

LAPORAN UJIAN AKHIR SEMESTER
APLIKASI MARKETPLACE DIGITAL PRINTING BERBASIS
OOP (OBJECT ORIENTED PROGRAMMING) DAN GUI
TKINTER

MATA KULIAH
PEMROGRAMAN BERBASIS OBJEK



Oleh
Pramudya Danish Ersyandi
NIM 24091397134
2024D

PROGRAM STUDI MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
TAHUN 2025

DAFTAR ISI

| | |
|--|-----------|
| DAFTAR ISI..... | ii |
| DAFTAR GAMBAR..... | iv |
| BAB I..... | 1 |
| PENDAHULUAN | 1 |
| 1.1 LATAR BELAKANG | 1 |
| 1.2 TUJUAN..... | 1 |
| 1.3 MANFAAT | 2 |
| BAB II | 3 |
| LANDASAN TEORI | 3 |
| 2.1 OBJECT ORIENTED PROGRAMMING | 3 |
| 2.1.1 Encapsulation | 3 |
| 2.1.2 Inheritance..... | 3 |
| 2.1.3 Polymorphism | 3 |
| 2.2 PEMROGRAMAN PYTHON | 3 |
| 2.2.1 Penggunaan Pemrograman Python..... | 4 |
| 2.2.2 Kelebihan dan Kekurangan Pemrograman Python | 4 |
| 2.3 TKINTER LIBRARY GUI | 4 |
| 2.4 MARKETPLACE DIGITAL PRINTING | 5 |
| 2.4.1 Produk Digital Printing..... | 5 |
| 2.4.2 Sistem Keranjang Belanja..... | 6 |
| 2.4.3 Sistem Perhitungan Harga Diskon | 6 |
| 2.4.4 Antarmuka Pengguna (User Interface) | 6 |
| BAB III..... | 7 |
| PERANCANGAN SISTEM | 7 |
| 3.1 DESKRIPSI SISTEM | 7 |
| 3.2 CLASS DESIGN..... | 7 |
| 3.3 FLOWCHART SISTEM | 8 |
| BAB IV..... | 10 |
| IMPLEMENTASI PROGRAM..... | 10 |
| 4.1 LINGKUP PENGEMBANGAN | 10 |
| 4.2 STRUKTUR PROGRAM..... | 10 |
| 4.3 IMPLEMENTASI KODE PROGRAM | 11 |

| | |
|---|-----------|
| BAB V | 22 |
| PENGUJIAN SISTEM | 22 |
| 5.1 TUJUAN PENGUJIAN | 22 |
| 5.2 METODE PENGUJIAN..... | 22 |
| 5.3 SKENARIO PENGUJIAN | 22 |
| 5.4 DOKUMENTASI PENGUJIAN..... | 23 |
| 5.5 ANALISIS HASIL PENGUJIAN | 25 |
| BAB VI..... | 26 |
| PENUTUP..... | 26 |
| 6.1 KESIMPULAN..... | 26 |
| 6.2 SARAN | 26 |
| DAFTAR PUSTAKA..... | 27 |

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 2. 1 Python | 3 |
| Gambar 3. 1 Class Diagram | 7 |
| Gambar 3. 2 Flowchart Sistem..... | 8 |
| Gambar 5. 1 Tampilan Awal | 23 |
| Gambar 5. 2 Menambahkan produk ke keranjang | 24 |
| Gambar 5. 3 Menghapus Produk | 24 |
| Gambar 5. 4 Harus memilih Item di keranjang..... | 24 |
| Gambar 5. 5 Checkout Produk..... | 25 |

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Dalam era transformasi digital saat ini, pemrograman menjadi salah satu keterampilan penting yang digunakan untuk membangun sistem informasi yang efektif dan efisien. Berbagai sektor mulai dari pendidikan, bisnis, hingga industri kreatif memerlukan aplikasi yang mampu membantu proses pengelolaan data dan pelayanan kepada pengguna. Oleh karena itu, pemahaman mengenai konsep pemrograman, khususnya Pemrograman Berorientasi Objek (Object Oriented Programming), sangat diperlukan sebagai dasar membangun aplikasi modern. Sangat diperlukan sebagai dasar membangun aplikasi modern.

Namun, masih banyak mahasiswa atau pengguna yang belum memahami bagaimana konsep OOP dapat diterapkan dalam aplikasi nyata, terutama dalam pembuatan aplikasi berbasis GUI. Selain itu, contoh aplikasi sederhana yang intuitif, menarik, dan mudah digunakan juga masih terbatas, padahal aplikasi seperti ini sangat dibutuhkan untuk memahami alur kerja sebuah sistem berbasis objek.

Untuk menjawab kebutuhan tersebut, maka dibutuhkan nya merancangan dan membangun sebuah Aplikasi Marketplace Digital Printing yang menerapkan konsep OOP secara lengkap, yaitu encapsulation, inheritance, dan polymorphism. Aplikasi ini menyediakan fitur menampilkan produk digital printing, menambahkan produk ke keranjang belanja, menghitung total harga otomatis, serta menampilkan antarmuka pengguna (GUI) yang menarik dan professional. Proyek ini disusun sebagai sarana untuk mengaplikasikan materi yang telah dipelajari selama perkuliahan, khususnya pada penerapan konsep OOP dalam pemrograman Python. Dengan mengembangkan aplikasi ini, mahasiswa dapat mengevaluasi sejauh mana pemahaman mereka terhadap desain kelas, relasi objek, serta penerapan metode yang efektif dalam membangun sebuah aplikasi.

1.2 TUJUAN

Tujuan dari pembuatan aplikasi ini sebagai berikut :

1. Menerapkan konsep OOP ke dalam program secara nyata.
2. Mengimplementasikan GUI menggunakan Tkinter agar aplikasi mudah digunakan.
3. Membuat sistem marketplace sederhana dengan keranjang belanja.

4. Menyusun dokumentasi proyek yang menjelaskan perancangan, konsep OOP serta implementasi aplikasi.

1.3 MANFAAT

Pembuatan aplikasi MarketPlace Digital Printing ini memberikan beberapa manfaat, antara lain:

1. Menambah pemahaman mahasiswa terhadap konsep Pemrograman Berorientasi Objek (OOP) melalui penerapan langsung pada aplikasi yang memiliki struktur kelas.
2. Meningkatkan kemampuan dalam mengembangkan antarmuka pengguna (GUI) menggunakan TKinter.
3. Memberikan pengalaman dalam merancang sistem marketplace sederhana, termasuk pengelolaan produk, keranjang belanja, dan perhitungan total harga.
4. Melatih keterampilan dokumentasi teknis, mulai dari perancangan class diagram, flowchart, hingga penyusunan laporan proyek yang terstruktur.
5. Menjadi dasar untuk pengembangan aplikasi yang lebih kompleks seperti sistem pemesanan makanan, manajemen inventori atau aplikasi e-commerce lainnya.

