TUGAS 8

PEMROGRAMAN BERORIENTASI OBJEK

“Format Apk PREMIUM”

Di Susun Oleh:

ELSA AMBARWATI (5230411238)

Program Studi Informatika

Fakultas Sains dan Teknologi

Universitas Teknologi Yogyakarta

Tahun Akademik

2024/2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aplikasi ini dirancang untuk memenuhi kebutuhan pengguna dalam melakukan pemesanan layanan aplikasi premium, seperti Netflix dan Canva. Dalam era digital saat ini, banyak pengguna yang mencari cara yang efisien dan mudah untuk mengakses layanan langganan. Dengan adanya aplikasi ini, pengguna dapat dengan cepat dan praktis mengisi informasi yang diperlukan untuk melakukan pemesanan.

Aplikasi ini menggunakan Tkinter, pustaka GUI untuk Python, yang memungkinkan pengembangan antarmuka pengguna yang intuitif dan responsif. Melalui aplikasi ini, pengguna dapat memilih jenis aplikasi, menentukan durasi langganan, dan melakukan pembayaran dengan mudah. Selain itu, aplikasi ini dilengkapi dengan fitur validasi input untuk memastikan bahwa semua informasi yang diberikan adalah akurat dan sesuai dengan format yang diharapkan.

Dengan adanya fitur untuk menampilkan daftar pesanan, pengguna dapat dengan mudah melacak dan mengelola pesanan mereka. Aplikasi ini tidak hanya bertujuan untuk mempermudah proses pemesanan, tetapi juga untuk memberikan pengalaman pengguna yang menyenangkan dan efisien. Melalui aplikasi ini, diharapkan pengguna dapat lebih mudah mendapatkan akses ke layanan aplikasi premium yang mereka inginkan.

1.2 Tujuan dan Ruang Lingkup Laporan

Laporan ini bertujuan untuk mendokumentasikan pengembangan aplikasi pemesanan layanan aplikasi premium yang dirancang untuk memberikan kemudahan dan efisiensi bagi pengguna. Ruang lingkup laporan mencakup analisis kebutuhan pengguna, desain sistem, implementasi kode, pengujian aplikasi, serta dokumentasi yang diperlukan untuk pengguna dan pengembang. Dengan laporan ini, diharapkan dapat memberikan gambaran yang jelas tentang proses pengembangan aplikasi serta manfaat yang dapat diperoleh oleh pengguna.

1.3 Deskripsi Singkat Tentang Studi Kasus “Format Apk PREMIUM”

Aplikasi pemesanan layanan aplikasi premium ini merupakan sebuah platform berbasis GUI yang dibangun menggunakan Python dan Tkinter. Aplikasi ini memungkinkan pengguna untuk memilih layanan aplikasi premium, menentukan durasi langganan, dan melakukan pembayaran dengan mudah. Antarmuka pengguna dirancang sederhana dan intuitif, sehingga pengguna dari berbagai latar belakang dapat dengan cepat memahami cara penggunaannya. Selain itu, aplikasi ini dilengkapi dengan fitur validasi input untuk memastikan bahwa semua informasi yang diberikan adalah akurat, serta fitur untuk menampilkan daftar pesanan yang telah dilakukan. Dengan aplikasi ini, diharapkan pengguna dapat lebih mudah dan cepat dalam mengakses layanan langganan yang mereka inginkan.

