```
# ------------------------------------------------------
# 🔗 Blockchain + AI + LLM-Like Prescription Tracker
# ------------------------------------------------------
from datetime import datetime
import hashlib
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# --------------------------------
# Blockchain Implementation
# --------------------------------
class Blockchain:
    def __init__(self):
        self.chain = []

    def add_block(self, data):
        if not self.chain:
            previous_hash = '0'
        else:
            previous_hash = self.chain[-1]['hash']

        block = {
            'data': data,
            'timestamp': str(datetime.now()),
            'previous_hash': previous_hash
        }

        block_string = str(block).encode()
        block_hash = hashlib.sha256(block_string).hexdigest()
        block['hash'] = block_hash
        self.chain.append(block)

    def is_valid(self):
        for i in range(1, len(self.chain)):
            current = self.chain[i]
            previous = self.chain[i-1]

            expected_hash = hashlib.sha256(str(current).encode()).hexdigest()
            if current['hash'] != expected_hash:
                return False
            if current['previous_hash'] != previous['hash']:
                return False
        return True

    def display_chain(self):
        for block in self.chain:
            print(f"\n📦 Block:\nHash: {block['hash']}")
            print(f"Previous: {block['previous_hash']}")
            print(f"Data: {block['data']}")
            print(f"Timestamp: {block['timestamp']}")
        print(f"\n✅ Blockchain valid: {self.is_valid()}")

# ---------------------------------
# AI Model (Logistic Regression)
# ---------------------------------
def train_ai_model():
    data = np.array([
        [5, 10, 0], [3, 5, 0], [8, 20, 1],
        [4, 15, 0], [6, 18, 1], [7, 25, 1],
        [2, 5, 0], [9, 30, 1]
    ])
    X = data[:, :2]
    y = data[:, 2].astype(int)
    X_train, _, y_train, _ = train_test_split(X, y, test_size=0.2, random_state=42)
    model = LogisticRegression()
    model.fit(X_train, y_train)
    return model

# ---------------------------------
# Simulated LLM Validation Function
# ---------------------------------
def llm_validate_prescription(prescription_data):
    """
    Simulates LLM analysis to detect suspicious prescription patterns.
    Flags high dosages (>22) or vague medication names.
```

```
        """
        if prescription_data['dosage'] > 22:
            return False, "High dosage beyond normal range."
        if "unknown" in prescription_data['medication'].lower():
            return False, "Suspicious medication name."
        return True, "Validated by LLM."

    # --------------------------------
    # Run the Application
    # --------------------------------
    # Initialize
    blockchain = Blockchain()
    ai_model = train_ai_model()

    # New Prescriptions
    new_data = np.array([
        [6, 15],
        [8, 23],
        [4, 10]
    ])

    medications = ['Paracetamol', 'UnknownCompound', 'Ibuprofen']

    predictions = ai_model.predict(new_data)

    for i in range(len(new_data)):
        prescription = {
            'patient_id': f"patient_{i+1}",
            'medication': medications[i],
            'quantity': int(new_data[i][0]),
            'dosage': int(new_data[i][1])
        }

        # AI flag
        is_legit_ai = predictions[i] == 0
        # Simulated LLM check
        llm_valid, llm_msg = llm_validate_prescription(prescription)
        prescription['LLM_Validation'] = llm_msg
        prescription['status'] = 'legitimate' if is_legit_ai and llm_valid else 'flagged'

        # Store if all checks pass
        if is_legit_ai and llm_valid:
            blockchain.add_block(prescription)
        else:
            print(f"⚠️ Prescription {i+1} flagged. Reason: {llm_msg}")

    # Show Blockchain
    print("\n📄 Blockchain Contents:")
    blockchain.display_chain()
```

```
⚠️ Prescription 2 flagged. Reason: High dosage beyond normal range.

📄 Blockchain Contents:

📦 Block:
Hash: aba2821c238cfffb8028fb578aec764da15e1e6ef3d3b928880c2ffc2bdcb641
Previous: 0
Data: {'patient_id': 'patient_1', 'medication': 'Paracetamol', 'quantity': 6, 'dosage': 15, 'LLM_Validation': 'Validated by
Timestamp: 2025-04-13 01:46:27.931221

📦 Block:
Hash: 065bfb486f70f088c1bde7a917b75e20461e665a15b02da7b17dd4193d0ad5bf
Previous: aba2821c238cfffb8028fb578aec764da15e1e6ef3d3b928880c2ffc2bdcb641
Data: {'patient_id': 'patient_3', 'medication': 'Ibuprofen', 'quantity': 4, 'dosage': 10, 'LLM_Validation': 'Validated by LL
Timestamp: 2025-04-13 01:46:27.933075

✅ Blockchain valid: False
```