### Documentation: Customer Order Tracking System

#### Overview:

The **Customer Order Tracking System** is a Python-based information system designed to manage and visualize customer orders. It allows users to input data about customers and their orders, process this data to produce meaningful summaries, and visualize the results using Python libraries such as matplotlib.

#### Objectives:

* **Input**: Collect customer order data, including customer names, order IDs, product details, and order statuses.
* **Processing**: Calculate and display:
  1. The total number of orders per customer.
  2. The count of each order status (e.g., Processing, Shipped, Delivered).
* **Visualization**: Generate two types of visualizations:
  1. A bar chart showing the distribution of order statuses.
  2. A timeline of the order progress, showing the cumulative number of orders delivered over time.

### System Design:

#### 1. ****Input Collection (****accept\_order\_data ****Function):****

This function collects order details from the user interactively. The following information is gathered:

* **Customer Names**: The number of customers and their names are input by the user.
* **Orders per Customer**: The system asks for the number of orders each customer has made. For each order, it collects:
  + Order ID (a unique number).
  + Product name (a string describing the item).
  + Order status (either "Processing", "Shipped", or "Delivered").

The collected data is stored in a dictionary where each key represents a customer, and the value is a list of order details for that customer.

#### 2. ****Processing Functions:****

total\_orders\_per\_customer: This function calculates the total number of orders each customer has placed. It takes the orders dictionary and returns a new dictionary where each customer is mapped to the number of orders they made.

order\_status\_count: This function calculates how many orders fall into each status category (Processing, Shipped, Delivered). It returns a dictionary with the status as the key and the count of that status as the value.

#### 3. ****Display Functions:****

* display\_total\_orders: This function prints a summary of the total number of orders per customer.
* display\_status\_count: This function prints a summary of how many orders are in each status category.

#### 4. ****Visualization:****

Two key visualizations are generated to make the data more meaningful:

* plot\_order\_status: A bar chart that shows the distribution of orders in each status category (Processing, Shipped, Delivered).
* plot\_order\_progress: A line chart that simulates the progress of delivered orders over time, showing cumulative deliveries on the y-axis and time (in days) on the x-axis.

#### 5. ****Main Program Flow (****main ****Function):****

* **Step 1**: Calls accept\_order\_data to collect customer and order information from the user.
* **Step 2**: Calculates the total number of orders for each customer using total\_orders\_per\_customer.
* **Step 3**: Calculates the count of orders in each status category using order\_status\_count.
* **Step 4**: Displays the results using display\_total\_orders and display\_status\_count.
* **Step 5**: Generates a bar chart of order statuses using plot\_order\_status.
* **Step 6**: Generates a timeline of order progress using plot\_order\_progress.

### Assumptions:

1. All order statuses are valid and belong to one of the three categories: Processing, Shipped, Delivered.
2. The timeline visualization assumes a simplified progression of order delivery over a fixed period.

### Conclusion:

This system offers a complete solution for managing and visualizing customer orders. It accepts input dynamically, processes the data efficiently, and provides meaningful visualizations to track order statuses and progress over time.

**The code**

import matplotlib.pyplot as plt

import numpy as np

# Function to accept user input for orders

def accept\_order\_data():

orders = {}

num\_customers = int(input("Enter the number of customers: "))

for \_ in range(num\_customers):

customer\_name = input("Enter the customer's name: ")

orders[customer\_name] = []

num\_orders = int(input(f"Enter the number of orders for {customer\_name}: "))

for \_ in range(num\_orders):

order\_id = int(input("Enter the order ID: "))

product = input("Enter the product name: ")

status = input("Enter the status (Processing, Shipped, Delivered): ")

order = {'order\_id': order\_id, 'product': product, 'status': status}

orders[customer\_name].append(order)

return orders

# Function to calculate total orders per customer

def total\_orders\_per\_customer(orders):

total\_orders = {customer: len(order\_list) for customer, order\_list in orders.items()}

return total\_orders

# Function to calculate order status count

def order\_status\_count(orders):

status\_count = {'Processing': 0, 'Shipped': 0, 'Delivered': 0}

for order\_list in orders.values():

for order in order\_list:

status\_count[order['status']] += 1

return status\_count

# Function to display total orders per customer

def display\_total\_orders(total\_orders):

print("\n--- Total Orders per Customer ---")

for customer, total in total\_orders.items():

print(f"{customer}: {total} orders")

# Function to display order status count

def display\_status\_count(status\_count):

print("\n--- Order Status Count ---")

for status, count in status\_count.items():

print(f"{status}: {count} orders")

# Visualization 1: Bar chart of order status

def plot\_order\_status(status\_count):

statuses = list(status\_count.keys())

counts = list(status\_count.values())

plt.bar(statuses, counts, color=['orange', 'blue', 'green'])

plt.xlabel('Order Status')

plt.ylabel('Number of Orders')

plt.title('Order Status Distribution')

plt.show()

# Visualization 2: Timeline of order progress (line chart)

def plot\_order\_progress():

# Sample Timeline Data (order progress over time)

time\_period = np.arange(1, 7) # Example time points (could be days or weeks)

orders\_progress = [1, 2, 3, 3, 4, 6] # Orders delivered over time

plt.plot(time\_period, orders\_progress, marker='o', color='purple')

plt.xlabel('Time (Days)')

plt.ylabel('Total Orders Delivered')

plt.title('Order Progress Timeline')

plt.grid(True)

plt.show()

# Main function to run the customer order tracking system

def main():

print("Customer Order Tracking System")

# Step 1: Accept user input for orders

orders = accept\_order\_data()

# Step 2: Calculate total orders per customer

total\_orders = total\_orders\_per\_customer(orders)

# Step 3: Calculate order status count

status\_count = order\_status\_count(orders)

# Step 4: Display total orders and status count

display\_total\_orders(total\_orders)

display\_status\_count(status\_count)

# Step 5: Visualize order status as a bar chart

plot\_order\_status(status\_count)

# Step 6: Visualize order progress as a timeline

plot\_order\_progress()

# Run the system

if \_\_name\_\_ == "\_\_main\_\_":

main()