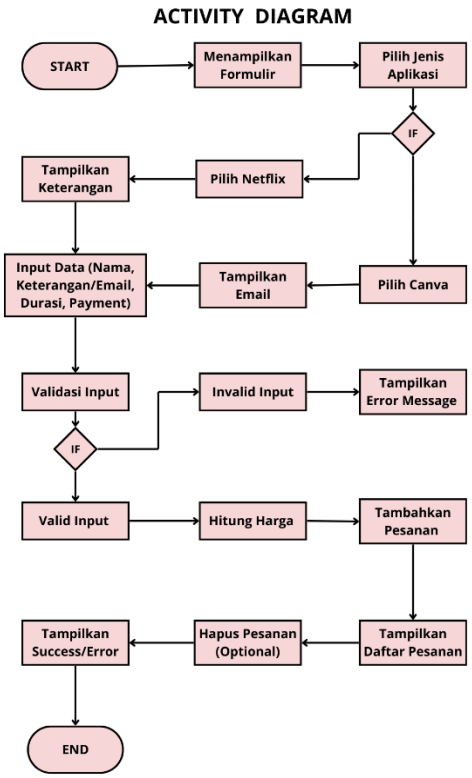
BAB II
PEMBAHASAN

2.1 Alur Kerja Format APK PREMIUM

Aplikasi **Format APK PREMIUM** memiliki alur kerja yang dirancang untuk memastikan kemudahan penggunaan, keakuratan data, dan efisiensi pengelolaan pesanan. Berikut adalah tahapan utama dalam alur kerja aplikasi:

1. **Pengisian Formulir Pemesanan**:
    - Pengguna diminta untuk mengisi nama, memilih jenis aplikasi, durasi, dan metode pembayaran.
    - Untuk aplikasi *Netflix*, pengguna harus memilih keterangan (*Shar* atau *Priv*).
    - Untuk aplikasi *Canva*, pengguna diminta mengisi alamat email yang valid (harus berformat @gmail.com).
2. **Validasi Input**:
    - Sistem memeriksa kelengkapan data.
    - Sistem juga memvalidasi email jika aplikasi yang dipilih adalah *Canva*.
    - Jika ada data yang tidak lengkap atau tidak valid, sistem akan memberikan pesan kesalahan.
3. **Perhitungan Harga Otomatis**:
    - Harga dihitung berdasarkan kombinasi jenis aplikasi, keterangan (khusus *Netflix*), dan durasi layanan.
    - Jika kombinasi tidak valid, seperti memilih durasi 1 tahun untuk *Netflix*, maka sistem akan memberikan peringatan.
4. **Pengelolaan Pesanan**:
    - Pesanan yang valid ditambahkan ke daftar pesanan dalam bentuk tabel.
    - Pengguna dapat menghapus pesanan tertentu dari daftar.
5. **Tampilan Informasi Harga dan Pesanan**:
    - Harga total untuk setiap pesanan ditampilkan di layar.
    - Daftar semua pesanan yang telah dimasukkan dapat dilihat, lengkap dengan rincian seperti nama, jenis aplikasi, keterangan/email, durasi, harga, dan metode pembayaran.

2.2 Activity Diagram



Penjelasan Alur:

1. **Start**: Pengguna membuka aplikasi.
2. **Menampilkan Formulir**: Tampilan formulir aplikasi muncul.
3. **Pilih Jenis Aplikasi**: Pengguna memilih jenis aplikasi (Netflix atau Canva).
    - Jika memilih **Netflix**: Menampilkan pilihan keterangan (Shar atau Priv).
    - Jika memilih **Canva**: Menampilkan input email.
4. **Input Data**: Pengguna mengisi semua data yang diperlukan (nama, jenis aplikasi, keterangan/email, durasi, pembayaran).
5. **Validasi Input**: Sistem memeriksa apakah semua kolom sudah terisi dengan benar.
    - Jika input **valid**, lanjut ke proses selanjutnya.
    - Jika **invalid**, tampilkan pesan kesalahan dan kembali ke input.
6. **Hitung Harga**: Berdasarkan input, sistem menghitung harga yang sesuai.
7. **Tambahkan Pesanan**: Jika semua input valid, pesanan ditambahkan ke daftar pesanan.
8. **Tampilkan Daftar Pesanan**: Daftar pesanan yang berhasil ditambahkan ditampilkan.
9. **Hapus Pesanan** (opsional): Pengguna dapat memilih dan menghapus pesanan.
10. **Tampilkan Success/Error**: Pesan sukses atau kesalahan ditampilkan kepada pengguna.

11. **End**: Proses selesai.

