**Test Document for get\_product,checkout and check\_cart functions**

1) **Module: Get\_Product**

The **get\_product** function reads a CSV file and loads product data into a list of Product objects. The function handles various scenarios including valid files, invalid inputs, and malformed CSV files, ensuring proper data processing and error handling.

**1. Valid Scenarios:**

* The function loads products correctly from a valid CSV file with 71 products and valid data.
* It handles files with special characters, whitespace, and case-sensitive product names.
* The function ignores extra fields in the CSV file, ensuring only valid product data is loaded.

**2. Invalid Scenarios:**

* The function raises appropriate errors for invalid input types such as integers, floats, and lists.
* It raises errors when the file path is a string that doesn't exist or is malformed.
* The function handles CSV files with missing columns, malformed data, or invalid data types (non-numeric values for price).

**3. Edge Cases:**

* The function handles empty CSV files by returning an empty list.
* It handles CSV files with extra fields, ignoring irrelevant data.
* The function strips whitespace from product names and loads products with special characters.

**4. Test cases:**

* **Test Case 1**: Check if the function raises a TypeError when an integer is passed as the input .
* **Test Case 2**: Check if the function raises a TypeError when a float is passed as the input .
* **Test Case 3**: Check if the function raises a FileNotFoundError when a string is passed as the file path .
* **Test Case 4**: Check if the function raises a TypeError when a list is passed as the file path .
* **Test Case 5**: Check if the function successfully loads products from a valid CSV file (products.csv) and returns 71 products with valid data.
* **Test Case 6**: Check if the function handles an empty CSV file by returning an empty list.
* **Test Case 7**: Check if the function raises a KeyError when the CSV file is malformed .
* **Test Case 8**: Check if the function raises a KeyError when the CSV file has missing data .
* **Test Case 9**: Check if the function raises a KeyError when the CSV file contains invalid data types .
* **Test Case 10**: Check if the function correctly handles product names with special characters .
* **Test Case 11**: Check if the function strips whitespace from product names .
* **Test Case 12**: Check if the function loads product names with case sensitivity .
* **Test Case 13**: Check if the function ignores extra fields in the CSV file and loads only valid products.
* **Test Case 14**: Check if the function correctly loads products with valid attributes from a properly formatted CSV file.

**2)Module: Checkout**

The **checkout** function processes the user's shopping cart and adjusts the user's wallet balance based on the total cost of the items in the cart. It also handles edge cases like insufficient funds, empty cart, and invalid inputs, ensuring smooth transactions.

**1. Valid Scenarios:**

* The user has enough balance in their wallet to cover the total cost.
* The shopping cart contains valid products with positive price and quantity.
* The checkout function correctly adjusts the user's wallet balance and clears the cart.

**2. Invalid Scenarios:**

* The user's wallet balance is insufficient to cover the total cost.
* The cart is empty.
* The product prices or quantities are zero or negative.
* Invalid product data (e.g., missing price or quantity) is present in the cart.

**3. Edge Cases:**

* The user's wallet balance exactly matches the total cost of the cart.
* The cart contains a large quantity of a single product.
* The cart has a mix of expensive and inexpensive products.
* A zero-priced product is included in the cart.
* The checkout function handles a cart with multiple products and exact wallet balance.

**4. Test Cases:**  
• **Test Case 1:** Check if the cart remains uncleared when the user has a negative wallet balance.  
• **Test Case 2:** Check if the cart clears but the wallet remains unchanged for zero price and quantity products.  
• **Test Case 3:** Check if the function handles an empty cart gracefully, leaving the wallet unchanged.  
• **Test Case 4:** Check if the user's wallet is debited correctly and the cart is cleared for sufficient balance and products in the cart.  
• **Test Case 5:** Check if the wallet decreases correctly and the cart clears for a single product checkout.  
• **Test Case 6:** Check if the wallet decreases by the correct amount and the cart clears for multiple products in the cart.  
• **Test Case 7:** Check if the wallet becomes zero and the cart clears when the balance matches the total cost exactly.  
• **Test Case 8:** Check if the wallet decreases correctly and the cart clears for a large quantity of a single product.  
• **Test Case 9:** Check if the cart remains uncleared when the wallet balance is zero.  
• **Test Case 10:** Check if the cart remains uncleared when the wallet balance is insufficient for an expensive product.  
• **Test Case 11:** Check if the wallet becomes zero and the cart clears when the product price matches the wallet balance exactly.  
• **Test Case 12:** Check if the wallet remains unchanged and the cart clears for zero-priced products.  
• **Test Case 13:** Check if the wallet becomes zero and the cart clears for multiple products with a total cost matching the wallet balance.  
• **Test Case 14:** Check if the cart remains uncleared when the wallet balance is zero and the product is expensive.

**3)Module- check\_cart**

The **check\_cart** function evaluates the contents of the user's cart and determines whether checkout is possible based on the wallet balance, product prices, and quantities. The function ensures that invalid data, empty carts, and edge cases like zero or negative balances are handled gracefully.

#### 1. Valid Scenarios:

* The user has enough balance to cover the total cost of products in the cart.
* The cart contains valid products with positive price and quantity.
* The wallet balance is correctly adjusted, and the cart is cleared post-checkout.

#### 2. Invalid Scenarios:

* The user's wallet balance is insufficient to cover the total cost.
* The cart contains products with zero or negative price or quantity.
* The user's wallet balance is negative or zero.

#### 3. Edge Cases:

* The wallet balance exactly matches the total product cost.
* The cart contains a mix of expensive and inexpensive products.
* The cart has products with zero price or quantity.
* The cart is empty.

#### 4. Test Cases:

* **Test Case 1:** Check if the user's wallet is reduced correctly and the cart is cleared for valid balance and valid products in the cart.
* **Test Case 2:** Check if the wallet is reduced correctly and the cart is cleared when there is a single product in the cart and sufficient wallet balance.
* **Test Case 3:** Check if the wallet is reduced correctly and the cart is cleared for multiple products with sufficient wallet balance.
* **Test Case 4:** Check if the wallet becomes zero and the cart clears when the wallet balance exactly equals the total product price.
* **Test Case 5:** Check if the wallet reduces correctly and the cart is cleared for a large quantity of a single product.
* **Test Case 6:** Check if the wallet remains unchanged and the cart remains empty when attempting checkout with an empty cart.
* **Test Case 7:** Check if the wallet remains unchanged and the cart is not cleared when the balance is insufficient for an expensive product.
* **Test Case 8:** Check if the wallet becomes zero and the cart clears when the exact wallet balance matches the total product price.
* **Test Case 9:** Check if the wallet remains unchanged and the cart is cleared for products with zero price.
* **Test Case 10:** Check if the wallet reduces by the total cost and the cart clears when multiple products’ total cost matches the wallet balance.
* **Test Case 11:** Check if the wallet remains negative and the cart is not cleared when the user has a negative wallet balance.
* **Test Case 12:** Check if the wallet remains unchanged and the cart is cleared when the cart contains products with zero price and quantity.
* **Test Case 13:** Check if the wallet remains unchanged and the cart remains empty when attempting checkout with an already empty cart.
* **Test Case 14:** Check if the wallet remains unchanged and the cart is not cleared when the wallet balance is insufficient for an expensive product.