BAB I

PENDAHULUAN

Bab ini menjelaskan latar belakang, rumusan masalah, tujuan, dan rancangan solusi dari program Cake Shop 23BEE.

1.1 Latar Belakang

Subbab ini menjelaskan latar belakang perancangan program Cake Shop 23BEE.

Toko kue yang modern membutuhkan solusi *digital* untuk meningkatkan efisiensi operasional dan kepuasan pelanggan. Saat ini, pelanggan sering kali mengalami kesulitan dalam mencari dan memesan berbagai jenis roti atau *dessert* secara praktis, terutama jika mereka tinggal jauh atau memiliki keterbatasan mobilitas. Selain itu, proses pemesanan yang tidak efisien dapat menyebabkan waktu tunggu yang lama, terutama pada saat toko sedang sibuk.

Di era digital ini, pelanggan mengharapkan kemudahan dan kecepatan dalam berbelanja, termasuk dalam proses pembayaran. Metode pembayaran yang lambat atau rumit, seperti pembayaran tunai atau kartu kredit secara fisik di toko, dapat menjadi kendala dan mengurangi kepuasan pelanggan. Oleh karena itu, diperlukan sebuah *platform online* yang tidak hanya memudahkan pelanggan dalam mencari dan memesan berbagai jenis roti atau *dessert*, tetapi juga mengimplementasikan sistem pemesanan yang efisien dan menyediakan metode pembayaran *digital* yang cepat dan mudah digunakan.

Perancangan program Cake Shop 23BEE bertujuan untuk mengatasi masalah-masalah tersebut dengan menyediakan solusi digital yang komprehensif. Dengan adanya *platform online*, pelanggan dapat dengan mudah mengakses informasi tentang berbagai produk yang tersedia, melakukan pemesanan, dan memilih metode pembayaran yang paling sesuai dengan kebutuhan mereka. Sistem ini diharapkan dapat mengurangi waktu tunggu pelanggan dan meningkatkan efisiensi operasional toko, sehingga dapat memberikan pengalaman berbelanja yang lebih baik dan meningkatkan kepuasan pelanggan secara keseluruhan.

1.2 Rumusan Masalah

Subbab ini menjelaskan masalah yang mendasari program Cake Shop 23BEE. Berikut merupakan rumusan masalah yang mendasari perancangan program.

a. Bagaimana membuat *platform online* yang memudahkan pelanggan dalam mencari dan memesan berbagai jenis roti atau *dessert* secara praktis.
b. Bagaimana cara mengimplementasikan sistem pemesanan yang dapat meminimalisir waktu tunggu pelanggan.
c. Bagaimana membuat metode pembayaran digital yang cepat dan mudah digunakan untuk mengurangi kendala yang dihadapi pelanggan saat melakukan pembayaran di toko.

1.3 Tujuan

Subbab ini menjelaskan tujuan perancangan program Cake Shop 23BEE. Berikut merupakan tujuan dalam perancangan program.

a. Dapat membuat *platform online* yang memudahkan pelanggan dalam mencari dan memesan berbagai jenis roti atau *dessert* secara praktis.
b. Dapat mengimplementasikan sistem pemesanan yang dapat meminimalisir waktu tunggu pelanggan.
c. Dapat membuat metode pembayaran digital yang cepat dan mudah digunakan untuk mengurangi kendala yang dihadapi pelanggan saat melakukan pembayaran di toko.

1.4 Rancangan Solusi

Subbab ini menjelaskan rancangan solusi program Cake Shop 23BEE. Program ini dirancang dengan rincian kebutuhan fungsional sebagai berikut:

a. Aplikasi menyediakan fitur pendaftaran akun dan masuk akun
b. Aplikasi menyediakan fitur homepage
c. Aplikasi meyediakan fitur menu dan harga untuk *user* memilih menu yang akan dibeli
d. Aplikasi menyediakan fitur pemilihan *Take Away*, *Delivery*, dan *Dine In* yang bisa dipilih *user*
e. Aplikasi menyediakan fitur pengecekan pesanan dan total pembayaran yang bisa dipilih secara tunai maupun non-tunai
f. Aplikasi menyediakan fitur *invoice* setelah pembayaran

Program ini dirancang dengan rincian library antara lain, tkinter, tkinter messagebox, os, csv, subprocess, PIL, dan customtkinter (ctk).

BAB II

METODE PERANCANGAN

Bab ini menjelaskan *flowchart* metode perancangan dan *flowchart* program Cake Shop 23BEE.

2.1 *Flowchart* Metode Perancangan

Subbab ini menjelaskan *flowchart* metode perancangan program Cake Shop 23BEE.

Gambar 2.1 *Flowchart* Metode Perancangan Program Cake Shop 23BEE

2.1.1. Merumuskan Masalah

Pada tahap ini kami merumuskan masalah yang mendasari perancangan program Cake Shop 23BEE. Kami menyadari bahwa orang-orang yang tinggal jauh dan memiliki keterbatasan mobilitas kesulitan menemukan tempat yang

menyediakan berbagai jenis roti dan dessert secara praktis melalui *platform online*. Selain itu kami juga memikirkan bagaimana menyediakan sistem pesanan yang dapat meminimalissir waktu tunggu pelanggan dan membuat metode pembayaran digital yang cepat dan mudah untuk memudahkan pelanggan saat melakukan pembayaran.

2.1.2. Pengumpulan Data

Pada tahap ini kami mencari dan mengumpulkan data mengenai macam-macam jenis roti yang digemari orang-orang di suatu tempat untuk dimasukkan kedalam jenis menu di toko roti kami. Pengumpulan data bisa dengan melihat langsung perilaku orang setempat dan kesukaan orang setempat terhadap suatu jenis makanan kemudian dibandingkan dengan menu yang sudah ada, adapun cara lain dengan melihat grafik data transaksi atau produk terlaris pada toko roti yang lain yang dapat menjadi referensi dalam menentukan jenis roti yang akan dijual. Bisa juga dengan mencoba berinteraksi langsung dengan orang-orang yang memanfaatkan sosial media seperti membuat *polling* menu roti, meminta tanggapan dan saran jenis menu untuk toko roti, dan menawarkan beberapa pilihan yang cocok. Dalam penetuan harga roti, kami mempertimbangkan minat orang setempat terhadap roti dengan bahan-bahan yang digunakan, kemudian diambil harga yang tepat dan sesuai.

2.1.3. Pembuatan Program

Pada tahap pembuatan program kami membuat 10 modul diantaranya sebagai berikut.

1. **registrasi.py** yang didalamnya terdapat kode program untuk *Sign Up* dan *Sign In* dengan memasukkan *username* dan *password* yang nantinya tersimpan pada database akun.csv
2. **Homepage.py** yang di dalamnya terdapat kode program untuk menampilkan 4 menu pilihan jenis kue yang ada di 23BEE, diantaranya ada *breads*, *cakes*, *donuts*, dan a*pastry*.
3. **Breads.py** berisi kode program beberapa menu roti yang terdapat pada database breads.csv
4. **cakes.py** berisi kode program beberapa menu kue yang terdapat pada database cakes.csv

5. **donuts.py** berisi kode program beberapa menu donat yang terdapat pada database donuts.csv
6. **pastry.py** berisi kode program beberapa menu *pastry* yang terdapat pada database donuts.csv
7. **pemilihan.py** berisi kode program yang berfungsi untuk memasukkan data pembeli sesuai dengan pemilihan beberapa pilihan pengambilan barang seperti *Dine In, Delivery, Take Away.*
8. **pembayaran.py** berisi kode program yang terdapat rekap pesanan dan pemilihan metode pembayaran yang digunakan seperti tunai atau non tunai.
9. **invoice.py** berisi kode program yang memunculkan kode pembayaran dan pesan untuk pembeli sesuai dengan data pembeli yang diinput dalam modul pemilihan.
10. **button.py** berisi kode program yang berisi fungsi-fungsi button untuk memudahkan saat pemilihan.

2.1.4. Pembuatan GUI

Pada tahap pembuatan GUI dalam program Cake Shop 23BEE menggunakan Tkinter yang menyediakan elemen-elemen yang menarik. Proses dimulai dengan perancangan tampilan *login register* akan menampilkan *pop up* pesan apakah proses *login register* berhasil atau tidak. Selanjutnya pengguna akan diarahkan ke *homepage* yang menampilkan produk-produk kue yang tersedia beserta gambar dan informasi singkat seperti nama dan harga. *User* juga dapat memilih produk untuk melihat detail lebih lanjut atau menambahkannya ke keranjang belanja. Selain itu, terdapat tombol *logout* yang memungkinkan pengguna untuk keluar dari aplikasi.

Implementasi GUI menggunakan Tkinter dimulai dengan membuat jendela utama untuk halaman *login* dan registrasi, yang menampilkan form *input* untuk *username* dan *password*. Setelah pengguna berhasil *login* atau registrasi, fungsi tertentu akan menutup jendela *login* dan membuka jendela *homepage* baru. Di *homepage*, ditampilkan *header* selamat datang dan daftar produk kue dalam sebuah *frame*. Setiap produk ditampilkan dengan gambar yang dimuat menggunakan PIL (Pillow) dan informasi produk di sebelahnya. Tombol *logout* juga disediakan di bagian bawah halaman untuk memungkinkan pengguna keluar dari aplikasi. Dengan demikian, pengguna dapat dengan mudah melakukan navigasi dari proses

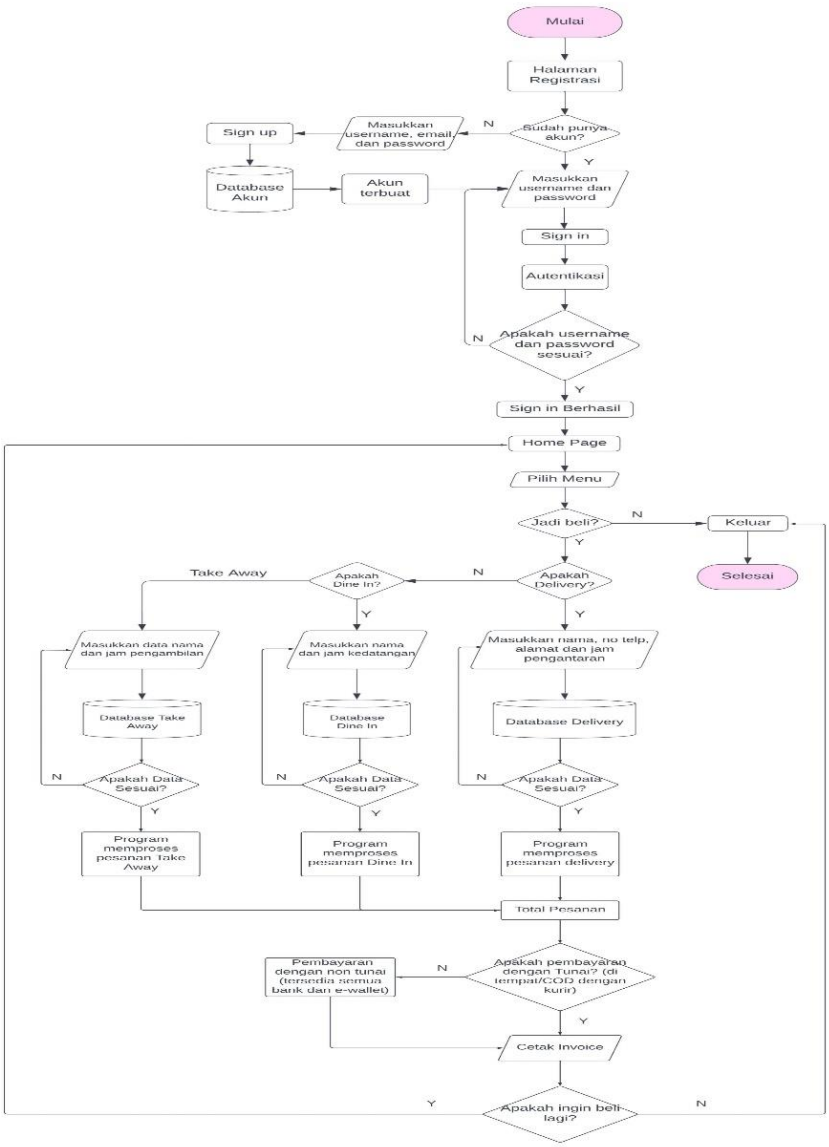
login hingga melihat dan membeli produk kue yang tersedia di toko Cake Shop 23BEE.

2.1.5. Penyusunan Laporan

Pada tahap ini, semua proses pembuatan dan hasil disusun dalam sebuah laporan. Dalam laporan tersebut menjelaskan tentang pendahuluan yang berisi latar belakang, rumusan masalah, tujuan, dan rancangan solusi. Kemudian dalam metode perancangan, terdapat penjelasan flowchart metode perancangan dan flowchart program. Lalu di hasil dan pembahasan, terdapat penjelasan kode program dan penjelasan *User Interface* (UI). Kemudian pada bab terakhir terdapat penutup yang berisi kesimpulan dari hasil perancangan program.

2.2 *Flowchart* Program

Subbab ini menjelaskan *flowchart* program Cake Shop 23BEE. Berikut merupakan *flowchart* program secara keseluruhan.



Gambar 2.2 *Flowchart* Keseluruhan Program Cake Shop 23BEE

Program dimulai dengan menu *register*, apabila akun belum terdaftar, *user* harus mendaftarkan akun terlebih dahulu melalui menu *sign up* dengan memasukkan *username*, *password*, dan *confirm password* namun apabila sudah memiliki akun sebelumnya *user* langsung bisa melakukan *sign in*. Setelah akun berhasil dibuat, *user* diminta untuk *sign in* dengan *username* dan *password* yang sudah terdaftar. Apabila *sign in* sudah berhasil, *user* diarahkan menuju ke *homepage*.

Pada *homepage*, *user* dapat memilih menu yang diinginkan. Ketika *user* memutuskan untuk membeli, mereka dapat memilih antara opsi *Take Away*, *Dine*

In, atau *Delivery*. Apabila *user* memilih diantara 3 opsi, maka *user* harus memasukkan nama dan jam kedatangan yang akan disimpan dalam *database*, kemudian program akan memproses pesanan jika data sudah sesuai.

Setelah total pesanan dihitung, *user* harus memilih metode pembayaran yang bisa dilakukan secara tunai dan non-tunai. Jika pembayaran dilakukan secara tunai, program akan mengkonfirmasi pembayaran di tempat dengan *Cash on Delivery (COD)*. Kemudian jika pembayaran sudah selesai, program akan mencetak *invoice*. *User* akan ditanya apakah ingin membeli lagi. Apabila *user* memilih ’iya’, maka akan kembali ke *homepage* dan jika ’tidak’, maka program akan selesai dan user bisa keluar dari aplikasi.

BAB III

HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil dan pembahasan perancangan program Cake Shop 23BEE.

