اونيۏرسيتي مليسيا ڤهڠ السلطان عبدالله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BSD2213 DATA SCIENCE PROGRAMMING I

TITLE:

E-Commerce Platform: SparkTech.Co

LECTURER'S NAME:

DR NORAZIAH BINTI ADZHAR

SECTION 01:

GROUP 07

MATRIC ID	NAME
SD22003	NUR ATIEKA RAFIEKAH BINTI RAZAK
SD22038	NURIN NADHIRAH IZZAH BINTI ZAINUDDIN
SD21019	AINA ISMA NAJWA BINTI AZARZAINI
SD22057	MUHAMMAD FIRAS BIN IRWAN

Table of Contents

1.0 Introduction

SparkTech.Co is an online platform where you can buy gadget accessories. The platform consists of personal information, products, order summary, payment, and admin also can access to get the data of all customers. Through a simple registration process, clients may safely give their personal data. Customers can also search and add items to the cart with just in a few clicks. In an era that is defined by technological evolution, SparkTech.Co is not just a store. It is also a platform that has been designed to cater to tech and gadget enthusiasts. SparkTech.Co also provide a wide selection of electronic accessories to simplify and improve digital life. Since our inception, we have been driven to provide cutting-edge solutions that complement with the dynamic needs demands of our tech-savvy customer.

Objective:

- To develop a user-friendly in our interface.
- To ensure that our customers can effortlessly explore, compare, and make informed decisions.
- To track the number of customers that purchase on our online platform.

Limitation:

- Users are encouraged to have a stable internet connection to access to our platform.
- Some products may be released in limited quantities, leading to high demand.

2.0 Why This Project Is Important?

.
The reason why we decided to proceed with this project is we all as a team believe that it is the trend nowadays people tend to simplify their shopping process via e-commerce. Therefore, by introducing a good interface we are able to fulfill the customer needs smoothly in the shopping process. Since the profits not only from customers who walk into the shops but also from online customers, it will increase and optimize our SparkTech.Co sales and revenue.

One of the most innovative features of GUI SparkTech.co is its advanced search features. This user-friendly search function that customers can easily to find the desired items by allowing them to quickly locate and explore preferred items. This innovative approach not only simplifies navigation but also ensures that users can quickly find exactly the items they want. As a result, customers have higher satisfaction and a high probability to repeat purchasing at our platform.

In addition, our platform includes several analysis tools to aid administrators in their tasks. For instance, in the end, admins are able to observe the charts of which product is the highest or lowest sold, percentages of male and female customers and more. These tools provide critical insights into the administrators to observe trends and patterns in sales data. This helps them make informed decisions based on sales data in a short time. Thus administrators can make informed choices on how to optimize operations, boost profits and sales, and build excellent management behavior.

3.0 How Can This Project Be Extended?

However, these limitations provide opportunities for improvements and extensions. One way this project can be augmented is by adding advanced analysis tools, for example, predictive sales forecasting and user behaviour analysis. These additions could elevate this project's functionality significantly by offering the administrators deeper insights into the sales trends and enabling more informed and less time-consuming decision-making processes.

This project can also be extended by making the GUI bilingual, which will enable the GUI to cater to a broader audience and ensure inclusivity. This functionality could reach diverse user bases and create a more accessible and user-friendly interface. Bilingual support not only expands this project's reach but also aligns with the global nature of modern business environments.

Additionally, the enhancement for this project could also include the incorporation of new interesting features that will aim to improve users' experience when using the platform. For instance, a product reviews feature, and a recommendation system based on the users' algorithm can be introduced as substantial values. These will engage users and contribute to a more personalized and interactive experience for them. Users and sellers can benefit from the insights shared by other users through reviews, and the recommendation system will also add a new layer of convenience by suggesting products to customers based on their search preferences.

To sum up, this project can be expanded by including sophisticated analysis tools, guaranteeing language inclusivity with bilingual assistance, and adding more features that are focused on the needs of the user. These additions make the platform more resilient, flexible, and user-friendly and at the same time remove or lessen the current limitations of this project.