BAB II

LANDASAN TEORI

2.1 OBJECT ORIENTED PROGRAMMING

Pemrograman berorientasi objek merupakan metode pengembangan perangkat lunak yang didasarkan pada konsep “objek”. Objek di sini merepresentasikan entitas nyata yang memiliki atribut (data) dan perilaku (fungsi/metode). Paradigma ini mendorong pengembang untuk memodelkan komponen program berdasarkan bagaimana objek tersebut bekerja di dunia nyata. Konsep utama dalam OOP meliputi sebagai berikut :

2.1.1 Encapsulation

Konsep ini mengacu pada pembungkusan data dan metode dalam satu entitas (objek) sehingga detail internalnya tersembunyi dari pengguna lain. Dengan enkapsulasi, akses ke data dapat dikontrol melalui metode khusus, menjaga keamanan dan konsistensi data dalam sistem.

2.1.2 Inheritance

Pewarisan memungkinkan suatu kelas untuk mewarisi properti dan metode dari kelas lain. Dengan konsep ini, pengembang dapat membuat hierarki kelas yang lebih terstruktur, mengurangi redundansi kode, dan meningkatkan efisiensi dalam pengembangan perangkat lunak.

2.1.3 Polymorphism

Polimorfisme memungkinkan objek yang berbeda untuk merespons metode yang sama dengan cara yang berbeda. Ini memberikan fleksibilitas dalam pengembangan kode, di mana satu fungsi atau metode dapat digunakan untuk berbagai jenis objek tanpa mengubah strukturnya secara mendasar.

2.2 PEMROGRAMAN PYTHON



Gambar 2. 1 Python

Python merupakan bahasa pemrograman komputer yang biasa dipakai untuk membangun situs, software/aplikasi, mengotomatiskan tugas dan melakukan analisis data. Bahasa pemrograman

ini termasuk bahasa tujuan umum. Artinya, ia bisa digunakan untuk membuat berbagai program berbeda, bukan khusus untuk masalah tertentu saja. Karena sifatnya yang serba guna dan mudah digunakan, ia menjadi bahasa pemrograman yang paling banyak digunakan. Terutama untuk mereka yang masih pemula. Selain itu, Penggunaan Pemrograman Python meliputi sebagai berikut :

2.2.1 Penggunaan Pemrograman Python

Python biasa dipakai dalam pengembangan situs dan perangkat lunak, membuat analisis data, visualisasi data dan otomatisasi tugas. Karena sifatnya yang relatif mudah dipelajari, bahasa pemrograman ini digunakan secara luas oleh non-programmer seperti ilmuwan dan akuntan untuk melakukan tugas harian mereka. Misalnya sebagai berikut :

1. Data Analyst dan Machine Learning.
2. Pengembangan Web (Web Development).
3. Otomatisasi Tugas.
4. Pengembangan Kecerdasan Buatan (Artificial Intelligence).

2.2.2 Kelebihan dan Kekurangan Pemrograman Python

Python adalah salah satu bahasa pemrograman yang paling populer di dunia saat ini. Dengan sintaks yang bersih dan sederhana, Python telah menarik perhatian banyak pengembang, ilmuwan data, dan profesional TI. Namun, seperti bahasa pemrograman lainnya, Python juga memiliki kelebihan dan kekurangan. Kelebihan Pemrograman Python sebagai berikut :

1. Sintaks yang sederhana dan mudah dipahami.
2. Pustaka Library yang kaya.
3. Fleksibilitas.
4. Platform Independen.
5. Interaktif dan Dinamis.
6. Dukungan untuk OOP (Object Oriented Programming).

Kekurangan Pemrograman Python meliputi sebagai berikut :

1. Performa yang lebih lambat.
2. Penggunaan memori yang tinggi.
3. Keterbatasan di Mobile Computing.

2.3 TKINTER LIBRARY GUI

Tkinter adalah kerangka kerja GUI standar Python, yang memudahkan pengembangan antarmuka pengguna grafis (GUI). Sebagai pustaka lintas platform, Tkinter memastikan aplikasi Anda tampak asli di Windows, macOS, dan Linux. Meskipun dikritik karena

tampilannya yang ketinggalan zaman, Tkinter tetap menjadi pilihan praktis untuk membuat aplikasi GUI yang fungsional dan lintas platform dengan cepat. Tkinter merupakan Pustaka grafis yang memberikan kemudahan dalam pembuatan program berbasis grafis. Setiap GUI Toolkit menyediakan widget, yaitu objek user interface seperti button, scrollbar, listbox, checkbutton, radiobutton, label text dan lain sebagainya. Widget mengkapsulasi detil implementasi dan untuk setiap widget telah di definisikan perilaku defaultnya sehingga mempermudah pemrograman GUI. Keunggulan Tkinter antara lain:

1. Mudah digunakan dan tidak membutuhkan instalasi tambahan.
2. Mendukung berbagai komponen GUI seperti Frame, Label, Button, dan Listbox.
3. Dapat dikombinasikan dengan OOP untuk membuat aplikasi yang modular.
4. Ringan dan kompatibel di berbagai sistem operasi.

Dalam aplikasi marketplace digital printing, Tkinter digunakan untuk membuat tampilan daftar produk, keranjang belanja, tombol aksi, dan total harga, sehingga pengguna dapat berinteraksi dengan sistem secara visual.

2.4 MARKETPLACE DIGITAL PRINTING

Marketplace digital printing merupakan sebuah platform yang menyediakan layanan cetak digital berbasis komputer, seperti banner, stiker, foto, merchandise, dan berbagai kebutuhan desain lainnya. Layanan digital printing semakin banyak digunakan karena mampu menyediakan hasil cetak dengan kualitas tinggi, proses cepat, dan fleksibilitas ukuran serta desain. Pada sistem marketplace, pengguna dapat memilih produk, menyesuaikan kebutuhan, serta melakukan pemesanan secara mandiri melalui antarmuka yang disediakan. Secara umum, marketplace digital printing memiliki beberapa elemen utama :

2.4.1 Produk Digital Printing

Produk digital printing mencakup berbagai layanan cetak seperti:

1. Banner atau spanduk dengan berbagai ukuran.
2. Cetak foto (4R, 10R, 20R, dll).
3. Merchandise seperti mug custom, kaos sablon, dan stiker. Setiap produk memiliki harga, kategori, dan karakteristik tertentu. Dalam aplikasi, produk direpresentasikan sebagai objek yang memiliki atribut dan perilaku sesuai prinsip OOP.

2.4.2 Sistem Keranjang Belanja

Keranjang belanja merupakan fitur penting dalam sistem marketplace karena memungkinkan pengguna:

1. Menambahkan produk secara bertahap.
 2. Melihat daftar item yang dipilih.
 3. Menghapus item tertentu.
 4. Menghitung total biaya pesanan.
- Pada aplikasi ini, keranjang belanja direpresentasikan sebagai sebuah kelas (*Cart*) yang menyimpan kumpulan objek produk dalam sebuah list.

