LAPORAN PROYEK AKHIR PEMOGRAMAN DASAR

Disusun oleh:

Jovan Rio Fernando	(25031554017)
Maydian Ratu Valencia	(25031554121)
Syabina Suwardhana Poetri	(25031554150)

Dosen pengampu:

Hassanuddin Al-Habib, S.Si., M.Si

PROGAM STUDI SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi informasi saat ini telah membawa perubahan besar dalam berbagai aspek kehidupan, termasuk dalam bidang usaha dan layanan. Digitalisasi menjadi salah satu kebutuhan utama bagi pelaku usaha untuk meningkatkan efisiensi kerja, ketepatan pencatatan data, serta kualitas pelayanan kepada pelanggan. Salah satu contoh penerapan teknologi tersebut adalah penggunaan aplikasi kasir dalam kegiatan transaksi penjualan.

Kafe sebagai salah satu bentuk usaha di bidang kuliner memiliki aktivitas transaksi yang cukup tinggi setiap harinya. Proses transaksi meliputi pencatatan menu yang dipesan, perhitungan total harga, pemilihan metode pembayaran, serta penyimpanan data transaksi. Apabila proses ini dilakukan secara manual, seperti mencatat di kertas atau menghitung secara langsung, maka risiko terjadinya kesalahan perhitungan, kehilangan data transaksi, serta keterlambatan pelayanan kepada pelanggan akan semakin besar.

Selain itu, pencatatan transaksi secara manual juga menyulitkan pemilik usaha dalam melakukan evaluasi penjualan. Data transaksi yang tidak tersusun dengan baik akan menyulitkan proses rekapitulasi, terutama ketika ingin mengetahui total pendapatan dalam jangka waktu tertentu, seperti laporan mingguan dan bulanan. Akan tetapi, laporan tersebut sangat penting untuk mengetahui perkembangan usaha, menentukan strategi penjualan, dan mengambil Keputusan bisnis.

Berdasarkan permasalahan tersebut, penulis mengembangkan aplikasi kasir kafe brewlytica sebagai proyek akhir pada mata kuliah pemograman dasar. Aplikasi ini dirancang untuk membantu proses transaksi penjualan dengan fitur penginputan menu, perhitungan total harga secara otomatis, pemilihan metode diharapkan dapat memberikan solusi sederhana namun efektif dalam mendukung operasional kafe.

Selain bertujuan untuk membantu proses transaksi, pembuatan aplikasi ini juga menjadi sarana bagi penulis untuk menerapkan dan memahami konsep-konsep dasar pemograman yang telah dipelajari selama perkuliahan. Dengan mengerjakan proyek ini, penulis dapat mengaplikasikan teori ke dalam bentuk program nyata, sehigga pemahaman terhadap pemograman dasar menjadi lebih mendalam dan aplikatif.

1.2. Rumusan Masalah

1. Bagaimana merancang sebuah aplikasi kasir sederhana yang sesuai dengan kebutuhan operasional kafe brewlytica?
2. Bagaimana mengimplementasikan proses penginputan menu dan perhitungan total harga secara otomatis agar meminimalkan kesalahan perhitungan?
3. Bagaimana sistem dapat menyediakan beberapa metode pembayaran yang mudah digunakan oleh kasir?
4. Bagaimana aplikasi dapat menyimpan dan menampilkan data transaksi dalam bentuk rekap mingguan dan bulanan?
5. Bagaimana penerapan konsep dasar pemograman dapat digunakan untuk membangun aplikasi kasir yang fungsional dan mudah dipahami?

1.3. Tujuan Pembuatan Sistem Kasir

1. Menghasilkan sebuah aplikasi kasir sederhana yang dapat membantu proses transaksi penjualan di kafe brewlytica.
2. Menerapkan konsep-konsep dasar pemograman seperti variabel, tipe data, percabangan, perulangan, dan fungsi dalam sebuah aplikasi.
3. Mengurangi risiko kesalahan perhitungan total pembayaran melalui proses perhitungan otomatis.
4. Memudahkan pencatatan dan pengelolaan data transaksi penjualan secara terstruktur.
5. Menyediakan rekap transaksi mingguan dan bulanan sebagai bahan evaluasi penjualan.

BAB 2
ISI

2.1. Analisis Kebutuhan Aplikasi

Pengembangan aplikasi kasir sederhana dilakukan untuk menganalisis dan mengetahui alasan dan kebutuhan pengguna aplikasi ini dalam menunjang proses pengelolaan transaksi penjualan agar lebih efektif dan tersusun dengan baik. Pada program kasir kafe, analisis disusun menjadi dua bagian:

2.1.1. Permasalahan pada pencatatan transaksi manual

- Pencatatan transaksi secara manual masih banyak digunakan pada usaha kecil seperti kafe dan tempat makan.
- Perhitungan manual rentan terhadap kesalahan, terutama saat transaksi berlangsung ramai.
- Data transaksi sering tidak tersusun rapi dan sulit ditelusuri kembali.
- Catatan berbasis kertas berisiko hilang atau rusak.
- Proses penjualan menjadi lambat dan kurang efisien.
- Kurangnya sistem terstruktur menyulitkan evaluasi penjualan.

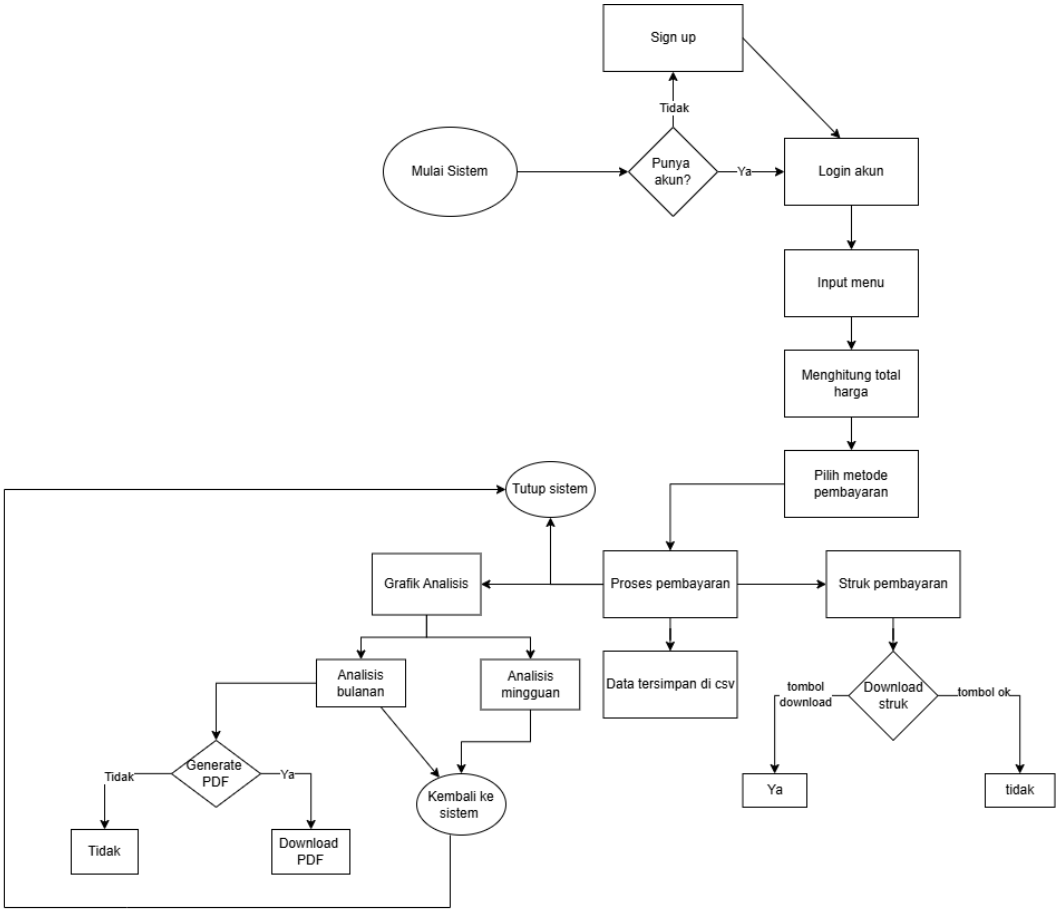
2.1.2. Kebutuhan akan aplikasi kasir digital sederhana

- Aplikasi kasir digital dapat menghitung total transaksi secara otomatis.
- Data penjualan dapat disimpan secara terstruktur dan aman.
- Riwayat transaksi dapat diakses kembali dengan mudah.
- Sistem sederhana dapat digunakan tanpa keahlian teknis khusus.
- Proses transaksi menjadi lebih cepat dan efisien.
- Data digital mendukung analisis penjualan dasar untuk pengembalian keputusan.