4.0 Source Code

```
import tkinter as tk
from tkinter import messagebox, Scrollbar
import matplotlib.pyplot as plt
import pandas as pd
from PIL import ImageTk, Image
from tkinter import Label
from datetime import datetime
import csv


class SparkTechApp:
    def __init__(self):
        super().__init__()
        self.prev_frame = None
        self.prev_frame_time = None
        self.prev_centroid = None
        self.window = tk.Tk()

    def mainloop(self):
        self.window.title("SparkTech.Co")
        self.window.geometry("1920x1080")
        # Use a raw string for the file path
        logo = r'C:\Users\nurra\Downloads\WhatsApp Image 2024-01-09 at 4.32.00 PM.jpeg'
        image = Image.open(logo)
        bg_image = ImageTk.PhotoImage(image)
        background_label = Label(self.window, image=bg_image, bg="#F4E572")
        background_label.place(x=0, y=0, relwidth=1, relheight=1)

        label = tk.Label(self.window, text="WELCOME TO SPARKTECH.CO", font=("Vogue", 30,
'bold'),
                         bg="#F4E572", fg="black")
        label.pack(pady=10)
        label = tk.Label(self.window, text="Click the Start button to begin",
font=("Helvetica", 12, "bold"), bg="#F4E572", fg="cyan4")
        label.pack(pady=10)
        start_button = tk.Button(self.window, text="Start", command=self.front_page,
font=("Helvetica", 12),
                                 bg="#000000", fg="white",
                                 width=30, height=2)
        start_button.pack(pady=30, padx=30)
        self.window.mainloop()

    def front_page(self):
        front_page = FrontPage()
        self.window.withdraw()







class FrontPage(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("SparkTech.Co")
        self.configure(bg="#F4E572")  # Light peach background color
        self.geometry("1920x1080")
        self.selected_date = datetime.now().strftime('%d-%m-%Y')
        tk.Label(self, text="Date:", bg="#F4E572", fg="#000000").pack(pady=5)
        self.date_label = tk.Label(self, text=self.selected_date, bg="#F4E572", fg="#3D405B",
font=("Helvetica", 12))
        self.date_label.pack(pady=10)

    # Adding Menu Bar
        menubar = tk.Menu(self)
        about_menu = tk.Menu(menubar, tearoff=0)
        about_menu.add_command(label="Help", command=self.show_about)
        menubar.add_cascade(label="Menu", menu=about_menu)
        self.config(menu=menubar)

        tk.Label(self, text="Welcome to SparkTech.Co", bg="#F4E572", font=("Helvetica",
16)).pack(pady=20)
        tk.Button(self, text="Register as Customer", command=self.customer_mode, bg="#FF857F",
fg="black",
              font=("Helvetica", 16),
              width=30, height=2).pack(pady=10)
        tk.Button(self, text="Admin Login", command=self.admin_mode, bg="#66BFBF", fg="black",
font=("Helvetica", 16),
              width=30, height=2).pack(pady=10)

        exit_button = tk.Button(self, text="Exit", command=self.destroy, bg="#000000",
fg="white",
                              width=30, height=2)
        exit_button.pack(pady=5, padx=10)

    def customer_mode(self):
        self.destroy()
        PersonalInfoPage()

    def admin_mode(self):
        self.destroy()
        AdminLoginPage()

    def show_about(self):
    about_message = (
        "Welcome to SPARKTECH.CO, where cutting-edge meets convenience in the world of
technology."
        "Established with a passion for enhancing your digital lifestyle, we curate a range of
high-quality and innovative accessories to complement your devices.\n"
        "Since 2023\n\n"

        "For any inquiries:\n"
        "Email: sparktech@gmail.com\n"
        "Call : 09-8676324\n\n"

        "Thank you for choosing SPARKTECH.CO. We're excited to serve you and assist with any
questions you may have."
    )
    messagebox.showinfo("About SparkTech.Co", about_message)



class PersonalInfoPage(tk.Toplevel):  # Change to Toplevel
    def __init__(self):
        super().__init__()
        self.title("Personal Information")
        self.configure(bg="#F4E572")
        self.geometry("1920x1080")

        self.user_name = tk.StringVar()
        self.user_email = tk.StringVar()
        self.user_gender = tk.StringVar()
        self.user_gender.set(" ")
        self.user_phone = tk.StringVar()
        self.user_address = tk.StringVar()

        tk.Label(self, text="Name:").pack(pady=10)
        self.name_entry = tk.Entry(self, textvariable=self.user_name)
        self.name_entry.pack(pady=15)
        self.name_entry.bind("<Return>", lambda event: self.next_entry(self.email_entry))

        tk.Label(self, text="Gender:").pack(pady=10)
        male = tk.Radiobutton(self, text='Male', variable=self.user_gender, value='Male')
        male.pack(pady=15)
        female = tk.Radiobutton(self, text='Female', variable=self.user_gender,
value='Female')
        female.pack(pady=15)

        tk.Label(self, text="Email:").pack(pady=10)
        self.email_entry = tk.Entry(self, textvariable=self.user_email)
        self.email_entry.pack(pady=15)
        self.email_entry.bind("<Return>", lambda event: self.next_entry(self.phone_entry))

        tk.Label(self, text="Phone:").pack(pady=10)
        self.phone_entry = tk.Entry(self, textvariable=self.user_phone)
        self.phone_entry.pack(pady=15)
        self.phone_entry.bind("<Return>", lambda event: self.next_entry(self.address_entry))

        tk.Label(self, text="Address:").pack(pady=10)
        self.address_entry = tk.Entry(self, textvariable=self.user_address)
        self.address_entry.pack(pady=15)

        self.subscription_status = tk.BooleanVar(value=False)  # Initialize subscription
status

        # Subscription toggle
        self.subscription_toggle = tk.Checkbutton(self, text="Subscribe to our news",
variable=self.subscription_status,
                                                  onvalue=True, offvalue=False, bg="#F4E572")
        self.subscription_toggle.pack(pady=15)

        tk.Button(self, text="Shop Now!", command=self.next_page).pack(pady=10)
        self.cart = []  # Add the cart attribute

    def next_entry(self, entry_widget):
        entry_widget.focus()

    def next_page(self):
        if not self.user_name.get() or not self.user_email.get() or not self.user_phone.get():
            messagebox.showerror("Error", "Please fill out all required fields.")
            return
        personal_info = {
            "name": self.user_name.get(),
            "email": self.user_email.get(),
            "gender": self.user_gender.get(),
            "phone": self.user_phone.get(),
            "address": self.user_address.get(),
            "subscription": self.subscription_status.get()
        }

        ProductSelectionPage(self, personal_info)


class ProductSelectionPage(tk.Toplevel):
    def __init__(self, master, personal_info, entry=None):
        super().__init__(master)
        self.title("Product Selection")
        self.master = master
        self.personal_info = personal_info
        self.config(bg="#F4E572")
        self.geometry("1920x1080")
        self.sales_data = []
        self.image_labels = []

        master.withdraw()
        self.products = [
            {"id": 1, "name": "Phone Case", "price": 10.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\phone case.jpg'},
            {"id": 2, "name": "Power Banks", "price": 30.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\power bank.jpg'},
            {"id": 3, "name": " Wireless Earphone", "price": 25.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\wireless earphone.jpg'},
            {"id": 4, "name": "Charging Cables", "price": 18.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\charging cable.jpg'},
            {"id": 5, "name": "Bluetooth Speakers", "price": 35.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\bluetooth speaker.jpg'},
            {"id": 6, "name": " Wired Earphones ", "price": 18.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\wired earphones.jpg'},
            {"id": 7, "name": "Charging Adapter", "price": 15.00, "quantity": tk.IntVar(),
            "picture": r'C:\Users\nurra\Downloads\charging adapter.jpg'},
        ]

        self.filtered_products = self.products
        tk.Label(self, text="Search Product:").pack(pady=5)
        self.search_entry = tk.Entry(self)
        self.search_entry.pack(pady=5)
        self.search_entry.bind("<KeyRelease>", self.filter_products)
        self.product_frames = []

        scrollable_container = tk.Canvas(self, bg="#F4E572")  # Or use a Frame if preferred
        scrollable_container.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        sb = Scrollbar(self, width=30)
        sb.pack(side=tk.RIGHT, fill=tk.BOTH)

        container = tk.Frame(scrollable_container, bg="#F4E572")
        container.bind("<Configure>",
                   lambda e:
scrollable_container.configure(scrollregion=scrollable_container.bbox("all")))

        scrollable_container.create_window((0, 0), window=container, anchor="nw")

        for product in self.filtered_products:
            product_frame = tk.Frame(container, bg="#F4E572")

            # Load image using PhotoImage
            image = Image.open(product['picture'])
            image = image.resize((100, 100))  # Adjust the size as needed
            photo = ImageTk.PhotoImage(image)

            # Create a label to display the image
            image_label = tk.Label(product_frame, image=photo, bg="#F4E572")
            image_label.image = photo  # Keep a reference to avoid garbage collection
            image_label.grid(row=0, column=0, padx=100, pady=30)
            self.image_labels.append(image_label)

            tk.Label(product_frame, bg="#F4E572", text=f"{product['name']} - RM
{product['price']: .2f}").grid(row=0, column=1, padx=100,

pady=5)
            tk.Label(product_frame, bg="#F4E572", text="Quantity:").grid(row=0, column=2,
padx=5, pady=5)
            entry = tk.Entry(product_frame, textvariable=product['quantity'])
            entry.grid(row=0, column=3, padx=5, pady=5)
            entry.bind("<Return>", lambda event, entry=tk.Entry: self.next_entry(entry))
            product_frame.pack(pady=5)

            sb.config(command=scrollable_container.yview)
            scrollable_container.config(yscrollcommand=sb.set)

        tk.Label(self, text="Cart:").pack(pady=10)
        self.cart_listbox = tk.Listbox(self, selectmode=tk.MULTIPLE, width=40, height=5)
        self.cart_listbox.pack(pady=10)
        self.update_cart_listbox()

        tk.Button(self, text="Remove from Cart", command=self.remove_from_cart).pack(pady=5)
        tk.Button(self, text="Add to Cart", command=self.add_to_cart).pack(pady=10)
        tk.Button(self, text="View Summary", command=self.view_summary).pack(pady=10)
        tk.Button(self, text="Back", command=self.back_to_first_page).pack(pady=10)

        sb.config(command=scrollable_container.yview)
        scrollable_container.config(yscrollcommand=sb.set)

        # Inside the ProductSelectionPage class
    def update_product_widgets(self):
        for frame in self.product_frames:
            frame.destroy()
        self.product_frames = []

        for product in self.filtered_products:
            product_frame = tk.Frame(self, bg="#F4E572")

            # Load image using PhotoImage
            image = Image.open(product['picture'])
            image = image.resize((100, 100))  # Adjust the size as needed
            photo = ImageTk.PhotoImage(image)

            # Create a label to display the image
            image_label = tk.Label(product_frame, image=photo, bg="#F4E572")
            image_label.image = photo  # Keep a reference to avoid garbage collection
            image_label.grid(row=0, column=0, padx=10, pady=10)
            self.image_labels.append(image_label)

            tk.Label(product_frame, text=f"{product['name']} - RM
{product['price']:.2f}").grid(row=0, column=1,

padx=10, pady=10)
            tk.Label(product_frame, bg="#F4E572", text="Quantity:").grid(row=0, column=2,
padx=10, pady=10)
            entry = tk.Entry(product_frame, bg="#F4E572", textvariable=product['quantity'])
            entry.grid(row=0, column=3, padx=10, pady=10)
            entry.bind("<Return>", lambda event, entry=tk.Entry: self.next_entry(entry))

            product_frame.pack(pady=5)

            sb.config(command=scrollable_container.yview)
            scrollable_container.config(yscrollcommand= sb.set)

    def next_page(self):
        if not self.personal_info['name'] or not self.personal_info['email'] or not
self.personal_info['phone']:
            messagebox.showerror("Error", "Please fill out all required fields.")
            return
        self.customer_data = self.generate_customer_data()

        personal_info = {
            "name": self.user_name.get(),
            "email": self.user_email.get(),
            "gender": self.user_gender.get(),
            "phone": self.user_phone.get(),
            "address": self.user_address.get()
        }
        SummaryPage(self.master, self.personal_info, self.customer_data)

    def filter_products(self, event):
        search_query = self.search_entry.get().lower()
        self.filtered_products = [product for product in self.products if search_query in
product['name'].lower()]
        self.update_product_widgets()
        self.photo_images()

    def update_product_widgets(self):
        for frame in self.product_frames:
            frame.destroy()
        self.product_frames = []

        for product in self.filtered_products:
            product_frame = tk.Frame(self, bg="#F4E572")
            tk.Label(product_frame, bg="#F4E572", text=f"{product['name']} - RM
{product['price']:.2f}").grid(row=0, column=0,

padx=5, pady=5)
            tk.Label(product_frame, bg="#F4E572", text="Quantity:").grid(row=0, column=1,
padx=5, pady=5)
            entry = tk.Entry(product_frame, textvariable=product['quantity'])
            entry.grid(row=0, column=2, padx=5, pady=5)
            entry.bind("<Return>", lambda event, entry=tk.Entry: self.next_entry(entry))
            product_frame.pack(pady=10)
            self.product_frames.append(product_frame)

    def next_entry(self, entry_widget):
        entry_widget.tk_focusNext().focus()

    def add_to_cart(self):
        for product in self.filtered_products:
            quantity = product['quantity'].get()
            if quantity > 0:
                selected_product = {
                    "id": product["id"],
                    "name": product["name"],
                    "price": product["price"],
                    "quantity": quantity
                }
                self.master.cart.append(selected_product)

            # Calculate total price for the current order
            total_price = sum(product['price'] * product['quantity'].get() for product in
self.filtered_products)

        # Append daily sales data with the current date
        current_date = datetime.now().strftime("%Y-%m-%d")
        self.sales_data.append({"date": current_date, "total_sales": total_price})

        self.update_cart_listbox()

        if not self.master.cart:
            messagebox.showinfo("Error", "Please specify the quantity for at least one
product.")
        else:
            messagebox.showinfo("Success", f"Products added to the cart for
{self.personal_info['name']}.")

    def update_cart_listbox(self):
        self.cart_listbox.delete(0, tk.END)

        for product in self.master.cart:
            self.cart_listbox.insert(tk.END, f"{product['name']} - Quantity:
```

```
{product['quantity']}")

    def remove_from_cart(self):
        selected_indices = self.cart_listbox.curselection()
        for index in reversed(selected_indices):
            del self.master.cart[index]
        self.update_cart_listbox()

    def back_to_first_page(self):
        self.destroy()
        # Show the PersonalInfoPage again
        self.master.deiconify()

    def pay_now(self):
        self.destroy()
        PaymentPage(self.master, self.personal_info, self.master.cart)

    def save_customer_data(self, personal_info):
        fieldnames = ['Date', 'Name', 'Email', 'Phone', 'Address', 'Subscription', 'Gender',
'Product', 'Quantity',
                      'Price', 'TotalPrice']
        filename = 'customer_data.csv'

        with open(filename, mode='a', newline='') as file:
            writer = csv.DictWriter(file, fieldnames=fieldnames)

            # Check if the file is empty and write headers if needed
            file.seek(0, 2)  # Move the cursor to the end of the file
            if file.tell() == 0:
                writer.writeheader()

            # Write customer data to the CSV file
            for product in self.master.cart:
                quantity = product['quantity']  # Get the quantity directly
                writer.writerow({
                    'Date': datetime.now().strftime('%Y-%m-%d'),  # Save the current date and
time
                    'Name': personal_info['name'],
                    'Email': personal_info['email'],
                    'Phone': personal_info['phone'],
                    'Address': personal_info['address'],
                    'Subscription': personal_info.get('subscription', ''),
                    'Gender': personal_info.get('gender', ''),
                    'Product': product['name'],
                    'Quantity': quantity,  # Use the fetched quantity
                    'Price': product['price'],
                    'TotalPrice': product['price'] * quantity  # Calculate total price here
                })

    def view_summary(self):
        if not self.master.cart:
            messagebox.showerror("Error", "Please add items to the cart.")
            return
        self.destroy()
        self.save_customer_data(self.personal_info)
        SummaryPage(self.master, self.personal_info, self.master.cart)


class PaymentPage(tk.Toplevel):
    def __init__(self, master, personal_info, cart):
        super().__init__(master)
        self.title("Payment Information")
        self.configure(bg="#F4E572")
        self.geometry("1920x1080")

        self.personal_info = personal_info
        self.cart = cart
        tk.Label(self, text="Payment Information", font=("Vogue", 20, "bold")).pack(pady=10)

        self.payment_method = tk.StringVar()
        self.payment_method.set("Select Payment Method".ljust(40))

        Label(self, text="Select Payment Method", font=("Helvetica", 16),
bg="#F4E572").pack(pady=30)
        tk.OptionMenu(self, self.payment_method, "Select", "Credit / Debit Card", "Online
Banking", "Cash on Delivery",
                      "Touch 'n Go eWallet", "GrabPay").pack(pady=10)
        self.payment_method.trace("w", self.show_payment_fields)

        tk.Button(self, text="Back to Product Selection",
command=self.back_to_product_selection, font=("Helvetica", 14), width=20,
height=1).pack(pady=120)

    def show_payment_fields(self, *args):
        for widget in self.winfo_children():
            widget.destroy()

            # Show different fields based on the selected payment method
        payment_method = self.payment_method.get()

        if payment_method == "Credit / Debit Card":
            tk.Label(self, text="Card Number:").pack(pady=5)
            self.card_number_entry = tk.Entry(self)
            self.card_number_entry.pack(pady=5)

            tk.Label(self, text="Expiration Date:").pack(pady=5)
            self.expiry_date_entry = tk.Entry(self)
            self.expiry_date_entry.pack(pady=5)

            tk.Label(self, text="CVV:").pack(pady=5)
            self.cvv_entry = tk.Entry(self)
            self.cvv_entry.pack(pady=5)
            tk.Button(self, text="Pay Now ", command=self.pay_now, bg="#75FF5F").pack(pady=7,
padx=10)
            tk.Button(self, text="Back to Payment Selection",
command=self.back_to_payment_selection).pack(pady=5,padx=10)

        elif payment_method == "Online Banking":
            tk.Label(self, text="Bank Name:", font=("Helvetica", 14)).pack(pady=5)
            bank_frame = tk.Frame(self)
            bank_frame.pack(pady=10)

            # Define the available banks
            self.banks = [
                "Select Bank", "Bank Islam", "Maybank", "Bank Rakyat", "RHB", "CIMB",
                "Hong Leong Bank", "BSN", "Bank Muamalat", "Alliance Bank", "HSBC Bank"
            ]

            # Create a Spinbox for bank selection
            self.selected_bank = tk.StringVar()
            self.bank_spinbox = tk.Spinbox(bank_frame, values=self.banks,
textvariable=self.selected_bank)
            self.bank_spinbox.pack(padx=10, pady=5, side=tk.LEFT)

            tk.Label(self, text="Username:", font=("Helvetica", 14)).pack(pady=5)
            self.username_entry = tk.Entry(self)
            self.username_entry.pack(pady=5)

            tk.Label(self, text="Password:", font=("Helvetica", 14)).pack(pady=5)
            self.password_entry = tk.Entry(self, show="*")
            self.password_entry.pack(pady=5)
            tk.Button(self, text="Pay Now ",
command=self.pay_now,bg="#75FF5F").pack(pady=7,padx=10)
            tk.Button(self, text="Back to Payment Selection",
command=self.back_to_payment_selection).pack(pady=5,padx=10)
            pass

        elif payment_method == "Cash on Delivery":
            tk.Label(self, text=f"We will COD at {self.personal_info['address']}",
font=("Helvetica", 14)).pack(pady=10)
            tk.Button(self, text="Pay Now ", command=self.pay_now, bg="#75FF5F").pack(pady=7,
padx=10)
            tk.Button(self, text="Back to Payment Selection",
command=self.back_to_payment_selection).pack(pady=5, padx=10)

        elif payment_method == "Touch 'n Go eWallet" or payment_method == "GrabPay":
            tk.Label(self, text="Username:", font=("Helvetica", 14)).pack(pady=5)
            self.username_entry = tk.Entry(self)
            self.username_entry.pack(pady=5)

            tk.Label(self, text="Password:", font=("Helvetica", 14)).pack(pady=5)
            self.password_entry = tk.Entry(self, show="*")
            self.password_entry.pack(pady=5)
            tk.Button(self, text="Pay Now ", command=self.pay_now, bg="#75FF5F").pack(pady=7,
padx=10)
            tk.Button(self, text="Back to Payment Selection",
command=self.back_to_payment_selection).pack(pady=5, padx=10)
            pass
        else:
            print("Please select a payment method.")

    def pay_now(self):
        payment_method = self.payment_method.get()
        if payment_method == "Select":
            messagebox.showerror("Error", "Please select a payment method.")
            return
        else:
            messagebox.showinfo("Payment Successful",
                                 f"Payment successful with {payment_method}. Thank you for your
purchase!")

        self.destroy()
        front_page = FrontPage()
        front_page.mainloop()

    def back_to_payment_selection(self):
        self.destroy()
        PaymentPage(self.master, self.personal_info, self.cart)

    def back_to_product_selection(self):
        self.destroy()
        ProductSelectionPage(self.master, self.personal_info)


class SummaryPage(tk.Toplevel):
    def __init__(self, master, personal_info, cart):
        super().__init__(master)
        self.title("Order Summary")
        self.master = master
        self.personal_info = personal_info
        self.geometry("1920x1080")
        self.cart = cart

        background_color = "#F4E572"
        text_color = "#000000"
        self.configure(bg=background_color)

        tk.Label(self, text="Order Summary", bg=background_color, fg=text_color,
font=("Vogue", 20, "bold")).pack(
            pady=10)
        tk.Label(self, text=f"Name: {personal_info['name']}", bg=background_color,
fg=text_color, font=("Times New Roman", 14)).pack()
        tk.Label(self, text=f"Email: {personal_info['email']}", bg=background_color,
fg=text_color, font=("Times New Roman", 14)).pack()
        tk.Label(self, text=f"Phone: {personal_info['phone']}", bg=background_color,
fg=text_color, font=("Times New Roman", 14)).pack()
        tk.Label(self, text=f"Address: {personal_info['address']}", bg=background_color,
fg=text_color, font=("Times New Roman", 14)).pack()
        subscription_status = "Subscribed" if personal_info.get("subscription") else "Not
Subscribed"
        tk.Label(self, text=f"Subscription Status: {subscription_status}",
bg=background_color, fg=text_color,font=("Times New Roman", 14)).pack()
        tk.Label(self, text="Ordered Items:", bg=background_color, fg=text_color, font=("Times
New Roman", 14)).pack(pady=10)
        total_price = 0

        for product in cart:
            item_price = product['price'] * product['quantity']
            total_price += item_price
            tk.Label(self, text=f"{product['name']} - Quantity: {product['quantity']} - Price:
RM {item_price: .2f}",
                  bg=background_color, fg=text_color, font=("Times New Roman", 14)).pack()

        tk.Label(self, text=f"Total Price: RM {total_price: .2f}", bg=background_color,
fg=text_color, font=("Times New Roman", 14)).pack(pady=10)
        tk.Button(self, text="Proceed to Payment", command=self.proceed_to_payment,
bg="#FF857F", fg="Black", font=("Helvetica", 14), width=20, height=2).pack(pady=10)
        tk.Button(self, text="Back to Product Selection",
command=self.back_to_product_selection, bg="#66BFBF", fg="Black", font=("Helvetica", 14),
width=20, height=2).pack(pady=10)

    def proceed_to_payment(self):
        self.destroy()
        PaymentPage(self.master, self.personal_info, self.cart)

    def back_to_product_selection(self):
        self.destroy()
        ProductSelectionPage(self.master, self.personal_info)

class AdminLoginPage(tk.Toplevel):
    def __init__(self):
        super().__init__()
        self.title("Admin Login")
        self.configure(bg="#F4E572")
        self.geometry("1920x1080")
        self.admin_id = tk.StringVar()
        self.admin_password = tk.StringVar()

        tk.Label(self, text="Admin ID:", font=("Helvetica", 14)).pack(pady=5)
        self.id_entry = tk.Entry(self, textvariable=self.admin_id)
        self.id_entry.pack(pady=5)
        tk.Label(self, text="Password:", font=("Helvetica", 14)).pack(pady=5)
        self.password_entry = tk.Entry(self, textvariable=self.admin_password, show="*")
        self.password_entry.pack(pady=5)
        tk.Button(self, text="Login", command=self.login, font=("Helvetica",
14)).pack(pady=10)

    def login(self):
        valid_id = "SparkTech"
        valid_password = "00000"

        entered_id = self.admin_id.get()
        entered_password = self.admin_password.get()
        if entered_id == valid_id and entered_password == valid_password:
            self.destroy()
            AdminPage()
        else:
            messagebox.showerror("Login Failed", "Invalid Admin ID or Password")

class AdminPage(tk.Toplevel):
    def __init__(self):
        super().__init__()
        self.title("Admin Dashboard")
        self.configure(bg="#F4E572")
        self.geometry("1920x1080")
        self.customer_data = pd.read_csv(r'C:\Users\nurra\OneDrive - ump.edu.my\UMP\SEM
3\PYTHON\LAB\PYCHAM CODE\Lib\customer_data.csv')

        tk.Label(self, text="Select Chart Type:", font=("Helvetica", 14), width=30,
height=2).pack(pady=10)
        tk.Button(self, text="Show Daily Sales Chart", command=self.daily_sales_chart,
font=("Helvetica", 14), width=30, height=2).pack(pady=10)
        tk.Button(self, text="Show Customer Gender Chart", command=self.customer_gender_chart,
font=("Helvetica", 14), width=30, height=2).pack(pady=10)
        tk.Button(self, text="Show Product Quantity Chart",
command=self.product_quantity_chart,  font=("Helvetica", 14), width=30,
height=2).pack(pady=10)
        tk.Button(self, text="Export Order Data to Excel",
command=self.export_order_data_to_excel, font=("Helvetica", 14), width=30,
height=2).pack(pady=10)
        tk.Button(self, text="Home", command=self.show_home, font=("Helvetica", 14), width=30,
height=2).pack(pady=15)

    def update_customer_data(self, customer_data):
        self.customer_data = customer_data

    def product_quantity_chart(self):
        product_quantity_sold = self.customer_data.groupby('Product')['Quantity'].sum()
        # Plotting a horizontal bar chart for product quantities
        plt.figure(figsize=(10, 6))
        product_quantity_sold.sort_values().plot(kind='barh', color='skyblue')
        plt.xlabel('Quantity Sold')
        plt.ylabel('Product')
        plt.title('Quantity Sold for Each Product')
        plt.tight_layout()
        plt.show()

    def export_order_data_to_excel(self):
        is_admin = True  # Replace with your admin check logic
        if not is_admin:
            messagebox.showerror("Access Denied", "Only admin can export order data.")
            return
        else:
             # Assuming self.customer_data contains the order data
            if not self.customer_data.empty:
                timestamp = datetime.now().strftime("%d%m%Y")
                excel_filename = f"order_data_{timestamp}.xlsx"
                self.customer_data.to_excel(excel_filename, index=False)
                messagebox.showinfo("Success", f"Order data exported to {excel_filename}")
            else:
                 messagebox.showinfo("Info", "No order data available.")

    def daily_sales_chart(self):
        # Code to generate a bar chart for daily sales
        daily_sales_data = self.customer_data.groupby('Date')['TotalPrice'].sum()

        plt.bar(daily_sales_data.index, daily_sales_data.values)
        plt.xlabel('Date')
        plt.ylabel('Sales')
        plt.title('Daily Sales Chart')
        plt.show()

    def customer_gender_chart(self):
        gender_distribution = self.customer_data['Gender'].value_counts()
        labels = gender_distribution.index
        sizes = gender_distribution.values

        plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
        plt.title('Customer Gender Distribution')
        plt.show()

    def show_home(self):
        self.withdraw()
        FrontPage()

def main():
    SparkTech_app = SparkTechApp()
    # Your additional setup and logic here
    SparkTech_app.mainloop()


if __name__ == "__main__":
    main()
```