2.3 Implementasi Kode

1. **Memulai dengan Import dan Persiapan**

```
import tkinter as tk

from tkinter import ttk, messagebox

import re
```

Penjelasan:

- tkinter: Merupakan modul untuk membuat antarmuka pengguna grafis (GUI) di Python.
- ttk: Merupakan submodul dari tkinter yang menyediakan widget dengan tampilan yang lebih modern dan rapi.
- messagebox: Digunakan untuk menampilkan kotak pesan, seperti pesan kesalahan atau pemberitahuan kepada pengguna.
- re: Merupakan modul untuk bekerja dengan ekspresi reguler, di sini digunakan untuk validasi email.

2. **Membuat Kelas Aplikasi**

```
class FromApp:

    def __init__(self, root):

        self.root = root

        self.root.title("Format Apk PREMIUM")

        self.root.geometry("600x600")


        self.orders = []

        self.create_widget()
```

Penjelasan:

- Kelas FromApp adalah inti dari aplikasi ini. Saat aplikasi dijalankan, kelas ini yang akan mengatur alur tampilan dan fungsionalitas.
- self.root adalah objek utama dari jendela aplikasi (Tk()).
- self.orders digunakan untuk menyimpan daftar pesanan yang sudah dibuat.
- self.create_widget() memanggil metode untuk membuat widget GUI.

3. **Membuat Widget**

```
def create_widget(self):

        # Title label

        title_label = tk.Label(self.root, text="Format APK PREMIUM",
font=("Times New Roman", 15, 'bold'))
```

```
        title_label.pack(pady=5)


        # Nama Pemesan
        nama_label = tk.Label(self.root, text="Nama Pemesan:")
        nama_label.pack()
        self.nama_entry = tk.Entry(self.root, width=25)
        self.nama_entry.pack(pady=5)


        # Jenis Apk
        jenis_apk_label = tk.Label(self.root, text="Jenis Aplikasi:")
        jenis_apk_label.pack()
        self.jenis_var = tk.StringVar(value="")
        self.jenis_apk = ttk.Combobox(self.root, textvariable=self.jenis_var,
state="readonly", width=25)
        self.jenis_apk['values'] = ["Netflix", "Canva"]
        self.jenis_apk.pack(pady=5)
        self.jenis_apk.bind("<<ComboboxSelected>>", self.update_from)


        # Keterangan (Netflix)
        ket_label = tk.Label(self.root, text="Keterangan (Netflix):")
        ket_label.pack()
        self.ket_var = tk.StringVar(value="")
        self.ket_apk = ttk.Combobox(self.root, textvariable=self.ket_var,
state="readonly", width=25)
        self.ket_apk['values'] = ["Shar", "Priv"]
        self.ket_apk.pack(pady=5)


        # Email (Canva)
        email_label = tk.Label(self.root, text="Email (Canva):")
        email_label.pack()
        self.email_entry = tk.Entry(self.root, width=25)
        self.email_entry.pack(pady=5)
```

```
        # Durasi
        durasi_label = tk.Label(self.root, text="Durasi :")
        durasi_label.pack()
        self.durasi_var = tk.StringVar(value="")
        self.durasi_apk = ttk.Combobox(self.root,
textvariable=self.durasi_var, state="readonly", width=25)
        self.durasi_apk['values'] = ["1 Tahun", "1 Bulan", "1 Minggu", "1
Hari"]
        self.durasi_apk.pack(pady=5)

        # Payment
        payment_label = tk.Label(self.root, text="Payment :")
        payment_label.pack()
        self.payment_var = tk.StringVar(value="")
        self.payment_apk = ttk.Combobox(self.root,
textvariable=self.payment_var, state="readonly", width=25)
        self.payment_apk['values'] = ["BRI", "DANA"]
        self.payment_apk.pack(pady=5)

        # Harga
        self.price_label = tk.Label(self.root, text="Harga: Rp. 0",
font=("Times New Roman", 12, 'bold'))
        self.price_label.pack(pady=5)

        # Tombol kirim format
        self.kirim_button = tk.Button(self.root, text="Kirim format",
command=self.add_order)
        self.kirim_button.pack(pady=5)

        # Tombol hapus pesanan
        self.hapus_button = tk.Button(self.root, text="Hapus Pesanan",
command=self.delete_order)
```

```
        self.hapus_button.pack(pady=5)


        # Daftar Pesanan
        daftar_pesanan_label = tk.Label(self.root, text="Daftar Pesanan:",
font=("Times New Roman", 12, 'bold'))
        daftar_pesanan_label.pack()
        self.order_table = ttk.Treeview(self.root, columns=("Nama", "Jenis",
"Keterangan/Email", "Durasi", "Harga", "Payment"), show="headings",
selectmode="extended")
        self.order_table.heading("Nama", text="Nama")
        self.order_table.heading("Jenis", text="Jenis")
        self.order_table.heading("Keterangan/Email", text="Keterangan/Email")
        self.order_table.heading("Durasi", text="Durasi")
        self.order_table.heading("Harga", text="Harga")
        self.order_table.heading("Payment", text="Payment")


        self.order_table.column("Nama", width=100, anchor="center")
        self.order_table.column("Jenis", width=100, anchor="center")
        self.order_table.column("Keterangan/Email", width=150,
anchor="center")
        self.order_table.column("Durasi", width=100, anchor="center")
        self.order_table.column("Harga", width=100, anchor="center")
        self.order_table.column("Payment", width=100, anchor="center")
        self.order_table.pack(pady=5)


        # Trace variables
        self.jenis_var.trace_add("write", self.update_price)
        self.ket_var.trace_add("write", self.update_price)
        self.durasi_var.trace_add("write", self.update_price)
```