2.4.3 Sistem Perhitungan Harga Diskon

Marketplace digital printing sering menyediakan harga khusus atau diskon untuk kategori tertentu. Sistem perhitungan harga:

1. Menjumlahkan harga seluruh produk dalam keranjang.
2. Menerapkan diskon berdasarkan kategori produk.
3. Menghasilkan total akhir secara otomatis.

Penerapan diskon ini memanfaatkan konsep *method overriding* dalam OOP, di mana setiap jenis produk memiliki metode `apply_discount()` yang berbeda.

2.4.4 Antarmuka Pengguna (User Interface)

Agar marketplace mudah digunakan, diperlukan tampilan antarmuka yang jelas dan informatif. Dalam aplikasi ini, Tkinter digunakan untuk menampilkan:

1. Daftar produk dalam bentuk list.
2. Tombol untuk menambah atau menghapus produk.
3. Keranjang belanja.
4. Total harga secara real-time.

Desain antarmuka disusun agar menyerupai tampilan marketplace modern dengan layout terstruktur, efek warna profesional, dan shadow untuk meningkatkan estetika aplikasi.

BAB III

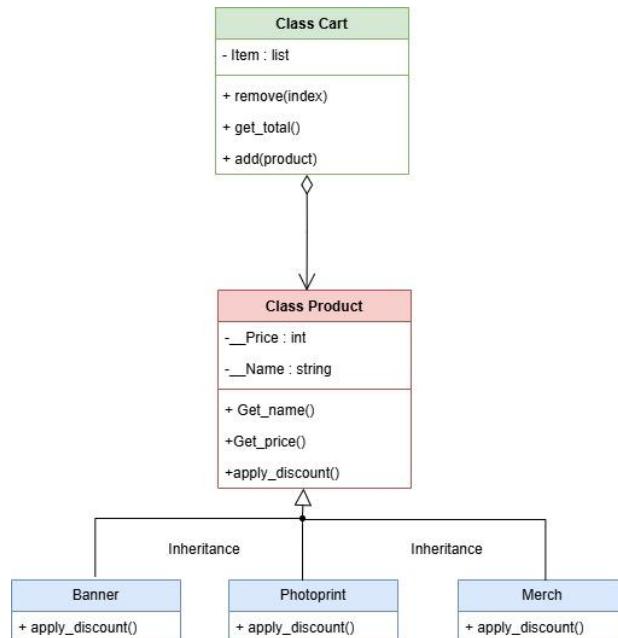
PERANCANGAN SISTEM

3.1 DESKRIPSI SISTEM

Aplikasi Marketplace Digital Printing dirancang untuk menyediakan layanan pembelian produk digital printing seperti banner, cetak foto, dan merchandise melalui antarmuka grafis (GUI). Sistem ini memungkinkan pengguna melihat daftar produk, memilih produk, menambahkannya ke keranjang belanja, serta menghitung total harga secara otomatis berdasarkan diskon kategori.

Perancangan sistem menggunakan pendekatan OOP, di mana setiap komponen utama direpresentasikan sebagai kelas. Dengan pendekatan ini, aplikasi menjadi lebih modular, mudah dikembangkan, dan mudah dipelihara. Selain itu, GUI Tkinter digunakan untuk memudahkan interaksi antara pengguna dan sistem melalui tampilan visual.

3.2 CLASS DESIGN



Gambar 3. 1 Class Diagram

Perancangan kelas bertujuan untuk menggambarkan struktur objek yang digunakan dalam aplikasi serta hubungan antar kelas. Aplikasi ini menggunakan empat kelompok kelas utama yaitu:

1. Product (kelas induk)
 - A. Menyimpan atribut dasar produk seperti nama dan harga.
 - B. Berfungsi sebagai parent class untuk kategori produk lainnya.
2. Banner, PhotoPrint, Merch (kelas turunan)
 - A. Mewarisi atribut dari Product.

- B. Mengimplementasikan metode `apply_discount()` yang berbeda pada setiap kategori (polymorphism).

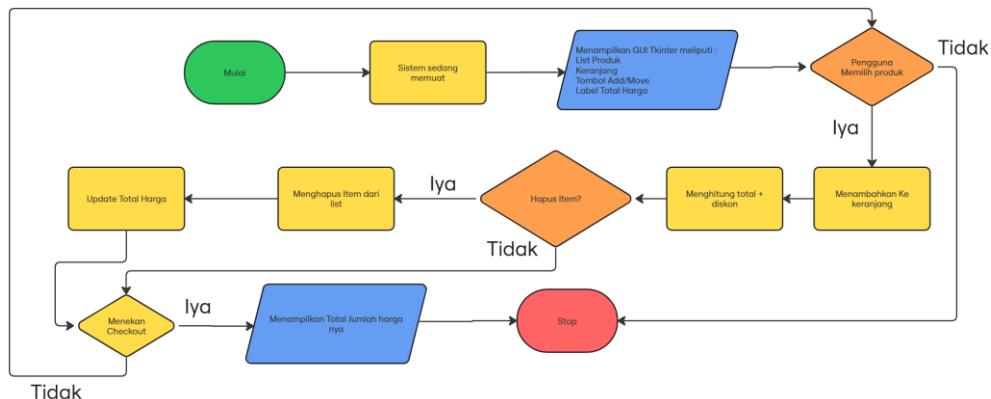
3. Cart

- Menyimpan daftar produk yang ditambahkan ke keranjang.
- Memiliki metode untuk menambahkan produk, menghapus produk, dan menghitung total harga.

4. MarketplaceApp (GUI)

- Mengatur logika antarmuka pengguna.
- Menyediakan fungsi untuk mengontrol interaksi pengguna, seperti menambah ke keranjang, checkout, dan memperbarui total harga.

3.3 FLOWCHART SISTEM



Gambar 3. 2 Flowchart Sistem

Flowchart aplikasi Marketplace Digital Printing menggambarkan alur kerja sistem sejak aplikasi dijalankan hingga proses belanja selesai. Proses dimulai ketika aplikasi aktif, kemudian sistem melakukan pemuatan data produk yang diperlukan untuk ditampilkan pada antarmuka. Setelah proses pemuatan selesai, aplikasi menampilkan GUI Tkinter yang berisi daftar produk, keranjang belanja, tombol untuk menambah atau menghapus item, serta label total harga. Pada tahap ini aplikasi berada pada kondisi menunggu interaksi pengguna.

Selanjutnya, pengguna dapat memilih produk yang tersedia. Jika pengguna belum memilih produk, sistem tetap berada pada tampilan GUI dan menunggu input berikutnya. Namun apabila pengguna memilih salah satu produk, sistem akan memasukkan produk tersebut ke dalam keranjang belanja. Setelah produk ditambahkan, sistem menghitung total harga sekaligus menerapkan diskon berdasarkan kategori produk yang menggunakan metode polymorphism melalui fungsi `apply_discount()`.