2.2. Diagram Alir

Proses transaksi pada aplikasi kasir ini dirancang untuk menggantikan pencatatan manual yang kurang efisien dengan alur digital yang lebih terstruktur. Melalui aplikasi ini, pengguna dapat melakukan transaksi dengan langkah-langkah yang jelas, mulai dari pemilihan menu hingga penyimpanan data transaksi secara otomatis. Dengan diagram alir (*flowchart*) digunakan untuk menggambarkan urutan proses terjadi dalam sistem aplikasi kasir. *Flowchart* ini menujukkan alur kerja sistem secara logis dari awal hingga akhir, sehingga memudahkan

pemahaman terhadap proses transaksi dan memastikan setiap langkah berjalan secara sistematis.



Gambar 2. 1

2.3. Sketsa *User Interface Design*

1. Halaman login
   Komponen:
   - Judul: “Login kasir”
   - Input pengguna:
     - Kotak input username
     - Kotak input password
     - Tombol:
       - Login
       - Buat akun
2. Halaman signup
   Komponen:
   - Input pengguna
     - Nama kasir
     - Username
     - Password
   - Tombol:
     - Signup
     - Kembali login
3. Halaman kasir
   Komponen:
   - *Header*:
     - Judul: “Brewlytica Cafe”
     - Label kasir
     - Tombol analisis
     - Tombol logout
   - Panel kiri:
     - Pemilihan kategori

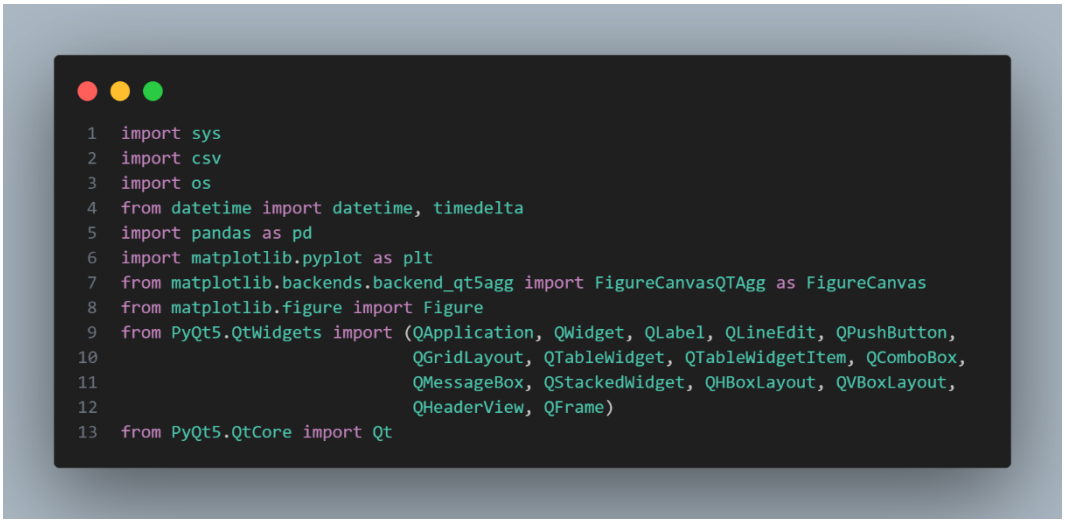
        - Pemilihan menu
        - Kotak input jumlah
        - Tombol tambah keranjang
        - Pemilihan metode pembayaran
    - Panel kanan:
      Tabel tampilan:
        - Kategori
        - Menu
        - Jumlah
        - Harga
        - Subtotal
        - Tombol hapus item
        - Tombol total harga
        - Tombol proses pembayaran
        - Tampilan struk
        - Unduh struk
4. Hasil transaksi/struk
   Komponen:
    - Tampilan informasi:
        - Nomor transaksi
        - Tanggal dan waktu
        - Nama kasir
        - Daftar item yang dibeli
        - Total harga
        - Metode pembayaran
5. Halaman analisis
   Komponen:
    - Judul: “Analisis penjualan”
    - Tombol:
        - Analisis mingguan
        - Analisis bulanan
        - Kembali ke kasir
    - Visualisasi analisis bulanan
        - Grafik analisis penjualan bulanan (*pie chart/bar chart*)
    - Fitur:
        - Ringkasan penjualan bulanan
        - Unduh laporan analisis bulanan

BAB 3
IMPLEMENTASI

3.1. Penjelasan Kode

Implementasi sistem kasir brewlytica terdiri dari beberapa modul yang saling teritegrasi untuk menangani proses autentikasi, transaksi penjualan, penyimpanan data, serta analisis penjualan. Setiap bagian kode dirancang dengan pembagian tanggung jawab yang jelas antara pengolahan data, logika sistem, dan antarmuka pengguna. Berikut ini merupakan penjelasan dari bagian utama kode.

3.1.1. *Library pada sistem kasir*



```
import sys
import csv
import os
from datetime import datetime, timedelta
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
from PyQt5.QtWidgets import (QApplication, QWidget, QLabel, QLineEdit, QPushButton,
                             QGridLayout, QTableWidget, QTableWidgetItem, QComboBox,
                             QMessageBox, QStackedWidget, QHBoxLayout, QVBoxLayout,
                             QHeaderView, QFrame)
from PyQt5.QtCore import Qt
```

Gambar 3. 1

1. sys: Digunakan untuk mengatur alur eksekusi program, khususnya dalam menjalankan dan menghentikan aplikasi PyQt5 dengan aman melalui *event loop*.
2. csv: Digunkan untuk membaca dan menulis data akun kasir serta data transaksi ke dalam file CSV. *Library* ini dipilih karena sederhana dan sesuai untuk penyimpanan data pada skala kecil.
3. os: Digunakan untuk berinteraksi dengan sistem operasi, terutama dalam pengelolaan *path file* agar aplikasi dapat berjalan pada berbagai sistem operasi tanpa perubahan kode.
4. *datetime* dan *timedelta*: Digunakan untuk mengelola data tanggal dan waktu, seperti pencatatan waktu transaksi serta perhitungan analisis penjualan mingguan dan bulanan.
5. pandas: Digunakan untuk memuat, membersihkan, dan mengelola data transaksi dari file CSV. *Library* ini mempermudah proses *filtering*, pengelompokkan, dan perhitungan data penjualan.

6. matplotlib.pyplot: Digunakan untuk membuat grafik tren penjualan, baik dalam bentuk grafik mingguan maupun bulanan.
7. *figure*canvasQTAgg dan *figure*: Digunakan untuk mengintegrasikan grafik matplotlib ke dalam antarmuka PyQt5 sehingga grafik dapat ditampilkan langsung pada aplikasi desktop.
8. PyQt5.QtWidgets: Menyediakan berbagai komponen antarmuka grafis seperti jendela aplikasi, label, input teks, tombol, tabel, *combo box*, *layout*, serta kotak dialog pesan.
9. PyQt5.QtCore(Qt): Digunakan untuk mengatur properti inti antarmuka, seperti perataan komponen, pengaturan *layout*, dan interaksi antar widget.

3.1.2. *Base directory* file

Gambar 3. 2

1. Pengaturan *base directory*
   Variabel BASE_DIR digunakan untuk menentukan direktori utama tempat file program dijalankan. Direktori ini menjadi acuan utama dalam mengakses seluruh file data yang digunakan oleh sistem. Tujuan penggunaan *base directory*
   - Menjamin konsistensi akses file meskipun lokasi folder proyek pindahkan.
   - Menghindari penggunaan *path absolut* yang bergantung pada satu perangkat tertentu.
   - Meningkatkan portabilitas dan kestabilan aplikasi.
2. Konfigurasi base data
   File users.csv digunakan untuk menyimpan data akun kasir yang meliputi *username, password,* dan nama kasir. File ini diakses oleh sistem autentikasi untuk proses pendaftaran akun dan login pengguna.
3. File data transaksi
   File data_transaksi.csv digunakan untuk menyimpan seluruh data transaksi penjualan. Data yang disimpan mencakup nomor struk, tanggal,

waktu, nama kasir, menu, kategori, harga satuan, subtotal, total transaksi, serta metode pembayaran.

3.1.3. Sistem Autentikasi

```python
class SistemAuth:
    def __init__(self, user_account_file=USERS_FILE):

        self.user_account_file = user_account_file

        if not os.path.exists(self.user_account_file):
            with open(self.user_account_file, "w", newline="", encoding="utf-8") as f:
                writer = csv.writer(f)
                writer.writerow(["username", "password", "nama_kasir"])

    def signup(self, username, password, nama_kasir):
        if self._is_username_exists(username):
            return False, "Username telah digunakan"

        with open(self.user_account_file, "a", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow([username, password, nama_kasir])
        return True, "Signup berhasil"

    def login(self, username, password):
        with open(self.user_account_file, "r", newline="", encoding="utf-8") as f:
            reader = csv.DictReader(f)
            for row in reader:
                if row["username"] == username and row["password"] == password:
                    return True, row["nama_kasir"]
        return False, "Username atau password salah"

    def _is_username_exists(self, username):
        try:
            with open(self.user_account_file, "r", newline="", encoding="utf-8") as f:
                reader = csv.DictReader(f)
                for row in reader:
                    if row["username"] == username:
                        return True
        except:
            pass
        return False
```

Gambar 3. 3

1. *class* SistemAuth
   Kelas SistemAuth berfungsi sebagai modul autentikasi pada sistem kasir brewlytica. Kelas ini bertanggung jawab dalam pengelolaan akun kasir, termasuk proses pendaftaran (signup), login, serta validasi data pengguna. Seluruh data akun disimpan dalam bentuk defile CSV sehingga mudah diakses dalam dikelola.
2. Inisialisasi kelas (__init__)
   Method __init__ digunakan untuk melakukan insialisasi awal sistem autentikasi. Parameter user_account_file berisi *path* menuju file CSV yang menyimpan data akun kasir. Proses yang dilakukan:

- Menyimpan *path* file akun kasir ke dalam atribut kelas.
- Mengecek apakah file users.csv sudah tersedia.
- Jika file belum ada, sistem akan membuat file baru dan menambahkan header kolom.

