**Project Request: E-commerce Shopping Cart System Development**

**Date:** July 10, 2025

**Prepared For:** [Relevant Department/Team, e.g., IT Development, Product Management]

**Prepared By:** [Your Name/Department]

**1. Executive Summary:**

This document outlines the requirement for developing a foundational e-commerce shopping cart system. The primary objective is to enable customers to select and quantify desired items, store these selections, and subsequently generate an accurate bill based on predefined product pricing. This system is crucial for streamlining the customer purchasing process and enhancing operational efficiency within our retail environment.

**2. Business Need:**

To support our retail operations and provide a seamless customer experience, we require a digital solution that automates the item selection and billing process. Manual or disparate systems lead to inefficiencies, potential errors, and a suboptimal customer journey. A robust shopping cart system will directly address these challenges.

**3. Functional Requirements:**

The proposed system must incorporate the following core functionalities:

* **Item Selection and Quantity Specification:**
  + Customers shall be able to input item names and specify the desired purchase quantity for each item.
  + The system must accommodate the addition of multiple unique items to a single shopping cart instance.
* **Dynamic Cart Management:**
  + All selected items and their corresponding quantities must be dynamically stored within a structured data format (e.g., a key-value pair collection where the item name is the key and the quantity is the value, such as { "Rice": 5, "Wheat": 10, "Sugar": 5 }).
  + The system should allow for continuous updates to the cart as customers add more items.
* **Automated Billing Generation:**
  + Upon customer request, the system shall compare the items and quantities in the shopping cart against a predefined store inventory and pricing database.
  + A comprehensive bill must be generated, detailing each purchased item, its quantity, unit price, and the calculated subtotal.
  + A grand total reflecting the sum of all purchased items must be prominently displayed.

**4. Data Specifications (Illustrative):**

The system will interact with at least two primary data structures:

* **Customer Cart Data:**
  + Format: Dictionary
  + Example: {"Rice": 5, "Wheat": 10, "Sugar": 5}
* **Store Inventory & Pricing Data:**
  + Format: Dictionary
  + Example: {"Wheat": 50, "Rice": 50, "Sugar": 30} (where values represent unit prices)

**5. Technical Considerations (Initial Recommendation):**

Given the requirement for rapid prototyping and ease of development, Python is recommended as the programming language for initial development. The use of dictionaries for data storage aligns with the functional requirements for dynamic cart and inventory management.

**Write a program to create a shopping cart for a mall. Customers should be able to add up to any number of items to their cart, specifying the item name and quantity. The items and their quantities should be stored in a Python dictionary.**

**You  need to write a program for Shopping mall where customer should be able enter items details including item name and quantity to be purchased and entered items would be part of the dictionary.**

**i.e   { Rice :5,Wheat:10, Sugar:5 }**

**Purchased items should be compared with existing dictionary having items in the shop for sale with sale price. Total amount to be paid by customer (Bill should generated)**