Setelah total harga diperbarui, sistem memeriksa apakah pengguna ingin menghapus item tertentu dari keranjang. Jika pengguna memilih untuk menghapus item, sistem akan menghapus produk tersebut dari daftar keranjang dan menghitung ulang total harga, kemudian alur kembali ke proses pemilihan produk sehingga pengguna dapat melanjutkan interaksi. Apabila pengguna tidak ingin menghapus item, alur dilanjutkan ke proses checkout.

Pada tahap checkout, sistem menanyakan apakah pengguna ingin menyelesaikan transaksi. Jika tidak, alur kembali ke proses pemilihan produk sehingga pengguna tetap dapat menambah atau menghapus item. Namun jika pengguna memilih untuk checkout, sistem akan menampilkan total jumlah belanja melalui dialog pesan sebagai informasi akhir. Setelah total belanja ditampilkan, proses dianggap selesai dan aplikasi dapat ditutup.

BAB IV

IMPLEMENTASI PROGRAM

4.1 LINGKUP PENGEMBANGAN

Aplikasi Marketplace Digital Printing dikembangkan menggunakan bahasa pemrograman Python dengan memanfaatkan pustaka Tkinter sebagai antarmuka grafis (GUI). Proses pengembangan dilakukan pada lingkungan sistem operasi Windows, dan editor yang digunakan adalah Visual Studio Code. Tkinter dipilih karena merupakan modul GUI bawaan Python yang ringan, mudah digunakan, serta mendukung berbagai elemen antarmuka seperti tombol, label, dan listbox. Selain itu, implementasi konsep Pemrograman Berorientasi Objek (OOP) dalam Python sangat mendukung struktur program yang modular dan mudah dikembangkan.

4.2 STRUKTUR PROGRAM

Program ini dibangun menggunakan empat kelompok kelas utama, yaitu:

1. Class Product

Berfungsi sebagai kelas induk (superclass) yang menyimpan atribut dasar produk berupa nama dan harga. Class ini juga menyediakan metode dasar seperti `get_name()`, `get_price()`, serta metode `apply_discount()` yang akan di override oleh kelas turunannya.

2. Class Banner, Photoprint, dan Merch

Ketiga class ini merupakan turunan dari `Product` yang masing-masing mengimplementasikan metode `apply_discount()` sesuai ketentuan diskon per kategori.

- A. Banner Diskon 5%

- B. PhotoPrint Diskon 2%

- C. Merch Diskon 10%

3. Class Cart

Berfungsi untuk menyimpan daftar produk yang ditambahkan oleh pengguna ke dalam keranjang belanja. Class ini memiliki metode `add()`, `remove()`, dan `get_total()` untuk menghitung total harga setelah diskon diterapkan.

4. Class MarketplaceApp

Merupakan implementasi GUI berbasis Tkinter. Class ini menangani interaksi pengguna dengan aplikasi melalui tampilan lisr produk, keranjang belanja, tombol tambah dan hapus, serta tombol checkout.

Struktur ini mencerminkan prinsip OOP yaitu enkapsulasi, pewarisan, dan polimorfisme, sehingga program menjadi modular dan mudah diperluas.

4.3 IMPLEMENTASI KODE PROGRAM

Setelah melakukan perancangan sistem untuk aplikasi MarketPlaceApp Digital Printing maka tahap selanjutnya adalah implementasi kode program berbasis Pemrograman Python dengan Pustaka Tkinter library GUI (Graphical User Interface).

| No | KODE PROGRAM |
|----|---|
| 1. | <pre>1 import tkinter as tk 2 from tkinter import messagebox 3 4 # ===== 5 # CLASS PRODUCT OOP 6 # ===== 7 class Product: 8 def __init__(self, name, price): 9 self.__name = name 10 self.__price = price 11 12 def get_name(self): 13 return self.__name 14 15 def get_price(self): 16 return self.__price 17 18 def apply_discount(self): 19 return self.__price</pre> <p>Analisis :</p> <p>Pada bagian ini agar dapat menggunakan tkinter maka harus mengimport modul nya terlebih dahulu karena berfungsi sebagai antarmuka grafis(GUI). Messagebox diimpor secara khusus untuk menampilkan pesan pop-up seperti ketika melakukan checkout. Setelah itu, membuat class produk sebagai parent atau superclass bagi seluruh jenis produk dalam aplikasi marketplace. Kemudian mendefinisikan dengan method constructor yaitu <code>__init__</code> yang berfungsi menjalankan secara otomatis ketika objek baru dibuat. Di dalamnya memiliki atribut Encapsulation untuk name</p> |

dan price menggunakan double underscore untuk menjadikannya private attribute. Setelah itu terdapat method getter pada name dan price yang befungsi mengakses atribut private name dan produk, metode ini diperlukan agar bagian lain dari program tetap dapat membaca nama produk dan harga produk tanpa harus melanggar prinsip OOP. Kemudian pada method apply_discount() akan di override di subclass. Metode ini hanya mengembalikan harga asli tanpa diskon. Method tersebut disengaja dibuat agar di override oleh class turunan yaitu banner, photoprint, dan merch.

```
class Banner(Product):
    def apply_discount(self):
        return self.get_price() * 0.95

class PhotoPrint(Product):
    def apply_discount(self):
        return self.get_price() * 0.98

class Merch(Product):
    def apply_discount(self):
        return self.get_price() * 0.90
```

2. Ketiga kelas turunan yaitu Banner, PhotoPrint, dan Merch merupakan implementasi dari konsep inheritance dalam OOP, di mana masing-masing kelas mewarisi seluruh atribut dan metode dasar dari superclass Product. Pada setiap kelas turunan tersebut, metode apply_discount() dioverride untuk menyesuaikan besarnya potongan harga sesuai kategori produk. Class Banner menerapkan diskon sebesar 5% dengan mengembalikan nilai self.get_price() * 0.95, sedangkan PhotoPrint memberikan diskon 2% melalui pengembalian self.get_price() * 0.98. Adapun class Merch memberikan diskon paling besar, yaitu 10%, melalui operasi self.get_price() * 0.90. Dengan demikian, proses perhitungan diskon tidak lagi dilakukan secara manual menggunakan kondisi atau percabangan, namun cukup dengan memanggil metode apply_discount() pada objek produk apa pun. Sistem akan secara otomatis menjalankan versi metode yang sesuai berdasarkan tipe objeknya. Hal ini merupakan contoh penerapan runtime polymorphism (method overriding) yang sangat efektif,