3. Fungsi signup
   Method ini digunakan untuk mendaftarkan akun kasir baru ke dalam sistem kasir. Berikut alur kerjanya:
   a. Sistem memeriksa apakah username yang dimasukkan sudah terdaftar menggunakan fungsi _is_username_exists.
   b. Jika username sudah digunakan, proses signup dibatalkan dan sistem mengembalikan pesan kegagalan.
   c. Jika username belum terdaftar, data akun baru akan ditambahkan ke file users.csv.
   d. Sistem mengembalikan status keberhasilan pendaftaran.
4. Fungsi login
   Method ini digunakan untuk memverifikasi data login kasir saat proses masuk ke sistem. Berikut alur kerjanya:
   a. File users.csv dibaca menggunakan csv.*DictReader*
   b. Sistem membandingkan *username* dan *password* yang dimasukkan dengan data yang tersimpan.
   c. Jika data cocok, sistem mengembalikan status berhasil beserta nama kasir.
   d. Jika tidak ditemukan kecocokan, sistem mengembalikan pesan kesalahan
5. Fungsi pengecekan username
   Merupakan fungsi internal (*helper*) yang digunakan untuk memeriksa apakah *username* tertentu sudah terdaftar di sistem. Berikut alur kerjanya:
   a. File users.csv dibaca baris demi baris.
   b. Sistem membandingkan nilai *username* pada setiap baris dengan input pengguna.
   c. Jika ditemukan kecocokan, fungsi mengembalikan nilai *True.*
   d. Jika tidak ditemukan, fungsi mengembalikan nilai *False*.

3.1.4. Sistem Kasir

Kelas SistemKasir merupakan inti dari logika bisnis pada sistem kasir brewlytica. Kelas ini bertanggung jawab dalam pengelolaan menu, keranjang belanja, perhitungan transaksi, pembuatan struk, serta penyimpanan data transaksi ke dalam file CSV. Seluruh proses perhitungan dilakukan pada kelas ini, sehingga antarmuka pengguna hanya berfungsi sebagai media input dan output.

1. Inisialisasi Kelas

```python
class SistemKasir:
    def __init__(self):
        self.keranjang = []
        self.metode_pembayaran = ""
        self.menu_kafe = {
            "Kopi": {
                "Espresso": 10000,
                "Americano": 12000,
                "Cappucino": 15000,
                "Latte": 15000,
                "Macchiato": 17000,
                "Kopi Tubruk": 8000,
            },
            "Smoothie": {
                "Smoothie Kiwi": 17000,
                "Smoothie Berrie": 18000,
                "Smoothie Alpukat": 16000,
            },
            "Pasta": {
                "Spagetti": 20000,
                "Fettuccine": 22000,
                "Lasagna": 25000,
            },
            "Nasi dan Mie": {
                "Nasi Goreng": 15000,
                "Mie Goreng": 15000,
                "Mie Kuah": 15000,
            },
            "Camilan": {
                "Kentang Goreng": 12000,
                "Onion Rings": 12000,
            },
            "Dessert": {
                "Es Krim": 10000,
                "Puding": 8000,
                "Tart Lemon": 15000,
                "Cheesecake": 15000,
                "Gelato": 12000,
            }
        }
        self.csv_file = TRANSAKSI_FILE
        self._init_csv_file()
```

Gambar 3. 4

Methode __init__ digunakan untuk melakukan inisialisasi awal sistem kasir. Pada tahap ini, seluruh komponen penting yang berkaitan dengan transaksi disiapkan. Berikut alur kerjanya:

a. Menginisialisasi variabel keranjang sebagai *list* kosong untuk menyimpan item pesanan.
b. Menginsialisasi variabel metode pembayaran.
c. Mendefinisikan struktur menu kafe dalam bentuk dictionary bertingkat (menu kafe) yang berisi kategori, nama menu, dan harga.
d. Menentukan file penyimpanan transaksi (data_transaksi.csv).
e. Memanggil fungsi _init_csv_file untuk memastikan file transaksi tersedia.

2. Inisialisasi file transaksi

```
def _init_csv_file(self):
    if not os.path.exists(self.csv_file):
        with open(self.csv_file, "w", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow(["No_Struk", "Tanggal", "Waktu", "Kasir", "Menu",
                             "Kategori", "Harga_Satuan", "Jumlah", "Subtotal",
                             "Total_Transaksi", "Metode_Pembayaran"])
```

Gambar 3. 5

Method ini digunakan untuk memastikan file penyimpanan transaksi tersedia sebelum sistem digunakan. Berikut alur kerjanya:

a. Mengecek keberadaan file data_transaksi.csv.
b. Jika file belum ada, sistem akan membuat file baru beserta header kolom.

3. Pengelolaan menu

```
def get_kategori_menu(self):
    return list(self.menu_kafe.keys())

def get_menu_by_kategori(self, kategori):
    return self.menu_kafe.get(kategori, {})
```

Gambar 3. 6

a. get_kategori_menu digunakan untuk mengambil daftar kategori menu yang tersedia. Fungsi ini digunakan oleh antarmuka pengguna untuk menampilkan pilihan kategori.

b. get_menu_by_kategori digunakan untuk mengambil daftar menu berdasarkan kategori yang dipilih pengguna.

4. Penambahan item ke keranjang

```python
def tambah_item(self, kategori, nama_menu, jumlah):
    if kategori not in self.menu_kafe:
        return {"status": False, "message": f"Kategori '{kategori}' tidak ditemukan!"}

    if nama_menu not in self.menu_kafe[kategori]:
        return {"status": False, "message": f"Menu '{nama_menu}' tidak ditemukan!"}

    if jumlah <= 0:
        return {"status": False, "message": "Jumlah harus lebih dari 0!"}

    harga = self.menu_kafe[kategori][nama_menu]
    subtotal = harga * jumlah

    for item in self.keranjang:
        if item["menu"] == nama_menu:
            item["jumlah"] += jumlah
            item["subtotal"] = item["jumlah"] * item["harga"]
            return {"status": True, "message": f"{nama_menu} ditambahkan lagi."}

    self.keranjang.append({
        "kategori": kategori,
        "menu": nama_menu,
        "harga": harga,
        "jumlah": jumlah,
        "subtotal": subtotal
    })

    return {"status": True, "message": f"{nama_menu} x {jumlah} berhasil ditambahkan!"}
```

Gambar 3. 7

Fungsi menambahkan item pesanan ke dalam keranjang belanja. Berikut alur kerjanya:

a. Memvalidasi kategori dan nama menu.
b. Memastikan jumlah pesanan lebih dari nol.
c. Mengambil harga menu dari struktur menu_kafe.
d. Menghitung subtotal berdasarkan harga dan jumlah.
e. Jika menu sudah ada di keranjang, jumlah dan subtotal diperbarui.
f. Jika menu belum ada, item baru ditambahkan ke dalam keranjang.

5. Penghapus dan pembaruan item
   a. hapus_item digunakan untuk menghapus item dari keranjang berdasarkan indeks yang dipilih pada tabel antarmuka.

b. update_jumlah_item digunakan untuk memperbarui jumlah item dalam keranjang. Jika jumlah baru bernilai nol atau kurang, item akan dihapus secara otomatis.

```python
def hapus_item(self, index):
    if index < 0 or index >= len(self.keranjang):
        return {"status": False, "message": "Index tidak valid"}

    item = self.keranjang.pop(index)
    return {"status": True, "message": f"{item['menu']} dihapus"}

def update_jumlah_item(self, index, jumlah_baru):
    if index < 0 or index >= len(self.keranjang):
        return {"status": False, "message": "Index tidak valid"}

    if jumlah_baru <= 0:
        return self.hapus_item(index)

    item = self.keranjang[index]
    item["jumlah"] = jumlah_baru
    item["subtotal"] = item["harga"] * jumlah_baru
    return {"status": True, "message": f"{item['menu']} diperbarui"}

def get_keranjang(self):
    return self.keranjang
```

Gambar 3. 8

6. Perhitungan total transaksi

```python
def hitung_total(self):
    return sum(item["subtotal"] for item in self.keranjang)
```

Gambar 3. 9

Fungsi ini menghitung total harga seluruh item yang ada di dalam keranjang dengen menjumlahkan nilai subtotal masing-masing item.