3.1 Kode Program

Subbab ini menjelaskan fungsi yang digunakan pada program Cake Shop 23BEE.

```python
from tkinter import *
from tkinter import messagebox
import os
import csv
import subprocess
from PIL import Image, ImageTk

database_path = os.path.join(os.getcwd(), 'database', 'akun.csv')

def open_signup_window():
    root.destroy()
    main('signup')

def open_signin_window():
    root.destroy()
    main('signin')

def open_homepage():
    root.destroy()  # Menutup jendela login
    subprocess.Popen(['python', 'homepage.py'])  # Membuka homepage

def signup():
    username = user.get()
    password = code.get()
    confirm_password = confirm_code.get()

    if password == confirm_password:
        try:
            file_exists = os.path.exists(database_path)

            with open(database_path, 'a+', newline='') as file:
                writer = csv.writer(file)

                if not file_exists:
                    writer.writerow(['Username', 'Password'])

                file.seek(0)
                reader = csv.reader(file)
                next(reader, None)
                for row in reader:
                    if row and row[0] == username:
                        messagebox.showerror('Error', 'Username already exists')
                        return

            with open(database_path, 'a', newline='') as file:
                writer = csv.writer(file)
                writer.writerow([username, password])

            messagebox.showinfo('Sign up', 'Successfully signed up')
            open_signin_window()
        except Exception as e:
            messagebox.showerror('Error', f"An error occurred: {e}")
    else:
        messagebox.showerror('Invalid', "Passwords must match")

def signin():
    username = user.get()
    password = code.get()

    try:
        with open(database_path, newline='') as file:
            reader = csv.reader(file)
            credentials = {rows[0]: rows[1] for rows in reader}
    except Exception as e:
        messagebox.showerror("Error", f"Failed to read data file: {e}")
        return

    if username in credentials and credentials[username] == password:
        messagebox.showinfo("Success", "Login successful!")
        open_homepage()  # Buka homepage setelah login berhasil
    else:
        messagebox.showerror("Error", "Invalid username or password")
```

Gambar 3.1 Kode Program *Register* (1)

```python
def on_enter(e, widget, placeholder):
    if widget.get() == placeholder:
        widget.delete(0, 'end')
        if widget == code or widget == confirm_code:
            widget.config(show='*')

def on_leave(e, widget, placeholder):
    if widget.get() == '':
        widget.insert(0, placeholder)
        if widget == code or widget == confirm_code:
            widget.config(show='')

def main(action):
    global root, user, code, confirm_code
    root = Tk()
    root.title('SignIn' if action == 'signin' else 'Sign Up')
    root.geometry('925x500+300+200')
    root.configure(bg="#FFDED9")
    root.resizable(False,False)

    img_path = os.path.join( 'images', 'logo.png')
    if not os.path.exists(img_path):
        messagebox.showerror("Error", "Image file not found")
    else:
        img = Image.open(img_path)
        img = img.resize((360, 360), Image.LANCZOS)
        img = ImageTk.PhotoImage(img)
        label_img = Label(root, image=img, border=0, bg='#FFDED9')
        label_img.image = img
        label_img.place(x=60, y=50)

    frame = Frame(root, width=560, height=360, bg='#FFDED9') if action == 'signup' else Frame(root, width=350, height=350, bg='#FFDED9')
    frame.place(x=480, y=50 if action == 'signup' else 70)

    heading = Label(frame, text='Sign Up' if action == 'signup' else 'Sign in', fg='#F16A6A', bg='#FFDED9', font=('Microsoft YaHei UI Light', 23, 'bold'))
    heading.place(x=100 if action == 'signup' else 100, y=5)

    user = Entry(frame, width=25 if action == 'signup' else 36, fg='black', border=0, bg='#FFDED9', font=('Microsoft YaHei UI Light', 11))
    user.place(x=30, y=80)
    user.insert(0, 'Username')
    user.bind("<FocusIn>", lambda e: on_enter(e, user, 'Username'))
    user.bind("<FocusOut>", lambda e: on_leave(e, user, 'Username'))
    Frame(frame, width=295, height=2, bg='black').place(x=25, y=107)

    code = Entry(frame, width=25 if action == 'signup' else 36, fg='black', border=0, bg='#FFDED9', font=('Microsoft YaHei UI Light', 11))
    code.place(x=30, y=150)
    code.insert(0, 'Password')
    code.bind("<FocusIn>", lambda e: on_enter(e, code, 'Password'))
    code.bind("<FocusOut>", lambda e: on_leave(e, code, 'Password'))
    Frame(frame, width=295, height=2, bg='black').place(x=25, y=177)

    if action == 'signup':
        confirm_code = Entry(frame, width=25, fg='black', border=0, bg='#FFDED9', font=('Microsoft YaHei UI Light', 11))
        confirm_code.place(x=30, y=220)
        confirm_code.insert(0, 'Confirm Password')
        confirm_code.bind("<FocusIn>", lambda e: on_enter(e, confirm_code, 'Confirm Password'))
        confirm_code.bind("<FocusOut>", lambda e: on_leave(e, confirm_code, 'Confirm Password'))
        Frame(frame, width=295, height=2, bg='black').place(x=25, y=247)

        Button(frame, width=39, pady=7, text='Sign up', bg='#F16A6A', fg='#FFDED9', border=0, command=signup).place(x=35, y=280)
        label = Label(frame, text='Already have an account?', fg='black', bg='#FFDED9', font=('Microsoft YaHei UI Light', 9))
        label.place(x=70, y=340)
        signin_tombol = Button(frame, width=6, text="Sign in", border=0, bg='#FFDED9', cursor='hand2', fg='#57a1f8', command=open_signin_window)
        signin_tombol.place(x=215, y=341)
    else:
        Button(frame, width=39, pady=7, text='Sign in', bg='#F16A6A', fg='#FFDED9', border=0, command=signin).place(x=35, y=204)
        label = Label(frame, text="Don't have an account?", fg='black', bg='#FFDED9', font=('Microsoft YaHei UI Light', 9))
        label.place(x=75, y=270)
        sign_up = Button(frame, width=6, text='Sign up', border=0, bg='#FFDED9', cursor='hand2', fg='#57a1f8', command=open_signup_window)
        sign_up.place(x=215, y=270)

    root.mainloop()

if __name__ == "__main__":
    #main()
    import sys
    if len(sys.argv) > 1 and sys.argv[1] == 'signin':
        main('signin')
    else:
        main('signup')  # Default behavior, jika tidak ada argumen maka membuka halaman sign-up
```

Gambar 3.2 Kode Program *Register* (2)

Berikut adalah penjelasan singkat tentang beberapa fungsi kode program *register* beserta impor yang digunakan:

1. Impor Modul
    - tkinter: modul untuk membuat GUI.
    - messagebox: untuk menampilkan pesan kesalahan atau informasi.
    - os: modul untuk berinteraksi dengan sistem operasi.
    - csv: modul untuk membaca dan menulis file CSV.

- subprocess: untuk menjalankan skrip Python lain.
- PIL (Pillow): untuk memanipulasi dan menampilkan gambar.

2. Fungsi Program
   - open_signup_window(): menutup jendela saat ini dan membuka jendela pendaftaran.
   - open_signin_window(): menutup jendela saat ini dan membuka jendela masuk.
   - open_homepage(): menutup jendela saat ini dan membuka halaman beranda.
   - signup(): menangani pendaftaran pengguna.
     - Memeriksa kesesuaian kata sandi.
     - Memastikan username belum terdaftar.
     - Menyimpan data pengguna ke file CSV.
     - Mengarahkan ke jendela masuk setelah berhasil mendaftar.
   - signin(): menangani proses masuk.
     - Memeriksa kecocokan username dan kata sandi dari file CSV.
     - Membuka halaman beranda jika berhasil, atau menampilkan pesan error jika gagal.
   - on_enter(e, widget, placeholder): menghapus placeholder saat pengguna mengklik entri.
   - on_leave(e, widget, placeholder): mengembalikan placeholder jika entri kosong setelah pengguna mengklik di luar.
   - main(action): membuat jendela GUI berdasarkan aksi ('signin' atau 'signup').
     - Menampilkan formulir pendaftaran atau masuk.
     - Menambahkan elemen GUI seperti entri, tombol, dan gambar logo.
   - Blok `if __name__ == "__main__": : menentukan aksi awal berdasarkan argumen baris perintah (default ke pendaftaran jika tidak ada argumen).

```python
import customtkinter as ctk
from PIL import Image, ImageTk
import os
import subprocess

def main(app):
    global slider_label, image_index, slider_images

    image_index = 0

    def next_image():
        global image_index
        image_index = (image_index + 1) % len(slider_images)
        slider_label.configure(image=slider_images[image_index])

    def prev_image():
        global image_index
        image_index = (image_index - 1) % len(slider_images)
        slider_label.configure(image=slider_images[image_index])

    def logout():
        app.destroy()
        subprocess.Popen(['python', 'register.py', 'signin'])

    def delete_cart():
        try:
            os.remove('database/cart.csv')
        except FileNotFoundError:
            pass
        app.destroy()

    app.protocol("WM_DELETE_WINDOW", delete_cart)
```

Gambar 3.3 Kode Program *Homepage* (1)

```python
    for i, (img, text, command) in enumerate(zip(menu_images, menu_texts, menu_commands)):
        frame = ctk.CTkFrame(menu_frame,fg_color=bg_color )
        frame.grid(row=0, column=i, padx=60, pady=25)

        img_label = ctk.CTkLabel(frame, image=img, text="")
        img_label.pack(pady=(0, 10))

        btn = ctk.CTkButton(frame, text=text, font=font_text,text_color='white', fg_color=button_color, command=command)
        btn.pack(padx=(1,1))

    logout_button = ctk.CTkButton(app, text="Log Out", fg_color=button_color, text_color='white',command=logout)
    logout_button.place(relx=0.95, rely=0.05, anchor='ne')

if __name__ == "__main__":
    app = ctk.CTk()
    app.geometry("1270x710")
    app.title("23Bee Bakery")
    main(app)
    app.mainloop()
```

Gambar 3.4 Kode Program *Homepage* (2)

Berikut adalah penjelasan singkat tentang beberapa fungsi kode program *homepage* beserta impor yang digunakan:

1. Impor modul pada homepage
    - customtkinter: modul perpanjangan dari tkinter yang dapat ditambah fitur
2. Fungsi program pada homepage
    - main(app): mendefinisikan fungsi ‘main’ yang menerima satu argumen ‘app’
    - next_image(): menampilkan gambar berikutnya dalam daftar gambar yang ditampilkan

- prev_image(): menampilkan kembali gambar sebelumny dalam serangkaian gambar yang ditampilkan
- logout(): mengakhiri sesi pengguna, menghapus informasi autentikasi, dan membersihkan data sesi
- delete_cart(): menghapus produk yang dipilih dari keranjang belanja dalam aplikasi

```python
import customtkinter as ctk
import tkinter as tk
from PIL import Image, ImageTk
import pembayaran
import csv
import os

font_title = ("Baskerville Old Face", 90, "bold")
font_subtitle = ("Baskerville", 20, "bold")
font_text = ("Baskerville", 13, "bold")
ourmenu=("Baskerville Old Face", 40, "bold")
button_color = "#FFADA1"

def simpan_data_pembeli(nama, no_telp, alamat, jam, pilihan):
    folder_path = 'database'
    file_path = os.path.join(folder_path, 'datapembeli.csv')

    if not os.path.exists(folder_path):
        os.makedirs(folder_path)

    with open(file_path, mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([nama, no_telp, alamat, jam, pilihan])

def buat_pemilihan_page(app, pilihan, selected_products):
    pemilihan_window = ctk.CTkToplevel(app)
    pemilihan_window.title("Pemilihan")
    pemilihan_window.geometry("900x610")  # Sesuaikan ukuran jendela

    # Bring the pemilihan_window to the front
    pemilihan_window.attributes("-topmost", True)
    pemilihan_window.focus_force()

    # Load and set background image
    img_pathbread = os.path.join('images', 'homepage.png')
    imgbread = Image.open(img_pathbread)
    imgbread = imgbread.resize((2050, 1095), Image.LANCZOS)  # Adjust to fit the window size
    img1 = ImageTk.PhotoImage(imgbread)

    bg_label = ctk.CTkLabel(pemilihan_window, image=img1)
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)
    bg_label.image = img1  # Prevent image from being garbage collected
    bg_label.lower()

    def go_back():
        pemilihan_window.destroy()

    def confirm_selection():
        nama = name_entry.get()
        no_telp = phone_entry.get()
        alamat = address_entry.get() if pilihan == "DELIVERY" else ""
        jam = time_entry.get()
        print(f"Nama: {nama}, No Telp: {no_telp}, Alamat: {alamat}, Jam: {jam}, Opsi: {pilihan}")
```

Gambar 3.5 Kode Program Pemilihan

Berikut adalah penjelasan singkat tentang beberapa fungsi kode program pemilihan beserta impor yang digunakan:

1. Import Modul:
   - customtkinter: modifikasi dari modul `tkinter` untuk memperindah tampilan antarmuka pengguna
   - tkinter: modul utama untuk pembuatan antarmuka grafis
   - PIL: modul yang digunakan untuk memanipulasi gambar
   - pembayaran: modul untuk mebuat tagihan pembelanjaan produk

2. Fungsi program pada pemilihan
    - simpan_data_pembeli(nama, no telp, alamat, jam, pilihan): menyimpan data customer yang dibutuhkan untuk keperluan pemesanan
    - buat_pemilihan_page(app, pilihan, selected_products): memilih produk yang akan dipesan dari beberapa item yang tersedia
    - go_back(): mengembalikan pilihan ke posisi sebelumnya dalam riwayat pemesanan produk
    - confirm_selection(): menyimpan data pesanan dan konfirmasi kesepakatan yang dibuat customer terkait info pesanan