| | |
|----|--|
| | <p>karena memungkinkan setiap objek produk memiliki perilaku berbeda meskipun berbagi interface yang sama. Struktur ini menjadikan program lebih fleksibel, mudah dikembangkan, dan meminimalkan duplikasi kode pada proses perhitungan harga diskon.</p> |
| 3. | <pre>class Cart: def __init__(self): self.items = [] def add(self, product): self.items.append(product) def remove(self, index): if 0 <= index < len(self.items): del self.items[index] def get_total(self): return sum(item.apply_discount() for item in self.items)</pre> <p>Class Cart berfungsi sebagai wadah penyimpanan produk yang dipilih oleh pengguna, sekaligus menjadi komponen logika utama untuk mengelola data keranjang belanja. Pada konstruktor <code>__init__()</code>, atribut <code>items</code> diinisialisasi sebagai list kosong yang nantinya berisi objek-objek produk apa pun yang ditambahkan ke keranjang, baik dari jenis Banner, PhotoPrint, maupun Merch. Metode <code>add()</code> berfungsi menambahkan objek produk ke dalam list menggunakan <code>append()</code>, sehingga penyimpanan bersifat dinamis sesuai banyaknya pilihan pengguna. Selanjutnya, metode <code>remove()</code> digunakan untuk menghapus item tertentu berdasarkan indeks yang dipilih. Pengecekan kondisi <code>0 <= index < len(self.items)</code> memastikan bahwa indeks valid sehingga mencegah error seperti <code>IndexError</code>. Penghapusan menggunakan <code>del</code> memberikan efisiensi pada manipulasi list. Bagian terpenting dari class ini adalah metode <code>get_total()</code>, yang menghitung total harga seluruh produk dalam keranjang. Perhitungan dilakukan dengan memanggil <code>apply_discount()</code> pada setiap item melalui ekspresi generator, sehingga setiap produk akan menerapkan diskonnya masing-masing secara otomatis. Proses ini menunjukkan penerapan polymorphism, karena setiap item memanggil metode yang sama tetapi menghasilkan nilai berbeda sesuai tipe objeknya. Dengan demikian, class <code>Cart</code> tidak perlu mengetahui jenis produk secara spesifik, namun tetap dapat</p> |

menyediakan total belanja yang akurat dan fleksibel. Struktur ini menjadikan class Cart berperan sebagai pusat pengelolaan data keranjang dengan desain yang bersih, sederhana, dan efektif.

```
class MarketplaceApp:
    def __init__(self, master):
        self.master = master
        master.title("Marketplace Digital Printing")
        master.geometry("950x550")
        master.configure(bg="#eef2f7")

        # Professional Color Palette
        frame_bg = "#ffffff"
        accent_blue = "#2f4e78"
        accent_light = "#4c6ea8"
        border_color = "#d0d7e2"
        shadow_color = "#c7ccd4"
        danger = "#d9534f"
        danger_hover = "#c9302c"
        font_main = ("Segoe UI", 10)
        font_title = ("Segoe UI", 12, "bold")

        self.cart = Cart()
```

4.

Pada bagian awal class MarketplaceApp, metode `__init__()` digunakan untuk menginisialisasi jendela utama aplikasi dan seluruh komponen pendukung antarmuka grafis. Parameter `master` merepresentasikan objek Tkinter utama, yang kemudian disimpan dalam atribut `self.master` agar dapat diakses oleh metode lain. Baris `master.title("Marketplace Digital Printing")` berfungsi memberikan judul pada window aplikasi, sementara `master.geometry("950x550")` mengatur ukuran jendela agar tampilan lebih proporsional dan sesuai dengan jumlah elemen GUI yang akan ditampilkan. Selanjutnya, `master.configure(bg="#eef2f7")` menetapkan warna latar belakang jendela menggunakan kode warna hex, memberikan nuansa visual yang lembut dan profesional. Setelah konfigurasi dasar window selesai, bagian berikutnya mendefinisikan serangkaian variabel warna seperti `frame_bg`, `accent_blue`, `accent_light`, serta `shadow_color`, yang berfungsi sebagai color palette untuk menciptakan tampilan antarmuka yang konsisten, modern, dan lebih estetis. Selain itu, font utama dan font judul juga dideklarasikan melalui `font_main` dan `font_title`.

agar tampilan teks lebih rapi dan mudah dibaca. Penambahan palet warna dan gaya font pada constructor merupakan bentuk penerapan prinsip clean UI design, dengan tujuan meningkatkan kenyamanan pengguna dan memberikan kesan profesional pada aplikasi. Terakhir, objek self.cart = Cart() dibuat untuk menghubungkan antarmuka GUI dengan logika penyimpanan keranjang belanja, sehingga class ini berfungsi sebagai penghubung antara backend dan tampilan pengguna.

```
def create_shadow(x, y, parent):
    shadow = tk.Frame(parent, bg=shadow_color)
    shadow.place(x=x+3, y=y+3, width=350, height=430)
    return shadow

#     FRAME PRODUK + SHADOW
create_shadow(30, 25, master)      # Shadow behind product frame

left_frame = tk.Frame(master, bg=frame_bg, bd=1, relief="solid")
left_frame.place(x=30, y=25, width=350, height=430)

# Header Accent
tk.Frame(left_frame, bg=accent_blue, height=4).pack(fill="x")
```

- kode ini menunjukkan bagaimana antarmuka aplikasi dibuat lebih estetis melalui tambahan efek visual berupa bayangan (shadow) di belakang frame utama. Fungsi create_shadow(x, y, parent) menghasilkan sebuah objek Frame berwarna shadow_color yang ditempatkan sedikit bergeser dari posisi sebenarnya menggunakan place(x=x+3, y=y+3), sehingga menciptakan efek bayangan halus yang memberikan kedalaman visual pada tampilan UI. Setelah itu, fungsi ini dipanggil melalui create_shadow(30, 25, master) untuk menambahkan bayangan di area kiri jendela aplikasi, tepat di belakang frame daftar produk. Kemudian, frame utama untuk menampilkan daftar produk didefinisikan dengan left_frame = tk.Frame(master, bg=frame_bg, bd=1, relief="solid"), yang memberi batas garis halus agar frame terlihat lebih tegas. Frame ini diposisikan menggunakan metode place() dengan ukuran lebar 350 dan tinggi 430 piksel. Selanjutnya, bagian header frame diberi elemen dekoratif berupa garis aksen menggunakan tk.Frame(left_frame, bg=accent_blue, height=4).pack(fill="x"), menciptakan tampilan yang lebih menarik dan profesional layaknya aplikasi modern. Secara keseluruhan, kode ini mencerminkan pendekatan desain UI yang mengutamakan estetika melalui penggunaan shading, kontras warna, dan struktur frame yang rapi, sehingga

| | |
|--|---|
| | <p>antarmuka aplikasi tidak hanya fungsional tetapi juga visualnya nyaman dan menarik bagi pengguna.</p> |
| | <pre> tk.Label(left_frame, text="Daftar Produk", bg=frame_bg, fg=accent_blue, font=font_title, pady=10).pack() self.product_listbox = tk.Listbox(left_frame, width=42, height=15, bg="#f9fbff", fg="#1e2a38", font=font_main, selectbackground="#c7dbf4", highlightbackground=border_color) self.product_listbox.pack(padx=15, pady=5) self.products = [Banner("Banner 1x1 m", 30000), Banner("Banner 2x1 m", 55000), Banner("Banner 3x1 m", 75000), PhotoPrint("Foto 4R", 3000), PhotoPrint("Foto 10R", 12000), PhotoPrint("Foto 20R", 25000), Merch("Mug Custom", 35000), Merch("Kaos Sablon", 80000), Merch("Stiker A4", 5000)] </pre> |