7. Pengelolaan metode pembayaran

```python
def set_metode_pembayaran(self, metode):
    metode_valid = ["Tunai", "QRIS", "Debit"]
    if metode not in metode_valid:
        return {"status": False, "message": "Metode pembayaran tidak valid"}

    self.metode_pembayaran = metode
    return {"status": True, "message": "Metode pembayaran diperbarui"}

def get_metode_pembayaran(self):
    return self.metode_pembayaran
```

Gambar 3. 10

a. set_metode_pembayaran digunakan untuk menetapkan metode pembayaran yang dipilih pengguna, dengan validasi metode yang diperbolehkan.
b. get_metode_pembayaran digunakan untuk mengambil metode pembayaran yang sedang aktif.

8. Pembuatan struk pembayaran

```python
def membuat_struk(self, nama_kasir="Kasir"):
    if not self.keranjang:
        return "Keranjang kosong, tidak ada transaksi"

    now = datetime.now()
    struk = "=" * 23 + "\n"
    struk += "          STRUK PEMBAYARAN\n"
    struk += "           BREWALYTICA CAFE\n"
    struk += "=" * 23 + "\n\n"
    struk += f"Tanggal: {now.strftime('%d-%m-%Y')}\n"
    struk += f"Waktu  : {now.strftime('%H:%M:%S')}\n"
    struk += f"Kasir  : {nama_kasir}\n"
    struk += "=" * 23 + "\n"

    for item in self.keranjang:
        struk += f"{item['menu']} x {item['jumlah']}\n"
        struk += f" Rp {item['harga']:,} = Rp {item['subtotal']:,}\n"

    total = self.hitung_total()
    struk += "=" * 23 + "\n"
    struk += f"Total : Rp {total:,}\n"
    struk += f"Metode: {self.metode_pembayaran}\n"
    struk += "=" * 23 + "\n"
    struk += "Terima kasih!\n"
    struk += "=" * 23 + "\n"

    return struk
```

Gambar 3. 11

Fungsi ini menghasilkan struk pembayaran dalam format teks yang rapi dan mudah dibaca. Kemudian sistem akan menampikan informasi melalui dialog antarmuka pengguna setelah transaksi berhasil.

9. Pembuatan nomor struk

```python
def _membuat_no_struk(self):
    hari_ini = datetime.now().strftime("%Y%m%d")
    count = 0

    try:
        with open(self.csv_file, "r", encoding="utf-8") as f:
            for row in csv.DictReader(f):
                if row["No_Struk"].startswith(f"-{hari_ini}"):
                    count += 1
    except:
        pass

    return f"-{hari_ini}-{count+1:04d}"
```

Gambar 3. 12

Fungsi ini membuat nomor struk unik berdasarkan tanggal transaksi dan urutan transaksi pada hari tersebut. Kemudian sistem akan membuat format nomor struk (--YYYYMMDD-XXX), format ini memastikan setiap transaksi memiliki identitas yang unik dan terstruktur.

10. enyimpanan transaksi

```python
def simpan_transaksi(self, nama_kasir="Kasir"):
    if not self.keranjang:
        return {"status": False, "message": "Keranjang kosong"}

    no_struk = self._membuat_no_struk()
    now = datetime.now()
    total = self.hitung_total()

    try:
        with open(self.csv_file, "a", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            for item in self.keranjang:
                writer.writerow([
                    no_struk,
                    now.strftime("%Y-%m-%d"),
                    now.strftime("%H:%M:%S"),
                    nama_kasir,
                    item["menu"],
                    item["kategori"],
                    item["harga"],
                    item["jumlah"],
                    item["subtotal"],
                    total,
                    self.metode_pembayaran
                ])
        return {"status": True, "nomor_struk": no_struk}
    except Exception as e:
        return {"status": False, "message": str(e)}
```

Fungsi ini menyimpan data transaksi ke dalam file data_transaksi.csv. Berikut alur kerjanya:

a. Membuat nomor struk.
b. Mengambil waktu dan tanggal transaksi.
c. Menyimpan setiap item dalam keranjang sebagai satu baris data.
d. Menyertakan total transaksi dan metode pembayaran.

11. Penyelesaian transaksi

```python
def selesai_transaksi(self, nama_kasir="Kasir"):
    if not self.keranjang:
        return {"status": False, "message": "Keranjang kosong"}

    struk = self.membuat_struk(nama_kasir)
    result = self.simpan_transaksi(nama_kasir)

    if not result["status"]:
        return result

    self.keranjang = []
    self.metode_pembayaran = ""

    return {
        "status": True,
        "struk": struk,
        "nomor_struk": result["nomor_struk"]
    }
```

Gambar 3. 216

Fungsi ini mengelola seluruh proses akhir transaksi. Berikut alur kerjanya:

a. Membuat struktur pembayaran.
b. Menyimpan data transaksi ke file CSV.
c. Mengosongkan keranjang dan metode pembayaran.
d. Mengembalikan hasil transaksi ke antarmuka pengguna.

12. Reset keranjang

```python
def reset_keranjang(self):
    self.keranjang = []
    self.metode_pembayaran = ""
    return {"status": True, "message": "Keranjang dikosongkan"}
```

Gambar 3. 420

Fungsi ini digunakan untuk mengosongkan keranjang belanja dan mereset metode pembayaran, biasanya saat pengguna melakukan logout.

3.1.5. Sistem Analisis

Kelas SistemAnalisis berfungsi sebagai modul pengolahan dan analisis data transaksi pada sistem kasir brewlytica. Kelas ini bertanggung jawab dalam membaca data transaksi dari file CSV, membersihkan data, serta menghasilkan informasi penjualan dalam bentuk ringkasan dan grafik tren penjualan mingguan dan bulanan.

1. Insialisasi kelas

```python
class SistemAnalisis:
    def __init__(self, csv_file=TRANSAKSI_FILE):
        self.csv_file = csv_file
```

Gambar 3. 624

Fungsi __init__ digunakan untuk menginisalisasi sistem analisis dengan menentukan sumber data transaksi yang akan digunakan. Berikut alur kerjanya:

   a. Menyimpan path file transaksi ke dalam atribut kelas.
   b. Menjadikan file CSV sebagai satu-satunya sumber data analisis penjualan.

2. Pemrosesan dan pemuatan data

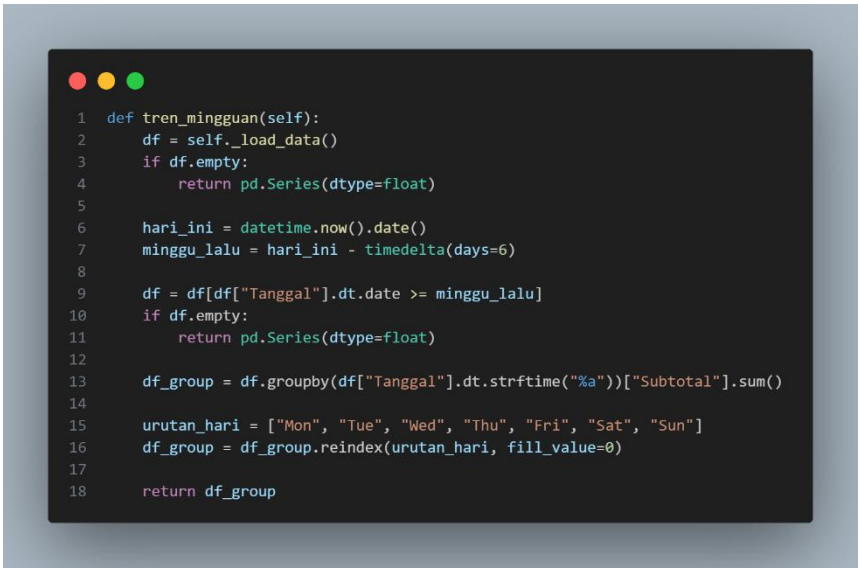
```python
def _load_data(self):
    if not os.path.exists(self.csv_file):
        return pd.DataFrame()

    df = pd.read_csv(self.csv_file, encoding="utf-8", on_bad_lines="skip")

    if df.empty:
        return df

    df["Tanggal"] = df["Tanggal"].astype(str).str.strip()
    df["Tanggal"] = pd.to_datetime(
        df["Tanggal"],
        format="%Y-%m-%d",
        errors="coerce"
    )
    df["Subtotal"] = pd.to_numeric(df["Subtotal"], errors="coerce")
    df["Jumlah"] = pd.to_numeric(df["Jumlah"], errors="coerce")
    df = df.dropna(subset=["Subtotal", "Jumlah"])

    return df
```

Gambar 3. 828

Method ini digunakan untuk memuat data transaksi dari file CSV ke dalam bentuk data *frame* menggunakan *library* pandas. Berikut alur kerjanya:

a. Mengecek keberadaan file transaksi.
b. Membaca data CSV dengan penanganan baris yang tidak valid.
c. Membersihkan data tanggal denga mengonversi kolom tanggal ke format *datetime*.
d. Mengonversi kolom subtotal dan jumlah menjadi data numerik.
e. Menghapus data yang tidak valid atau kosong.