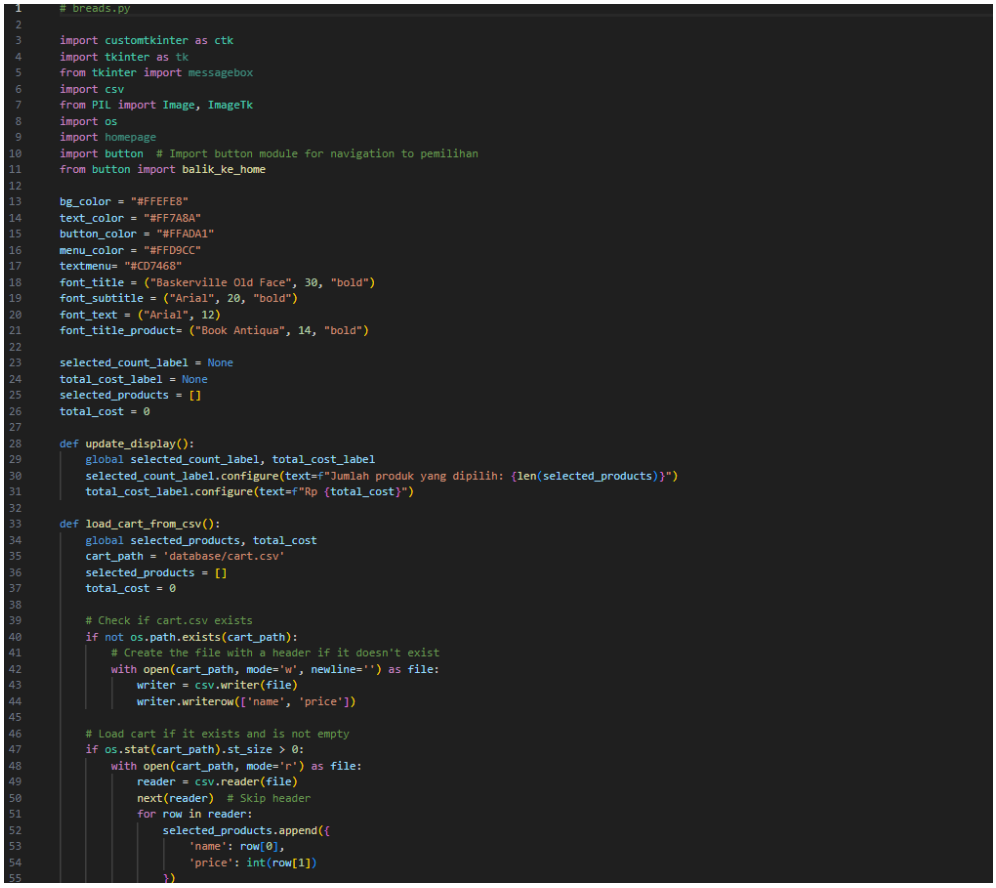
```python
# button.py
import pemilihan
import pembayaran

def menuju_ke_pemilihan(app, pilihan, selected_products):
    pemilihan.buat_pemilihan_page(app, pilihan, selected_products)

def menuju_ke_pembayaran(app, products, pilihan):
    pembayaran.buat_pembayaran_page(app, products, pilihan)

# Fungsi-fungsi navigasi lainnya bisa ditambahkan di sini
def menuju_ke_breads(app):
    for widget in app.winfo_children():
        widget.destroy()
    import breads
    breads.buat_breads_page(app)

def balik_ke_home(app):
    for widget in app.winfo_children():
        widget.destroy()
    import homepage
    homepage.main(app)

def menuju_ke_register(app):
    for widget in app.winfo_childern():
        widget.destroy()
        import Register
        Register.main()

def menuju_ke_cakes(app):
    for widget in app.winfo_children():
        widget.destroy()
    import cakes
    cakes.buat_cakes_page(app)

def menuju_ke_donuts(app):
    for widget in app.winfo_children():
        widget.destroy()
    import donuts
    donuts.buat_donuts_page(app)

def menuju_ke_pastry(app):
    for widget in app.winfo_children():
        widget.destroy()
    import pastry
    pastry.buat_pastry_page(app)
```

Gambar 3.6 Kode Program *Button*

Berikut adalah penjelasan singkat tentang beberapa fungsi kode program *button* yang digunakan:

Fungsi program pada *button*:

- menuju_ke_pemilihan(app, pilihan, selected_products): masuk ke bagian pemilihan produk yang akan dipesan
- menuju_ke_(app, products, pilihan): masuk ke bagian checkout produk yang sebelumnya sudah dimasukkan keranjang dan segera dibayar
- menuju_ke_breads(app): masuk ke bagian memilih jenis breads yang ingin dipesan
- balik_ke_home(app): kembali ke menu awal yaitu homepage, ke lobby sebelum pemesanan
- menuju_ke_register(app): masuk ke sesi pendaftaran akun sebelum order produk roti dan kue
- menuju_ke_cakes(app): masuk ke bagian memilih jenis cakes yang ingin dipesan
- menuju_ke_donuts(app): masuk ke bagian memilih jenis donuts yang ingin dipesan
- menuju_ke_pastry(app): masuk ke bagian memilih jenis pastry yang ingin dipesan



```python
# breads.py

import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox
import csv
from PIL import Image, ImageTk
import os
import homepage
import button  # Import button module for navigation to pemilihan
from button import balik_ke_home

bg_color = "#FFEFE8"
text_color = "#FF7A8A"
button_color = "#FFADA1"
menu_color = "#FFD9CC"
textmenu= "#CD7468"
font_title = ("Baskerville Old Face", 30, "bold")
font_subtitle = ("Arial", 20, "bold")
font_text = ("Arial", 12)
font_title_product= ("Book Antiqua", 14, "bold")

selected_count_label = None
total_cost_label = None
selected_products = []
total_cost = 0

def update_display():
    global selected_count_label, total_cost_label
    selected_count_label.configure(text=f"Jumlah produk yang dipilih: {len(selected_products)}")
    total_cost_label.configure(text=f"Rp {total_cost}")

def load_cart_from_csv():
    global selected_products, total_cost
    cart_path = 'database/cart.csv'
    selected_products = []
    total_cost = 0

    # Check if cart.csv exists
    if not os.path.exists(cart_path):
        # Create the file with a header if it doesn't exist
        with open(cart_path, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(['name', 'price'])

    # Load cart if it exists and is not empty
    if os.stat(cart_path).st_size > 0:
        with open(cart_path, mode='r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip header
            for row in reader:
                selected_products.append({
                    'name': row[0],
                    'price': int(row[1])
                })
```

Gambar 3.7 Kode Program *Pastry* (1)

```python
def save_cart_to_csv():
    with open('database/cart.csv', mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['name', 'price'])  # Write header
        for product in selected_products:
            writer.writerow([product['name'], product['price']])

def buat_pastry_page(app):

    load_cart_from_csv()
    # update_display()

    def load_products(file_path):
        products = []
        with open(file_path, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                products.append({
                    'name': row['name'],
                    'image': os.path.join('images', row['image']),
                    'price': int(row['price'])
                })
        return products

    def select_product(product):
        global selected_products, total_cost
        selected_products.append(product)
        total_cost += product['price']
        update_display()

    # def go_back():
    #     save_cart_to_csv()  # Save cart to CSV before going back
    #     for widget in app.winfo_children():
    #         widget.destroy()
    #     os.system('python homepage.py')

    #def go_back():
    save_cart_to_csv()  # Save cart to CSV before going back
        #app.destroy()
        #os.system('python homepage.py')

    def display_menu(csv_file):

        load_cart_from_csv()

        products = load_products(os.path.join('database', csv_file))

        product_frame = ctk.CTkFrame(app,fg_color='white')
        product_frame.grid(row=1, column=0, columnspan=3, pady=0, padx=65, sticky="nsew")
        img_pathbread = os.path.join( 'images', 'pastryframe.png')
        imgbread = Image.open(img_pathbread)
        imgbread = imgbread.resize((2050, 1095), Image.LANCZOS)
        img1 = ImageTk.PhotoImage(imgbread)
```

Gambar 3.8 Kode Program *Pastry* (2)

Berikut adalah penjelasan singkat tentang beberapa fungsi kode program *pastry* beserta impor yang digunakan:

1. Impor Modul
    - customtkinter as ctk: untuk membuat antarmuka modern.
    - tkinter as tk: untuk membuat GUI dasar.
    - messagebox: untuk menampilkan kotak pesan.
    - csv: untuk membaca dan menulis file CSV.
    - PIL (Image, ImageTk): untuk memproses gambar.
    - os: untuk berinteraksi dengan sistem file.
    - homepage: modul beranda atau halaman awal sebelum memasuki program
    - button: modul tambahan untuk navigasi, termasuk `balik_ke_home`.
2. Global Variables
    - Warna: `bg_color`, `text_color`, `textmenu`, `button_color`, `menu_color`.
    - Font: `font_title`, `font_subtitle`, `font_text`, `font_title_product`.

- Label dan Data: `selected_count_label`, `total_cost_label`, `selected_products`, `total_cost`.

3. Fungsi
   - `update_display()`: memperbarui jumlah produk yang dipilih dan total biaya.
   - `load_cart_from_csv()`: memuat keranjang belanja dari `cart.csv`.
   - `save_cart_to_csv()`: menyimpan keranjang belanja ke `cart.csv`.
   - `buat_pastry_page(app)`: membuat halaman produk kue, menampilkan produk dari CSV, dan menyediakan tombol untuk memilih produk dan opsi pemesanan.
4. Main Program
   - `if __name__ == "__main__":`: memulai aplikasi, membuat jendela utama (`app`), memanggil `buat_pastry_page(app)`, dan menjalankan loop utama Tkinter (`app.mainloop()`).

```python
import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox
import csv
from PIL import Image, ImageTk
import os
import button
from button import balik_ke_home # Import button module for navigation to pemilihan

bg_color = "#FFEFE8"
text_color = "#FF7A8A"
textmenu= "#CD7468"
button_color = "#FFADA1"
menu_color = "#FFD9CC"
font_title = ("Baskerville Old Face", 30, "bold")
font_subtitle = ("Arial", 20, "bold")
font_text = ("Arial", 12)
font_title_product= ("Book Antiqua", 18, "bold")

selected_count_label = None
total_cost_label = None
selected_products = []
total_cost = 0

def update_display():
    global selected_count_label, total_cost_label
    selected_count_label.configure(text=f"Jumlah produk yang dipilih: {len(selected_products)}")
    total_cost_label.configure(text=f"Rp {total_cost}")

def load_cart_from_csv():

    global selected_products, total_cost
    cart_path = 'database/cart.csv'
    selected_products = []
    total_cost = 0

    # Check if cart.csv exists
    if not os.path.exists(cart_path):
        # Create the file with a header if it doesn't exist
        with open(cart_path, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(['name', 'price'])

    # Load cart if it exists and is not empty
    if os.stat(cart_path).st_size > 0:
        with open(cart_path, mode='r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip header
            for row in reader:
                selected_products.append({
                    'name': row[0],
                    'price': int(row[1])
                })
                total_cost += int(row[1])
    else:
        # If cart.csv is empty, set default values
        selected_products = []
        total_cost = 0

def save_cart_to_csv():
    with open('database/cart.csv', mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['name', 'price'])  # Write header
        for product in selected_products:
            writer.writerow([product['name'], product['price']])
```

Gambar 3.9 Kode Program *Breads* (1)

```python
def buat_breads_page(app):

    load_cart_from_csv()

    def load_products(file_path):
        products = []
        with open(file_path, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                products.append({
                    'name': row['name'],
                    'image': os.path.join('images', row['image']),
                    'price': int(row['price'])
                })
        return products

    def select_product(product):
        global selected_products, total_cost
        selected_products.append(product)
        total_cost += product['price']
        update_display()

    #def go_back():
    # Save cart to CSV before going back
        #app.destroy()
        #os.system('python homepage.py')

    def display_menu(csv_file):
        products = load_products(os.path.join('database', csv_file))
        img_pathbread = os.path.join( 'images', 'breadsframe.png')
        imgbread = Image.open(img_pathbread)
        imgbread = imgbread.resize((2050, 1095), Image.LANCZOS)
        img1 = ImageTk.PhotoImage(imgbread)

        bg_label = ctk.CTkLabel(app, image=img1)
        bg_label.place(x=0, y=0, relwidth=1, relheight=1)
        bg_label.image = img1
        bg_label.lower()

        product_frame = ctk.CTkFrame(app, fg_color='white')
        product_frame.grid(row=1, column=0, columnspan=3, pady=0, padx=65, sticky="nsew")


        for index, product in enumerate(products):
            row = index // 5
            column = index % 5

            product_card = ctk.CTkFrame(product_frame,fg_color='white')
            product_card.grid(row=row, column=column,  padx=25, pady=15)

            image = Image.open(product['image'])
            image = image.resize((270, 270), Image.LANCZOS) ## kalo error ganti jadi Image.Resampling.LANCZOS
            img = ImageTk.PhotoImage(image)

            img_label = ctk.CTkLabel(product_card, image=img, text="")
            img_label.image = img
            img_label.grid(row=0, column=0, pady=(0, 10))
            text_menu = ctk.CTkLabel(product_card, text=product['name'],font=font_title_product,fg_color='transparent', text_color=textmenu)
            text_menu.grid(row=1,column=0, pady=(0, 5))

            button = ctk.CTkButton(product_card, text=f"Rp {product['price']}", command=lambda p=product: select_product(p), fg_color='#C49388', text_color='white')
            button.grid(row=2, column=0, sticky="ew")

    header_frame = ctk.CTkFrame(app,fg_color='#C27767')
    header_frame.grid(row=0, column=0, columnspan=3, padx=50, pady=13,sticky="ew")

    back_button = ctk.CTkButton(header_frame, text="Back", command=lambda:(save_cart_to_csv() ,balik_ke_home(app)), fg_color='white',text_color='#C27767')
    back_button.grid(row=0, column=0, padx=10, pady=10, sticky="w")

    header_frame.grid_columnconfigure(1, weight=1)

    title_label = ctk.CTkLabel(header_frame, text="BREADS", justify="center",text_color='#422C28', font=font_title)
    title_label.grid(row=0, column=1, pady=15, padx=275, sticky="ew")

    header_frame.grid_columnconfigure(3, weight=1)

    display_menu('breads.csv')

    bottom_frame = ctk.CTkFrame(app,fg_color='#C27767')
    bottom_frame.grid(row=2, column=0,columnspan=3, pady=12, padx=50,  sticky="ew")
```