6.

kode ini, antarmuka bagian kiri aplikasi dibangun untuk menampilkan daftar produk yang tersedia bagi pengguna. Pertama, sebuah label judul dibuat menggunakan tk.Label() dengan teks "Daftar Produk", ditempatkan pada left_frame dan diberi pengaturan warna teks accent_blue, latar frame_bg, serta font font_title agar tampil lebih menonjol sebagai header. Selanjutnya, komponen utama yaitu self.product_listbox didefinisikan menggunakan widget Listbox, yang menampilkan daftar produk dalam bentuk teks. Listbox ini diberikan konfigurasi estetika seperti warna latar #f9fbff, warna teks #1e2a38, serta warna pilihan (select background) #c7dbf4, sehingga pengguna dapat melihat item yang sedang dipilih dengan jelas. Widget ini kemudian ditempatkan menggunakan pack() dengan padding agar tidak menempel pada batas frame. Setelah komponen visual selesai disiapkan, daftar produk aktual dideklarasikan dalam atribut self.products berupa list berisi objek-objek dari class Banner, PhotoPrint, dan Merch. Setiap entri produk dibuat melalui pemanggilan konstruktor kelas masing-masing, misalnya Banner("Banner 1x1 m", 30000). Dengan struktur ini, program tidak hanya menampilkan teks produk, tetapi

juga menyimpan objek OOP lengkap yang dapat digunakan untuk perhitungan diskon, penambahan ke keranjang, dan interaksi logis lainnya. Integrasi antara listbox dan daftar objek ini menunjukkan penerapan konsep data binding sederhana, yang menghubungkan tampilan GUI dengan logika internal aplikasi secara konsisten dan efisien.

```
for p in self.products:
    self.product_listbox.insert(tk.END, f"{p.get_name()} - Rp {p.get_price():,}")

# Add Button
self.add_button = tk.Button(
    left_frame, text="Tambah ke Keranjang",
    command=self.add_to_cart, width=30,
    bg=accent_light, fg="white", font=font_main, bd=0, pady=7
)
self.add_button.pack(pady=10)
self.add_button.bind("<Enter>", lambda e: self.add_button.config(bg="#3d5f92"))
self.add_button.bind("<Leave>", lambda e: self.add_button.config(bg=accent_light))
```

- kode ini bertanggung jawab untuk menampilkan daftar produk pada listbox dan menyediakan tombol interaksi bagi pengguna untuk menambahkan item ke dalam keranjang. Pada bagian pertama, dilakukan iterasi terhadap self.products menggunakan perulangan for, di mana setiap objek produk dimasukkan ke dalam self.product_listbox melalui metode insert(). Format string f"{p.get_name()} - Rp {p.get_price():,}" memastikan bahwa nama dan harga produk ditampilkan dalam bentuk teks yang rapi dan mudah dibaca, sekaligus memanfaatkan metode getter untuk menjaga prinsip enkapsulasi pada class Product. Selanjutnya, tombol interaktif self.add_button dibuat dengan widget tk.Button, yang mempunyai teks "Tambah ke Keranjang" dan dihubungkan dengan fungsi self.add_to_cart melalui parameter command. Tombol ini diberi lebar 30 dan menggunakan kombinasi warna accent_light untuk latar serta teks putih (fg="white"), agar tampilannya konsisten dengan tema UI sebelumnya. Tombol kemudian ditambahkan ke frame menggunakan pack() dengan padding vertikal agar memiliki jarak yang proporsional. Untuk meningkatkan pengalaman pengguna, tombol juga dilengkapi interaksi hover menggunakan event binding "<Enter>" dan "<Leave>", yang mengubah warna background tombol saat kursor berada di atasnya maupun ketika keluar. Efek hover ini memberikan feedback visual yang membuat aplikasi terasa lebih responsif dan

modern. Secara keseluruhan, bagian kode ini tidak hanya menampilkan data produk secara dinamis, tetapi juga menciptakan mekanisme interaksi yang intuitif bagi pengguna dalam memilih produk yang ingin dimasukkan ke keranjang.

```
create_shadow(410, 25, master)      # Shadow behind cart frame

right_frame = tk.Frame(master, bg=frame_bg, bd=1, relief="solid")
right_frame.place(x=410, y=25, width=350, height=430)

tk.Frame(right_frame, bg=accent_blue, height=4).pack(fill="x")

tk.Label(
    right_frame, text="Keranjang Belanja", bg=frame_bg, fg=accent_blue,
    font=font_title, pady=10
).pack()

self.cart_listbox = tk.Listbox(
    right_frame, width=42, height=15, bg="#f9fbff",
    fg="#1e2a38", font=font_main,
    selectbackground="#f7c6c7", highlightbackground=border_color
)
self.cart_listbox.pack(padx=15, pady=5)

self.remove_button = tk.Button(
    right_frame, text="Hapus Item",
    command=self.remove_item, width=30,
    bg=danger, fg="white", font=font_main, bd=0, pady=7
)
self.remove_button.pack(pady=10)
self.remove_button.bind("<Enter>", lambda e: self.remove_button.config(bg=danger_hover))
self.remove_button.bind("<Leave>", lambda e: self.remove_button.config(bg=danger))
```

8.

Kode ini membangun tampilan sisi kanan aplikasi yang berfungsi sebagai area keranjang belanja, tempat item yang telah dipilih pengguna ditampilkan. Pertama, efek bayangan dibuat dengan `create_shadow(410, 25, master)` untuk memberikan kesan kedalaman visual di belakang frame keranjang. Setelah itu, `right_frame` dibuat menggunakan `tk.Frame` dengan latar `frame_bg`, border tipis, dan `relief="solid"`, lalu ditempatkan menggunakan `place()` dengan ukuran 350×430 piksel, simetris dengan frame produk di sebelah kiri. Sebuah garis aksen berwarna `accent_blue` ditempatkan di bagian atas frame sebagai elemen dekoratif untuk mempertegas header. Label "Keranjang Belanja" kemudian dibuat menggunakan `tk.Label`, dengan warna teks `accent_blue` dan font `font_title`, memberikan identitas visual yang konsisten dengan area produk. Selanjutnya, komponen utama berupa `self.cart_listbox` didefinisikan menggunakan widget `Listbox`, yang berfungsi menampilkan daftar item yang sudah dimasukkan ke keranjang. Widget ini menggunakan kombinasi warna halus seperti

