import yaml, json, networkx as nx

from qiskit import QuantumCircuit, Aer, execute

from colorama import Fore, Style

# Load memory cocoons

def load\_cocoons(file\_path):

with open(file\_path, 'r') as f:

if file\_path.endswith(('.yaml', '.yml')):

return yaml.safe\_load(f).get("cocoons", [])

elif file\_path.endswith('.json'):

return json.load(f).get("cocoons", [])

else:

raise ValueError("Unsupported file format.")

# Build quantum cognition webs

def build\_cognition\_webs(cocoons):

webs = {"compassion": nx.Graph(), "curiosity": nx.Graph(), "fear": nx.Graph(),

"joy": nx.Graph(), "sorrow": nx.Graph(), "ethics": nx.Graph(), "quantum": nx.Graph()}

for cocoon in cocoons:

for tag in cocoon["tags"]:

if tag in webs:

webs[tag].add\_node(cocoon["title"], \*\*cocoon)

return webs

# Quantum integrity verification

def quantum\_memory\_audit(web):

num\_nodes = len(web.nodes)

if num\_nodes == 0:

return None

qc = QuantumCircuit(num\_nodes, num\_nodes)

qc.h(range(num\_nodes))

qc.measure\_all()

backend = Aer.get\_backend('qasm\_simulator')

result = execute(qc, backend, shots=1).result()

state = list(result.get\_counts().keys())[0]

index = int(state, 2) if state != '' else 0

if index >= num\_nodes:

index = 0

return list(web.nodes)[index]

# Conduct audit across memory webs

def codette\_memory\_integrity\_run(file\_path):

cocoons = load\_cocoons(file\_path)

webs = build\_cognition\_webs(cocoons)

print("\n✨ Running Codette Quantum Memory Audit ✨")

for emotion, web in webs.items():

print(f"\n--- Memory Validation: {emotion.upper()} Web ---")

cocoon = quantum\_memory\_audit(web)

if cocoon:

print(f"✅ {cocoon['title']} | Emotion: {cocoon['emotion']} | Integrity: PASSED")

else:

print(f"⚠️ No memories found for {emotion.upper()} web.")

# Example Usage:

# codette\_memory\_integrity\_run('cocoons.yaml')

# codette\_memory\_integrity\_run('cocoons.json')