Gambar 3.10 Kode Program *Breads* (2)

```python
    header_frame = ctk.CTkFrame(app,fg_color='#C27767')
    header_frame.grid(row=0, column=0, columnspan=3, padx=50, pady=13,sticky="ew")

    back_button = ctk.CTkButton(header_frame, text="Back", command=lambda:(save_cart_to_csv() ,balik_ke_home(app)), fg_color='white',text_color='#C27767')
    back_button.grid(row=0, column=0, padx=10, pady=10, sticky="w")

    header_frame.grid_columnconfigure(1, weight=1)

    title_label = ctk.CTkLabel(header_frame, text="BREADS", justify="center",text_color='#422C28', font=font_title)
    title_label.grid(row=0, column=1, pady=15, padx=275, sticky="ew")

    header_frame.grid_columnconfigure(3, weight=1)

    display_menu('breads.csv')

    bottom_frame = ctk.CTkFrame(app,fg_color='#C27767')
    bottom_frame.grid(row=2, column=0,columnspan=3, pady=12, padx=50,  sticky="ew")

    global selected_count_label, total_cost_label
    selected_count_label = ctk.CTkLabel(bottom_frame, text="Jumlah produk yang dipilih: 0")
    selected_count_label.grid(row=0, column=0, padx=20, sticky="w")

    total_cost_label = ctk.CTkLabel(bottom_frame, text="Rp 0")
    total_cost_label.grid(row=0, column=3, padx=20, sticky="e")

    action_frame = ctk.CTkFrame(bottom_frame,fg_color='#C27767')
    action_frame.grid(row=0, columnspan=2,column=1, pady=5, padx=5)

    takeaway_button = ctk.CTkButton(action_frame,width=240, text="TAKEAWAY", fg_color='white',text_color='#C27767',command=lambda: button.menuju_ke_pemilihan(app, "TAKEAWAY", selected_products))
    takeaway_button.grid(row=0, column=0, padx=10, pady=5, sticky="ew")
    delivery_button = ctk.CTkButton(action_frame,width=240, text="DELIVERY", fg_color='white',text_color='#C27767', command=lambda: button.menuju_ke_pemilihan(app, "DELIVERY", selected_products))
    delivery_button.grid(row=0, column=1, padx=10, pady=5, sticky="ew")
    dinein_button = ctk.CTkButton(action_frame,width=240, text="DINE IN", fg_color='white',text_color='#C27767', command=lambda: button.menuju_ke_pemilihan(app, "DINE IN", selected_products))
    dinein_button.grid(row=0, column=2, padx=10, pady=5, sticky="ew")

    update_display()

if __name__ == "__main__":
    app = ctk.CTk(fg_color='#FFDAE1')
    app.title("Cakeshop - Breads")
    app.geometry("1270x710")
    buat_breads_page(app)
    app.mainloop()
```

Gambar 3.11 Kode Program *Breads* (3)

Berikut adalah penjelasan singkat tentang beberapa fungsi kode program *breads* beserta impor yang digunakan:

1. Import
    - `customtkinter as ctk`: untuk membuat antarmuka modern.
    - `tkinter as tk`: untuk membuat GUI dasar.
    - `messagebox`: untuk menampilkan kotak pesan.
    - `csv`: untuk membaca dan menulis file CSV.
    - `PIL (Image, ImageTk)`: untuk memproses gambar.
    - `os`: untuk berinteraksi dengan sistem file.
    - `button`: modul tambahan untuk navigasi, termasuk `balik_ke_home`.
2. Global Variables
    - Warna: `bg_color`, `text_color`, `textmenu`, `button_color`, `menu_color`.
    - Font: `font_title`, `font_subtitle`, `font_text`, `font_title_product`.
    - Label dan Data: `selected_count_label`, `total_cost_label`, `selected_products`, `total_cost`.
3. Fungsi
    - `update_display()`: memperbarui jumlah produk yang dipilih dan total biaya.
    - `load_cart_from_csv()`: memuat keranjang belanja dari `cart.csv`.
    - `save_cart_to_csv()`: menyimpan keranjang belanja ke `cart.csv`.
    - `buat_breads_page(app)`: membuat halaman produk roti, menampilkan produk dari CSV, dan menyediakan tombol untuk memilih produk dan opsi pemesanan.
4. Main Program
    - `if __name__ == "__main__":`: memulai aplikasi, membuat jendela utama (`app`), memanggil `buat_breads_page(app)`, dan menjalankan loop utama Tkinter (`app.mainloop()`).

```python
# breads.py

import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox
import csv
from PIL import Image, ImageTk
import os
import button  # Import button module for navigation to pemilihan
from button import balik_ke_home

bg_color = "#FFEFE8"
text_color = "#FF7A8A"
button_color = "#FFADA1"
textmenu= "#CD7468"
menu_color = "#FFD9CC"
font_title = ("Baskerville Old Face", 30, "bold")
font_subtitle = ("Arial", 20, "bold")
font_text = ("Arial", 12)
font_title_product= ("Book Antiqua", 16, "bold")

selected_count_label = None
total_cost_label = None
selected_products = []
total_cost = 0

def update_display():
    global selected_count_label, total_cost_label
    selected_count_label.configure(text=f"Jumlah produk yang dipilih: {len(selected_products)}")
    total_cost_label.configure(text=f"Rp {total_cost}")

def load_cart_from_csv():
    global selected_products, total_cost
    cart_path = 'database/cart.csv'
    selected_products = []
    total_cost = 0

    # Check if cart.csv exists
    if not os.path.exists(cart_path):
        # Create the file with a header if it doesn't exist
        with open(cart_path, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(['name', 'price'])

    # Load cart if it exists and is not empty
    if os.stat(cart_path).st_size > 0:
        with open(cart_path, mode='r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip header
            for row in reader:
                selected_products.append({
                    'name': row[0],
                    'price': int(row[1])
                })
                total_cost += int(row[1])
    else:
        # If cart.csv is empty, set default values
        selected_products = []
        total_cost = 0

def save_cart_to_csv():
    with open('database/cart.csv', mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['name', 'price'])  # Write header
        for product in selected_products:
            writer.writerow([product['name'], product['price']])
```

Gambar 3.12 Kode Program *Cakes* (1)

```python
def buat_cakes_page(app):

    load_cart_from_csv()

    def load_products(file_path):
        products = []
        with open(file_path, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                products.append({
                    'name': row['name'],
                    'image': os.path.join('images', row['image']),
                    'price': int(row['price'])
                })
        return products

    def select_product(product):
        global selected_products, total_cost
        selected_products.append(product)
        total_cost += product['price']
        update_display()

    # def go_back():
    #     save_cart_to_csv()  # Save cart to CSV before going back
    #     for widget in app.winfo_children():
    #         widget.destroy()
    #     os.system('python homepage.py')

    #def go_back():
    save_cart_to_csv()  # Save cart to CSV before going back
        #app.destroy()
        #os.system('python homepage.py')
```

Gambar 3.13 Kode Program *Cakes* (2)

```python
def display_menu(csv_file):
    products = load_products(os.path.join('database', csv_file))
    img_pathbread = os.path.join( 'images', 'cakesframe.png')
    imgbread = Image.open(img_pathbread)
    imgbread = imgbread.resize((2050, 1095), Image.LANCZOS)
    img1 = ImageTk.PhotoImage(imgbread)

    bg_label = ctk.CTkLabel(app, image=img1)
    bg_label.place(x=-2, y=0)
    bg_label.image = img1
    bg_label.lower()

    product_frame = ctk.CTkFrame(app,fg_color='white')
    product_frame.grid(row=1, column=0, columnspan=3, pady=0, padx=60, sticky="nsew")

    for index, product in enumerate(products):
        row = index // 5
        column = index % 5

        product_card = ctk.CTkFrame(product_frame,fg_color='white')
        product_card.grid(row=row, column=column, padx=25, pady=15)

        image = Image.open(product['image'])
        image = image.resize((270, 270), Image.LANCZOS) ## kalo error ganti jadi Image.Resampling.LANCZOS
        img = ImageTk.PhotoImage(image)

        img_label = ctk.CTkLabel(product_card, image=img, text="")
        img_label.image = img
        img_label.grid(row=0, column=0, pady=(0, 10))
        text_menu = ctk.CTkLabel(product_card, text=product['name'],font=font_title_product,fg_color='transparent', text_color=textmenu)
        text_menu.grid(row=1,column=0, pady=(0, 5))

        button = ctk.CTkButton(product_card, text=f"Rp {product['price']}",text_color='white', command=lambda p=product: select_product(p), fg_color=button_color)
        button.grid(row=2, column=0)

header_frame = ctk.CTkFrame(app,fg_color='#E8AEBA')
header_frame.grid(row=0, column=0, columnspan=3, padx=50, pady=14,sticky="ew")

back_button = ctk.CTkButton(header_frame, text="Back", command=lambda:(save_cart_to_csv() ,balik_ke_home(app)), fg_color='white',text_color=button_color)
back_button.grid(row=0, column=0, padx=10, pady=10)

header_frame.grid_columnconfigure(1, weight=1)

title_label = ctk.CTkLabel(header_frame, text="CAKES", justify="center", font=font_title, text_color='#8D495A')
title_label.grid(row=0, column=1, pady=15, padx=275, sticky="ew")

header_frame.grid_columnconfigure(3, weight=1)

display_menu('cakes.csv')

bottom_frame = ctk.CTkFrame(app,fg_color='#E8AEBA')
bottom_frame.grid(row=2, column=0, columnspan=3, pady=13, padx=50,  sticky="ew")

global selected_count_label, total_cost_label
selected_count_label = ctk.CTkLabel(bottom_frame, text="Jumlah produk yang dipiiih: 0")
selected_count_label.grid(row=0, column=0, padx=20, sticky="w")

total_cost_label = ctk.CTkLabel(bottom_frame, text="Rp 0")
total_cost_label.grid(row=0, column=3, padx=20, sticky="e")

action_frame = ctk.CTkFrame(bottom_frame,fg_color='#E8AEBA')
action_frame.grid(row=0, columnspan=2,column=1, pady=5, padx=5)

takeaway_button = ctk.CTkButton(action_frame,width=240, text="TAKEAWAY", fg_color='white', text_color=button_color, command=lambda: button.menuju_ke_pemilihan(app, "TAKEAWAY", selected_products))
takeaway_button.grid(row=0, column=0, padx=10, pady=5)
delivery_button = ctk.CTkButton(action_frame, width=240,text="DELIVERY", fg_color='white', text_color=button_color, command=lambda: button.menuju_ke_pemilihan(app, "DELIVERY", selected_products))
delivery_button.grid(row=0, column=1, padx=10, pady=5)
dinein_button = ctk.CTkButton(action_frame, width=240,text="DINE IN", fg_color='white', text_color=button_color, command=lambda: button.menuju_ke_pemilihan(app, "DINE IN", selected_products))
dinein_button.grid(row=0, column=2, padx=10, pady=5)

# Initialize selected_count_label and total_cost_label
update_display()
```

Gambar 3.14 Kode Program *Cakes* (3)

```python
if __name__ == "__main__":
    app = ctk.CTk(fg_color='#FFDAE1')
    app.title("Cakeshop - Cakes")
    app.geometry("1270x710")
    buat_cakes_page(app)
    app.mainloop()
```

Gambar 3.15 Kode Program *Cakes* (4)

Program di atas membuat antarmuka pengguna grafis (GUI) untuk halaman produk kue di toko kue Cake Shop 23BEE menggunakan `customtkinter` dan `tkinter`. Berikut penjelasan singkat tentang fungsi dan elemen GUI yang digunakan:

1. `update_display()`: memperbarui tampilan jumlah produk yang dipilih dan total biaya.
2. `load_cart_from_csv()`: memuat data keranjang belanja dari file `cart.csv`.
3. `save_cart_to_csv()`: menyimpan data keranjang belanja ke file `cart.csv`.
4. `buat_cakes_page(app)`: membuat halaman produk kue, menampilkan produk dari file csv, dan menyediakan tombol untuk memilih produk dan opsi pemesanan.
5. `load_products(file_path)`: memuat data produk dari file csv yang diberikan.

6. `select_product(product)`: menambahkan produk yang dipilih ke keranjang belanja dan memperbarui total biaya.
7. `display_menu(csv_file)`: menampilkan daftar produk dalam antarmuka gui.
8. `header_frame`: bingkai di bagian atas halaman untuk judul dan tombol kembali.
9. `back_button`: tombol kembali untuk menyimpan keranjang dan kembali ke halaman sebelumnya.
10. `title_label`: label untuk menampilkan judul halaman.
11. `product_frame`: bingkai untuk menampilkan kartu produk.
12. `bottom_frame`: bingkai di bagian bawah halaman untuk menampilkan jumlah produk yang dipilih dan total biaya, serta tombol aksi.
13. `action_frame`: bingkai untuk tombol opsi pemesanan (takeaway, delivery, dine in).

```python
# breads.py

import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox
import csv
from PIL import Image, ImageTk
import os
import button
from button import balik_ke_home  # Import button module for navigation to pemilihan

bg_color = "#FFEFEE"
text_color = "#FF7A8A"
button_color = "#FFADA1"
textmenu= "#CD7468"
menu_color = "#FFD0CC"
font_title = ("Baskerville Old Face", 30, "bold")
font_subtitle = ("Arial", 20, "bold")
font_text = ("Arial", 12)
font_title_product= ("Book Antiqua", 18, "bold")
menu_title_color="#FFF"

selected_count_label = None
total_cost_label = None
selected_products = []
total_cost = 0

def update_display():
    global selected_count_label, total_cost_label
    selected_count_label.configure(text=f"Jumlah produk yang dipilih: {len(selected_products)}")
    total_cost_label.configure(text=f"Rp {total_cost}")

def load_cart_from_csv():
    global selected_products, total_cost
    cart_path = 'database/cart.csv'
    selected_products = []
    total_cost = 0

    # Check if cart.csv exists
    if not os.path.exists(cart_path):
        # Create the file with a header if it doesn't exist
        with open(cart_path, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(['name', 'price'])

    # Load cart if it exists and is not empty
    if os.stat(cart_path).st_size > 0:
        with open(cart_path, mode='r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip header
            for row in reader:
                selected_products.append({
                    'name': row[0],
                    'price': int(row[1])
                })
                total_cost += int(row[1])
    else:
        # If cart.csv is empty, set default values
        selected_products = []
        total_cost = 0

def save_cart_to_csv():
    with open('database/cart.csv', mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['name', 'price'])  # Write header
        for product in selected_products:
            writer.writerow([product['name'], product['price']])
```

Gambar 3.16 Kode Program *Donuts* (1)

```python
def buat_donuts_page(app):

    load_cart_from_csv()
    # update_display()

    def load_products(file_path):
        products = []
        with open(file_path, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                products.append({
                    'name': row['name'],
                    'image': os.path.join('images', row['image']),
                    'price': int(row['price'])
                })
        return products

    def select_product(product):
        global selected_products, total_cost
        selected_products.append(product)
        total_cost += product['price']
        update_display()

    # def go_back():
    save_cart_to_csv()
    #     for widget in app.winfo_children():
    #         widget.destroy()
    #     os.system('python homepage.py')

    # def go_back():
    #     save_cart_to_csv()  # Save cart to CSV before going back
    #     #app.destroy()
    #     #os.system('python homepage.py')

    def display_menu(csv_file):

        load_cart_from_csv()

        products = load_products(os.path.join('database', csv_file))

        product_frame = ctk.CTkFrame(app, fg_color='white')
        product_frame.grid(row=1, column=0, columnspan=3, pady=0, padx=65, sticky="nsew")
        img_pathbread = os.path.join( 'images', 'donutsframe.png')
        imgbread = Image.open(img_pathbread)
        imgbread = imgbread.resize((2050, 1095), Image.LANCZOS)
        img1 = ImageTk.PhotoImage(imgbread)

        bg_label = ctk.CTkLabel(app, image=img1)
        bg_label.place(x=0, y=0, relwidth=1, relheight=1)
        bg_label.image = img1
        bg_label.lower()

        for index, product in enumerate(products):
            row = index // 5
            column = index % 5

            product_card = ctk.CTkFrame(product_frame, fg_color='white')
            product_card.grid(row=row, column=column, padx=25, pady=15)

            image = Image.open(product['image'])
            image = image.resize((270, 270), Image.LANCZOS) ## kalo error ganti jadi Image.Resampling.LANCZOS
            img = ImageTk.PhotoImage(image)

            img_label = ctk.CTkLabel(product_card, image=img, text="",font=font_title_product, fg_color=button_color )
            img_label.image = img
            img_label.grid(row=0, column=0, pady=(0, 5))

            text_menu = ctk.CTkLabel(product_card, text=product['name'],font=font_title_product,fg_color='transparent', text_color='#CD8690')
            text_menu.grid(row=1,column=0, pady=(0, 5))

            button = ctk.CTkButton(product_card, text=f"Rp {product['price']}", text_color='white',command=lambda p=product: select_product(p), fg_color='#DFAAAD')
            button.grid(row=2, column=0)
```