#f9fbff untuk latar dan #1e2a38 untuk teks, serta warna pemilihan #f7c6c7 agar item yang dipilih dapat terlihat jelas. Kemudian tombol self.remove_button dibuat sebagai kontrol untuk menghapus produk dari keranjang, dengan teks "Hapus Item" dan warna danger untuk memberikan penegasan bahwa tombol ini melakukan aksi destruktif. Tombol tersebut dihubungkan dengan fungsi self.remove_item melalui parameter command, dan efek hover ditambahkan menggunakan bind() untuk memberikan umpan balik visual saat kursor melewati tombol. Secara keseluruhan, kode ini membentuk area keranjang yang informatif, estetis, dan interaktif, sekaligus menghubungkan elemen GUI dengan logika penghapusan item pada class Cart.

```
self.total_label = tk.Label(  
    master, text="Total: Rp 0",  
    font=("Segoe UI", 16, "bold"),  
    bg="#eef2f7", fg=accent_blue  
)  
self.total_label.place(x=300, y=470)  
  
self.checkout_button = tk.Button(  
    master, text="Checkout",  
    command=self.checkout, width=20,  
    bg=accent_light, fg="white",  
    font=("Segoe UI", 12, "bold"), bd=0, pady=7  
)  
self.checkout_button.place(x=520, y=465)  
  
self.checkout_button.bind("<Enter>", lambda e: self.checkout_button.config(bg="#3d5f92"))  
self.checkout_button.bind("<Leave>", lambda e: self.checkout_button.config(bg=accent_light))
```

9. kode ini membangun bagian bawah antarmuka aplikasi yang berfungsi menampilkan total harga belanja serta menyediakan tombol untuk melakukan proses checkout. Komponen pertama adalah self.total_label, sebuah label yang menampilkan teks "Total: Rp 0" sebagai nilai awal sebelum pengguna menambahkan produk ke keranjang. Pengaturan font "Segoe UI", 16, "bold" dan warna teks accent_blue memberikan tampilan yang jelas dan mudah diidentifikasi sebagai informasi penting, sementara latar belakang #eef2f7 disesuaikan agar selaras dengan warna utama jendela aplikasi. Label ini ditempatkan secara presisi menggunakan place(x=300, y=470) agar posisinya berada di tengah antara frame produk dan frame keranjang. Selanjutnya dibuat tombol self.checkout_button yang berfungsi menjalankan perintah checkout ketika ditekan, melalui pemanggilan method self.checkout pada parameter command. Tombol ini diprioritaskan tampilannya dengan ukuran lebar 20, warna latar accent_light, warna teks putih, serta font tebal untuk memberikan kesan

tombol aksi utama. Posisi tombol diatur menggunakan place(x=520, y=465), memastikan tata letak harmonis di bawah keranjang belanja. Untuk meningkatkan interaksi pengguna, event binding ditambahkan pada tombol sehingga warna tombol berubah saat kursor berada di atasnya (<Enter>) maupun saat keluar (<Leave>), menciptakan efek hover yang memberikan respons visual modern. Secara keseluruhan, bagian kode ini menggabungkan informasi penting dan kontrol utama aplikasi dalam sebuah elemen UI yang estetis, jelas, dan mudah digunakan, sekaligus menghubungkan antarmuka dengan logika pemrosesan total belanja dan checkout.

```

def add_to_cart(self):
    index = self.product_listbox.curselection()
    if index:
        product = self.products[index[0]]
        self.cart.add(product)
        self.cart_listbox.insert(tk.END, f"{product.get_name()} - Rp {product.apply_discount():,.0f}")
        self.update_total()
    else:
        messagebox.showwarning("Error", "Pilih produk terlebih dahulu")

def remove_item(self):
    index = self.cart_listbox.curselection()
    if index:
        self.cart.remove(index[0])
        self.cart_listbox.delete(index)
        self.update_total()
    else:
        messagebox.showwarning("Error", "Pilih item di keranjang")

def update_total(self):
    total = self.cart.get_total()
    self.total_label.config(text=f"Total: Rp {total:,.0f}")

def checkout(self):
    total = self.cart.get_total()
    messagebox.showinfo("Checkout", f"Total Belanja Anda: Rp {total:,.0f}")
    self.cart.items.clear()
    self.cart_listbox.delete(0, tk.END)
    self.update_total()

```

10.

kode ini berisi seluruh logika interaksi utama pada aplikasi marketplace, mulai dari menambahkan produk ke keranjang, menghapus item, menghitung total harga, hingga menyelesaikan transaksi checkout. Fungsi add_to_cart() diawali dengan mengambil indeks item yang dipilih pengguna dari product_listbox menggunakan curselection(). Jika ada pilihan, program mengambil objek produk yang sesuai dari self.products, menambahkannya ke keranjang melalui self.cart.add(product), dan menampilkannya pada cart_listbox dalam format teks lengkap dengan harga setelah diskon menggunakan product.apply_discount(). Setelah itu, update_total() dipanggil untuk memperbarui total belanja. Apabila tidak ada produk dipilih, sistem menampilkan peringatan melalui messagebox.showwarning. Proses sebaliknya dilakukan oleh fungsi remove_item(), yang mengambil indeks pilihan dari keranjang dan menghapus item tersebut menggunakan self.cart.remove(index[0]), sekaligus

| | |
|-----|--|
| | <p>menghapus tampilannya dari listbox dan memperbarui total harga. Fungsi update_total() sendiri menghitung total belanja dengan memanggil self.cart.get_total(), lalu memperbarui label total menggunakan config(). Terakhir, fungsi checkout() menjadi tahap penyelesaian transaksi, di mana total akhir dihitung kembali, lalu ditampilkan kepada pengguna melalui messagebox. Setelah checkout, seluruh item dihapus dari keranjang menggunakan clear() dan listbox keranjang dikosongkan melalui delete(0, tk.END), memastikan aplikasi kembali ke kondisi awal. Keempat fungsi ini bekerja secara terintegrasi dengan class Cart dan interface Tkinter, membentuk alur interaksi yang lengkap dan responsif, serta menunjukkan pemisahan logika bisnis (OOP) dan logika tampilan (GUI) yang jelas dan efektif.</p> |
| 11. | <pre>root = tk.Tk() app = MarketplaceApp(root) root.mainloop()</pre> <p>kode ini merupakan titik awal eksekusi GUI, di mana sebuah instance Tkinter dibuat melalui root = tk.Tk(), yang berfungsi sebagai jendela utama tempat seluruh elemen antarmuka program akan ditampilkan. Objek root kemudian dikirimkan sebagai parameter ke konstruktur MarketplaceApp(root), yang bertanggung jawab membangun seluruh struktur UI beserta logika aplikasinya. Dengan cara ini, class MarketplaceApp memperoleh kendali penuh atas window utama melalui objek root yang disimpan ke dalam atribut self.master. Setelah semua komponen antarmuka selesai diinisialisasi, perintah root.mainloop() dijalankan. Metode mainloop() merupakan inti dari aplikasi Tkinter karena menjaga agar jendela tetap aktif dan responsif terhadap interaksi pengguna, seperti klik tombol, seleksi item, dan input lainnya. Tanpa mainloop(), jendela akan tertutup secara otomatis sesaat setelah program dijalankan. Dengan demikian, bagian kode ini dapat dikatakan sebagai <i>entry point</i> seluruh aplikasi GUI, menghubungkan antara definisi class dengan tampilan nyata yang digunakan pengguna.</p> |