3. Analisis tren penjualan mingguan



```python
def tren_mingguan(self):
    df = self._load_data()
    if df.empty:
        return pd.Series(dtype=float)

    hari_ini = datetime.now().date()
    minggu_lalu = hari_ini - timedelta(days=6)

    df = df[df["Tanggal"].dt.date >= minggu_lalu]
    if df.empty:
        return pd.Series(dtype=float)

    df_group = df.groupby(df["Tanggal"].dt.strftime("%a"))["Subtotal"].sum()

    urutan_hari = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
    df_group = df_group.reindex(urutan_hari, fill_value=0)

    return df_group
```

Gambar 3. 1032

Fungsi ini menghasilkan data tren penjualan selama tujuh hari terakhir. Berikut alur kerjanya:

a. Memuat data transaksi menggunakan _load_data.
b. Memfilter data berdasarkan rentang tujuh hari terakhir.
c. Mengelompokkan data berdasarkan hari (Senin-Minggu).
d. Menjumlahkan nilai subtotal penjualan untuk setiap hari.
e. Mengurutkan hasil analisis agar sesuai dengan urutan hari.

4. Analisis tren penjualan bulanan

Fungsi ini menghasilkan data tren penjualan bulanan dalam satu tahun. Berikut alur kerjanya:

a. Memuat data transaksi.
b. Memfilter data berdasarkan tahun yang dipilih.
c. Mengelompokkan data berdasarkan bulan.
d. Menjumlahkan subtotal transaksi setiap bulan.
e. Menyesuaikan label bulan ke dalam format singkatan nama bulan.

```python
def tren_bulanan(self, bulan=None, tahun=None):
    df = self._load_data()
    if df.empty:
        return pd.Series(dtype=float)

    if tahun is None:
        tahun = df['Tanggal'].dt.year.max()

    df = df[df["Tanggal"].dt.year == tahun]
    if df.empty:
        return pd.Series(dtype=float)

    df_group = (df.groupby(df["Tanggal"].dt.month)["Subtotal"].sum().reindex(range(1, 13), fill_value=0))

    nama_bulan = {
        1: "Jan", 2: "Feb", 3: "Mar", 4: "Apr",
        5: "Mei", 6: "Jun", 7: "Jul", 8: "Agu",
        9: "Sep", 10: "Okt", 11: "Nov", 12: "Des"
    }

    df_group.index = df_group.index.map(nama_bulan)

    return df_group
```

Gambar 3. 1440

5. Analisis ringkasan mingguan

```python
def analisis_mingguan(self):
    df = self._load_data()
    if df.empty:
        return {"status": False, "message": "Belum ada data transaksi"}

    hari_ini = datetime.now().date()
    minggu_lalu = hari_ini - timedelta(days=7)

    df_satu_minggu = df[df["Tanggal"].dt.date >= minggu_lalu]

    if df_satu_minggu.empty:
        return {"status": True, "total": 0, "jumlah_item": 0}

    total_pendapatan = df_satu_minggu["Subtotal"].sum()
    total_item_terjual = df_satu_minggu["Jumlah"].sum()

    return {
        "status": True,
        "total": int(total_pendapatan),
        "jumlah_item": int(total_item_terjual),
        "rentang": f"{minggu_lalu} s.d. {hari_ini}"
    }
```

Gambar 3. 1236

Fungsi ini menghasilkan ringkasan data penjualan selama satu minggu terakhir.

6. Analisis ringkasan bulanan

Fungsi ini menghasilkan ringkasan data penjualan dalam satu bulan tertentu.

```python
def analisis_bulanan(self, bulan=None, tahun=None):
    df = self._load_data()
    if df.empty:
        return {"status": False, "message": "Belum ada data transaksi"}

    sekarang = datetime.now()
    if bulan is None:
        bulan = sekarang.month
    if tahun is None:
        tahun = sekarang.year

    df_month = df[(df["Tanggal"].dt.month == bulan) & (df["Tanggal"].dt.year == tahun)]

    if df_month.empty:
        return {"status": True, "total": 0, "jumlah_item": 0}

    total_pendapatan = df_month["Subtotal"].sum()
    total_item_terjual = df_month["Jumlah"].sum()

    return {
        "status": True,
        "bulan": bulan,
        "tahun": tahun,
        "total": int(total_pendapatan),
        "jumlah_item": int(total_item_terjual)
    }
```

Gambar 3. 1644

3.1.6. *Controller*

Controller pada sistem kasir brewlytica berfungsi sebagai penghubung antara modul sistem (logika bisnis dan pengolahan data) dengan antarmuka pengguna (GUI). *Controller* bertugas menerima input dari antarmuka, melakukan validasi awal, serta meneruskan proses ke sistem yang sesuai. Dengan adanya *controller*, antarmuka pengguna tidak berinteraksi langsung dengan data maupun logika inti sistem.

1. Auth *controller*

```python
class AuthController:
    def __init__(self, auth_system, main_app):
        self.auth_system = auth_system
        self.main_app = main_app

    def handle_login(self, username, password):
        if not username or not password:
            return False, "Username dan password harus diisi"

        success, result = self.auth_system.login(username, password)
        if success:
            self.main_app.current_kasir = result
            return True, result
        return False, result

    def handle_signup(self, username, password, nama_kasir):
        if not username or not password or not nama_kasir:
            return False, "Semua persyaratan harus diisi"

        return self.auth_system.signup(username, password, nama_kasir)
```

1. Inisialisasi auth *controller*

Method ini digunakan untuk menghubungkan controller dengan sistem autentikasi dan aplikasi utama. Berikut alur kerjanya:

1) Menyimpan referensi ke objek sistem auth.
2) Menyimpan referensi ke objek *main app* untuk mengatur navigasi antar halaman.
3) Menginsialisasi *controller* sebagai penghubung antara UI dan sistem.

2. Proses login

   Fungsi tersebut digunakan untuk mengelola proses login kasir yang berasal dari antarmuka pengguna. Berikut alur kerjanya:

   1) Memeriksa apakah *username* dan *password* telah diisi.
   2) Meneruskan data login ke metode login dan sistem auth.
   3) Jika login berhasil, menyimpan nama kasir ke dalam variabel *current_kasir* pada *main app*.
   4) Mengembalikan status login dan pesan hasil autentikasi ke antarmuka pengguna.

3. Proses signup

   Fungsi tersebut digunakan untuk mengelola proses pendaftaran akun kasir baru. Berikut alur kerjanya:

   1) Memastikan seluruh input pendaftaran telah diisi.
   2) Meneruskan data pendaftaran ke metode signup pada sistem auth.
   3) Mengembalikan hasil proses pendaftaran ke antarmuka pengguna.

2. Kasir *controll*

   a. Inisialisasi kasir *controller*

```python
class KasirController:
    def __init__(self, sistem_kasir, main_app):
        self.sistem_kasir = sistem_kasir
        self.main_app = main_app

    def tambah_ke_keranjang(self, kategori, menu, jumlah):
        try:
            jumlah_int = int(jumlah)
            return self.sistem_kasir.tambah_item(kategori, menu, jumlah_int)
        except ValueError:
            return {"status": False, "message": "Jumlah harus berupa angka"}

    def hapus_dari_keranjang(self, index):
        return self.sistem_kasir.hapus_item(index)

    def proses_pembayaran(self, metode):
        if not metode or metode == "Pilih Metode":
            return {"status": False, "message": "Pilih metode pembayaran terlebih dahulu"}

        result = self.sistem_kasir.set_metode_pembayaran(metode)
        if not result["status"]:
            return result

        return self.sistem_kasir.selesai_transaksi(self.main_app.current_kasir)
```

Method tersebut digunakan untuk menghubungkan *controller* dengan sistem kasir dan aplikasi utama. Berikut alur kerjanya:

1) Menyimpan referensi ke objek sistem kasir.
2) Menyimpan referensi ke objek *main app* untuk mendapatkan informasi kasir yang sedang aktif.

b. Menambahkan item ke keranjang

Fungsi tersebut mengelola input penambahan item dari antarmuka kasir. Berikut alur kerjanya:

1) Mengonversi input jumlah menjadi bilangan bulat.
2) Menangani kesalahan input jika jumlah bukan angka.
3) Meneruskan data ke metode tambah_item pada sistem kasir.
4) Mengembalikan hasil proses ke antarmuka pengguna.

c. Menghapus item dari keranjang

Fungsi tersebut meneruskan permintaan penghapus item dari antarmuka ke sistem kasir berdasarkan indeks label.

d. Proses pembayaran

Fungsi tersebut mengelola proses pembayaran transaksi. Berikut alur kerjanya:

1) Memastikan metode pemabayaran telah dipilih.
2) Menetapkan metode pembayaran pada sistem kasir.
3) Menyelesaikan transaksi dengan metode selesai_transaksi.
4) Mengembalikan hasil transaksi dan struk ke antarmuka pengguna.