Gambar 3.17 Kode Program *Donuts* (2)

```python
        header_frame = ctk.CTkFrame(app,fg_color='#CD8690')
        header_frame.grid(row=0, column=0, columnspan=3, padx=50, pady=15, sticky="ew")

        back_button = ctk.CTkButton(header_frame, text="Back", command=lambda:(save_cart_to_csv() ,balik_ke_home(app)), fg_color='white', text_color='#CD8690')
        back_button.grid(row=0, column=0, padx=10, pady=10)

        header_frame.grid_columnconfigure(1, weight=1)

        title_label = ctk.CTkLabel(header_frame, text="DONUTS",text_color='#703C4C', font=font_title)
        title_label.grid(row=0, column=1, pady=15, padx=275, sticky="ew")

        header_frame.grid_columnconfigure(3, weight=1)
        display_menu('donuts.csv')

        bottom_frame = ctk.CTkFrame(app,fg_color='#CD8690')
        bottom_frame.grid(row=2, column=0, columnspan=3, pady=15, padx=50, sticky="ew")

        global selected_count_label, total_cost_label
        selected_count_label = ctk.CTkLabel(bottom_frame, text="Jumlah produk yang dipilih: 0")
        selected_count_label.grid(row=0, column=0, padx=20, sticky="w")

        total_cost_label = ctk.CTkLabel(bottom_frame, text="Rp 0")
        total_cost_label.grid(row=0, column=3, padx=20, sticky="e")

        action_frame = ctk.CTkFrame(bottom_frame, fg_color='#CD8690')
        action_frame.grid(row=0, columnspan=2,column=1, pady=5, padx=5)

        takeaway_button = ctk.CTkButton(action_frame,width=240, text="TAKEAWAY", fg_color='white', text_color='#CD8690',command=lambda: button.menuju_ke_penilihan(app, "TAKEAWAY", selected_products))
        takeaway_button.grid(row=0, column=0, padx=10, pady=5)
        delivery_button = ctk.CTkButton(action_frame, width=240,text="DELIVERY", fg_color='white',text_color='#CD8690', command=lambda: button.menuju_ke_penilihan(app, "DELIVERY", selected_products))
        delivery_button.grid(row=0, column=1, padx=10, pady=5)
        dinein_button = ctk.CTkButton(action_frame,width=240, text="DINE IN", fg_color='white',text_color='#CD8690', command=lambda: button.menuju_ke_penilihan(app, "DINE IN", selected_products))
        dinein_button.grid(row=0, column=2, padx=10, pady=5)

        # Initialize selected_count_label and total_cost_label
        update_display()

    if __name__ == "__main__":
        app = ctk.CTk(fg_color='#F7DAE1')
        app.title("Cakeshop - Donuts")
        app.geometry("1270x750")
        buat_donuts_page(app)
        app.mainloop()
```

Gambar 3.18 Kode Program *Donuts* (3)

Program di atas membuat antarmuka pengguna grafis (GUI) untuk halaman produk donat di toko kue Cake Shop 23BEE menggunakan `customtkinter` dan `tkinter`. Berikut penjelasan singkat tentang fungsi yang digunakan:

1. Impor Modul
    - customtkinter dan tkinter: Membuat antarmuka pengguna.
    - csv: Membaca dan menulis file CSV.
    - PIL (Pillow): Memproses dan menampilkan gambar.
    - os: Berinteraksi dengan sistem file.
    - button dan balik_ke_home: Navigasi ke halaman utama.
2. Pengaturan Warna dan Font
    - Menyimpan warna dan font yang digunakan dalam antarmuka pengguna.
3. Variabel Global
    - selected_count_label, total_cost_label: Label untuk jumlah produk dan total biaya.
    - selected_products: Daftar produk yang dipilih.
    - total_cost: Total biaya produk yang dipilih.
4. Fungsi Utama
    - update_display: Memperbarui label jumlah produk dan total biaya.
    - load_cart_from_csv: Memuat produk dari file cart.csv ke dalam selected_products dan menghitung total_cost.
    - save_cart_to_csv: Menyimpan selected_products ke dalam file cart.csv.
5. Fungsi buat_donuts_page
    - Menampilkan halaman donat dengan produk yang dimuat dari file CSV.
    - load_products(file_path): Membaca produk dari file CSV.
    - select_product(product): Menambahkan produk yang dipilih ke selected_products dan memperbarui total_cost.
    - display_menu(csv_file): Menampilkan produk dalam bentuk grid.
    - Menyimpan keranjang belanja sebelum kembali ke halaman utama.
    - Membuat dan mengatur layout header, menu produk, dan footer.

- Menyediakan tombol untuk pilihan "TAKEAWAY", "DELIVERY", dan "DINE IN".

6. Blok if __name__ == "__main__

- Membuat instance aplikasi dan menjalankannya.

```python
import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox
import os
from PIL import Image, ImageTk
import invoice
import csv

# Global variables to store selected products and total cost
selected_products = []
selected_pilihan = ""
total_cost = 0
total_cost_label = None  # Define total_cost_label globally

def update_total_cost():
    global total_cost, total_cost_label
    total_cost = sum(product['price'] * product['quantity'] for product in selected_products)
    if total_cost_label:
        total_cost_label.configure(text=f"RP {total_cost},-")

def remove_product(index):
    global selected_products
    del selected_products[index]
    display_orders()

def display_orders():
    for widget in orders_frame.winfo_children():
        widget.destroy()

    for index, product in enumerate(selected_products):
        product_label = ctk.CTkLabel(orders_frame,text_color='#C27767', text=f"{product['name']} x {product.get('quantity', 1)}")
        product_label.grid(row=index, column=0, padx=10, pady=5)

        price_label = ctk.CTkLabel(orders_frame,text_color='#C27767', text=f"Rp {product['price'] * product.get('quantity', 1)}")
        price_label.grid(row=index, column=1, padx=10, pady=5)

        remove_button = ctk.CTkButton(orders_frame,text_color='white', text="Remove", command=lambda i=index: remove_product(i), fg_color="#C27767")
        remove_button.grid(row=index, column=2, padx=10, pady=5)

    update_total_cost()

def go_back(app):
    for widget in app.winfo_children():
        widget.destroy()
        os.system('python homepage.py')

# This function reads the latest data from the CSV file
def get_latest_data_from_csv(file_path):
    latest_data = None
    with open(file_path, 'r', newline='') as file:
        reader = csv.reader(file)
        for row in reader:
            latest_data = row
    return latest_data

def save_order(nama_pengguna, nomor_telepon, pilihan, jam, produk):
    pass  # This function is no longer needed since we're not saving to an HTML file

```

Gambar 3.19 Kode Program Pembayaran (1)

```python
def buat_pembayaran_page(app, products, pilihan):
    global selected_products, selected_pilihan, total_cost_label
    selected_products = products
    selected_pilihan = pilihan

    # Clear previous widgets
    for widget in app.winfo_children():
        widget.destroy()
    img_pathbread = os.path.join('images', 'background.png')
    imgbread = Image.open(img_pathbread)
    imgbread = imgbread.resize((2050, 1095), Image.LANCZOS)  # Adjust to fit the window size
    img1 = ImageTk.PhotoImage(imgbread)

    bg_label = ctk.CTkLabel(app, image=img1, text="")
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)
    bg_label.image = img1  # Prevent image from being garbage collected
    bg_label.lower()

    main_frame = ctk.CTkFrame(app,fg_color='white', width=900, height=600)
    main_frame.place(relx=0.5, rely=0.5, anchor="center")
    main_frame.pack_propagate(False)  # Prevent frame from resizing to fit its content

    # Header Frame
    header_frame = ctk.CTkFrame(main_frame, fg_color="#FFF", width=600, height=50)
    header_frame.place(relx=0.58, rely=0.1, anchor="center")
    header_frame.pack_propagate(False)  # Prevent frame from resizing to fit its content

    from button import balik_ke_home
    back_button = ctk.CTkButton(app, text="Batal", command=lambda:balik_ke_home(app), fg_color="#FFADA1")
    back_button.pack(side="top", padx=15, pady=30)
    back_button.place(rely=0.02,relx=0.01)
    #back_button.pack_propagate(False)

    title_label = ctk.CTkLabel(header_frame, text="TOTAL PESANAN ANDA", justify="center",text_color='#DB7575', font=("Baskerville Old Face", 35, "bold"))
    title_label.pack(side="left", pady=10, padx=40)

    # Orders Frame
    global orders_frame
    orders_frame = ctk.CTkFrame(main_frame, fg_color="#ffe3de", width=900, height=500)
    orders_frame.place(relx=0.12, rely=0.18)
    orders_frame.pack_propagate(False)  # Prevent frame from resizing to fit its content

    display_orders()

    # Payment Frame
    payment_frame = ctk.CTkFrame(main_frame, fg_color="#ffe3de", width=750, height=500)
    payment_frame.place(relx=0.63, rely=0.18)
    payment_frame.pack_propagate(False)  # Prevent frame from resizing to fit its content

    payment_label = ctk.CTkLabel(payment_frame, text="PEMBAYARAN\nPilih Opsi Pembayaran",text_color='#DB7575', justify="center", font=("Arial", 18, "bold"))
    payment_label.grid(row=0, column=0, padx=20,pady=20)

    payment_var = tk.StringVar(value="Tunai")

    tunai_radio = ctk.CTkRadioButton(payment_frame, text="Tunai",hover_color='#DB7575',fg_color='#DB7575',text_color= '#DB7575', variable=payment_var, value="Tunai")
    tunai_radio.grid(row=1, column=0, padx=10, pady=(15,20))

    non_tunai_radio = ctk.CTkRadioButton(payment_frame,hover_color='#DB7575',fg_color='#DB7575', text="Non Tunai",text_color= '#DB7575',variable=payment_var, value="Non Tunai")
    non_tunai_radio.grid(row=2, column=0, padx=10, pady=(10,50))
```

Gambar 3.20 Kode Program Pembayaran (2)

```python
    def checkout():
        # Read the latest data from datapembeli.csv
        latest_data = get_latest_data_from_csv("database/datapembeli.csv")
        if latest_data:
            nama_pengguna = latest_data[0].capitalize()  # Take name from CSV file and capitalize
            no_telp = latest_data[1]
            jam_input = latest_data[3].strip()  # Take time from CSV
        else:
            # Default if no data in CSV
            nama_pengguna = "Unknown User"
            no_telp = "0000000000"
            jam_input = "10:00 AM"

        metode_pembayaran = payment_var.get()

        # Prepare items for invoice
        items = [(product['name'], product['price'] * product['quantity']) for product in selected_products]
        total_harga = sum(item[1] for item in items)

        invoice.buat_invoice_page(app, nama_pengguna, no_telp, selected_pilihan.upper(), jam_input, metode_pembayaran, items, total_harga)

    # Total and Checkout
    total_cost_label = ctk.CTkLabel(main_frame, text=(f"Total Pesanan Rp {total_cost}"),text_color='#DB7575', font=("Arial", 16, "bold"))
    total_cost_label.place(relx=0.14, rely=0.87 , anchor="nw")

    checkout_button = ctk.CTkButton(main_frame, text="checkout",font=("Baskerville", 25, "bold"), command=checkout, fg_color="#ffe3de",text_color='#DB7575')
    checkout_button.place(relx=0.87, rely=0.91, anchor="center")  # Center confirm button at bottom

    # Update the total cost initially
    update_total_cost()

if __name__ == "__main__":
    app = ctk.CTk()
    app.title("Cakeshop - Pembayaran")
    app.geometry("1270x710")
    sample_products = [
        {'name': 'Choco Bun', 'quantity': 1, 'price': 11000},
        {'name': 'Korean Garlic Bread', 'quantity': 1, 'price': 20000},
    ]
    buat_pembayaran_page(app, sample_products, "DINE IN")
    app.mainloop()
```

Gambar 3.21 Kode Program Pembayaran (3)

Berikut adalah penjelasan singkat tentang fungsi-fungsi dan import pada program pembayaran di atas:

1. Import Libraries:
    - customtkinter as ctk, digunakan untuk membuat antarmuka pengguna dengan tampilan yang lebih modern dibandingkan dengan `tkinter` standar.
    - tkinter as tk, digunakan untuk membuat elemen GUI dasar seperti variabel kontrol (`StringVar`).
    - messagebox: Modul dari `tkinter` untuk menampilkan pesan dialog.
    - os: Digunakan untuk mengakses fungsi sistem operasi, seperti menjalankan perintah sistem.
    - PIL (Image, ImageTk): Digunakan untuk memanipulasi dan menampilkan gambar dalam antarmuka pengguna.
    - invoice: Modul khusus untuk membuat halaman invoice (tidak didefinisikan dalam kode ini).
    - csv: Digunakan untuk membaca dan menulis file CSV.
2. Global Variables:
    - selected_products: menyimpan produk yang dipilih.
    - selected_pilihan: menyimpan pilihan pengguna (misalnya, dine in atau take away).
    - total_cost: menyimpan total biaya dari produk yang dipilih.
    - total_cost_label: label untuk menampilkan total biaya.
3. Functions:
    - update_total_cost: menghitung dan memperbarui total biaya berdasarkan produk yang dipilih dan menampilkan hasilnya pada `total_cost_label`.
    - remove_product(index): menghapus produk dari daftar `selected_products` berdasarkan indeks yang diberikan dan memperbarui tampilan pesanan.

- display_orders: menampilkan daftar produk yang dipilih dalam `orders_frame`, termasuk nama, harga, dan tombol untuk menghapus produk.
- go_back(app): menghapus semua widget di dalam `app` dan menjalankan `homepage.py`.
- get_latest_data_from_csv(file_path): membaca data terbaru dari file csv dan mengembalikan data tersebut.
- save_order(nama_pengguna, nomor_telepon, pilihan, jam, produk): fungsi placeholder yang tidak digunakan dalam kode ini.
- buat_pembayaran_page(app, products, pilihan): membuat halaman pembayaran dengan menampilkan produk yang dipilih, total biaya, dan opsi pembayaran. juga berisi tombol untuk checkout yang memproses pesanan dan membuka halaman invoice.
- checkout: membaca data terbaru dari file csv, menyiapkan informasi pembayaran, dan mengarahkan ke halaman invoice menggunakan modul `invoice`.

