Pseudocode

Data Retrieval Process

1. BEGIN

   // STEP 1: Initialize system databases and user session
2. INITIALIZE ProductDatabase, UserDatabase, AdminDatabase, StoreDatabase, FlyerDatabase
3. SET userLocation = GET_USER_LOCATION()
4. SET isLoggedIn = CHECK_USER_LOGIN_STATUS()

   // STEP 2: Get user input for product search or grocery list selection
5. PROMPT userInput FOR product search OR SELECT from grocery list

   // STEP 3: Fetch product details based on user input
6. IF userInput EXISTS THEN
   a. FETCH productDetails FROM ProductDatabase WHERE product = userInput
   b. IF productDetails FOUND THEN

      // Display product details to user
      i. DISPLAY productDetails
   c. ELSE

      // Notify user that product is unavailable
      i. DISPLAY "Product not found"
      ii. RETURN
   d. END IF
7. ELSE
   a. DISPLAY "No input provided"
   b. RETURN
8. END IF

   // STEP 4: Fetch real-time price comparison from nearby stores
9. FETCH nearbyStores FROM StoreDatabase WHERE storeLocation WITHIN userLocation
10. FOR EACH store IN nearbyStores DO
    a. FETCH storeProductPrices FROM StoreDatabase WHERE product = userInput
11. END FOR
12. DISPLAY storeProductPrices FOR all nearby stores
    // STEP 5: Fetch inventory details for each store
13. FOR EACH store IN nearbyStores DO
    a. FETCH inventoryDetails FROM StoreDatabase WHERE product = userInput AND store = storeName
    b. IF inventoryDetails AVAILABLE THEN

        i. DISPLAY "Available quantity: " + inventoryDetails.quantity
    c. ELSE
        i. DISPLAY "Out of stock at " + storeName
    d. END IF
14. END FOR

// STEP 6: Fetch flyer information for selected store and product
15. FETCH flyerDetails FROM FlyerDatabase WHERE product = userInput AND store = storeName
16. IF flyerDetails EXISTS THEN
    a. DISPLAY "Flyer Price Validity: " + flyerDetails.startDate + " to " + flyerDetails.endDate
17. ELSE
    a. DISPLAY "No current flyer available for this product"
18. END IF

// STEP 7: Admin-specific actions (if logged in as admin)
19. IF isLoggedIn = "Admin" THEN
        // Admin can update product quantities for price match
    a. FETCH adminOptions FROM AdminDatabase
    b. IF adminOptions = "Manage Product Quantities" THEN
        i. PROMPT admin FOR updateQuantity
        ii. UPDATE ProductDatabase SET quantity = updateQuantity WHERE product = selectedProduct
    c. END IF
        // Admin can manage stores for price matching
    d. IF adminOptions = "Manage Stores for Price Matching" THEN
        i. PROMPT admin FOR updateStore
        ii. UPDATE StoreDatabase SET storeName = updateStore WHERE product = selectedProduct
    e. END IF
20. END IF

// STEP 8: User-specific data retrieval
21. IF isLoggedIn = "User" THEN
    a. FETCH userDetails FROM UserDatabase WHERE userID = loggedInUserID
    b. DISPLAY "Grocery List: " + userDetails.groceryList
    c. DISPLAY "Notifications: " + userDetails.notifications
22. END IF

// STEP 9: Finalize and display retrieved data
23. DISPLAY "Final Price Comparison, Inventory Availability, and Flyer Details"
24. END

Cart Management

1. **Item management**
    a. BEGIN
    b. PROMPT user to search for the item by name
    c. IF item exists in inventory THEN
        i. IF item exists in cart THEN
            1. PROMPT user to either update the quantity or cancel
            2. IF user chooses to update THEN
                a. UPDATE item quantity in the cart
            3. ELSE
                a. CANCEL the update process
            4. END IF
        ii. ELSE
            1. PROMPT user to add quantity
            2. ADD item to cart with the specified quantity
        iii. END IF
    d. ELSE
        i. PRINT "Item not available in inventory"
    e. END IF
    f. RECALCULATE the total amount in the cart
    g. END
2. **Checkout**
    a. BEGIN
    b. PROMPT user to confirm the order
    c. IF user confirms THEN
        i. PROCEED to the order gateway
        ii. COMPLETE the transaction
        iii. SEND confirmation to the user
        iv. CLEAR the cart
    d. ELSE
        i. PRINT "Order not confirmed"
    e. END IF
    f. END

User Authentication

1. Begin

    // step 1: prompt user for input
2. display "enter your email:"
3. input email
4. display "enter your password:"
5. input password

    // step 2: validate email format
6. if email does not contain "@" or a valid domain then

    a. display "invalid email format."
    b. return to step 1
7. endif

   // step 3: query the database for the user
8. user = query "select * from users where email = email"

   // step 4: check if user exists
9. if user is null then
    a. display "user not found."
    b. return to step 1
10. endif

    // step 5: verify the password
11. if user.password == hash(password) then

        // password is correct, set session as authenticated
    a. set session.authenticated = true
    b. display "login successful. redirecting to dashboard..."
    c. redirect to "dashboard"
12. else

        // password is incorrect
    a. display "invalid password."
    b. return to step 1
13. endif
14. end

Budget Check Feature

1. BEGIN
2. FUNCTION budget_check(grocery_list, budget_limit):
   # Initialize total cost to 0
3. total_cost = 0

   # Step 2: Calculate the total cost
4. FOR each item IN grocery_list:
    a. item_cost = item['quantity'] * item['price_per_unit']
    b. total_cost += item_cost # Add the item cost to the total

   # Step 3: Check if total cost exceeds the budget
5. IF total_cost > budget_limit:
    a. PRINT "Budget exceeded! Please adjust your list."

   # Step 4: Provide editing options until budget is within limit
6. WHILE total_cost > budget_limit:

    a. PRINT "Options: Edit quantity / Remove items"
    b. user_choice = GET_USER_INPUT()
    c. IF user_choice == 'edit':
        i. CALL edit_item(grocery_list)
    d. ELSE IF user_choice == 'remove':
        i. CALL remove_item(grocery_list)

    # Step 5: Recalculate the total cost
7. total_cost = recalculate_cost(grocery_list)

    # Step 6: Save the list if within budget
8. PRINT "List saved successfully."
9. RETURN True # List saved

    # Helper function to recalculate total cost
    FUNCTION recalculate_cost(grocery_list):
1. new_total = 0
2. FOR each item IN grocery_list:
    a. new_total += item['quantity'] * item['price_per_unit']
3. RETURN new_total

    # Helper function to edit an item
    FUNCTION edit_item(grocery_list):
1. item_name = GET_USER_INPUT("Enter item name to edit: ")
2. new_quantity = GET_USER_INPUT("Enter new quantity: ")
3. FOR each item IN grocery_list:
    a. IF item['name'] == item_name:
        i. item['quantity'] = new_quantity

    # Helper function to remove an item
    FUNCTION remove_item(grocery_list):
1. item_name = GET_USER_INPUT("Enter item name to remove: ")
2. FOR each item IN grocery_list:
3. IF item['name'] == item_name:
    a. REMOVE item FROM grocery_list
4. END