**grocerystore={"Wheat" :50,"Rice" : 50,"Sugar" : 30}**

**import gradio as gr**

**# Define the existing items and their sale prices in the grocery store**

**# This dictionary acts as our backend database for available products and prices.**

**GROCERY\_STORE\_PRICES = {**

**"Wheat": 50,**

**"Rice": 50,**

**"Sugar": 30,**

**"Milk": 45,**

**"Bread": 25,**

**"Eggs": 60,**

**"Apples": 120,**

**"Bananas": 40,**

**"Coffee": 150,**

**"Tea": 80,**

**"Pasta": 75,**

**"Cheese": 200**

**}**

**def add\_item\_to\_cart(item\_name\_input, quantity\_input, current\_cart):**

**"""**

**Adds a specified item and quantity to the shopping cart.**

**Updates the cart dictionary and returns a message and the updated cart display.**

**Args:**

**item\_name\_input (str): The name of the item entered by the user.**

**quantity\_input (int): The quantity of the item entered by the user.**

**current\_cart (dict): The current state of the shopping cart.**

**Returns:**

**tuple: A message string, the updated shopping cart dictionary, and a formatted string of the current cart.**

**"""**

**item\_name = item\_name\_input.strip()**

**if not item\_name:**

**return "Please enter an item name.", current\_cart, format\_cart\_for\_display(current\_cart)**

**if quantity\_input is None or quantity\_input <= 0:**

**return "Quantity must be a positive whole number.", current\_cart, format\_cart\_for\_display(current\_cart)**

**# Capitalize the first letter of each word for consistent key lookup**

**# This helps in matching user input like "rice" to "Rice" in the store**

**formatted\_item\_name = ' '.join([word.capitalize() for word in item\_name.split()])**

**# Update the cart: if item exists, add to quantity; otherwise, add new item**

**current\_cart[formatted\_item\_name] = current\_cart.get(formatted\_item\_name, 0) + quantity\_input**

**message = f"Added {quantity\_input} x {formatted\_item\_name} to your cart."**

**return message, current\_cart, format\_cart\_for\_display(current\_cart)**

**def format\_cart\_for\_display(cart):**

**"""**

**Formats the shopping cart dictionary into a human-readable string for display.**

**"""**

**if not cart:**

**return "Your cart is currently empty."**

**display\_str = "Current Cart:\n"**

**for item, quantity in cart.items():**

**display\_str += f"- {item}: {quantity}\n"**

**return display\_str**

**def generate\_bill(shopping\_cart):**

**"""**

**Generates a detailed bill for the items in the shopping cart.**

**Calculates total amount based on GROCERY\_STORE\_PRICES.**

**Args:**

**shopping\_cart (dict): The current state of the shopping cart.**

**Returns:**

**tuple: A markdown-formatted string of the bill and the total amount.**

**"""**

**if not shopping\_cart:**

**return "Your cart is empty. Please add items to generate a bill.", 0.0**

**bill\_output = "### Your Bill\n\n"**

**bill\_output += "| Item           | Quantity | Price/Unit | Subtotal |\n"**

**bill\_output += "|:---------------|:---------|:-----------|:---------|\n"**

**total\_amount = 0.0**

**items\_not\_found = []**

**for item, quantity in shopping\_cart.items():**

**if item in GROCERY\_STORE\_PRICES:**

**price\_per\_unit = GROCERY\_STORE\_PRICES[item]**

**subtotal = quantity \* price\_per\_unit**

**total\_amount += subtotal**

**bill\_output += f"| {item:<14} | {quantity:<8} | {price\_per\_unit:<10.2f} | {subtotal:<8.2f} |\n"**

**else:**

**items\_not\_found.append(f"- {item} (Quantity: {quantity})")**

**bill\_output += f"| {item:<14} | {quantity:<8} | {'N/A':<10} | {'Item not available':<8} |\n"**

**bill\_output += "|----------------|----------|------------|----------|\n"**

**bill\_output += f"| \*\*Total Amount:\*\* |          |            | \*\*{total\_amount:<8.2f}\*\* |\n"**

**if items\_not\_found:**

**bill\_output += "\n\*\*Note:\*\* The following items were in your cart but are not available in our store and were not charged:\n"**

**for item\_info in items\_not\_found:**

**bill\_output += f"{item\_info}\n"**

**return bill\_output, total\_amount**

**def clear\_cart():**

**"""**

**Clears the shopping cart and resets the display.**

**"""**

**return {}, "Cart cleared.", "Your cart is currently empty.", "", 0.0**

**# Gradio Interface Setup**

**with gr.Blocks(title="Shopping Mall Cart") as demo:**

**# State to hold the shopping cart dictionary across interactions**

**shopping\_cart\_state = gr.State({})**

**gr.Markdown(**

**"""**

**# 🛍️ Shopping Mall Cart**

**Welcome! Add items to your cart and generate your bill.**

**"""**

**)**

**with gr.Row():**

**with gr.Column(scale=1):**

**item\_name\_input = gr.Textbox(label="Item Name", placeholder="e.g., Rice, Wheat, Sugar")**

**quantity\_input = gr.Number(label="Quantity", value=1, precision=0)**

**add\_item\_btn = gr.Button("Add Item to Cart")**

**clear\_cart\_btn = gr.Button("Clear Cart")**

**with gr.Column(scale=2):**

**add\_item\_message = gr.Textbox(label="Status", interactive=False)**

**current\_cart\_display = gr.Textbox(label="Your Current Cart", interactive=False, lines=5)**

**generate\_bill\_btn = gr.Button("Generate Bill")**

**bill\_output\_markdown = gr.Markdown("### Bill will appear here.")**

**total\_amount\_output = gr.Number(label="Total Amount (Rs.)", interactive=False, precision=2)**

**# Define interactions**

**add\_item\_btn.click(**

**add\_item\_to\_cart,**

**inputs=[item\_name\_input, quantity\_input, shopping\_cart\_state],**

**outputs=[add\_item\_message, shopping\_cart\_state, current\_cart\_display]**

**)**

**generate\_bill\_btn.click(**

**generate\_bill,**

**inputs=[shopping\_cart\_state],**

**outputs=[bill\_output\_markdown, total\_amount\_output]**

**)**

**clear\_cart\_btn.click(**

**clear\_cart,**

**inputs=[],**

**outputs=[shopping\_cart\_state, add\_item\_message, current\_cart\_display, bill\_output\_markdown, total\_amount\_output]**

**)**

**# Launch the Gradio app**

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

**demo.launch()**