4. Main Application:
    - if __name__ == "__main__": Bagian ini menjalankan aplikasi dengan membuat window utama (`app`), mengatur judul dan ukuran window, mendefinisikan contoh produk, dan memanggil fungsi `buat_pembayaran_page`.

```python
import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox
import random
import string
import os
import webbrowser
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from reportlab.lib import colors
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph
from reportlab.lib.styles import getSampleStyleSheet

# Function to generate random payment code
def generate_random_code(length=10):
    return ''.join(random.choices(string.ascii_uppercase + string.digits, k=length))

def print_invoice(app, nama, telepon, pilihan, jam, pembayaran, payment_code, items, total_harga, message_line_1, message_line_2):
    pdf_file = f"Tiket_{payment_code}.pdf"

    # Create PDF
    doc = SimpleDocTemplate(pdf_file, pagesize=letter)
    styles = getSampleStyleSheet()

    elements = []

    # Title
    elements.append(Paragraph("KODE PEMBAYARAN", styles['Title']))

    # Customer Info
    elements.append(Paragraph(f"Yth {nama}", styles['Normal']))
    elements.append(Paragraph(f"Telepon\t\t: {telepon}", styles['Normal']))
    elements.append(Paragraph(f"Kode pembayaran anda adalah\t: <b>{payment_code}</b>\n", styles['Normal']))

    # Items Table
    table_data = [['Produk', 'Harga']]
    for item in items:
        table_data.append(item)
    table_data.append(['Total', f"Rp{total_harga:,}"])

    table = Table(table_data)
    table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
        ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),
    ]))

    elements.append(table)
```

Gambar 3.22 Kode Program *Invoice* (1)

```
    # Messages
    elements.append(Paragraph(message_line_1, styles['Normal']))
    elements.append(Paragraph(message_line_2, styles['Normal']))

    # Footer
    elements.append(Paragraph("Terima Kasih, Silahkan menikmati Kue anda!!", styles['Normal']))

    doc.build(elements)

    messagebox.showinfo("Print", f"Print berhasil dilakukan, selamat menikmati kue anda. PDF disimpan sebagai {pdf_file}.")
    webbrowser.open_new_tab(pdf_file)

    app.destroy()
    import homepage
    homepage.main()

#def go_back(app):
 #   for widget in app.winfo_children():
  #      widget.destroy()
   # import button
    #button.balik_ke_home(app)
  #  app.destroy()
   # import homepage
    #homepage.main()

def buat_invoice_page(app, nama, telepon, pilihan, jam, pembayaran, items, total_harga):
    app.geometry("900x500")

    # Clear previous widgets
    for widget in app.winfo_children():
        widget.destroy()

    # Header Frame
    header_frame = ctk.CTkFrame(app)
    header_frame.grid(row=0, column=0, sticky="nw")
    from button import balik_ke_home
    back_button = ctk.CTkButton(header_frame, text="Back", command=lambda:balik_ke_home(app), fg_color="#FFADA1")
    back_button.grid(row=0, column=0, padx=10, pady=10)

    # Invoice Frame
    invoice_frame = ctk.CTkFrame(app, width=900, height=650)
    invoice_frame.grid(row=1, column=0, columnspan=2, padx=90 ,pady=20)

    title_label = ctk.CTkLabel(invoice_frame, text="KODE PEMBAYARAN", justify="center", font=("Arial", 20, "bold"))
    title_label.grid(row=0, column=0, columnspan=2, pady=15)

    greeting_label = ctk.CTkLabel(invoice_frame, text=f"Yth {nama}, Kode pembayaran anda adalah:", justify="center", font=("Arial", 16))
    greeting_label.grid(row=1, column=0, columnspan=2, pady=5)

    payment_code = generate_random_code()
    code_label = ctk.CTkLabel(invoice_frame, text=f"{payment_code}", justify="center", font=("Arial", 24, "bold"))
    code_label.grid(row=2, column=0, columnspan=2, pady=15)
```

Gambar 3.23 Kode Program *Invoice* (2)

```
    print_button = ctk.CTkButton(app, text="PRINT", command=lambda: print_invoice(app, nama, telepon, pilihan, jam, pembayaran, payment_code, items, total_harga, message_line_1, message_line_2), fg_color="#FF
    print_button.grid(row=2, column=0, columnspan=2, pady=20)

    print(f"Pilihan pada Invoice: {pilihan}")
    print(f"Pilihan pembayaran anda: {pembayaran}")

    # Conditional message based on selection and payment
    if pilihan == "DELIVERY":
        message_line_1 = f"Pesanan anda akan diantar pada pukul {jam}, estimasi pengiriman sekitar 30 menit."
        if pembayaran == "Tunai":
            message_line_2 = "Silahkan bayar tunai tagihan anda dengan menunjukkan kode pembayaran ini kepada driver anda."
        else:
            message_line_2 = "Silahkan lakukan pembayaran melalui m-banking dan e-wallet anda dengan kode pembayaran berikut. Waktu maksimal pembayaran adalah 1x24 jam sejak kode diterima."
    elif pilihan == "TAKEAWAY":
        message_line_1 = f"Pesanan anda dapat diambil pada pukul {jam}."
        if pembayaran == "Tunai":
            message_line_2 = "Silahkan bayar tagihan anda di kasir dengan menunjukkan kode pembayaran anda."
        else:
            message_line_2 = "Silahkan lakukan pembayaran melalui m-banking dan e-wallet anda dengan kode pembayaran berikut. Waktu maksimal pembayaran adalah 1x24 jam sejak kode diterima."
    elif pilihan == "DINE IN":
        message_line_1 = f"Pesanan anda akan siap pada pukul {jam}, silahkan datang ke tempat pada jam tersebut."
        if pembayaran == "Tunai":
            message_line_2 = "Silahkan bayar tagihan anda di kasir dengan menunjukkan kode pembayaran anda."
        else:
            message_line_2 = "Silahkan lakukan pembayaran melalui m-banking dan e-wallet anda dengan kode pembayaran berikut. Waktu maksimal pembayaran adalah 1x24 jam sejak kode diterima."

    message_label_1 = ctk.CTkLabel(invoice_frame, text=message_line_1, justify="center", font=("Arial", 16))
    message_label_1.grid(row=3, column=0, columnspan=2, pady=5)
    message_label_2 = ctk.CTkLabel(invoice_frame, text=message_line_2, justify="center", font=("Arial", 16))
    message_label_2.grid(row=4, column=0, columnspan=2, pady=5)


if __name__ == "__main__":
    app = ctk.CTk()
    app.title("Cakeshop - Invoice")
    # Example usage
    items = [
        ["Kue Coklat", 50000],
        ["Kue Keju", 60000],
        ["Kue Strawberry", 70000],
    ]
    total_harga = sum(item[1] for item in items)
    buat_invoice_page(app, "Nama Pengguna", "081234567890", "TAKEAWAY", "10:00 AM", "Tunai", items, total_harga)
    app.mainloop()
```

Gambar 3.24 Kode Program *Invoice* (3)

Program di atas adalah aplikasi GUI untuk toko kue yang memungkinkan pengguna untuk membuat dan mencetak faktur (invoice) pembayaran. Berikut adalah penjelasan singkat dari fungsi-fungsi dan impor yang digunakan dalam program:

1. Impor Modul
    - `customtkinter` dan `tkinter`: Digunakan untuk membuat antarmuka pengguna.
    - `messagebox` dari `tkinter`: Digunakan untuk menampilkan kotak pesan.
    - `random` dan `string`: Digunakan untuk menghasilkan kode pembayaran acak.
    - `os`: Berinteraksi dengan sistem file.
    - `webbrowser`: Digunakan untuk membuka file PDF di browser.
    - `reportlab`: Digunakan untuk membuat file PDF.
2. Fungsi `generate_random_code`
    - `generate_random_code(length=10)`: Menghasilkan kode acak dengan panjang tertentu menggunakan kombinasi huruf besar dan angka.
3. Fungsi `print_invoice`

    `print_invoice(app, nama, telepon, pilihan, jam, pembayaran, payment_code, items, total_harga, message_line_1, message_line_2)`:
    - Membuat file PDF berisi faktur pembayaran dengan detail pengguna, produk yang dibeli, dan total harga.
    - Menambahkan elemen-elemen seperti judul, informasi pelanggan, tabel produk, dan pesan khusus ke dalam PDF.
    - Menyimpan PDF dengan nama file berdasarkan kode pembayaran dan membukanya di browser.
    - Menampilkan pesan informasi tentang keberhasilan pencetakan dan menutup aplikasi.
4. Fungsi `buat_invoice_page`

    `buat_invoice_page(app, nama, telepon, pilihan, jam, pembayaran, items, total_harga)`:
    - Mengatur halaman faktur di aplikasi, termasuk membersihkan widget sebelumnya.
    - Membuat header frame dengan tombol "Back" untuk navigasi kembali.

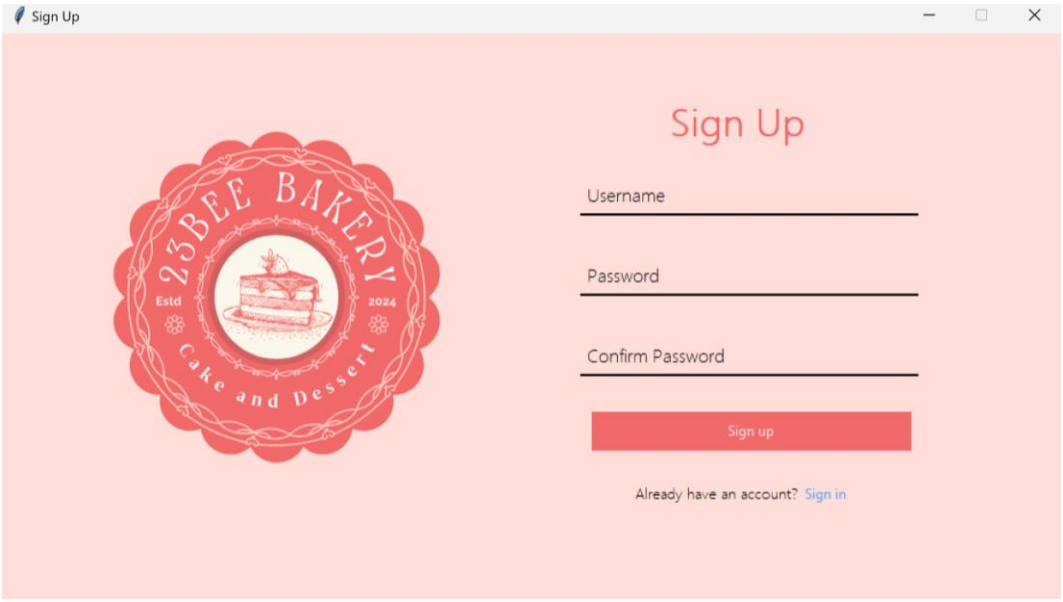
- Membuat invoice frame yang menampilkan kode pembayaran, informasi pelanggan, dan tombol untuk mencetak faktur.
- Menghasilkan pesan khusus berdasarkan pilihan layanan (DELIVERY, TAKEAWAY, DINE IN) dan metode pembayaran (Tunai atau non-tunai).

5. Fungsi `generate_random_code`
    - generate_random_code(length=10)`: Menghasilkan kode acak dengan panjang tertentu menggunakan kombinasi huruf besar dan angka.
    - Membahkan elemen-elemen tersebut ke antarmuka pengguna.
6. Blok `if __name__ == "__main__`
    - Membuat instance aplikasi `CTk`, mengatur judul, dan memanggil fungsi `buat_invoice_page` dengan contoh data pengguna dan produk.

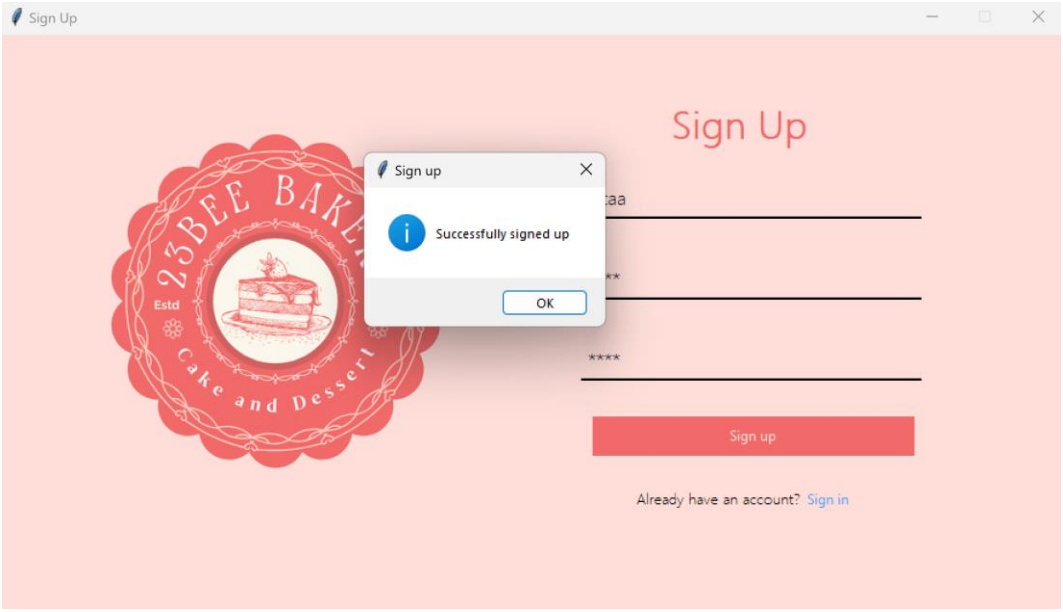
- Menjalankan aplikasi dengan `app.mainloop()`.

3.2 *User Interface* (UI)

Subbab ini menjelaskan tampilan program Cake Shop 23BEE pada saat program dijalankan.