*Note : Download all contents in this file before run the codes as it contains the wallpaper and the picture for this GUI.
https://drive.google.com/drive/folders/1UljS0RwMTuS_wUiNxpswcpJdlMgEaRI?usp=drive_link

5.0 Gui Screenshot

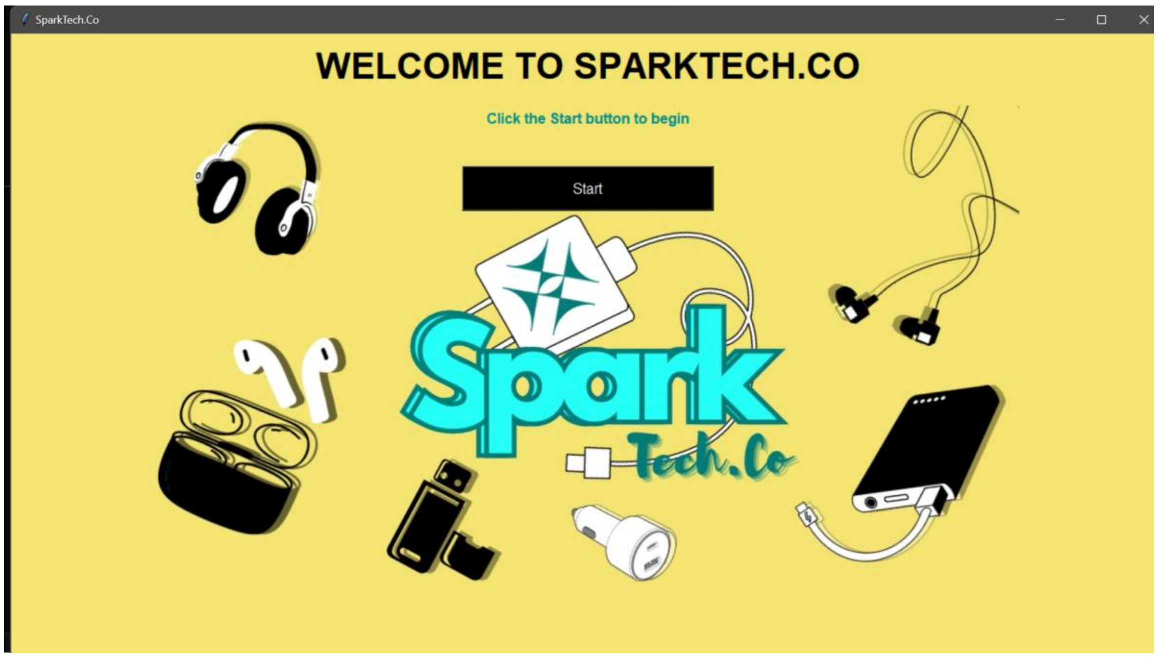
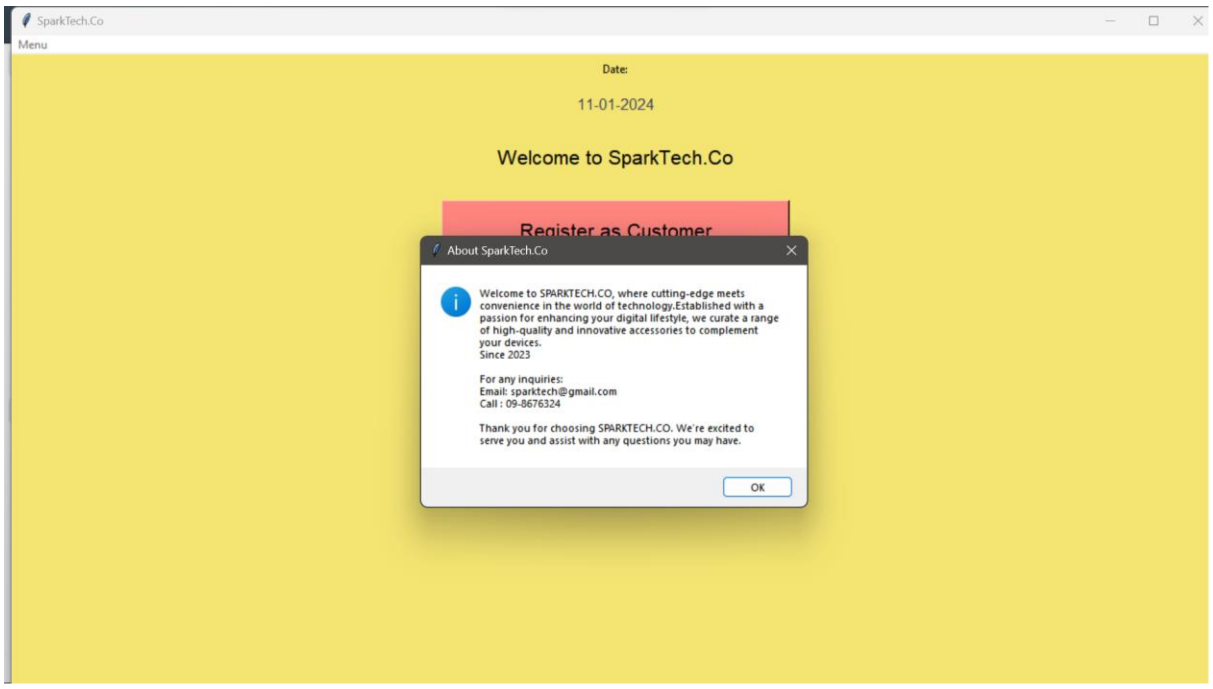


Figure 1: Front page of GUI Spark Tech Co.



Welcome to SPARKTECH.CO, where cutting-edge meets convenience in the world of technology.Established with a passion for enhancing your digital lifestyle, we curate a range of high-quality and innovative accessories to complement your devices.
Since 2023

For any inquiries:
Email: sparktech@gmail.com
Call : 09-8676324

Thank you for choosing SPARKTECH.CO. We're excited to serve you and assist with any questions you may have.

