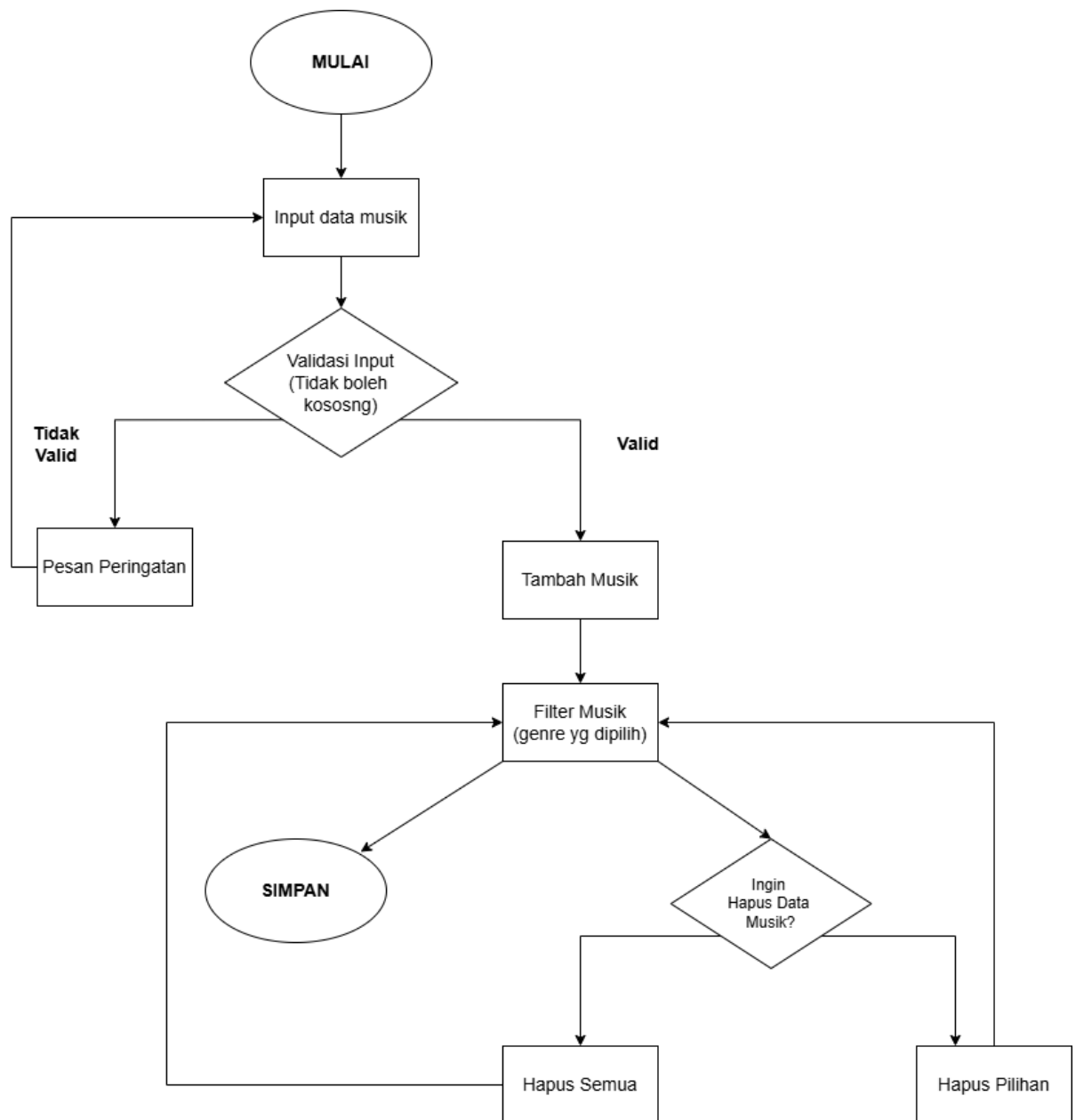
Kode ini berfungsi untuk menyimpan data koleksi musik ke dalam file CSV. Fungsi `save_to_file` akan meminta pengguna memilih lokasi dan nama file untuk menyimpan data menggunakan kotak dialog `filedialog.asksaveasfilename`.

## 7. Reset Input Form

```
def reset_input(self):  
    # Mengosongkan atau mereset semua input form  
    self.entry_title.delete(0, tk.END)  
    self.entry_artist.delete(0, tk.END)  
    self.entry_album.delete(0, tk.END)  
    self.combo_genre.set("Pilih Genre")
```

Tujuan utama dari fungsi ini adalah untuk membersihkan form setelah data diproses atau disimpan, sehingga pengguna dapat memulai input baru tanpa harus menghapus data secara manual.