Penjelasan :

1) **Label dan Entry Boxes:**

- Label digunakan untuk memberikan petunjuk pada pengguna, seperti "Nama Pemesan" dan "Jenis Aplikasi", diikuti dengan kotak input (Entry) atau dropdown (Combobox) untuk memasukkan atau memilih data.
- **Entry Box** memungkinkan pengguna untuk memasukkan nama mereka.
- **Combobox** digunakan untuk memberikan pilihan kepada pengguna dalam bentuk dropdown, seperti jenis aplikasi atau durasi.

2) **Combobox untuk Pilihan Dropdown:**
   - **Jenis Aplikasi:**
     Combobox ini memungkinkan pengguna untuk memilih jenis aplikasi yang diinginkan, seperti Netflix atau Canva. Fungsi ini juga terhubung dengan fungsi yang mengubah elemen formulir lainnya sesuai pilihan.
   - **Keterangan untuk Netflix:**
     Ketika pengguna memilih Netflix, keterangan terkait seperti "Shar" dan "Priv" akan muncul di combobox ini.
   - **Email untuk Canva:**
     Ketika pengguna memilih Canva, maka kotak input untuk email akan muncul, memberikan tempat bagi pengguna untuk memasukkan email mereka.
   - **Durasi dan Payment:**
     Combobox untuk durasi berlangganan dan metode pembayaran akan memberikan opsi seperti durasi langganan (contohnya 1 tahun atau 1 bulan) dan pilihan pembayaran (misalnya BRI atau DANA).

3) **Label untuk Harga:**
   Label ini menunjukkan harga yang dihitung secara dinamis berdasarkan pilihan yang diambil oleh pengguna. Harga akan diperbarui setiap kali ada perubahan pilihan pada combobox.

4) **Tombol Kirim dan Hapus Pesanan:**
   - **Tombol Kirim Format:**
     Tombol ini digunakan untuk mengirim data pesanan yang telah dimasukkan oleh pengguna. Setelah tombol ini diklik, data akan ditambahkan ke dalam daftar pesanan.
   - **Tombol Hapus Pesanan:**
     Tombol ini memungkinkan pengguna untuk menghapus pesanan yang sudah ditambahkan dari daftar.

5) **Treeview untuk Daftar Pesanan:**
   - Treeview digunakan untuk menampilkan data pesanan yang telah dimasukkan. Setiap kolom diatur untuk menampilkan informasi tertentu seperti nama, jenis aplikasi, durasi, harga, dan metode pembayaran. Kolom-kolom ini ditata dengan lebar yang sesuai agar tampilan tetap rapi dan terorganisir.