3. Analisis *controller*

Analisis *controller* berfungsi sebagai penghubung antarmuka

```python
class AnalisisController:
    def __init__(self, sistem_analisis):
        self.sistem_analisis = sistem_analisis

    def get_tren_mingguan(self):
        return self.sistem_analisis.tren_mingguan()

    def get_tren_bulanan(self, bulan=None, tahun=None):
        return self.sistem_analisis.tren_bulanan(bulan, tahun)

    def get_analisis_mingguan(self):
        return self.sistem_analisis.analisis_mingguan()

    def get_analisis_bulanan(self, bulan=None, tahun=None):
        return self.sistem_analisis.analisis_bulanan(bulan, tahun)
```

analisis dan modul sistem analisis. Kelas ini menyediakan akses

terkontrol terhadap data analisis penjualan. Berikut akses tren dan analisis penjualan:

a. get_tren_mingguan mengambil data tren penjualan mingguan dari sistem analisis.
b. get_tren_bulanan mengambil data tren penjualan bulanan dari sistem analisis.
c. get_analisis_mingguan mengambil ringkasan analisis penjualan mingguan.
d. get_analisis_bulanan mengambil ringkasan penjualan bulanan.

3.1.7. *Class* GUI

1. *class* GUI Login

Class tersebut digunakan untuk membuat GUI Login dan memanggil fungsi yang telah dibuat.

Pengaturan *layout*:

- Membuat *grid layout* untuk penempatan komponen yang terorganisir.
- Menambahkan label judul, *input username*, *input password*.
- Menyertakan tombol login dan *link signup*.
- Menggunakan *spacing* dan margin yang tepat.
- Mengambil nilai input dari *filed teks.*
- Memanggil metode *controller* untuk validasi.
- Menampilkan *message box* yang sesuai.
- Menavigasi ke halaman kasir jika berhasil.

Fungsi *reset form*:

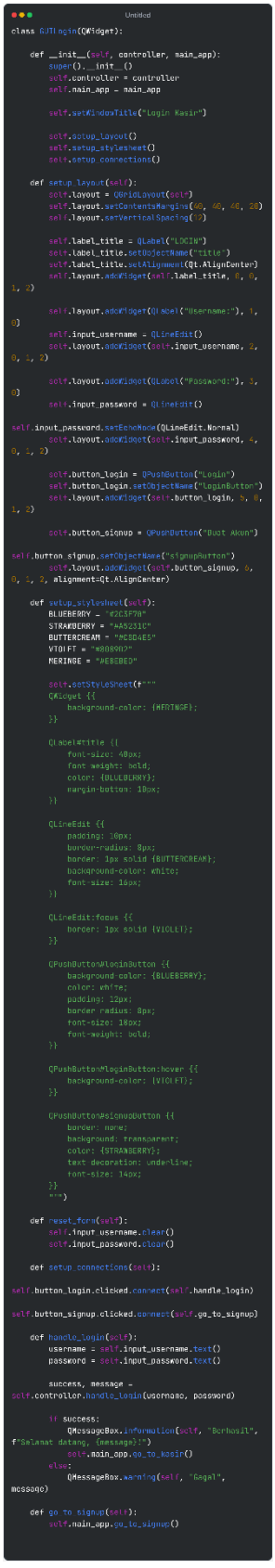
Fungsi ini digunakan untuk mengosongkan *field username* dan *password*. Dipanggil setalah login berhasil atau saat user kembali dari halaman signup supaya *field* tidak menyimpan data lama.

Penanganan Event

- Mengambil nilai input dari field teks
- Memanggil metode controller untuk validasi
- Menampilkan message box yang sesuai
- Menavigasi ke halaman kasir jika berhasil

Fungsi Go to Signup:

Fungsi untuk navigasi ke halaman signup. Dipanggil saat user klik tombol "Buat Akun".


Untitled
class GUILogin(QWidget):
def __init__(self, controller, main_app):
super().__init__()
self.controller = controller
self.main_app = main_app
self.setWindowTitle("Login Kasir")
self.setup_layout()
self.setup_stylesheet()
self.setup_connections()
def setup_layout(self):
self.layout = QVBoxLayout(self)
self.layout.setContentsMargins(40, 40, 40, 20)
self.layout.setVerticalSpacing(12)
self.label_title = QLabel("LOGIN")
self.label_title.setObjectName("title")
self.label_title.setAlignment(Qt.AlignCenter)
self.layout.addWidget(self.label_title, 0, 0,
1, 2)
self.layout.addWidget(QLabel("Username:"), 1,
0)
self.input_username = QLineEdit()
self.layout.addWidget(self.input_username, 2,
0, 1, 2)
self.layout.addWidget(QLabel("Password:"), 3,
0)
self.input_password = QLineEdit()
self.input_password.setEchoMode(QLineEdit.Normal)
self.layout.addWidget(self.input_password, 4,
0, 1, 2)
self.button_login = QPushButton("Login")
self.button_login.setObjectName("loginButton")
self.layout.addWidget(self.button_login, 5, 0,
1, 2)
self.button_signup = QPushButton("Buat Akun")
self.button_signup.setObjectName("signupButton")
self.layout.addWidget(self.button_signup, 6,
0, 1, 2, alignment=Qt.AlignCenter)
def setup_stylesheet(self):
BLUEBERRY = "#2C3F70"
STRAWBERRY = "#A9231C"
BUTTERCREAM = "#C5D4E5"
VIOLET = "#8089D7"
MERINGE = "#EBEBE0"
self.setStyleSheet(f"""
QWidget {{
background-color: {MERINGE};
}}
QLabel#title {{
font-size: 40px;
font-weight: bold;
color: {BLUEBERRY};
margin-bottom: 10px;
}}
QLineEdit {{
padding: 10px;
border-radius: 8px;
border: 1px solid {BUTTERCREAM};
background-color: white;
font-size: 16px;
}}
QLineEdit:focus {{
border: 1px solid {VIOLET};
}}
QPushButton#loginButton {{
background-color: {BLUEBERRY};
color: white;
padding: 12px;
border-radius: 8px;
font-size: 18px;
font-weight: bold;
}}
QPushButton#loginButton:hover {{
background-color: {VIOLET};
}}
QPushButton#signupButton {{
border: none;
background: transparent;
color: {STRAWBERRY};
text-decoration: underline;
font-size: 14px;
}}
""")
def reset_form(self):
self.input_username.clear()
self.input_password.clear()
def setup_connections(self):
self.button_login.clicked.connect(self.handle_login)
self.button_signup.clicked.connect(self.go_to_signup)
def handle_login(self):
username = self.input_username.text()
password = self.input_password.text()
success, message =
self.controller.handle_login(username, password)
if success:
QMessageBox.information(self, "Berhasil",
f"Selamat datang, {message}!")
self.main_app.go_to_kasir()
else:
QMessageBox.warning(self, "Gagal",
message)
def go_to_signup(self):
self.main_app.go_to_signup()

2. *Class* GUI Signup

Kelas GUISignup berfungsi untuk menampilkan dan mengelola halaman pendaftaran akun kasir baru pada aplikasi Sistem Kasir Brewalytica Café. Kelas ini menerima *controller* untuk menangani logika pendaftaran dan main_app untuk navigasi antar halaman. Pada saat inisialisasi, GUI Signup mengatur judul window "Sign Up Kasir" serta memanggil metode pengaturan layout, stylesheet, dan koneksi event. Layout halaman signup dibangun menggunakan QGridLayout yang berisi tombol kembali ke halaman login, judul "SIGN UP", tiga field input (nama kasir, username, dan password), serta tombol Sign Up untuk mengirim data pendaftaran. Setiap field dilengkapi placeholder text agar memudahkan pengguna dalam mengisi data.

Event handling menghubungkan tombol Sign Up dengan fungsi handle_signup, sedangkan tombol kembali dihubungkan dengan fungsi go_to_login. Fungsi handle_signup() bertugas mengambil data input, mengirimkannya ke controller untuk validasi dan penyimpanan akun, serta menampilkan pesan berhasil atau gagal. Jika pendaftaran berhasil, pengguna akan diarahkan kembali ke halaman login. Fungsi go_to_login() digunakan untuk kembali ke halaman login dengan mereset form login agar tetap bersih dan siap digunakan kembali.

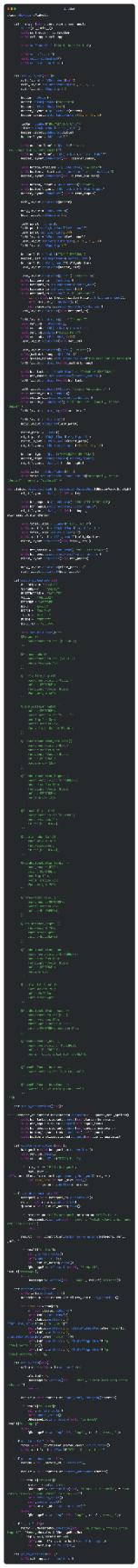
3. *Class* GUI Kasir

Kelas GUIKasir berfungsi sebagai antarmuka utama kasir untuk melakukan transaksi penjualan pada Sistem Kasir Brewalytica Café. Kelas ini menghubungkan interaksi pengguna dengan KasirController sebagai pengelola logika transaksi, serta MainApp sebagai pengatur navigasi halaman. Pada saat inisialisasi, GUIKasir menyimpan referensi controller dan main application, mengatur judul window, lalu memanggil metode setup_layout, setup_stylesheet, dan setup_connections untuk membangun tampilan, mengatur gaya visual, serta menghubungkan event antarmuka dengan fungsi yang sesuai.