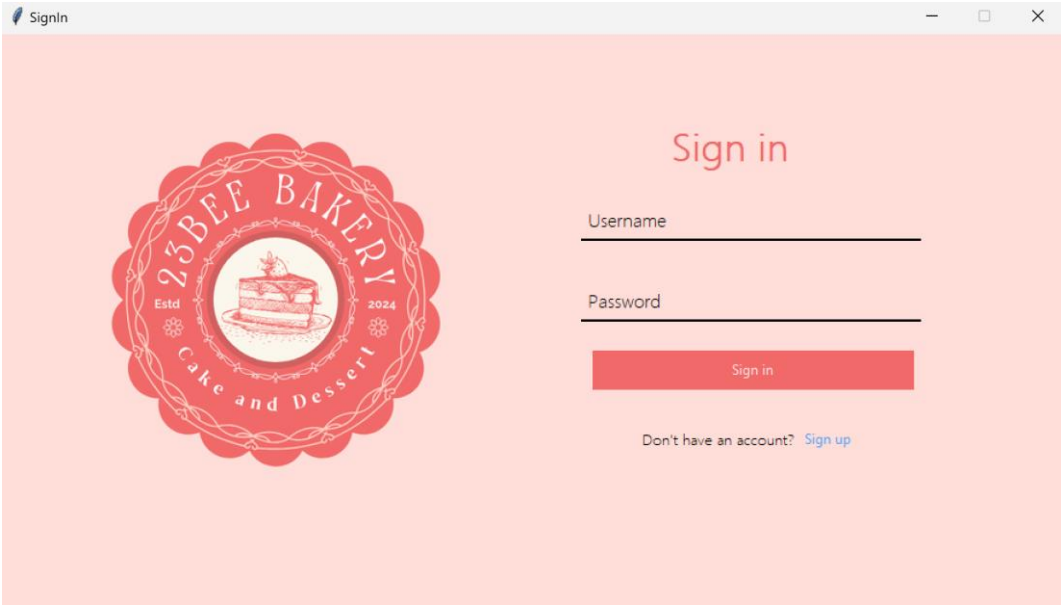


Gambar 3.25 Tampilan *(Sign Up)*



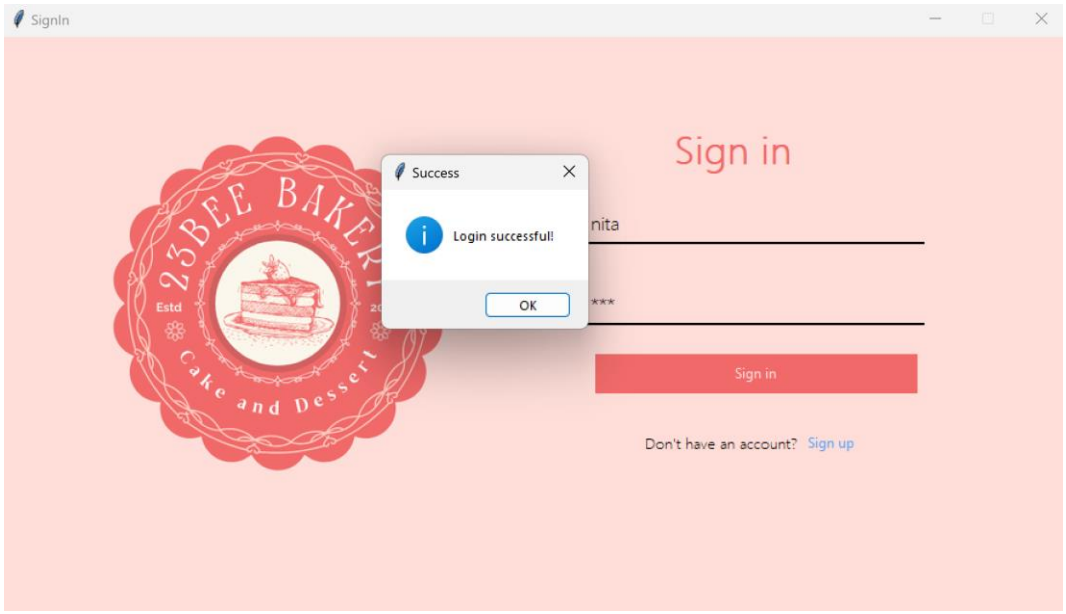
Gambar 3.26 Tampilan (*Sign Up* Berhasil)

Pengguna berhasil menginput *username* dan *password* yang nantinya akan tersimpan pada database akun.csv.



Gambar 3.27 Tampilan (Sign In)

Setelah melakukan *Sign Up, window Sign Up* akan tertutup dan *window Sign In* terbuka.



Gambar 3.28 Tampilan (Sign In Berhasil)

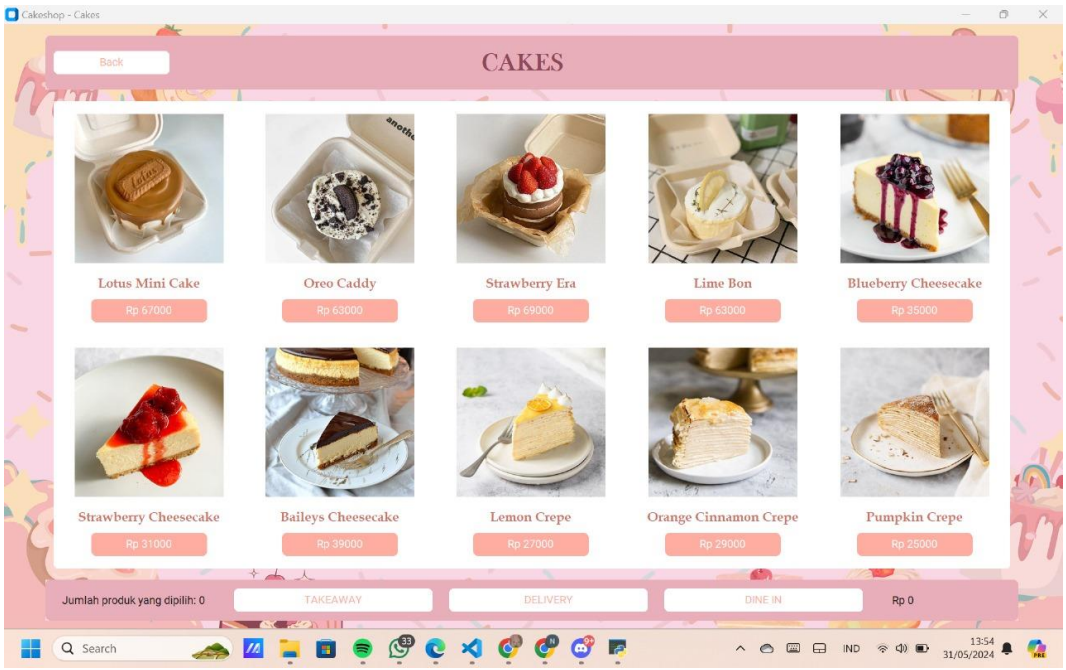
Setelah menginput *username* dan *password* yang sesuai maka akan muncul *pop up* bahwa *login* berhasil.

Gambar 3.29 Tampilan *(Homepage)*

Setelah melakukan *login*, *window sign in* akan tertutup dan digantikan dengan *window homepage*, dalam *window homepage* terdapat keterangan nama *cake shop* kami. Ada 4 jenis menu yang ditawarkan yaitu, *breads*, *cakes*, *donuts*, dan *pastry*. Keempatnya dapat diklik untuk memilih *breads*, *cakes*, *donuts* atau *pastry* yang diinginkan secara spesifik.

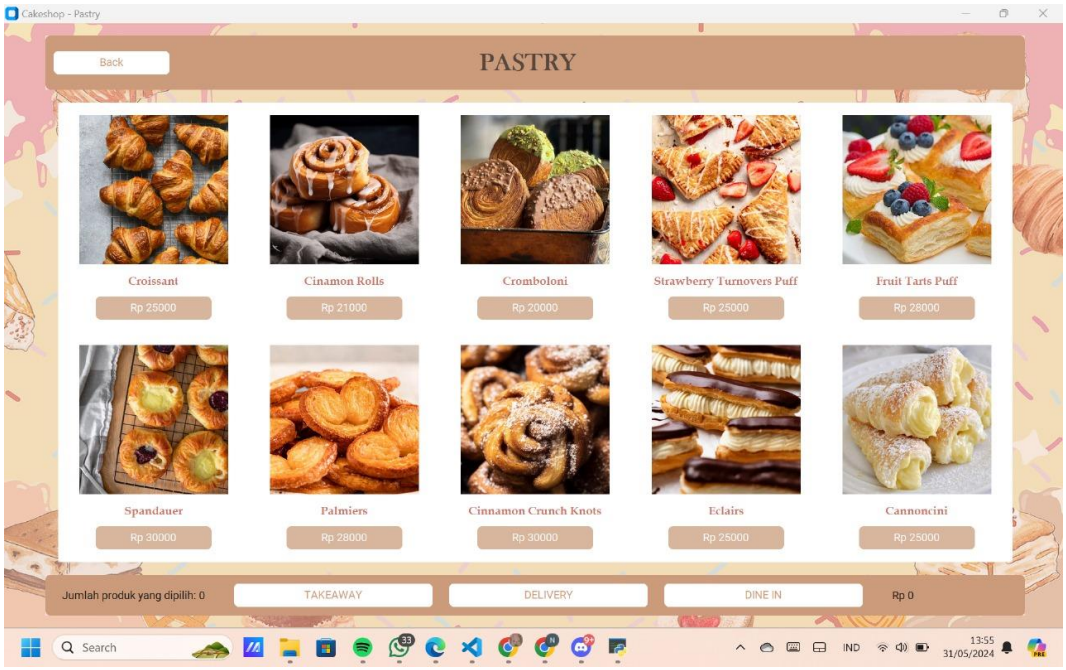
Gambar 3.30 Tampilan *(Menu Breads)*

Jika pengguna memilih *breads*, pengguna akan diberikan 10 macam jenis roti dengan berbagai rasa dan harga yang dapat dipilih.



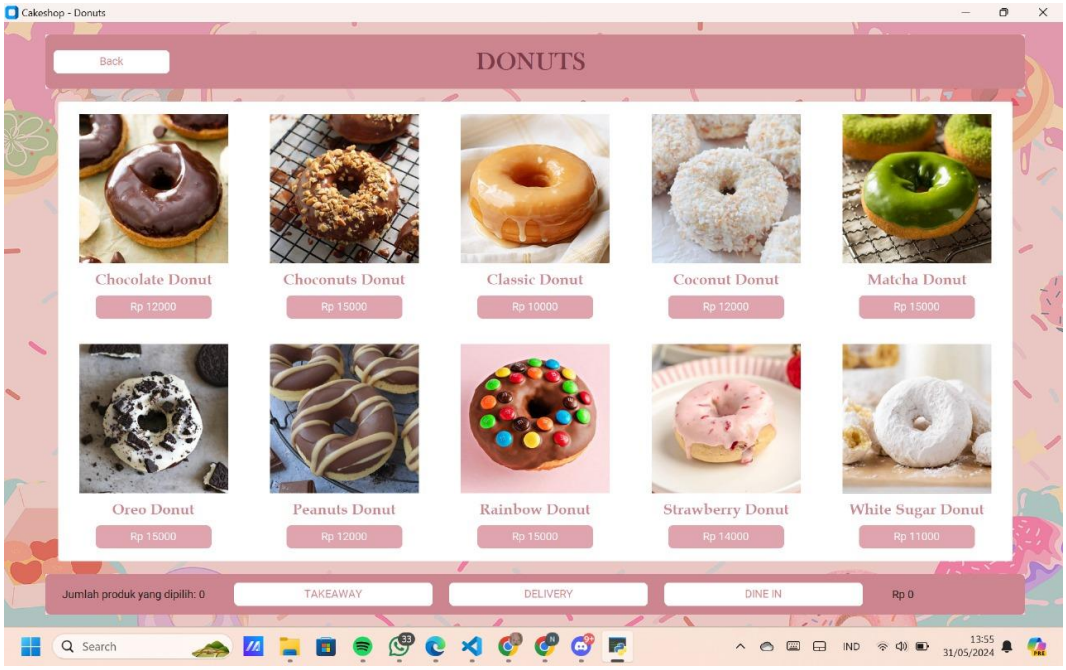
Gambar 3.31 Tampilan**31**Tampilan *(Menu Cakes)*

Jika pengguna memilih *cakes*, pengguna akan diberi tampilan dengan 10 macam jenis kue dengan berbagai rasa dan harga yang dapat dipilih.



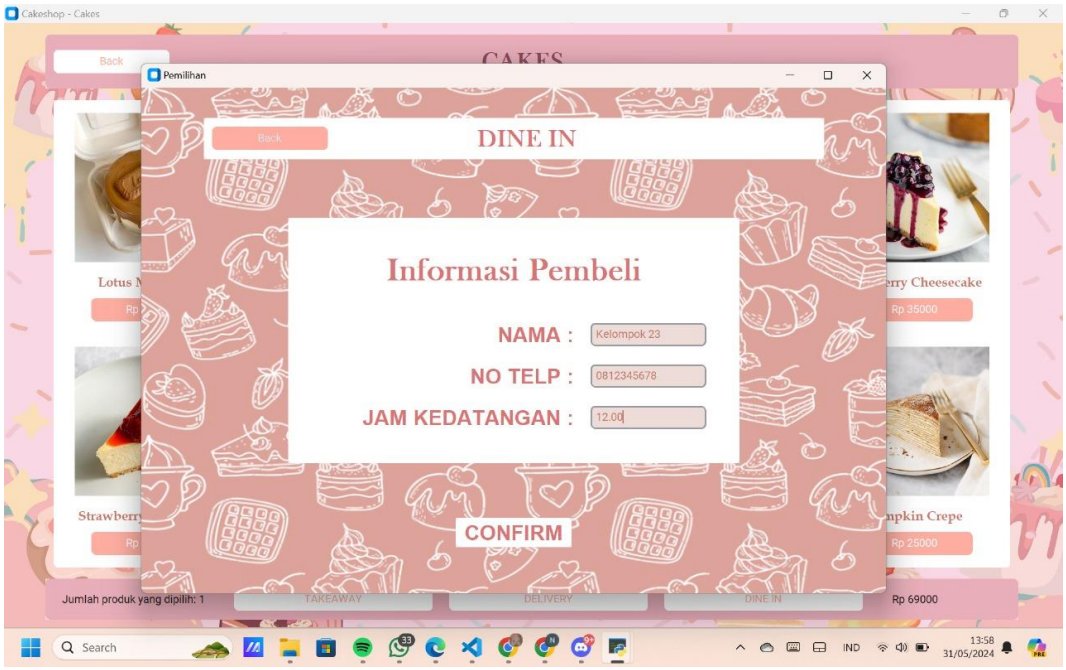
Gambar 3.32 Tampilan *(Menu Pastry)*

Jika pengguna memilih *pastry*, pengguna akan diberi tampilan dengan 10 macam jenis *pastry* dengan berbagai rasa dan harga yang dapat dipilih.



Gambar 3.33 Tampilan *(Menu Donuts)*

Jika pengguna memilih *donuts*, pengguna akan diberi tampilan dengan 10 macam jenis donat dengan berbagai rasa dan harga yang dapat dipilih.



