EX.NO.: 04
DATE: 07.07.2025

PYTEST FOR E-COMMERCE PLATFORM

AIM

To **design and implement an automated test suite using `pytest`** for validating the behavior of an `e-commerce checkout system`, specifically the function `checkout_cart()`, which processes cart items, checks stock and balance, and applies discount coupons during checkout.

ALGORITHM

1. Initialize `total = 0`.
2. For each item in `cart_items`:
3. Extract `item_id`, `quantity`, and `price`.
4. Check if `item_id` exists in `available_stock`.
    a. If not, return error: item not in stock.
5. Check if `available_stock[item_id] >= quantity`.
    a. If not, return error: not enough stock.
6. Add `price * quantity` to `total`.
7. If a `coupon_code` is provided:
8. If it's `"DISCOUNT10"`, apply 10% discount.
9. Compute `final_amount = total - discount`.
10. Check if `user_balance >= final_amount`.
11. If not, return error: insufficient balance.
12. Return a success message with the `final_amount`.

CODE AND OUTPUT

```
# checkout.py

def checkout_cart(cart_items, user_balance, available_stock, coupon_code=None):
    total = 0

    for item in cart_items:
        item_id = item['item_id']
        quantity = item['quantity']
        price = item['price']

        if item_id not in available_stock:
            return {"success": False, "message": f"Item {item_id} not in stock",
"final_amount": 0}

        if available_stock[item_id] < quantity:
            return {"success": False, "message": f"Not enough stock for item
{item_id}", "final_amount": 0}

        total += price * quantity

    discount = 0
    if coupon_code == "DISCOUNT10":
        discount = 0.10 * total
```

```
    final_amount = total - discount

    if user_balance < final_amount:
        return {"success": False, "message": "Insufficient balance", "final_amount":
final_amount}

    return {"success": True, "message": "Checkout successful!", "final_amount":
final_amount}
```

```
# test_checkout.py

import pytest
from checkout import checkout_cart

@pytest.fixture
def cart_data():
    return [
        {"item_id": "A1", "quantity": 2, "price": 100},
        {"item_id": "B2", "quantity": 1, "price": 200}
    ]

@pytest.fixture
def stock_data():
    return {"A1": 5, "B2": 2}

def test_checkout_success(cart_data, stock_data):
    result = checkout_cart(cart_data, user_balance=500, available_stock=stock_data)
    assert result["success"] is True
    assert result["final_amount"] == 400
    assert "successful" in result["message"].lower()

def test_checkout_insufficient_balance(cart_data, stock_data):
    result = checkout_cart(cart_data, user_balance=100, available_stock=stock_data)
    assert result["success"] is False
    assert result["message"] == "Insufficient balance"

def test_checkout_out_of_stock(cart_data):
    stock_data = {"A1": 1, "B2": 1}  # A1 required: 2, available: 1
    result = checkout_cart(cart_data, user_balance=1000, available_stock=stock_data)
    assert result["success"] is False
    assert "Not enough stock" in result["message"]

def test_checkout_item_not_in_stock(cart_data):
    stock_data = {"B2": 1}  # A1 not present
    result = checkout_cart(cart_data, user_balance=1000, available_stock=stock_data)
    assert result["success"] is False
    assert "not in stock" in result["message"]
```

```python
def test_checkout_with_coupon(cart_data, stock_data):
    result = checkout_cart(cart_data, user_balance=500, available_stock=stock_data,
coupon_code="DISCOUNT10")
    expected_discount = 0.10 * 400
    assert result["success"] is True
    assert result["final_amount"] == pytest.approx(400 - expected_discount, 0.01)

def test_invalid_coupon_code(cart_data, stock_data):
    result = checkout_cart(cart_data, user_balance=500, available_stock=stock_data,
coupon_code="FAKECODE")
    assert result["success"] is True
    assert result["final_amount"] == 400  # no discount applied
```

```
============================= test session starts =============================
platform win32 -- Python 3.13.5, pytest-8.4.1, pluggy-1.6.0 -- C:\Python313\python.exe
cachedir: .pytest_cache
rootdir: D:\TARU\V th year\Software Testing lab\Ex 4
collected 6 items

test_checkout.py::test_checkout_success PASSED                           [ 16%]
test_checkout.py::test_checkout_insufficient_balance PASSED              [ 33%]
test_checkout.py::test_checkout_out_of_stock PASSED                      [ 50%]
test_checkout.py::test_checkout_item_not_in_stock PASSED                 [ 66%]
test_checkout.py::test_checkout_with_coupon PASSED                       [ 83%]
test_checkout.py::test_invalid_coupon_code PASSED                        [100%]

============================== 6 passed in 0.02s ==============================
```

INFERENCE

The `checkout_cart()` function handles all major edge cases like **stock validation**, **coupon handling**, and **balance sufficiency** correctly. Discount codes are applied **only if valid**, and the function is **robust to invalid inputs**. The test suite ensures **reliable behavior** for both normal and exceptional checkout scenarios.