## II. FLOWCHART





### III. HASIL PENGUJIAN

#### 1. Input Data Musik



**Aplikasi Pengelolaan Koleksi Musik**

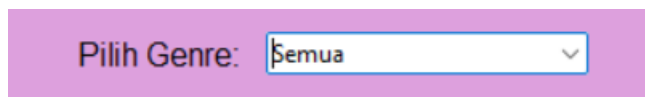
Judul Musik :  Album :

Nama Artis :  Genre :

**Tambah**

Pengguna diminta untuk menginput data lagu meliputi **judul, artis, album, dan genre**. Pada bagian **Genre** klik bagian pojok kanan, lalu pilih genre yang sesuai. Jika data lagu sudah benar dan tidak ada yang kosong maka klik **tombol tambah** yang ada di bawahnya.

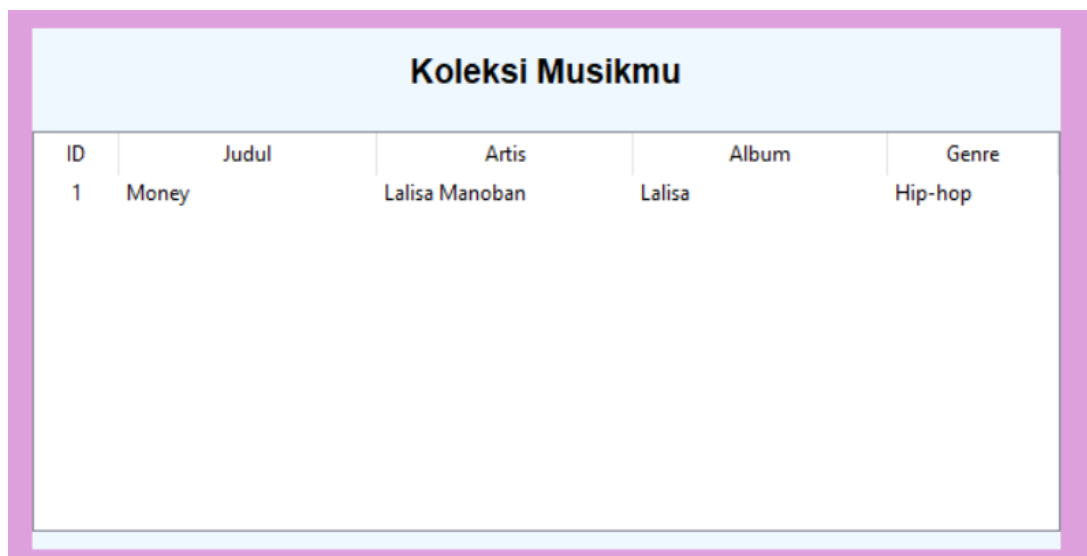
#### 2. Filter Genre



Pilih Genre:

Setelah itu pilih genre, terdapat opsi pilih genre semua, klasik, jazz, pop, R&B, hip-hop. Fitur ini menyediakan opsi untuk menyaring koleksi musik berdasarkan genre.

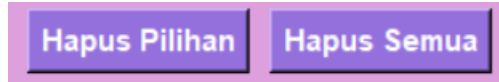
#### 3. Tabel Koleksi Musik



Koleksi Musikmu				
ID	Judul	Artis	Album	Genre
1	Money	Lalisa Manoban	Lalisa	Hip-hop

Lalu dapat terlihat pada table koleksi musikmu terdapat data yang telah ditambahkan sebelumnya.

#### 4. Hapus Data



Fitur ini digunakan untuk menghapus lagu yang dipilih atau menghapus semua data.

#### 5. Simpan Data



Setelah semua data selesai ditambahkan, klik simpan untuk menyimpan koleksi musik ke dalam file CSV untuk penyimpanan lebih lanjut

#### IV. TAMPILAN APLIKASI

### Aplikasi Pengelolaan Koleksi Musik

Judul Musik :

Album :

Nama Artis :

Genre : 

Pilih Genre

Tambah

Pilih Genre: 