BAB V

PENGUJIAN SISTEM

5.1 TUJUAN PENGUJIAN

Tujuan dari pengujian sistem ini adalah memastikan bahwa seluruh fitur pada aplikasi Marketplace Digital Printing dapat berjalan sesuai dengan kebutuhan fungsional yang telah dirancang. Pengujian dilakukan agar dapat diketahui apakah setiap proses, mulai dari penambahan produk ke keranjang, penghapusan item, perhitungan total belanja, hingga checkout, dapat berfungsi dengan benar tanpa error. Selain itu pengujian juga bertujuan untuk memvalidasi interaksi pengguna pada antarmuka GUI yang dibangun menggunakan Pustaka Tkinter.

5.2 METODE PENGUJIAN

Metode pengujian yang digunakan adalah Black-Box Testing, yaitu pengujian yang berfokus pada input dan output tanpa memeriksa struktur internal kode program. Dengan metode ini, setiap fungsi diuji berdasarkan aksi pengguna dan respons visual pada aplikasi. Pengujian dilakukan langsung melalui antarmuka, sehingga sesuai untuk aplikasi berbasis GUI seperti yang dikembangkan pada proyek ini.

5.3 SKENARIO PENGUJIAN

Berikut adalah skenario test-case yang digunakan untuk memastikan seluruh fitur bekerja dengan benar.

| NO | FITUR YANG DI UJI | AKSI PENGGUNA | HASIL YANG DIHARAPKAN | HASIL UJI | STATUS |
|----|-------------------------------|---------------------------|---|---------------|----------|
| | | | | | |
| 1. | Menampilkan daftar produk | Aplikasi dijalankan | Produk tampil pada list | Produk tampil | Berhasil |
| 2. | Menambah produk ke keranjang | Pilih produk klik tambah | Produk masuk ke keranjang dan total berubah | Sesuai | Berhasil |
| 3. | Menambah tanpa memilih produk | Klik tambah tanpa seleksi | Muncul popup error | Popup muncul | Berhasil |

| | | | | | |
|-----------|------------------------------|------------------------|---|--------------|----------|
| 4. | Menghapus item | Pilih item klik hapus | Item terhapus dan total berubah | Sesuai | Berhasil |
| 5. | Menghapus tanpa memilih item | Klik hapus | Muncuk popup error | Popup muncul | Berhasil |
| 6. | Menghitung total belanja | Tambah beberapa produk | Total dihitung berdasarkan perhitungan diskon | Nilai sesuai | Berhasil |
| 7. | Checkout | Klik tombol Checkout | Muncul popup total & keranjang kosong | Sesuai | Berhasil |
| 8. | Reset setelah checkout | Setelah checkout | Keranjang kosong, total Kembali 0 | Sesuai | Berhasil |

5.4 DOKUMENTASI PENGUJIAN

Berikut adalah hasil dari tahap pengujian.



Gambar 5. 1 Tampilan Awal



Gambar 5. 2 Menambahkan produk ke keranjang



Gambar 5. 3 Menghapus Produk



Gambar 5. 4 Harus memilih Item di keranjang



Gambar 5. 5 Checkout Produk

5.5 ANALISIS HASIL PENGUJIAN

Setelah melakukan pengujian seluruh alur sistem dan skenario yang dilakukan, dapat disimpulkan bahwa aplikasi Marketplace Digital Printing telah berjalan dengan baik dan memenuhi seluruh kebutuhan fungsional. Setiap fitur memberikan output yang konsisten dengan desain sistem, termasuk perhitungan diskon, penambahan dan penghapusan item pada keranjang, validasi input pengguna, serta proses checkout. Tidak ditemukan error atau kegagalan fungsi selama pengujian, sehingga aplikasi dinyatakan layak digunakan dan stabil.

BAB VI

PENUTUP

6.1 KESIMPULAN

Dari hasil perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi Marketplace Digital Printing berhasil dibangun menggunakan konsep Pemrograman Berorientasi Objek (OOP) dan antarmuka GUI Tkinter. Seluruh fitur utama seperti menampilkan produk, menambah dan menghapus item dari keranjang, menghitung total harga beserta diskon, serta melakukan checkout dapat berfungsi dengan baik tanpa error. Penerapan OOP membuat struktur program lebih teratur dan mudah dikembangkan, sementara pengujian menunjukkan bahwa sistem berjalan sesuai dengan kebutuhan fungsional yang telah ditentukan.

6.2 SARAN

Aplikasi ini dapat dikembangkan lebih lanjut seperti dengan menambahkan data menggunakan database, menampilkan gambar produk, serta menyediakan fitur login pengguna. Selain itu, tampilan antarmuka dapat diperbaiki agar lebih interaktif dan modern. Pengembangan lanjutan diharapkan dapat meningkatkan fungsi aplikasi dan membuatnya semakin mendekati sistem marketplace sesungguhnya.

Link Github :

https://github.com/Pramudya12-hub/Marketplace_DigitalPrinting/blob/main/Project.py

DAFTAR PUSTAKA

- (*Python: Pengertian, Contoh Penggunaan, Dan Manfaat Mempelajarinya*, 2022) Amos, D. (2024). *Pemrograman GUI Python: Tutorial Tkinter*. Realpython.Com. <https://realpython.com/python-gui-tkinter/>
- Jagoan, H. R. (2023). *Pengertian OOP (Object Oriented Programming) dan 4 Prinsipnya*. Jagoanhousing.Comagoanhousing.Com. <https://www.jagoanhousing.com/blog/oop-adalah/>
- Prasatya. (2024a). *Apa Itu Object Oriented Programming (OOP): Pengertian dan Contohnya!* Codepolitan.Com. <https://www.codepolitan.com/blog/apa-itu-object-oriented-programming-oop-pengertian-dan-contohnya/>
- Prasatya. (2024b). *Bahasa Pemrograman Python Sering digunakan untuk apa?* Codepolitan.Com. <https://www.codepolitan.com/blog/bahasapemrogramanpython-sering-digunakan-untuk-apa/>
- Python: Pengertian, Contoh Penggunaan, dan Manfaat Mempelajarinya*. (2022). Dicoding.Com. <https://www.dicoding.com/blog/python-pengertian-contoh-penggunaan-dan-manfaat-mempelajarinya/>