- Setiap kolom memiliki nama yang ditampilkan di bagian atas, seperti "Nama", "Jenis", "Durasi", "Harga", dan "Payment". Semua kolom disusun dengan rata tengah agar tampilannya lebih terstruktur.

6) **Trace Variables:**
Variabel yang digunakan untuk memilih jenis aplikasi, durasi, atau keterangan memiliki fungsi **trace**. Fungsi ini memungkinkan aplikasi untuk merespons perubahan dalam nilai variabel secara otomatis, seperti menghitung ulang harga ketika ada perubahan pilihan. Fungsi yang dipanggil ketika nilai variabel berubah akan memperbarui harga dan elemen-elemen formulir lainnya secara dinamis.

7) **Interaktivitas dan Dinamika Aplikasi:**

- **Elemen Formulir Dinamis:**
Fungsi-fungsi seperti update_from() berfungsi untuk mengubah elemen formulir lainnya tergantung pada pilihan pengguna. Sebagai contoh, jika pengguna memilih "Canva", maka field untuk email akan ditampilkan, sementara keterangan untuk Netflix akan disembunyikan.

- **Pembaharuan Harga:**
Setiap kali pilihan pada formulir diubah, aplikasi akan secara otomatis memperbarui harga berdasarkan pilihan jenis aplikasi, durasi, dan metode pembayaran yang telah dipilih. Pembaruan ini membuat pengalaman pengguna menjadi lebih interaktif dan responsif.

4. **Fungsi untuk Mengubah Formulir**

```
def update_from(self, event=None):
        jenis = self.jenis_var.get()
        if jenis == "Netflix":
            self.ket_apk.config(state="normal")
            self.email_entry.config(state="disabled")
        elif jenis == "Canva":
            self.email_entry.config(state="normal")
            self.ket_apk.config(state="disabled")
```

Penjelasan :

Fungsi ini mengubah elemen form berdasarkan jenis aplikasi yang dipilih.

- Jika **Netflix** dipilih, Maka kolom keterangan akan aktif dan kolom email dinonaktifkan.
- Jika **Canva** dipilih, Maka kolom email akan aktif dan kolom keterangan dinonaktifkan.

5. **Fungsi untuk Menghitung Harga**

```
def calculate_price(self):
```

```
        jenis = self.jenis_var.get()
        ket = self.ket_var.get()
        durasi = self.durasi_var.get()

        if jenis == "Netflix":
            if durasi == "1 Bulan":
                return "Rp. 19.000" if ket == "Shar" else "Rp. 27.000"
            if durasi == "1 Minggu":
                return "Rp. 14.500" if ket == "Shar" else "Rp. 15.500"
            if durasi == "1 Hari":
                return "Rp. 7.000" if ket == "Shar" else "Rp. 10.000"
            if durasi == "1 Tahun":
                return "Durasi tidak tersedia."

        elif jenis == "Canva":
            if durasi == "1 Tahun":
                return "Rp. 15.000"
            if durasi == "1 Bulan":
                return "Rp. 10.000"
            if durasi == "1 Minggu":
                return "Rp. 7.000"
            if durasi == "1 Hari":
                return "Rp. 5.000"
```

Penjelasan :

- Fungsi ini menghitung harga berdasarkan jenis aplikasi (Netflix/Canva), keterangan (untuk Netflix), dan durasi layanan yang dipilih.

6. **Fungsi untuk Memperbarui Harga**

```
def update_price(self, *args):
        price = self.calculate_price()
        self.price_label.config(text=f"Harga: {price}")
```

- Fungsi ini memperbarui harga yang ditampilkan di antarmuka pengguna setiap kali ada perubahan pada input formulir.