Figure 2: Pull Down Menu And This Is About Our Company

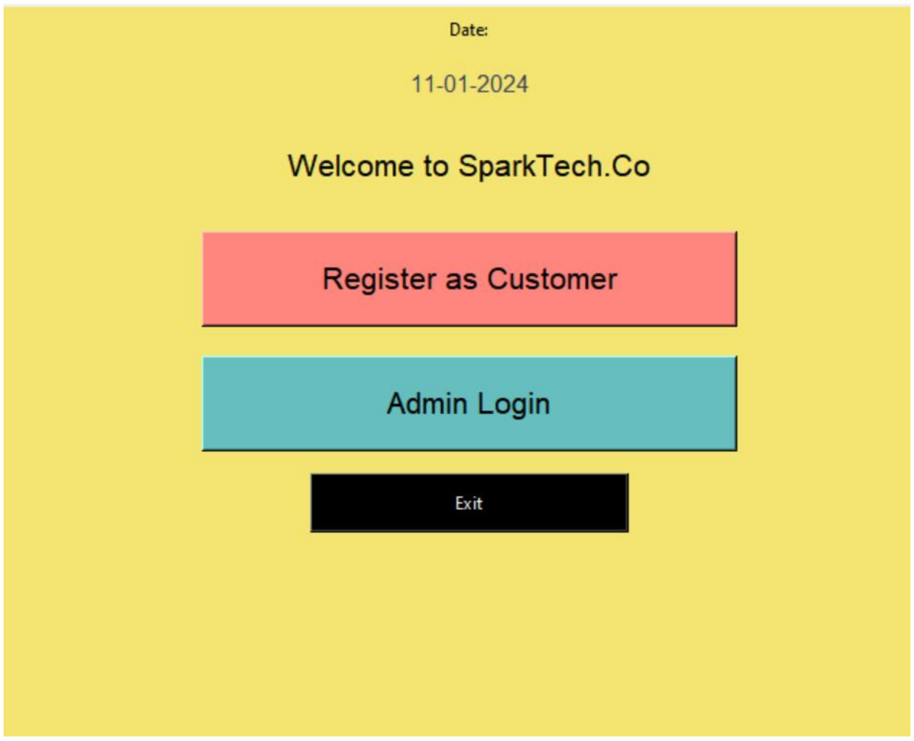


Figure 3: Option page as a customer or administrator

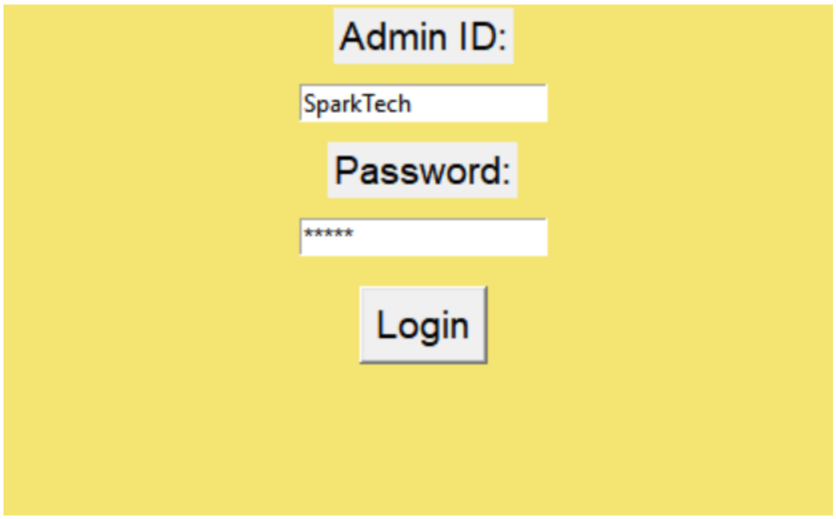


Figure 4: Procedure to login as administrator of SparkTech.Co

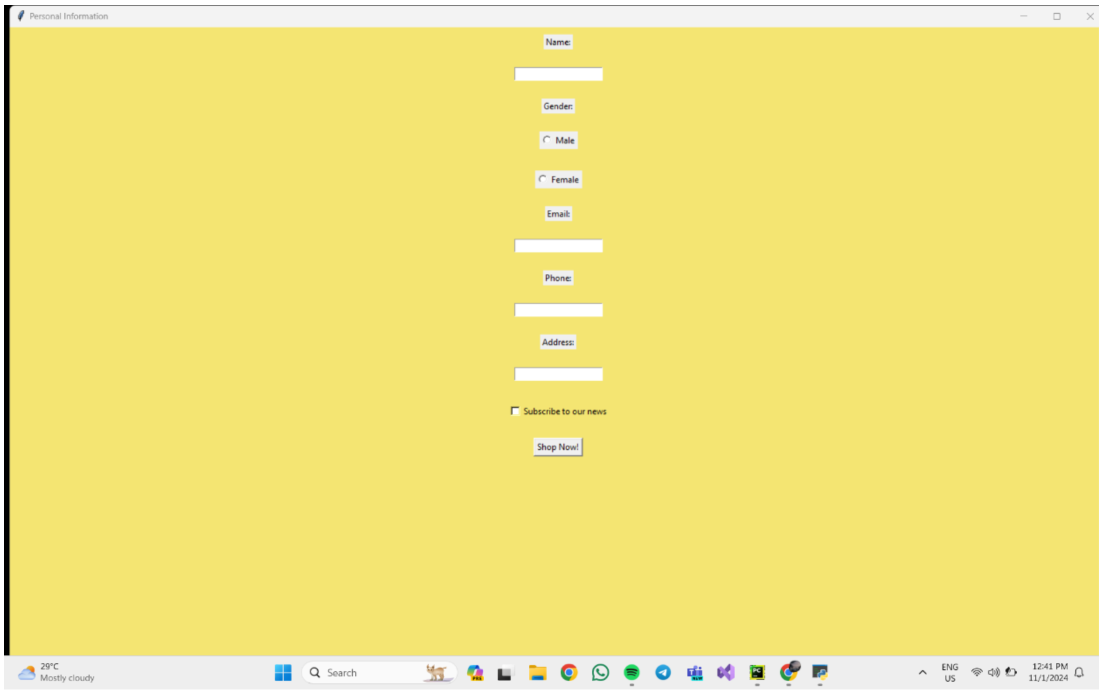


Figure 5: Registration form as a customer of SparkTech.Co

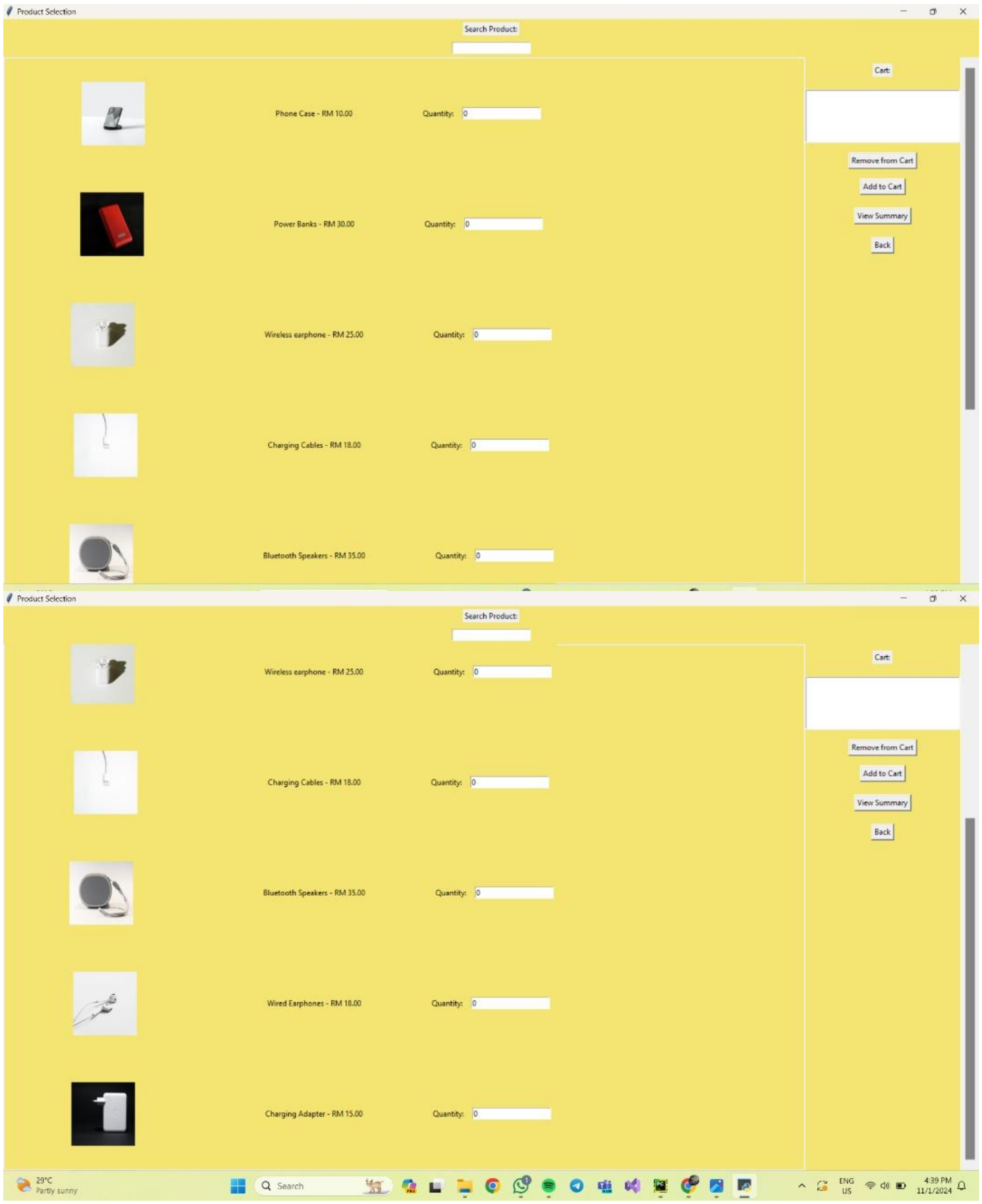
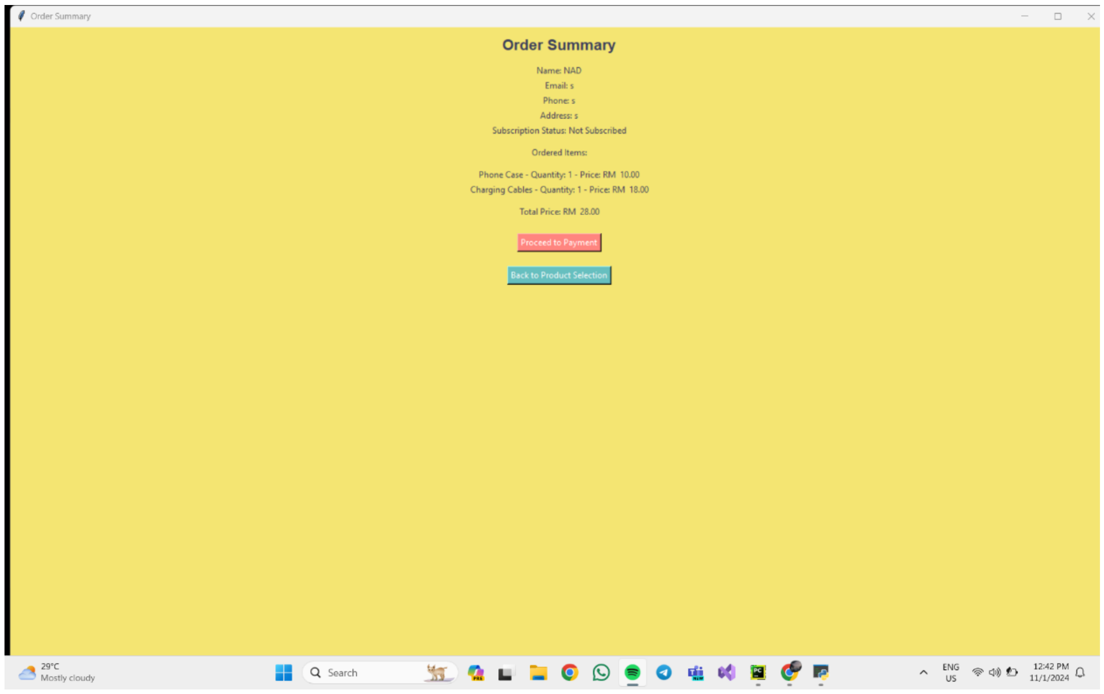


Figure 6: Product Selection Page


Order Summary

Name: NAD

Email: s

Phone: s

Address: s

Subscription Status: Not Subscribed

Ordered Items:

Phone Case - Quantity: 1 - Price: RM 10.00

Charging Cables - Quantity: 1 - Price: RM 18.00

Total Price: RM 28.00

Figure 7: Order Summary Page

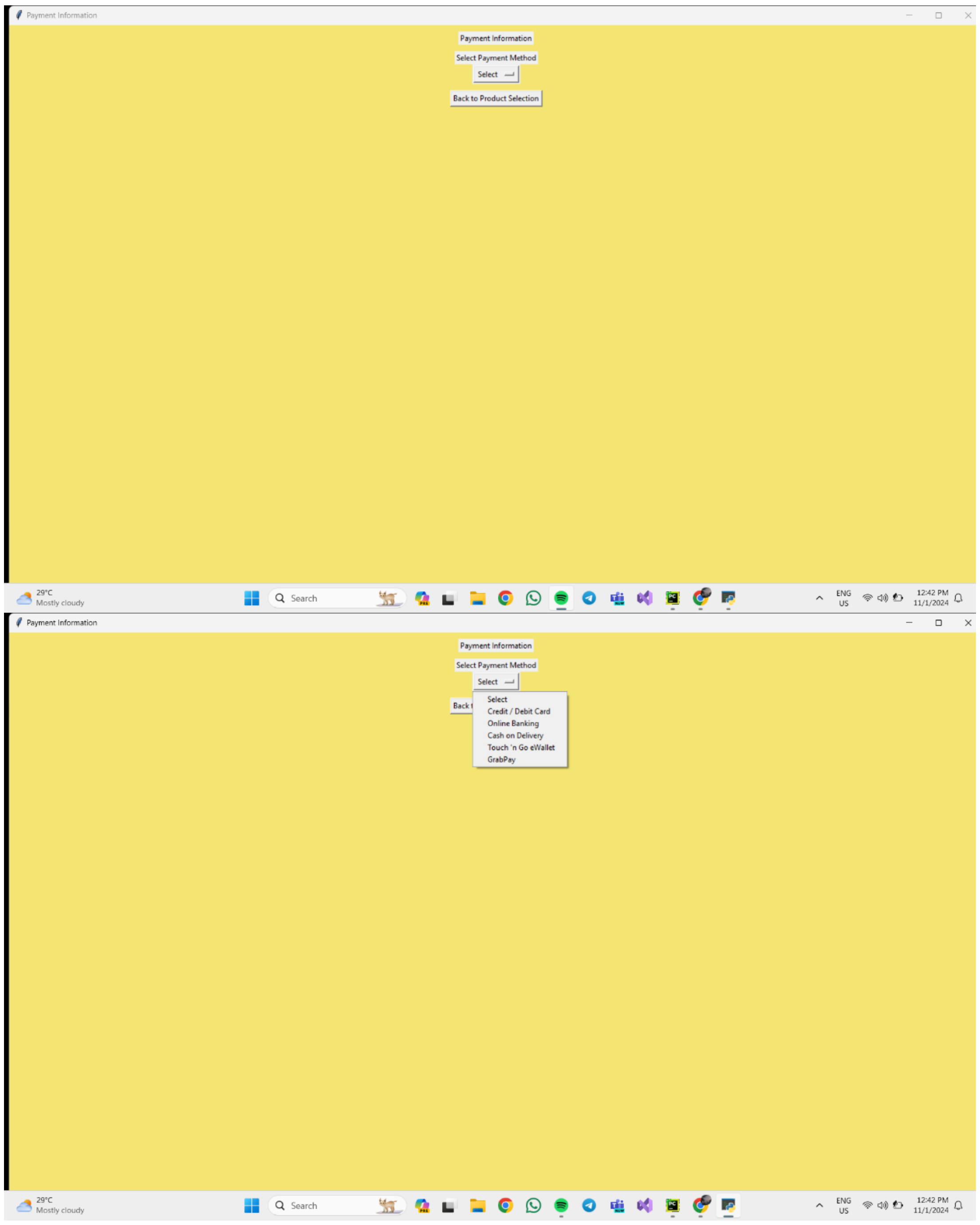
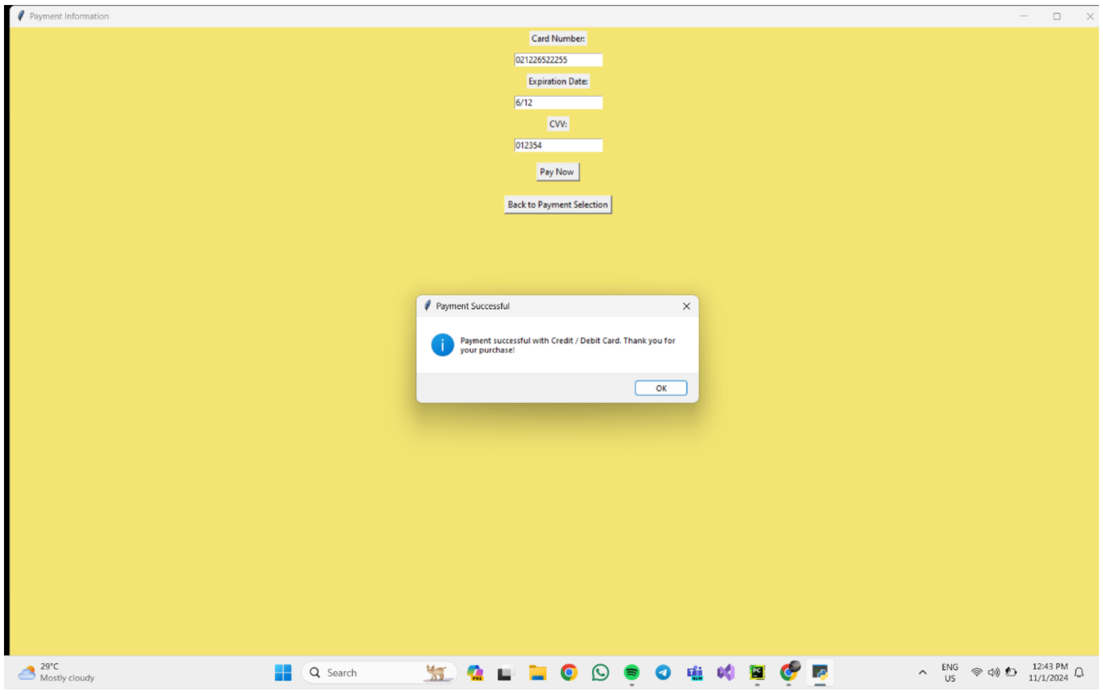


Figure 8: Payment Information Page

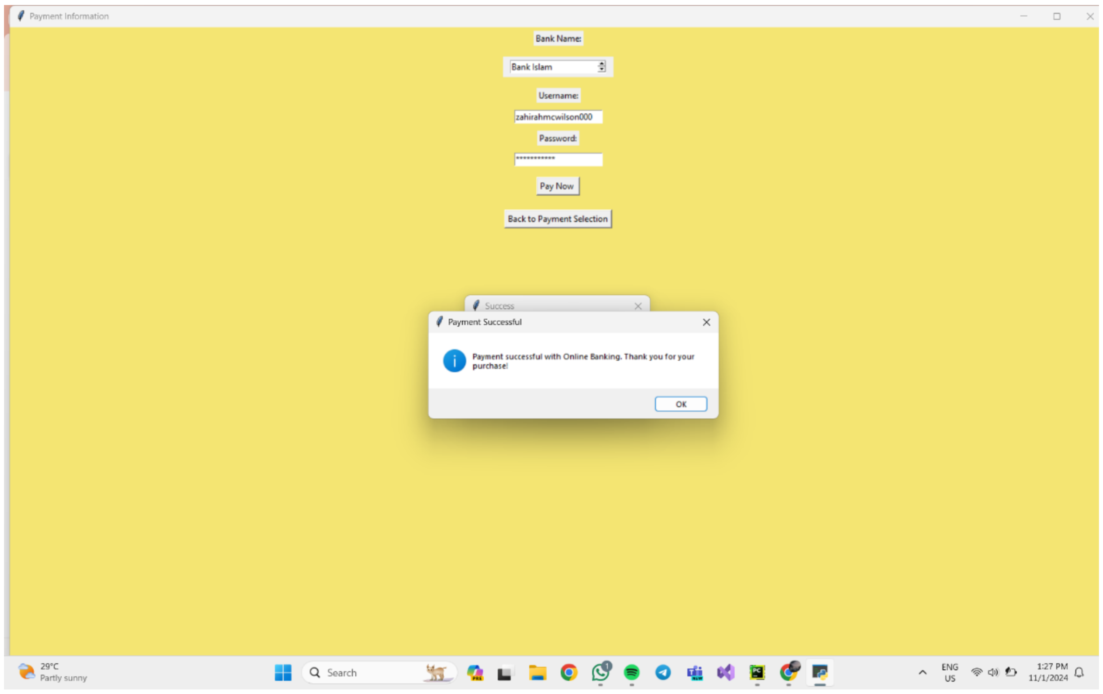

Payment Information

Card Number:
021226522255
Expiration Date:
6/12
CVV:
012354

Payment Successful

Payment successful with Credit / Debit Card. Thank you for your purchase!

Figure 9: Process Payment using Credit / Debit Card

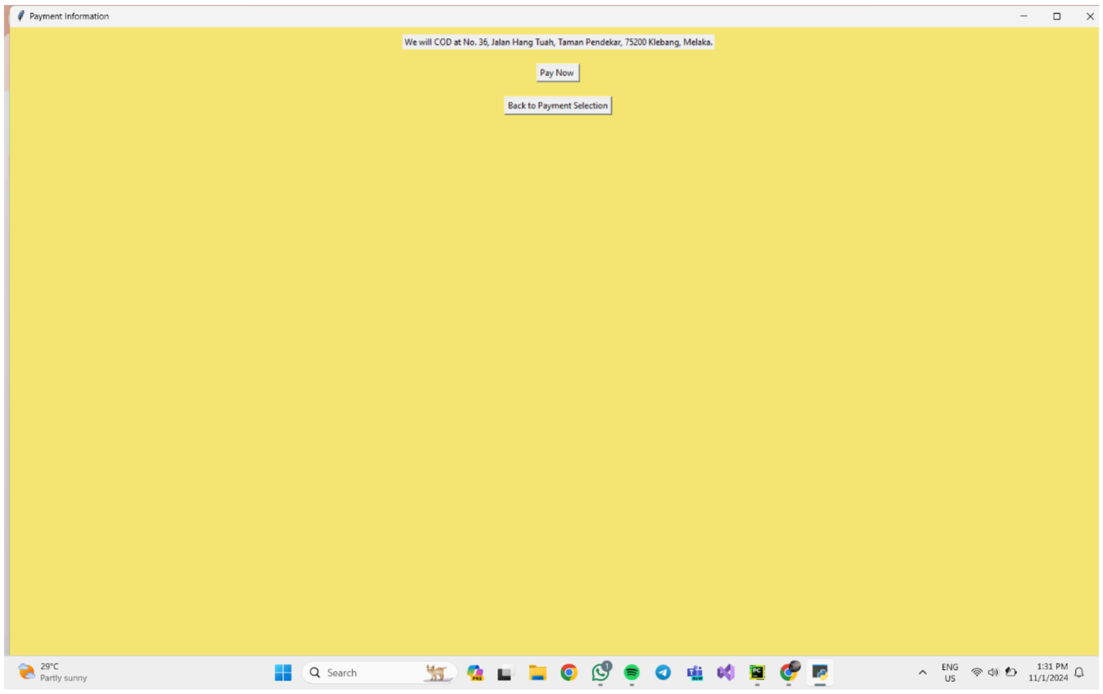

Payment Information

Bank Name:
Bank Islam
Username:
zahirahmcwilson000
Password:
****

Payment Successful

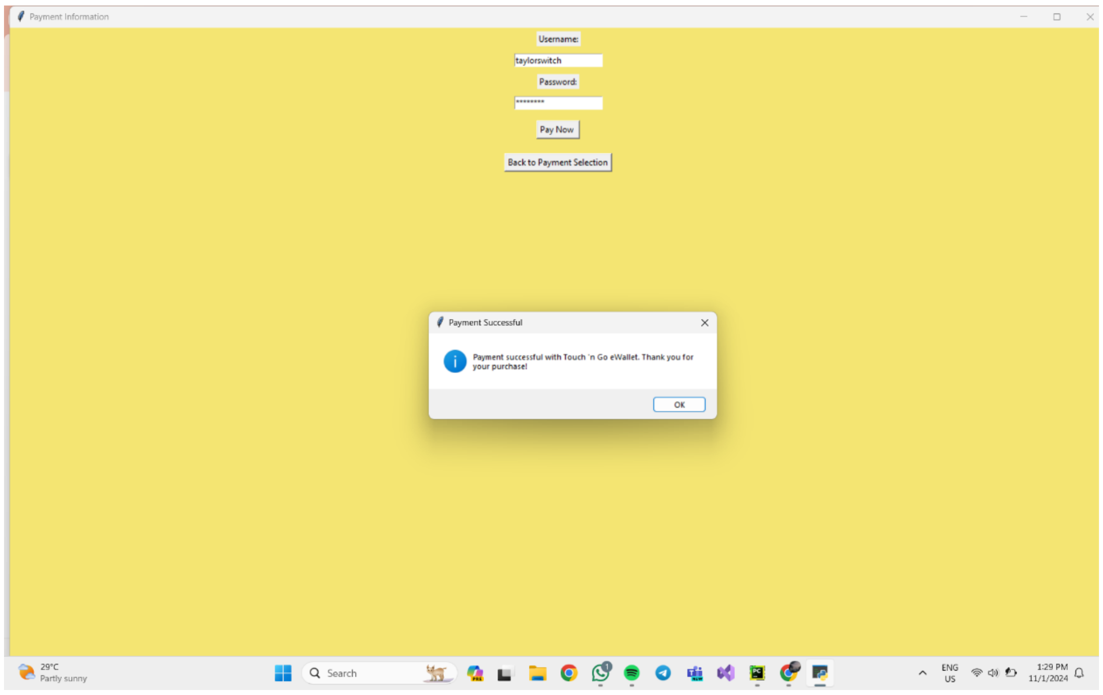
Payment successful with Online Banking. Thank you for your purchase!