Tampilan GUIKasir terdiri atas tiga bagian utama, yaitu header, panel input pesanan, dan panel keranjang belanja. Header menampilkan judul aplikasi, informasi kasir yang sedang aktif, serta tombol Analisis dan Logout. Panel input pesanan menyediakan pemilihan kategori menu, pemilihan menu berdasarkan kategori, input jumlah barang, tombol tambah ke keranjang, dan pemilihan metode pembayaran. Panel keranjang belanja menampilkan daftar item yang dipilih dalam bentuk tabel, tombol hapus item terpilih, tampilan total transaksi, serta tombol proses pembayaran.

Fungsi interaktif pada GUIKasir dijalankan melalui koneksi event, seperti perubahan kategori untuk memperbarui daftar menu, tombol tambah untuk memasukkan item ke keranjang, tombol hapus untuk menghapus item dari tabel, tombol proses pembayaran untuk menyelesaikan transaksi sekaligus menampilkan struk, tombol logout untuk keluar dari sesi kasir, dan tombol analisis untuk berpindah ke halaman analisis penjualan.

Dengan struktur tersebut, GUIKasir memungkinkan proses transaksi berjalan terarah dan konsisten, mulai dari input pesanan, pembaruan keranjang, perhitungan total, hingga penyelesaian pembayaran dan pencatatan transaksi.



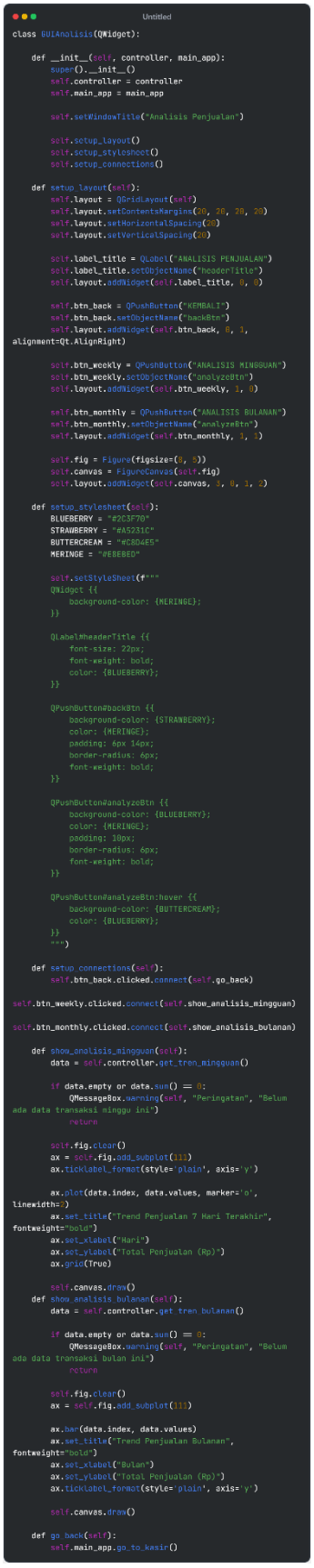
4. *Class* GUI Analisis

Kelas GUIAnalisis berfungsi sebagai antarmuka untuk menampilkan hasil analisis penjualan pada Sistem Kasir Brewalytica Café dalam bentuk grafik. Halaman ini memungkinkan pengguna melihat tren penjualan secara mingguan dan bulanan berdasarkan data transaksi yang tersimpan.

Pada saat inisialisasi, GUIAnalisis menerima controller analisis dan main application untuk mengatur logika pengolahan data serta navigasi halaman. Selanjutnya, kelas ini membangun tampilan menggunakan QGridLayout yang terdiri dari judul halaman, tombol kembali, tombol analisis mingguan, tombol analisis bulanan, dan area grafik.

Area grafik menggunakan Figure dan FigureCanvas dari Matplotlib yang terintegrasi dengan PyQt5 sehingga grafik dapat ditampilkan langsung di dalam aplikasi. Analisis mingguan ditampilkan dalam bentuk grafik garis untuk menunjukkan tren penjualan tujuh hari terakhir, sedangkan analisis bulanan ditampilkan dalam bentuk grafik batang untuk memperlihatkan total penjualan per bulan.

Tombol kembali digunakan untuk mengarahkan pengguna kembali ke halaman kasir melalui mekanisme navigasi pada main application. Dengan adanya GUIAnalisis, sistem kasir tidak hanya berfungsi untuk transaksi, tetapi juga menyediakan informasi evaluasi penjualan yang mudah dipahami dan bermanfaat bagi pengguna.



```
Untitled
class GUIAnalisis(QWidget):

    def __init__(self, controller, main_app):
        super().__init__()
        self.controller = controller
        self.main_app = main_app

        self.setWindowTitle("Analisis Penjualan")

        self.setup_layout()
        self.setup_stylesheet()
        self.setup_connections()

    def setup_layout(self):
        self.layout = QGridLayout(self)
        self.layout.setContentsMargins(20, 20, 20, 20)
        self.layout.setHorizontalSpacing(20)
        self.layout.setVerticalSpacing(20)

        self.label_title = QLabel("ANALISIS PENJUALAN")
        self.label_title.setObjectName("headerTitle")
        self.layout.addWidget(self.label_title, 0, 0)

        self.btn_back = QPushButton("KEMBALI")
        self.btn_back.setObjectName("backBtn")
        self.layout.addWidget(self.btn_back, 0, 1,
alignment=Qt.AlignRight)

        self.btn_weekly = QPushButton("ANALISIS MINGGUAN")
        self.btn_weekly.setObjectName("analyzeBtn")
        self.layout.addWidget(self.btn_weekly, 1, 0)

        self.btn_monthly = QPushButton("ANALISIS BULANAN")
        self.btn_monthly.setObjectName("analyzeBtn")
        self.layout.addWidget(self.btn_monthly, 1, 1)

        self.fig = Figure(figsize=(8, 5))
        self.canvas = FigureCanvas(self.fig)
        self.layout.addWidget(self.canvas, 2, 0, 1, 2)

    def setup_stylesheet(self):
        BLUEBERRY = "#2C3F70"
        STRAWBERRY = "#A5231C"
        BUTTERCREAM = "#C8D4E5"
        MERINGE = "#E8EBED"

        self.setStyleSheet(f"""
        QWidget {{
            background-color: {MERINGE};
        }}

        QLabel#headerTitle {{
            font-size: 22px;
            font-weight: bold;
            color: {BLUEBERRY};
        }}

        QPushButton#backBtn {{
            background-color: {STRAWBERRY};
            color: {MERINGE};
            padding: 6px 14px;
            border-radius: 6px;
            font-weight: bold;
        }}

        QPushButton#analyzeBtn {{
            background-color: {BLUEBERRY};
            color: {MERINGE};
            padding: 10px;
            border-radius: 6px;
            font-weight: bold;
        }}

        QPushButton#analyzeBtn:hover {{
            background-color: {BUTTERCREAM};
            color: {BLUEBERRY};
        }}
        """)

    def setup_connections(self):
        self.btn_back.clicked.connect(self.go_back)

self.btn_weekly.clicked.connect(self.show_analisis_mingguan)

self.btn_monthly.clicked.connect(self.show_analisis_bulanan)

    def show_analisis_mingguan(self):
        data = self.controller.get_tren_mingguan()

        if data.empty or data.sum() == 0:
            QMessageBox.warning(self, "Peringatan", "Belum
ada data transaksi minggu ini")
            return

        self.fig.clear()
        ax = self.fig.add_subplot(111)
        ax.ticklabel_format(style='plain', axis='y')

        ax.plot(data.index, data.values, marker='o',
linewidth=2)
        ax.set_title("Trend Penjualan 7 Hari Terakhir",
fontweight="bold")
        ax.set_xlabel("Hari")
        ax.set_ylabel("Total Penjualan (Rp)")
        ax.grid(True)

        self.canvas.draw()
    def show_analisis_bulanan(self):
        data = self.controller.get_tren_bulanan()

        if data.empty or data.sum() == 0:
            QMessageBox.warning(self, "Peringatan", "Belum
ada data transaksi bulan ini")
            return

        self.fig.clear()
        ax = self.fig.add_subplot(111)

        ax.bar(data.index, data.values)
        ax.set_title("Trend Penjualan Bulanan",
fontweight="bold")
        ax.set_xlabel("Bulan")
        ax.set_ylabel("Total Penjualan (Rp)")
        ax.ticklabel_format(style='plain', axis='y')

        self.canvas.draw()

    def go_back(self):
        self.main_app.go_to_kasir()
```

Gambar 3. 1848

3.1.8. *class Main App*

Kelas MainApp berfungsi sebagai pengendali utama aplikasi Sistem Kasir Brewalytica Café. Kelas ini bertanggung jawab menginisialisasi seluruh sistem, controller, dan halaman antarmuka pengguna, serta mengatur navigasi antar halaman aplikasi.