7. **Fungsi untuk Validasi Email**

```
def validate_email(self, email):

        return re.match(r"^[a-zA-Z0-9_.+-]+@gmail\.com$", email)
```

Penjelasan :

- Fungsi ini digunakan untuk memvalidasi email yang dimasukkan oleh pengguna. Fungsi ini menerima parameter email dan menggunakan regular expression untuk memeriksa apakah format email tersebut sesuai dengan pola email yang valid, khususnya yang memiliki domain @gmail.com.
- Jika email sesuai dengan format yang diinginkan (dengan @gmail.com), fungsi ini akan mengembalikan hasil yang cocok (True), sebaliknya akan mengembalikan None (False).

8. **Validasi Inputan**

```
def add_order(self):

        nama = self.nama_entry.get()

        jenis = self.jenis_var.get()

        ket = self.ket_var.get()

        email = self.email_entry.get()

        durasi = self.durasi_var.get()

        payment = self.payment_var.get()
```

Penjelasan :

- nilai dari setiap input (misalnya, nama, jenis, durasi, dll.) diambil dari widget GUI untuk mempersiapkan data pesanan.

```
        if not nama or not jenis or not durasi or not payment or (jenis ==
"Netflix" and not ket) or (jenis == "Canva" and not email):

            messagebox.showerror("Kesalahan", "Silakan isi semua form!")

            return
```

Penjelasan :

- Validasi ini memastikan bahwa semua kolom wajib diisi. Ini termasuk memeriksa apakah kolom keterangan (untuk Netflix) atau email (untuk Canva) telah diisi jika aplikasi yang dipilih membutuhkan input tambahan tersebut.

```
if jenis == "Canva" and not self.validate_email(email):

            messagebox.showerror("Kesalahan", "Email Canva harus menggunakan
@gmail.com!")

            return
```

- Jika jenis aplikasi adalah "Canva", fungsi ini memeriksa apakah email yang dimasukkan valid (harus berakhiran @gmail.com). Jika tidak valid, maka akan muncul pesan kesalahan.

```
price = self.calculate_price()

        if "tidak tersedia" in price:

            messagebox.showerror("Kesalahan", "Durasi tidak tersedia untuk
jenis aplikasi ini!")

            return
```

- Setelah memanggil fungsi calculate_price(), kode ini memeriksa apakah durasi yang dipilih tersedia untuk jenis aplikasi yang dipilih. Jika tidak tersedia, akan ada pesan kesalahan.

```
order = (nama, jenis, ket if jenis == "Netflix" else email, durasi, price,
payment)

        self.orders.append(order)

        self.order_table.insert("", "end", values=order)

        messagebox.showinfo("Sukses", "Pesanan berhasil ditambahkan!")
```

- Jika semua validasi lulus, pesanan yang valid akan dibuat dalam bentuk tuple yang berisi informasi pesanan (nama, jenis, keterangan/email, durasi, harga, dan metode pembayaran).
- Pesanan kemudian ditambahkan ke dalam list self.orders untuk disimpan, dan juga dimasukkan ke dalam Treeview (tabel) untuk ditampilkan di GUI.
- Pesan konfirmasi ditampilkan menggunakan messagebox.showinfo().

9. **menghapus pesanan**

```
def delete_order(self):

        selected_items = self.order_table.selection()

        if not selected_items:

            messagebox.showwarning("Peringatan", "Pilih pesanan yang ingin
dihapus!")

            return

        for item in selected_items:

            self.order_table.delete(item)

        messagebox.showinfo("Sukses", "Pesanan berhasil dihapus!")
```

Penjelasan :

- Di sini, selection() digunakan untuk mendapatkan ID item yang dipilih oleh pengguna di dalam Treeview. Fungsi ini mengembalikan sebuah tuple yang berisi ID dari item yang dipilih. Jika tidak ada item yang dipilih, tuple ini akan kosong.
- Jika pengguna tidak memilih item apapun (artinya selected_items kosong), maka aplikasi akan menampilkan pesan peringatan dengan menggunakan messagebox.showwarning(). Fungsi ini kemudian dihentikan dengan return, sehingga tidak ada tindakan lebih lanjut yang dilakukan.
- Setelah pesanan berhasil dihapus, aplikasi akan menampilkan pesan konfirmasi dengan menggunakan messagebox.showinfo(), memberi tahu pengguna bahwa pesanan telah berhasil dihapus.

