```python
import hashlib
import time
from random import randint


# Define the Block class to represent each block in the blockchain
class Block:
    def __init__(self, index, previous_hash, timestamp, data, hash):
        self.index = index
        self.previous_hash = previous_hash
        self.timestamp = timestamp
        self.data = data
        self.hash = hash

    def __repr__(self):
        return f"\nBlock {self.index} [Hash: {self.hash}]\nPrevious Hash: {self.previous_hash}\nData: {self.data}\nTimestamp: {time.ctime(self.timestamp)}\n"


# Define the Blockchain class to manage the chain of blocks
class Blockchain:
    def __init__(self):
        self.chain = [self.create_genesis_block()]

    def create_genesis_block(self):
        # Create the initial block in the blockchain
        return Block(0, "0", time.time(), "Genesis Block", self.hash_block(0, "0", time.time(), "Genesis Block"))

    def get_latest_block(self):
        # Retrieve the most recent block in the blockchain
        return self.chain[-1]

    def add_block(self, data):
        # Add a new block to the blockchain
        latest_block = self.get_latest_block()
        index = latest_block.index + 1
        timestamp = time.time()
        previous_hash = latest_block.hash
        hash = self.hash_block(index, previous_hash, timestamp, data)
        new_block = Block(index, previous_hash, timestamp, data, hash)
        self.chain.append(new_block)

    def hash_block(self, index, previous_hash, timestamp, data):
        # Generate a hash for a block
        block_string = f"{index}{previous_hash}{timestamp}{data}".encode()
        return hashlib.sha256(block_string).hexdigest()

    def is_chain_valid(self):
        # Verify the integrity of the blockchain
        for i in range(1, len(self.chain)):
            current_block = self.chain[i]
            previous_block = self.chain[i - 1]
            if current_block.hash != self.hash_block(current_block.index, current_block.previous_hash, current_block.timestamp, current_block.data):
                return False
            if current_block.previous_hash != previous_block.hash:
                return False
        return True

    def display_chain(self):
        # Print out the entire blockchain
        for block in self.chain:
            print(block)


# Define the PharmaceuticalSupplyChain class to simulate the supply chain
class PharmaceuticalSupplyChain:
    def __init__(self):
        self.blockchain = Blockchain()

    def simulate_iot_data(self):
        # Simulate IoT data for temperature and humidity
        temperature = randint(2, 8)  # Temperature between 2°C and 8°C
        humidity = randint(30, 50)   # Humidity between 30% and 50%
        return {"Temperature": f"{temperature}°C", "Humidity": f"{humidity}%"}

    def add_drug_shipment(self, shipment_id, drug_name):
        # Add a new drug shipment to the supply chain
        iot_data = self.simulate_iot_data()
        data = {
            "Shipment ID": shipment_id,
            "Drug Name": drug_name,
            "IoT Data": iot_data,
            "Timestamp": time.ctime()
        }
        self.blockchain.add_block(data)

    def display_supply_chain(self):
        # Display the entire supply chain
        print("\n--- Pharmaceutical Supply Chain ---")
        self.blockchain.display_chain()

    def validate_supply_chain(self):
        # Validate the integrity of the supply chain
        return self.blockchain.is_chain_valid()


# Main program to interact with the user
def main():
    print("Welcome to the Pharmaceutical Supply Chain Management System")
    supply_chain = PharmaceuticalSupplyChain()

    while True:
        print("\nOptions:")
        print("1. Add a new drug shipment")
        print("2. Display the supply chain")
        print("3. Validate the supply chain")
        print("4. Exit")

        choice = input("\nEnter your choice (1-4): ")

        if choice == '1':
            shipment_id = input("Enter Shipment ID: ")
            drug_name = input("Enter Drug Name: ")
            supply_chain.add_drug_shipment(shipment_id, drug_name)
            print("\nShipment added successfully!")

        elif choice == '2':
            supply_chain.display_supply_chain()

        elif choice == '3':
            is_valid = supply_chain.validate_supply_chain()
            print("\nSupply Chain Integrity: " + ("Valid" if is_valid else "Compromised"))

        elif choice == '4':
            print("\nExiting the system. Goodbye!")
            break

        else:
            print("\nInvalid choice! Please select a valid option.")

# Run the main program
if __name__ == "__main__":
    main()
```

```
Enter Shipment ID: 9563
Enter Drug Name: Hydroxyzine

Shipment added successfully!

Options:
1. Add a new drug shipment
2. Display the supply chain
3. Validate the supply chain
4. Exit

Enter your choice (1-4): 2

--- Pharmaceutical Supply Chain ---

Block 0 [Hash: 8d3f040f23351156b989f46827d8cdfe53664d5e9d04e847ad0c6e232c67df78]
Previous Hash: 0
Data: Genesis Block
Timestamp: Tue Aug 20 17:03:28 2024


Block 1 [Hash: 20712ac42f9c0d749a38eca4446d8c2beca0904e1beeb8adbdaeb27933e6b983]
Previous Hash: 8d3f040f23351156b989f46827d8cdfe53664d5e9d04e847ad0c6e232c67df78
Data: {'Shipment ID': '1234', 'Drug Name': 'Dolo-650', 'IoT Data': {'Temperature': '5°C', 'Humidity': '44%'}, 'Timestamp': 'Tue Aug 20 17:03:51 2024'}
Timestamp: Tue Aug 20 17:03:51 2024


Block 2 [Hash: 0c4620644fbdef4e973172aa234e224b7cf3ddf8c4ce447beb72b441cf1156b4]
Previous Hash: 20712ac42f9c0d749a38eca4446d8c2beca0904e1beeb8adbdaeb27933e6b983
Data: {'Shipment ID': '7894', 'Drug Name': 'Amoxicillin', 'IoT Data': {'Temperature': '2°C', 'Humidity': '46%'}, 'Timestamp': 'Tue Aug 20 17:04:54 2024'}
Timestamp: Tue Aug 20 17:04:54 2024


Block 3 [Hash: a9a7ec9e13e7daeb553af17cec216487f0aacbd27dfaffba350f6affb2abd389]
Previous Hash: 0c4620644fbdef4e973172aa234e224b7cf3ddf8c4ce447beb72b441cf1156b4
Data: {'Shipment ID': '9563', 'Drug Name': 'Hydroxyzine ', 'IoT Data': {'Temperature': '8°C', 'Humidity': '37%'}, 'Timestamp': 'Tue Aug 20 17:06:15 2024'}
Timestamp: Tue Aug 20 17:06:15 2024


Options:
1. Add a new drug shipment
2. Display the supply chain
3. Validate the supply chain
4. Exit

Enter your choice (1-4): 3

Supply Chain Integrity: Valid

Options:
1. Add a new drug shipment
2. Display the supply chain
3. Validate the supply chain
4. Exit

Enter your choice (1-4): 4

Exiting the system. Goodbye!
```