Pada saat inisialisasi, MainApp membuat sistem autentikasi, sistem kasir, dan sistem analisis, kemudian menghubungkannya dengan controller masing-masing. Atribut current_kasir digunakan untuk menyimpan nama kasir yang sedang login dan ditampilkan pada halaman kasir.

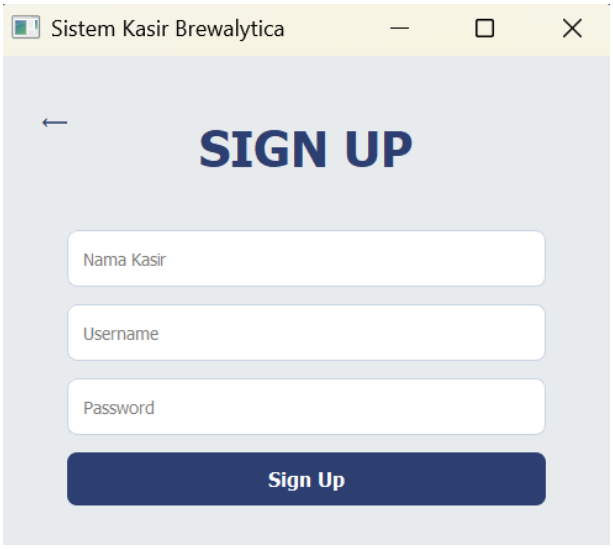
Seluruh halaman GUI, yaitu login, signup, kasir, dan analisis, dikelola menggunakan QStackedWidget sehingga perpindahan halaman dapat dilakukan dalam satu jendela. Fungsi navigasi seperti go_to_login, go_to_signup, go_to_kasir, dan go_to_analisis digunakan untuk mengatur perpindahan halaman sekaligus menyesuaikan ukuran tampilan window.
Dengan peran tersebut, kelas MainApp memastikan integrasi antara sistem, controller, dan antarmuka pengguna berjalan secara terstruktur dan konsisten.

Untitled

```
class MainApp(QWidget):

    def __init__(self):
        super().__init__()

        self.auth_system = SistemAuth()
        self.sistem_kasir = SistemKasir()
        self.sistem_analisis = SistemAnalisis()

        self.current_kasir = "Guest"

        self.auth_controller =
AuthController(self.auth_system, self)
        self.kasir_controller =
KasirController(self.sistem_kasir, self)
        self.analisis_controller =
AnalisisController(self.sistem_analisis)

        self.login_page = GUILogin(self.auth_controller,
self)
        self.signup_page = GUISignup(self.auth_controller,
self)
        self.kasir_page = GUIKasir(self.kasir_controller,
self)
        self.analisis_page =
GUIAnalisis(self.analisis_controller, self)

        self.stack = QStackedWidget()
        self.stack.addWidget(self.login_page)
        self.stack.addWidget(self.signup_page)
        self.stack.addWidget(self.kasir_page)
        self.stack.addWidget(self.analisis_page)

        main_layout = QVBoxLayout(self)
        main_layout.addWidget(self.stack)

        self.setWindowTitle("Sistem Kasir Brewalytica")
        self.resize(500, 400)
        self.setMinimumSize(500, 400)
        self.setStyleSheet("background-color: #E8EBED;")

        self.stack.setCurrentWidget(self.login_page)

    def go_to_login(self):
        self.stack.setCurrentWidget(self.login_page)
        self.resize(500, 400)

    def go_to_signup(self):
        self.stack.setCurrentWidget(self.signup_page)
        self.resize(500, 400)

    def go_to_kasir(self):
        self.kasir_page.cashier_label.setText(f"Kasir:
{self.current_kasir}")
        self.stack.setCurrentWidget(self.kasir_page)
        self.resize(1200, 800)

    def go_to_analisis(self):
        self.stack.setCurrentWidget(self.analisis_page)
        self.resize(900, 600)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    app.setStyle("Fusion")

    main_window = MainApp()
    main_window.show()

    sys.exit(app.exec_())
```

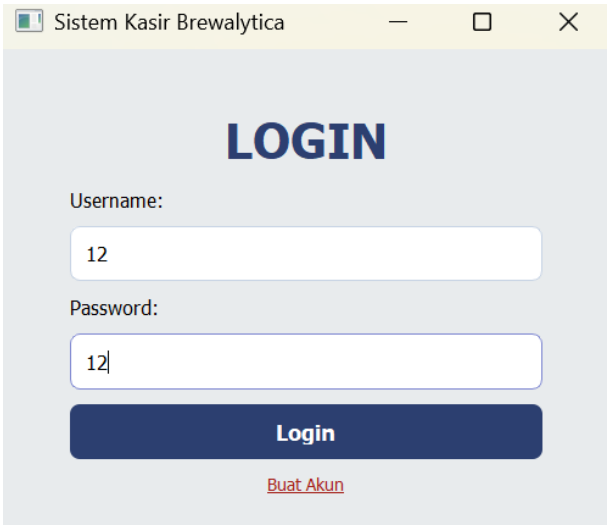
3.2. *Screenshoot* Sistem

1. Signup page



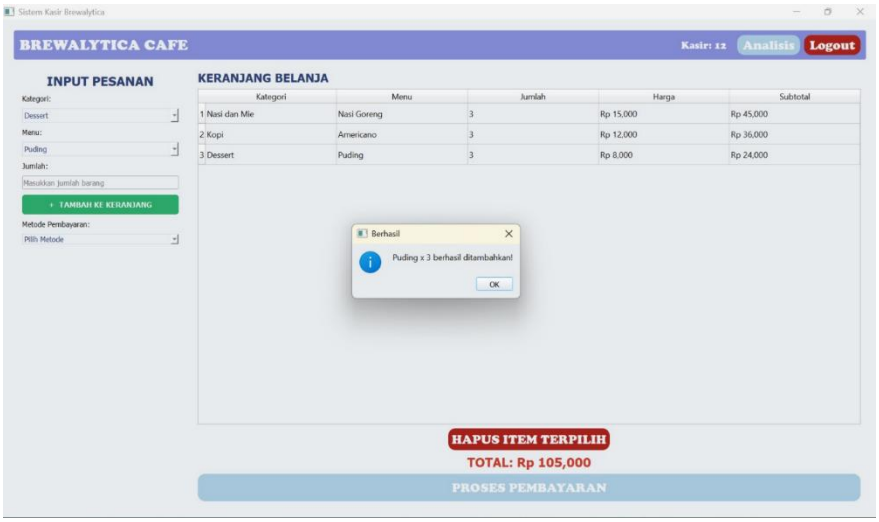
Gambar 3. 2052

2. Login page



Gambar 3. 2256

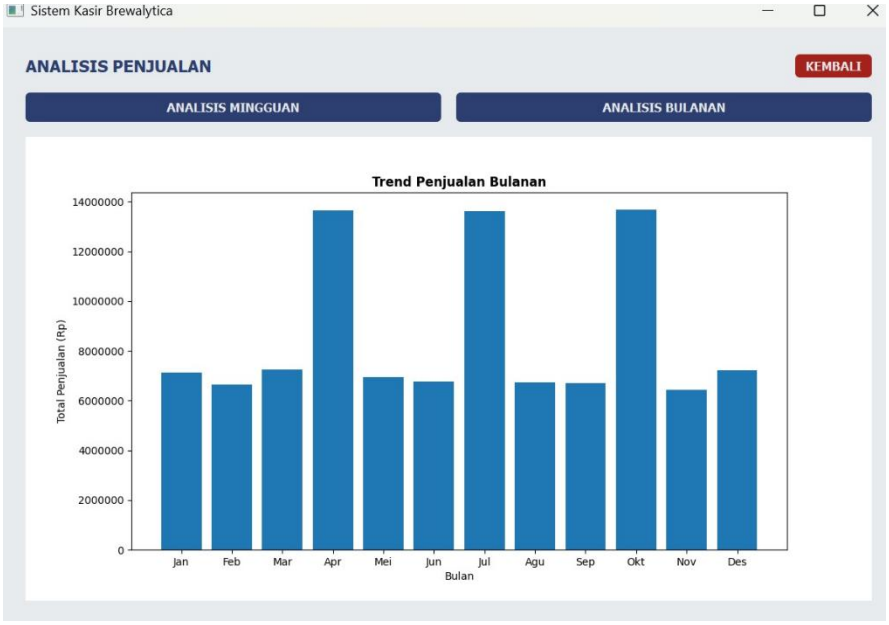
3. Kasir page



Gambar 3. 2460

4. Struk

5. Analisis page



DAFTAR PUSTAKA

Nuryamin, Y. (2025). *Perancangan aplikasi kasir pada kedai kopi*. Universitas Nusa Mandiri Repository. Nusamandiri Repository

Elavigne. (2025, March 21). *Kasir, stok, dan laporan otomatis! Begini cara modern kelola café!*. Karts.id. https://blog.karts.id/kasir-stok-laporan-cafe-otomatis