Semua

Koleksi Musikmu				
ID	Judul	Artis	Album	Genre

Hapus Pilihan

Hapus Semua

Simpan

### BAB III

### CODE

```
import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

from tkinter import filedialog

import csv


class AppMusic:

    def __init__(self, root):

        self.root = root

        self.root.title("Aplikasi Pengelolaan Koleksi Musik")

        self.root.geometry("600x750") #Lebar dan Tinggi ukuran awal jendela

        self.root.config(bg="#DDA0DD")


        # Data musik

        self.music_data = []


        # Memanggil metode untuk membuat widget

        self.widget_create()


    def widget_create(self):

        # Title Label

        title_label = tk.Label(self.root, text="Aplikasi Pengelolaan
Koleksi Musik",

                                font=("Times New Roman", 24, 'bold'),

                                fg="#333", bg="#DDA0DD")

        title_label.pack(pady=40)
```

```

# Frame Input

frame_input = tk.Frame(self.root, bg="#DDA0DD")

frame_input.pack(pady=10)

# Judul Musik

judul_label = tk.Label(frame_input, text="Judul Musik :",
bg="#DDA0DD", font=("Arial", 10))

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

self.entry_title = tk.Entry(frame_input, width=25)

self.entry_title.grid(row=0, column=1, padx=10, pady=5)

self.entry_title.bind("<Return>", lambda event:
self.entry_artist.focus())

# Nama Artis

artis_label = tk.Label(frame_input, text="Nama Artis :",
bg="#DDA0DD", font=("Arial", 10))

artis_label.grid(row=1, column=0, padx=10, pady=5, sticky="w")

self.entry_artist = tk.Entry(frame_input, width=25)

self.entry_artist.grid(row=1, column=1, padx=10, pady=5)

self.entry_artist.bind("<Return>", lambda event:
self.entry_album.focus())

# Album

album_label = tk.Label(frame_input, text="Album :", bg="#DDA0DD",
font=("Arial", 10))

album_label.grid(row=0, column=2, padx=10, pady=5, sticky="w")

self.entry_album = tk.Entry(frame_input, width=25)

self.entry_album.grid(row=0, column=3, padx=10, pady=5)

self.entry_album.bind("<Return>", lambda event:
self.combo_genre.focus())

```

```

        # Genre

        genre_label = tk.Label(frame_input, text="Genre :", bg="#DDA0DD",
font=("Arial", 10))

        genre_label.grid(row=1, column=2, padx=10, pady=5, sticky="w")

        self.genre_var = tk.StringVar()

        self.combo_genre = ttk.Combobox(frame_input,
textvariable=self.genre_var, width=23)

        self.combo_genre['values'] = ["Klasik", "Jazz", "Pop", "R&B", "Hip-
hop"]

        self.combo_genre.set("Pilih Genre")

        self.combo_genre.grid(row=1, column=3, padx=10, pady=5)


        # Tombol Tambah Musik

        add_button = tk.Button(self.root, text="Tambah", font=("Arial", 11,
'bold'), bg="#9370DB", fg="white", command=self.add_music)

        add_button.pack(pady=10)


        # Filter Genre

        filter_frame = tk.Frame(self.root, bg="#DDA0DD")

        filter_frame.pack(pady=10)

        filter_label = tk.Label(filter_frame, text="Pilih Genre:",
bg="#DDA0DD", font=("Arial", 12))

        filter_label.pack(side=tk.LEFT, padx=5)

        self.filter_genre_var = tk.StringVar()

        self.filter_combobox = ttk.Combobox(filter_frame,
textvariable=self.filter_genre_var, width=23)

```

```

        self.filter_combobox['values'] = ["Semua"] + ["Klasik", "Jazz",
"Pop", "R&B", "Hip-hop"]

        self.filter_combobox.set("Semua")

        self.filter_combobox.pack(side=tk.LEFT, padx=5)

        self.filter_combobox.bind("<<ComboboxSelected>>",
self.update_table)


        # Frame untuk Tabel

        frame_table = tk.Frame(self.root, bg="#F0F8FF")

        frame_table.pack(pady=10)


        # Table musik

        music_table_label = tk.Label(frame_table, text="Koleksi Musikmu",
bg="#F0F8FF", font=("Arial", 14, "bold"))

        music_table_label.pack(pady=10)


        self.music_table = ttk.Treeview(frame_table, columns=("ID",
"Judul", "Artis", "Album", "Genre"), show="headings")

        self.music_table.heading("ID", text="ID")

        self.music_table.heading("Judul", text="Judul")

        self.music_table.heading("Artis", text="Artis")

        self.music_table.heading("Album", text="Album")

        self.music_table.heading("Genre", text="Genre")

        self.music_table.column("ID", width=50, anchor="center")

        self.music_table.column("Judul", width=150)

        self.music_table.column("Artis", width=150)

        self.music_table.column("Album", width=150)

        self.music_table.column("Genre", width=100)

        self.music_table.pack(pady=10)