Figure 10: Process Payment using online Banking



We will COD at No. 36, Jalan Hang Tuah, Taman Pendekar, 75200 Klebang, Melaka.

Figure 11: Process Payment using COD



Username:

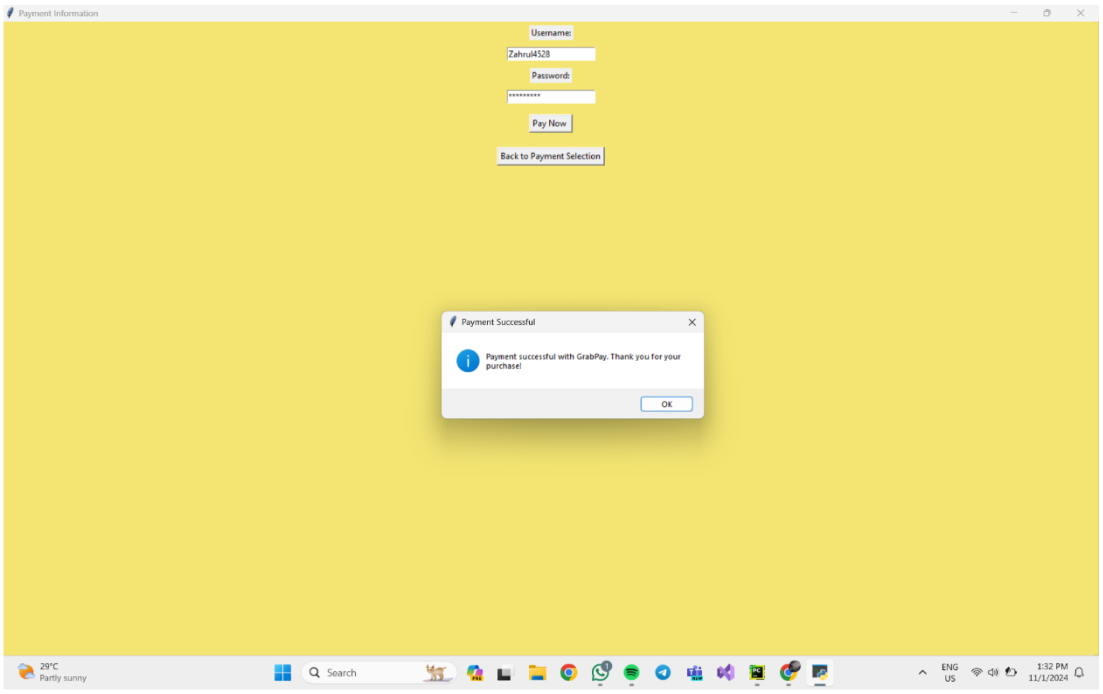
taylorswitch

Password:

****

Payment successful with Touch 'n Go eWallet. Thank you for your purchase!

Figure 12: Process Payment using Touch ‘n Go eWallet



Payment successful with GrabPay. Thank you for your purchase!

Figure 13: Process Payment using GrabPay

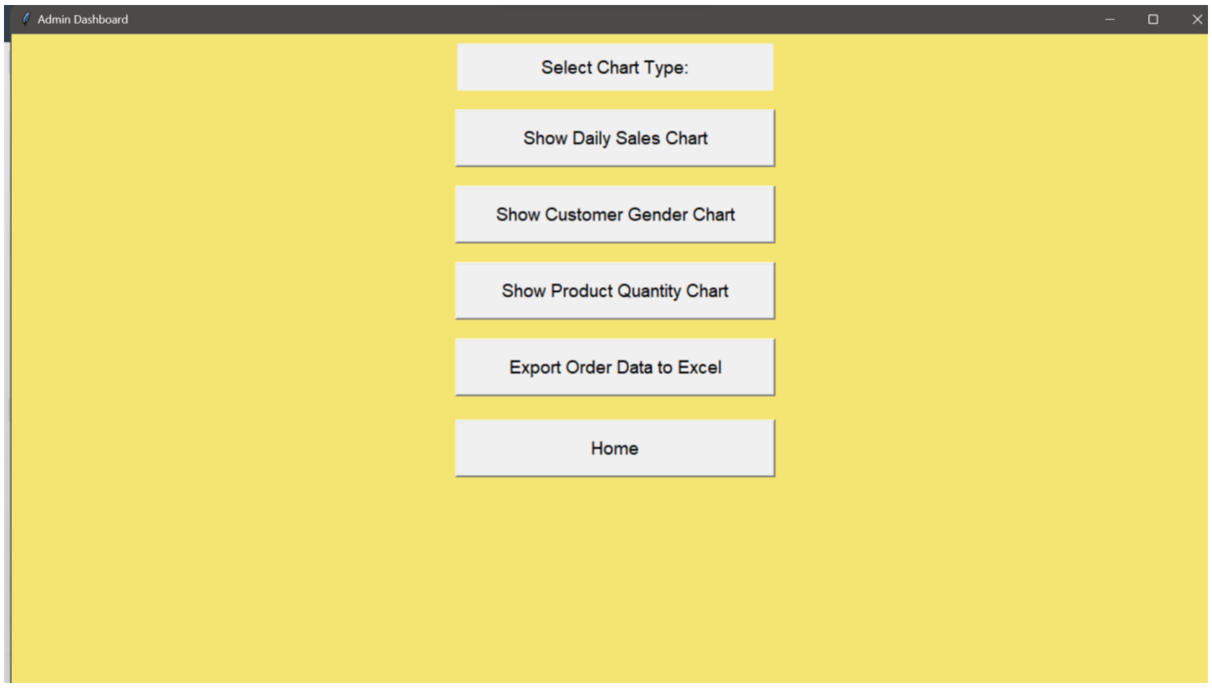

Admin Dashboard
Select Chart Type:
Show Daily Sales Chart
Show Customer Gender Chart
Show Product Quantity Chart
Export Order Data to Excel
Home

Figure 14: Admin dashboard to analysis.