10. **membangun antarmuka pengguna grafis (GUI)**

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

Penjelasan :

1) **if __name__ == "__main__":**
   - Baris ini memastikan bahwa kode yang ada di bawahnya hanya akan dijalankan jika file ini dijalankan sebagai skrip utama (bukan diimpor sebagai modul oleh skrip lain).
   - Ketika file Python dijalankan, Python akan mengeksekusi kode di dalam blok ini.
2) **root = tk.Tk()**
   - tk.Tk() adalah konstruktor untuk membuat objek **root window** (jendela utama) aplikasi GUI.
   - root adalah jendela utama aplikasi yang akan menampung semua elemen GUI seperti tombol, label, dan lainnya.
3) **app = FromApp(root)**
   - Di sini, objek app adalah instansiasi dari kelas FromApp, yang merupakan kelas utama aplikasi Anda.
   - root (jendela utama) diteruskan sebagai parameter ke konstruktor FromApp, sehingga aplikasi bisa mengakses jendela utama untuk menambahkan widget dan mengatur layout.
4) **root.mainloop()**
   - root.mainloop() adalah perintah untuk menjalankan loop utama aplikasi Tkinter.

- Fungsi ini akan terus berjalan dan menunggu interaksi pengguna, seperti klik tombol atau pengisian form. Selama aplikasi berjalan, event seperti klik tombol atau perubahan input akan diproses.
- Program akan terus berjalan di dalam mainloop() ini hingga pengguna menutup aplikasi.

2.3 Hasil Akhir

Hasil akhir dari program ini adalah sebuah aplikasi desktop berbasis Tkinter dengan fitur berikut:

1. **Input Form: Pengguna dapat mengisi data pesanan, seperti:**
   - Nama pemesan.
   - Jenis aplikasi (Netflix atau Canva).
   - Detail tambahan: Keterangan (Shar/Priv) untuk Netflix atau email untuk Canva.
   - Durasi langganan (1 Hari, 1 Minggu, 1 Bulan, atau 1 Tahun).
   - Metode pembayaran (BRI atau DANA).
2. **Penentuan Harga Otomatis:**
   - Harga langganan dihitung dan ditampilkan secara dinamis berdasarkan jenis aplikasi, keterangan, dan durasi yang dipilih.
3. **Daftar Pesanan:**
   Pesanan yang ditambahkan oleh pengguna akan ditampilkan dalam tabel dengan rincian:
   - Nama pemesan.
   - Jenis aplikasi.
   - Keterangan atau email.
   - Durasi.
   - Harga.
   - Metode pembayaran.
4. **Fitur Tambah dan Hapus Pesanan:**
   - Pesanan dapat ditambahkan ke tabel jika data valid.
   - Pesanan yang sudah ditambahkan dapat dihapus dari tabel.

5. **Validasi Input:**
    - Email untuk Canva harus berupa format Gmail.
    - Semua form wajib diisi, dan durasi tertentu tidak tersedia untuk aplikasi tertentu (misalnya, 1 Tahun untuk Netflix).

BAB III
PENUTUP

3.1 Kesimpulan

Aplikasi Format APK PREMIUM adalah sebuah sistem pengelolaan pesanan sederhana berbasis GUI yang mempermudah pengguna dalam mengatur dan mencatat pesanan layanan aplikasi premium seperti Netflix dan Canva. Dengan fitur seperti kalkulasi harga otomatis, validasi input, dan pengelolaan daftar pesanan, aplikasi ini memberikan pengalaman yang interaktif dan user-friendly. Meskipun sederhana, aplikasi ini cukup fungsional untuk mendukung kebutuhan simulasi pemrosesan pesanan dalam skala kecil.

Namun, aplikasi ini dapat dikembangkan lebih lanjut dengan menambahkan fitur seperti penyimpanan data permanen, kemampuan mengedit pesanan, dan peningkatan antarmuka untuk pengalaman pengguna yang lebih baik.