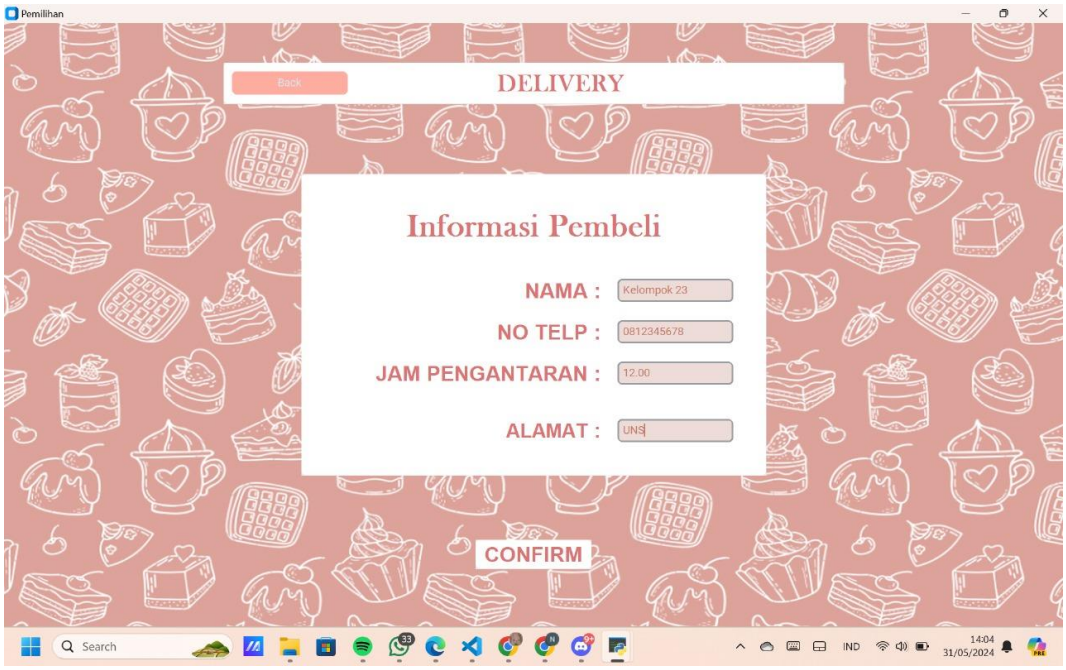
Gambar 3.34 Tampilan (Opsi Pembayaran)

Setelah memilih produk yang ingin dibeli, pengguna diberikan 3 opsi untuk pengambilan yaitu ada *Dine In, Take Away, Delivery*, apabila pengguna memilih *Dine In*, maka akan menghasilkan *pop up* seperti pada gambar di atas.

Gambar 3.35 Tampilan (Opsi Pengambilan Barang)

Gambar 3.36 Tampilan (Opsi Pengambilan Barang)

Jika pengguna memilih *Take Away*, pengguna wajib mengisi informasi pembeli, seperti nama, no telepon, dan jam kedatangan sama seperti jika pengguna memilih *Dine In.* Jika pengguna sudah mengisi informasi pembeli, pengguna bisa memencet tombol *confirm* untuk melanjutkan ke tahap pembayaran.



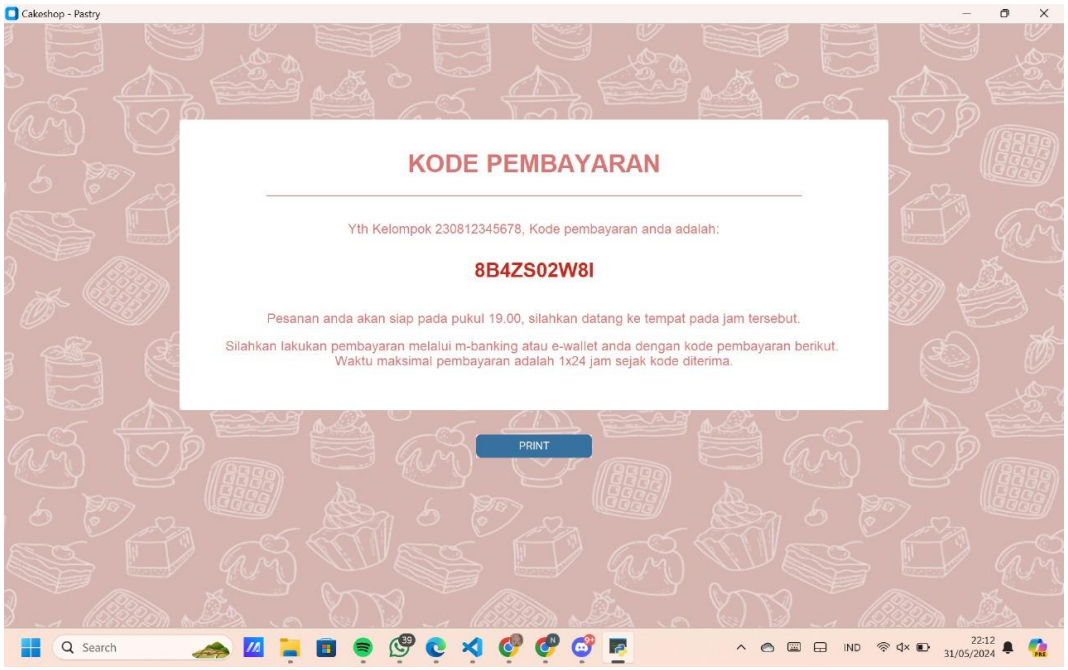
Gambar 3.37 Tampilan (Opsi Pengambilan Barang)

Sedangkan jika pengguna memilih *Delivery*, pengguna wajib mengisi informasi pembeli, seperti nama, no telepon, jam pengantaran dan alamat. Jika pengguna

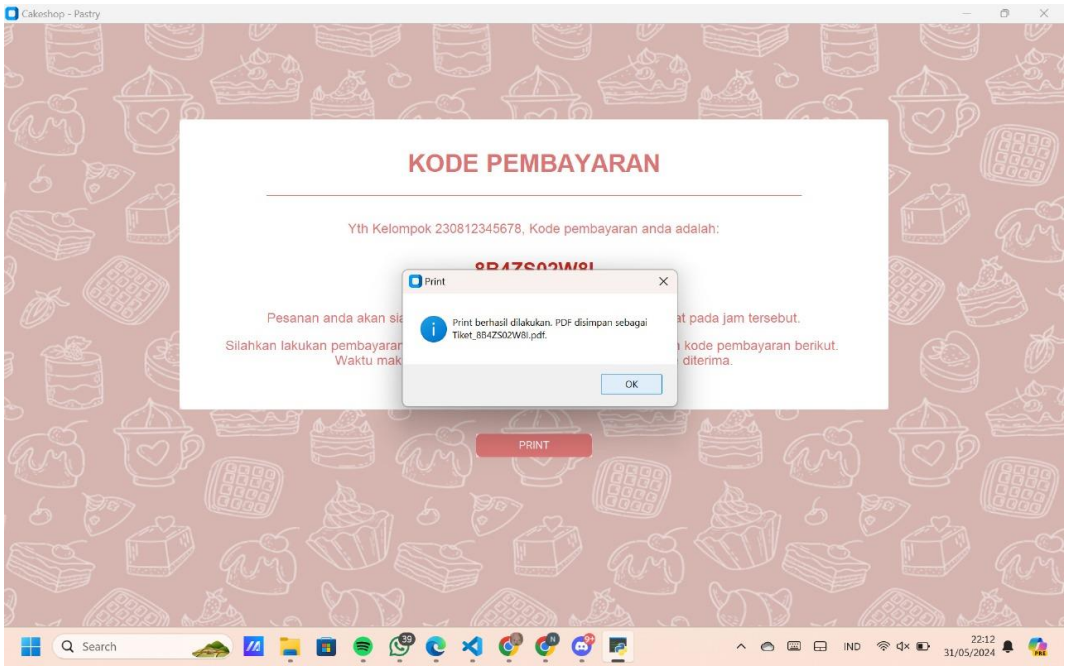
sudah mengisi informasi pembeli, pengguna bisa memencet tombol *confirm* untuk melanjutkan ke tahap pembayaran.

Gambar 3.38 Tampilan (Opsi Total Pesanan dan Pembayaran)

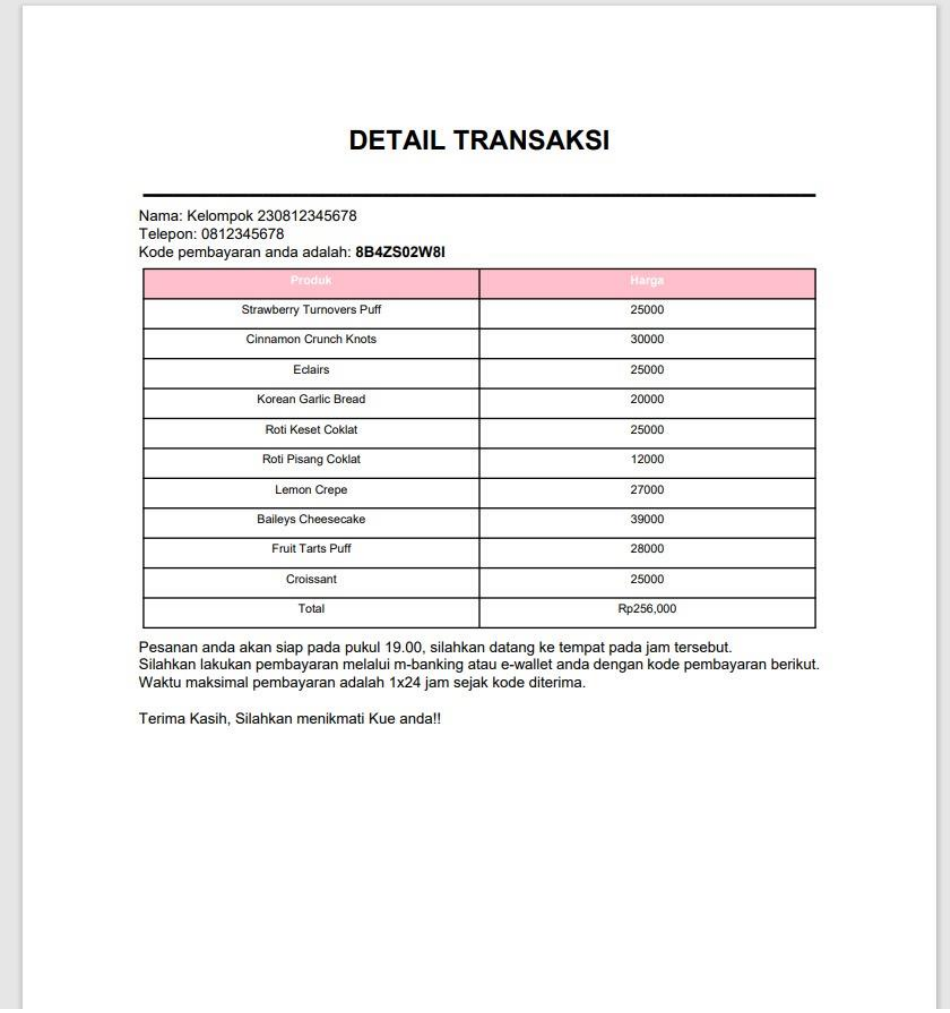
Setelah pengguna memilih produk menu dan memilih opsi pengambilan dan melakukan *confirm* informasi pembeli *window* akan menampilkan ringkasan pesanan dan total pesanan pengguna. Terdapat juga tombol *remove* jika pengguna ingin menghapus pilihan menu yang sudah dipilih sebelumya. Namun jika pengguna tidak ingin melanjutkan pembayaran maka pengguna bisa memencet tombol batal maka akan kembali ke halam sebelumnya yaitu informasi pembeli pada opsi pengambilan. Selain itu terdapat opsi pemilihan pembayaran yaitu tunai dan non-tunai yang dapat dipilih pengguna sesuai keinginan. Setelah keduanya sesuai keinginan pengguna, pengguna bisa memencet tombol *checkout* untuk lanjut ke halaman *invoice* dan mendapatkan kode pembayaran.



Gambar 3.39 Tampilan (*Invoice* Kode Pembayaran)



Gambar 3.40 Tampilan (*Pop Up* setelah klik *Print*)



DETAIL TRANSAKSI

Nama: Kelompok 230812345678
Telepon: 0812345678
Kode pembayaran anda adalah: **8B4ZS02W8I**

Produk	Harga
Strawberry Turnovers Puff	25000
Cinnamon Crunch Knots	30000
Eclairs	25000
Korean Garlic Bread	20000
Roti Keset Coklat	25000
Roti Pisang Coklat	12000
Lemon Crepe	27000
Baileys Cheesecake	39000
Fruit Tarts Puff	28000
Croissant	25000
Total	Rp256,000

Pesanan anda akan siap pada pukul 19.00, silahkan datang ke tempat pada jam tersebut.
Silahkan lakukan pembayaran melalui m-banking atau e-wallet anda dengan kode pembayaran berikut.
Waktu maksimal pembayaran adalah 1x24 jam sejak kode diterima.

Terima Kasih, Silahkan menikmati Kue anda!!

Gambar 3.41 Tampilan (*Invoice* pdf)

Setelah pengguna menentukan opsi pembayaran dan memencet tombol *checkout* maka akan ditampilkan halaman *invoice* yang berisi kode pembayaran dan informasi lanjutan tentang pengambilan maupun pengantaran pesanan. Setelah pengguna memencet tombol *print*, maka akan muncul pesan *pop up invoice* tersimpan. *Invoice* yang tercetak dalam bentuk pdf secara otomatis terbuka di *webbrowser* yang dapat disimpan oleh pengguna sebagai barang bukti saat akan mengambil pesanan ke toko. Setelah itu halaman akan secara otomatis kembali ke halaman *homepage* apabila pengguna ingin memesan kembali dan tersedia tombol *logout* apabila pengguna tidak ingin keluar dari aplikasi.

BAB IV

PENUTUP

Kesimpulan dari hasil perancangan program Cake Shop 23BEE adalah sebagai berikut:

1. Program Cake Shop 23BEE menawarkan sistem online dimana pelanggan dapat dengan mudah mencari dan memesan segala jenis roti atau *dessert.* Dengan fitur-fitur yang mudah digunakan, pelanggan dapat dengan mudah memperoleh informasi produk dan melakukan pemesanan dari mana saja dengan praktis.
2. Sistem pemesanan yang diaplikasikkan dalam program ini mampu mengurangi waktu tunggu pelanggan. Proses pemesanan yang tersusun dan otomatisasi dalam penanganan pesanan memastikan bahwa setiap pesanan diproses dengan cepat dan efisien.
3. Program Cake Shop 23BEE menyediakan metode pembayaran dengan beberapa pilihan pembayaran diantaranya tunai dan non-tunai sehingga pelanggan dapat memilih metode pembayaran mana yang sesuai dengan kebutuhan mereka. Hal tersebut dapat mengurangi permasalahan yang terjadi saat melakukan pembayaran secara fisik di toko.