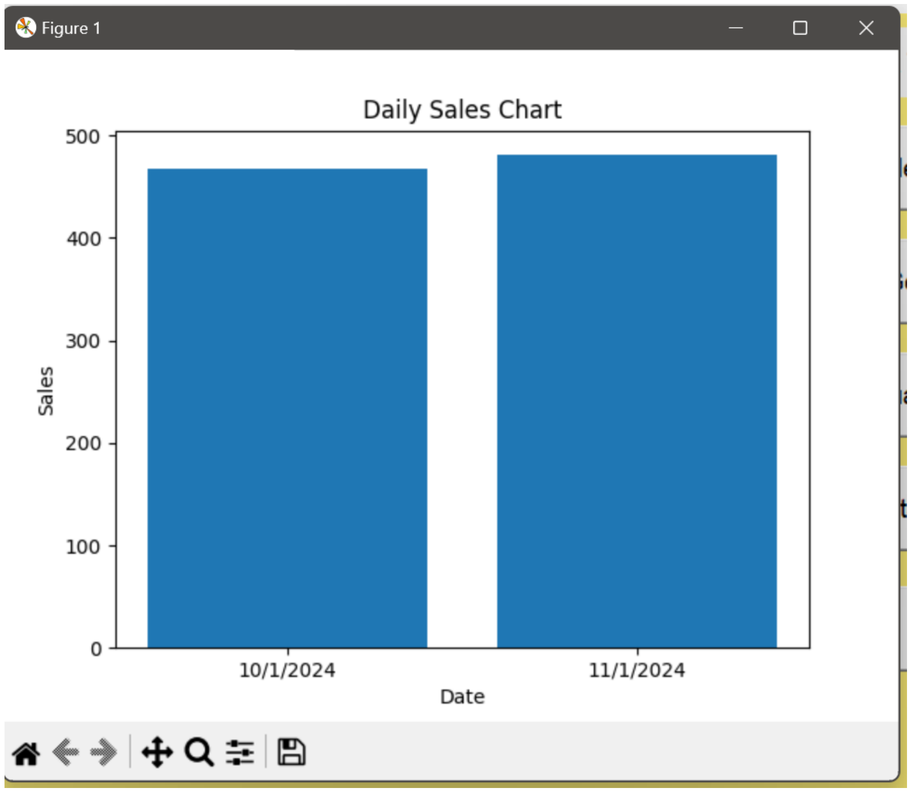


Figure 15: Daily sales Chart using bar chart

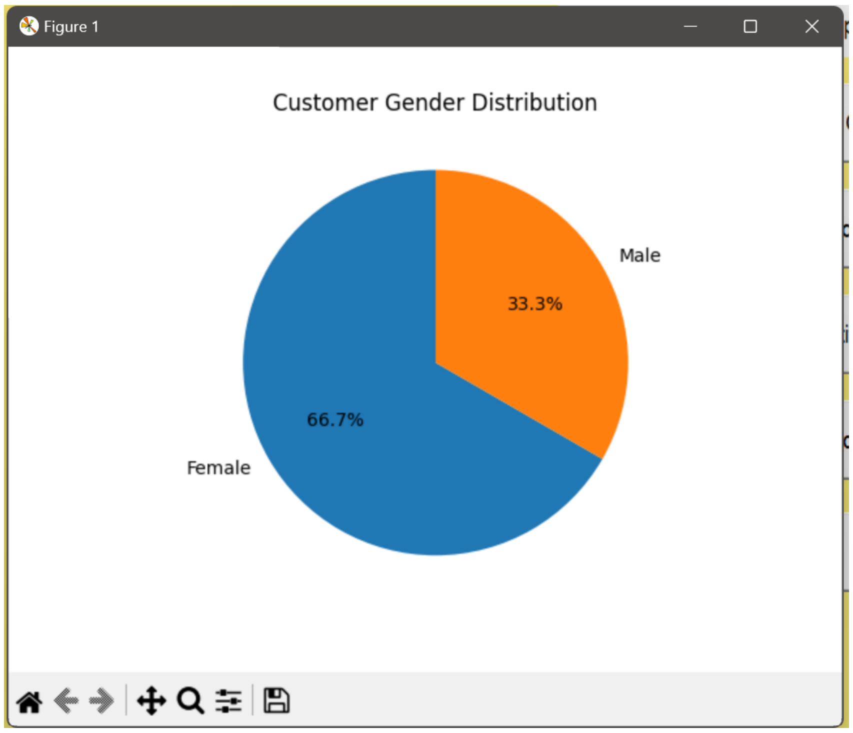


Figure 16: Customer Gender Distribution using pie chart

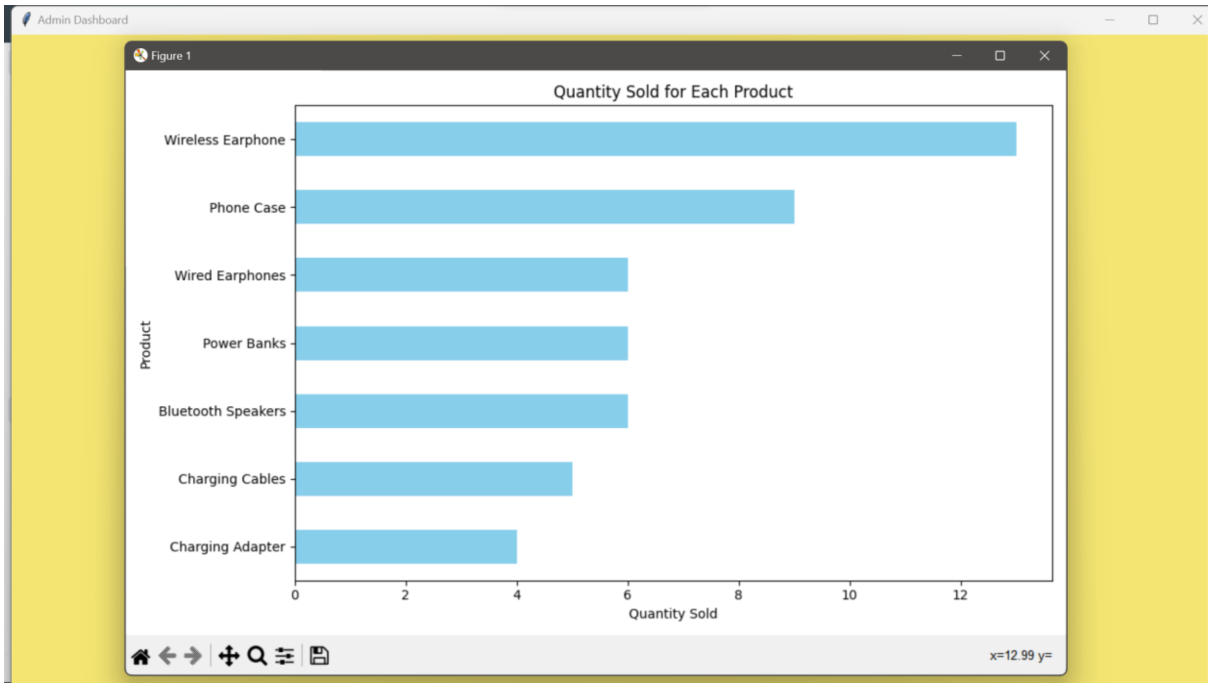
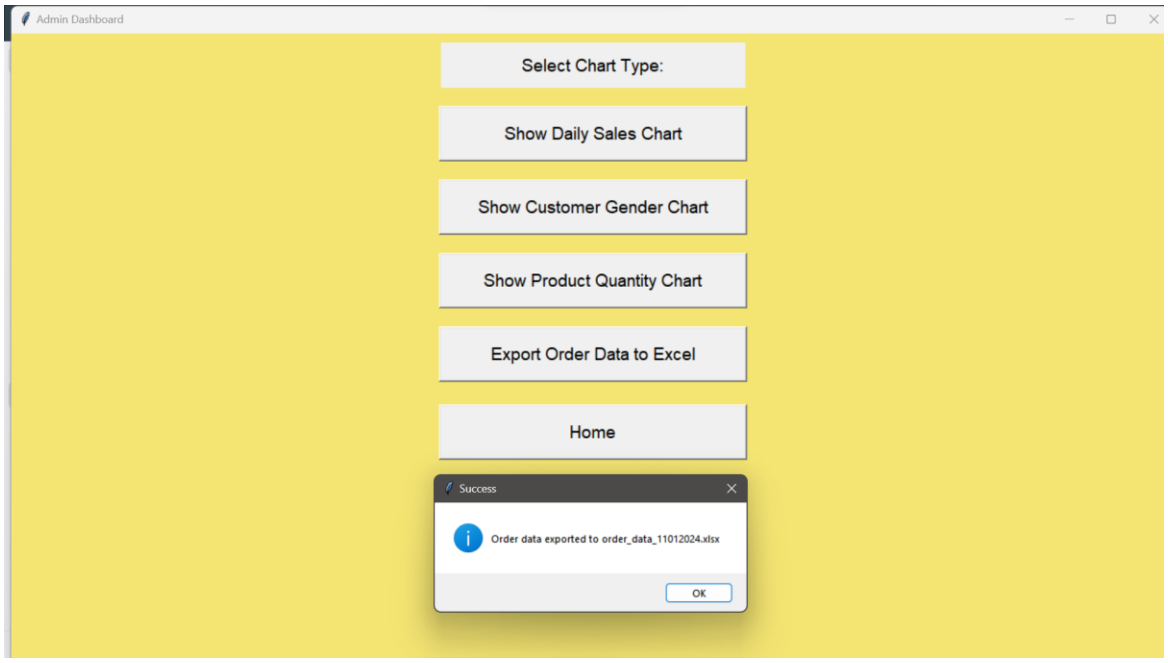


Figure 17: Quantity sold for each Product using horizontal bar Chart


Admin Dashboard
Select Chart Type:
Show Daily Sales Chart
Show Customer Gender Chart
Show Product Quantity Chart
Export Order Data to Excel
Home
Success
OK

Order data exported to order_data_11012024.xlsx

Figure 18: The export Data to excel

6.0 Marking Scheme

CLO	Description	PLO mapping	Percentage	Marks
CLO2	Use appropriate Python programming technique to solve problem.	PLO2: Cognitive Skills and Functional work skills with focus on Numeracy skills C3: Application	5%	10

LEVEL OF ACHIEVEMENT					
0 **None**	**1** **Inadequate**	**2** **Emerging**	**3** **Developing**	**4** **Good**	**5** **Excellent**

ELEMENTS	WEIGHTAGE	SCORE
Combination of appropriate controls and layout manager: • Input controls such as buttons, toggles, checkboxes etc. • Navigation controls such as pull-down menu. • Information components eg. message boxes etc. • Tkinter geometry manager (place/pack/grid manager).	**1**	
Task execution by each controls: • Each control is labelled using short and precise words representing the task. • The task for each controls is specified and written neatly. • The task for each control executed correctly and smoothly.	**1**	
TOTAL		

CLO	Description	PLO mapping	Percentage	Marks
CLO3	Construct and run program.	PLO3: Functional work skills with focus on Practical, and Digital skills P4: Mechanism	15%	30

CRITERIA	LEVEL OF ACHIEVEMENT						WEIGHTAGE	SCORE
	0 None	1 Inadequate	2 Emerging	3 Developing	4 Good	5 Excellent		
Theory/ Knowledge	No theoretical knowledge is observed.	Very little knowledge provided or information is incorrect.	Some knowledge or information is provided but missing all major points.	Some knowledge or information is provided but still missing some major points.	Good knowledge is observed, missing some minor points.	Excellent knowledge is observed; provides all necessary background principles.	1	
Assembly	Fail to demonstrate the given task.	Partly demonstrate the given task with errors.	Partly demonstrate the given task with wrong output.	Partly demonstrate the given task correctly.	Fully demonstrate the given task with some wrong output.	Demonstrate the given task correctly and perfectly.	2	
Technique used / Effectiveness	Fail to demonstrate the given task.	Demonstrate inappropriate techniques.	Partly correct techniques demonstrated.	Demonstrated technique is correct but not effective or efficient.	Demonstrated technique is partly effective and efficient.	Demonstrated technique is effective and efficient.	2	
GUI	Not submitting GUI.	The GUI presented was taken from the other sources with no modifications. The GUI presented was not effective in debugging the output with a lot of errors and displayed for an inappropriate time.	The GUI presented was modified from the other sources with minimal modifications. Shows less effective debugging on the output with with several errors and displayed for less appropriate time.	The GUI presented was modified well from the other sources. Shows effective debugging on the output with no error and displayed for an appropriate time.	The GUI presented was modified very well from the other sources. Shows effective debugging on the output with no error and displayed for an appropriate time.	The GUI presented was originally developed. Shows effective debugging on the output with no error and displayed for an appropriate time.	1	

CLO	Description	PLO mapping	Percentage	Marks
CLO4	Work collaboratively to solve assigned task.	PLO4: Functional work skills with focus on Interpersonal skills A3: Valuing	5%	10

CRITERIA	LEVEL OF ACHIEVEMENT						WEIGHTAGE	SCORE
	0 None	1 Inadequate	2 Emerging	3 Developing	4 Good	5 Excellent		
Foster Good Relationship	Show no good relationships and unable to work together effectively with other group members towards goal achievement.	No clear evidence of ability to foster good relationships and work together effectively with other group members towards goal achievement.	Able to foster relationship and work together with other group members towards goal achievement but with limited effect and require improvements.	Able to foster relationship and work together with other group members towards goal achievement with some effect(s) and require minor improvements.	Able to foster good relationship and work together with other group members towards goal achievement.	High ability to foster good relationship and work together effectively with other group members towards goal achievement.	1	
Alternate Roles	Show no ability to assume alternate roles as a group leader and group members.	No clear evidence of ability to assume alternate roles as a group leader and group members demonstrated in practice.	Attempt to demonstrate in practice the ability to alternate roles as a group leader and group members but with limited effect and require improvements.	Able to demonstrate in practice the ability to assume alternate roles as a group leader and group members with some effect(s) and require minor improvements.	Able to demonstrate in practice the ability to assume alternate roles as a group leader and a group member to achieve the same goal.	Show clear evidence to assume alternate roles as a group leader and a group member demonstrated in practice.	1	

CLO	Description	PLO mapping	Percentage	Marks
CLO5	Demonstrate innovative ideas in developing a graphical user interface.	PLO8: Entrepreneurial skills A3: Valuing	5%	10

CRITERIA	LEVEL OF ACHIEVEMENT						WEIGHTAGE	SCORE
	0 None	1 Inadequate	2 Emerging	3 Developing	4 Good	5 Excellent		
Analyzing an existing situation and identifying areas for improvement	Not providing any analysis of situation and areas for improvement were not identified.	The analysis of the situation was very limited and areas for improvement were not. identified	The analysis of the situation was limited and areas for improvement were not identified.	The analysis of the situation was appropriate but the identification of areas for improvement was limited.	The situation was appropriately analyzed and the identification of areas for improvement was completed.	The analysis of the situation and the identification of areas for improvement was completed and increases over time.	1	
Creativity/ Innovative ideas	Not presenting any GUI.	GUI presented contains lack of significance ideas, no innovative values, lack of creativity and not user friendly.	GUI presented contains lack of significance ideas, no innovative values, creative enough (catchy apps name & attractive) and user friendly.	GUI presented contains lack of significance ideas, but still have innovative values, creative enough (catchy apps name & attractive) and user friendly.	GUI presented contains significance ideas, innovative values, creative enough (catchy apps name & attractive) and user friendly.	GUI presented contains a very significance ideas, high innovative values, creative enough (catchy apps name & attractive) and user friendly.	1	