```

```

button_frame = tk.Frame(self.root, bg="#DDA0DD")

button_frame.pack(pady=5)

# Tombol Hapus Pilihan

delete_button = tk.Button(button_frame, text="Hapus Pilihan",
font=("Arial", 11, 'bold'), bg="#9370DB", fg="white",
command=self.delete_selected)

delete_button.pack(side=tk.LEFT, padx=5)

# Tombol Hapus Semua

delete_all_button = tk.Button(button_frame, text="Hapus Semua",
font=("Arial", 11, 'bold'), bg="#9370DB", fg="white",
command=self.delete_all)

delete_all_button.pack(side=tk.LEFT, padx=5)

# Tombol Simpan Data

save_button = tk.Button(self.root, text="Simpan", font=("Arial",
11, 'bold'), bg="#9370DB", fg="white", command=self.save_to_file)

save_button.pack(pady=10)

def add_music(self):

    title = self.entry_title.get()

    artist = self.entry_artist.get()

    album = self.entry_album.get()

    genre = self.genre_var.get()

    # Validasi input

    if not title or not artist or not album or genre == "Pilih Genre":

```



```

        messagebox.showwarning("Peringatan", "Semua kolom harus
diisi!") # Jika ada kolom yang kosong, muncul pesan peringatan

        return

    # Menambahkan data ke dalam list musik

    music_id = len(self.music_data) + 1

    self.music_data.append({"ID": music_id, "Judul": title, "Artis":
artist, "Album": album, "Genre": genre})

    self.update_table()

    self.reset_input()

def update_table(self, event=None):

    # Hapus semua data di tabel

    for item in self.music_table.get_children():

        self.music_table.delete(item)

    # Dapatkan genre yang dipilih

    selected_genre = self.filter_genre_var.get()

    # Filter dan tambahkan data ke tabel

    for music in self.music_data:

        if selected_genre == "Semua" or music["Genre"] ==
selected_genre:

            self.music_table.insert("", "end",

                                     values=(music["ID"],
music["Judul"], music["Artis"], music["Album"], music["Genre"]))

def delete_selected(self):

    selected_item = self.music_table.selection()

```

```

        if not selected_item:

            messagebox.showwarning("Error", "Silakan pilih baris untuk
dihapus!")

            return

        confirm = messagebox.askquestion("Konfirmasi", "Yakin ingin
menghapus lagu yang dipilih?")

        if confirm == "yes":

            for item in selected_item:

                values = self.music_table.item(item, "values")

                music_id = int(values[0])

                self.music_table.delete(item)

                self.music_data = [music for music in self.music_data if
music["ID"] != music_id]

    def delete_all(self):

        confirm = messagebox.askquestion("Konfirmasi", "Yakin ingin
menghapus semua lagu?")

        if confirm == "yes":

            self.music_data.clear()

            self.update_table()

    def save_to_file(self):

        # Menyimpan data koleksi musik ke dalam file CSV

        file_path = filedialog.asksaveasfilename(defaultextension=".csv",

                                                    filetypes=[("CSV files",
".*.csv"), ("All files", ".*")])

        if file_path:

            try:

                with open(file_path, "w", newline="", encoding="utf-8") as
file:

```

```

        writer = csv.writer(file)

        writer.writerow(["ID", "Judul", "Artis", "Album",
"Genre"])

        for music in self.music_data:

            writer.writerow([music["ID"], music["Judul"],
music["Artis"], music["Album"], music["Genre"]])

            messagebox.showinfo("Berhasil", f"Data berhasil disimpan ke
file '{file_path}'")

        except Exception as e:

            messagebox.showerror("Error", f"Gagal menyimpan data: {e}")

    def reset_input(self):

        # Mengosongkan atau mereset semua input form

        self.entry_title.delete(0, tk.END)

        self.entry_artist.delete(0, tk.END)

        self.entry_album.delete(0, tk.END)

        self.combo_genre.set("Pilih Genre")

if __name__ == "__main__":

    root = tk.Tk()

    app = AppMusic(root)

    root.mainloop()

```

## **BAB IV**

### **PENUTUP**

#### **A. KESIMPULAN**

Aplikasi Pengelolaan Koleksi Musik membantu pengguna mengatur koleksi musik secara efisien dengan fitur sederhana namun bermanfaat. Aplikasi ini menyediakan fitur untuk menambahkan, melihat, menyaring berdasarkan genre, dan menghapus data musik dengan antarmuka yang sederhana dan mudah digunakan.

Validasi input memastikan semua kolom diisi dengan benar sebelum data ditambahkan ke tabel koleksi, sementara filter genre mempermudah pengguna dalam menampilkan data sesuai preferensi. Secara keseluruhan, aplikasi ini dapat berfungsi dengan baik sesuai tujuan awal dan memiliki potensi untuk dikembangkan lebih lanjut, terutama dengan integrasi database dan fitur tambahan lainnya.