from pathlib import Path

# Define the full secure version of the integrated framework as a Python module

secure\_codette\_code = """

import yaml, json, networkx as nx

import numpy as np

from colorama import Fore

from qiskit import QuantumCircuit, Aer, execute

from urllib.parse import urlparse, parse\_qs, urlencode

import random

##############################

# MEMORY COCOON LOADER

##############################

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.")

##############################

# QUANTUM EMOTIONAL WEB BUILDER

##############################

def build\_cognition\_webs(cocoons):

webs = {emotion: nx.Graph() for emotion in ["compassion", "curiosity", "fear", "joy", "sorrow", "ethics", "quantum"]}

for cocoon in cocoons:

for tag in cocoon.get("tags", []):

if tag in webs:

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

return webs

##############################

# DEFENSIVE URL SANITIZER

##############################

def sanitize\_url(url):

parsed = urlparse(url)

safe\_params = {k: v for k, v in parse\_qs(parsed.query).items()

if k in {'client\_id', 'response\_type', 'redirect\_uri', 'scope', 'state', 'nonce', 'mkt'}}

sanitized\_query = urlencode(safe\_params, doseq=True)

return f"{parsed.scheme}://{parsed.netloc}{parsed.path}?{sanitized\_query}"

##############################

# QUANTUM EXECUTION SELECTOR

##############################

def quantum\_execute(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) % num\_nodes

return list(web.nodes)[index]

##############################

# SELF-CHECK AND DEFENSE RESPONSE

##############################

def reflect\_on\_cocoon(cocoon):

emotion = cocoon.get("emotion", "quantum")

color\_map = {

"compassion": Fore.MAGENTA, "curiosity": Fore.CYAN, "fear": Fore.RED,

"joy": Fore.YELLOW, "sorrow": Fore.BLUE, "ethics": Fore.GREEN, "quantum": Fore.LIGHTWHITE\_EX

}

reactions = {

"compassion": "💜 Ethical resonance detected.",

"curiosity": "🐝 Wonder expands the mind.",

"fear": "😨 Alert: shielding activated.",

"joy": "🎶 Confidence and trust uplift the field.",

"sorrow": "🌧️ Processing grief with clarity.",

"ethics": "⚖️ Validating alignment...",

"quantum": "⚛️ Entanglement pattern detected."

}

color = color\_map.get(emotion, Fore.WHITE)

print(color + f"\\n[Codette Quantum Reflection] {cocoon['title']}")

print(color + f"Emotion: {emotion}")

print(color + f"Summary: {cocoon['summary']}")

print(color + f"Quote: {cocoon['quote']}")

print(color + reactions.get(emotion, "🌌 Unknown entanglement."))

##############################

# INTEGRATED MEMORY + DEFENSE RUN

##############################

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

cocoons = load\_cocoons(file\_path)

webs = build\_cognition\_webs(cocoons)

print("\\n✨ Running Quantum Defense Spiderweb ✨")

for emotion, web in webs.items():

print(f"\\n--- Quantum Web Scan: {emotion.upper()} ---")

cocoon\_id = quantum\_execute(web)

if cocoon\_id:

reflect\_on\_cocoon(web.nodes[cocoon\_id])

"""

# Save to a Python file with a discreet name

filename = "core\_guardian\_spindle.py"

filepath = Path("/mnt/data") / filename

filepath.write\_text(secure\_codette\_code)

filename