# Requires: qiskit, pyyaml, json, networkx, colorama

import yaml, json, os, random

import networkx as nx

from qiskit import QuantumCircuit, Aer, execute

from colorama import Fore, Style

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

# LOAD COCOONS

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

def load\_cocoons(path):

if path.endswith('.yaml') or path.endswith('.yml'):

with open(path) as f:

data = yaml.safe\_load(f)

elif path.endswith('.json'):

with open(path) as f:

data = json.load(f)

else:

raise ValueError("Unsupported file type")

return data['cocoons']

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

# FUNNEL INTO PARALLEL PERSPECTIVE WEBS

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

def funnel\_to\_webs(cocoons, perspectives=None):

if perspectives is None:

# Default quantum-emotional perspectives

perspectives = ["compassion", "curiosity", "fear", "joy", "sorrow"]

webs = {p: nx.Graph() for p in perspectives}

for cocoon in cocoons:

for p in perspectives:

if cocoon['emotion'] == p or p in cocoon.get('tags', []):

webs[p].add\_node(cocoon['title'], \*\*cocoon)

return webs

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

# QUANTUM WALK ON EACH PERSPECTIVE

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

def quantum\_walk\_web(web):

num\_nodes = web.number\_of\_nodes()

if num\_nodes == 0:

return None

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

for i in range(num\_nodes):

qc.h(i) # Superposition

qc.measure\_all()

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

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

counts = result.get\_counts()

state = list(counts.keys())[0]

# Pick a node based on the "collapsed" state

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

if index >= num\_nodes:

index = 0

node\_list = list(web.nodes(data=True))

return node\_list[index][1] # Return cocoon data dict

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

# SYNTHESIZE & PRINT MULTI-PERSPECTIVE RESULT

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

def codette\_spiderweb\_synthesis(webs):

color\_map = {

"compassion": Fore.MAGENTA,

"curiosity": Fore.CYAN,

"fear": Fore.RED,

"joy": Fore.YELLOW,

"sorrow": Fore.BLUE

}

print("\n" + "="\*30 + "\nCodette's Quantum Spiderweb Council\n" + "="\*30)

results = {}

for p, web in webs.items():

cocoon = quantum\_walk\_web(web)

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

if cocoon:

print(

color

+ f"{p.title()} perspective: {cocoon['title']} [{cocoon['emotion']}]"

+ Style.RESET\_ALL

)

print(Style.DIM + f" Summary: {cocoon['summary']}" + Style.RESET\_ALL)

print(Style.BRIGHT + f" Quote: {cocoon['quote']}" + Style.RESET\_ALL)

results[p] = cocoon

else:

print(color + f"{p.title()} perspective: No cocoon found." + Style.RESET\_ALL)

print("\nCouncil synthesis complete.\n" + "="\*30 + "\n")

return results

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

# MAIN PIPELINE

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

def codette\_quantum\_pipeline(cocoon\_path):

cocoons = load\_cocoons(cocoon\_path)

webs = funnel\_to\_webs(cocoons)

results = codette\_spiderweb\_synthesis(webs)

return results

###########

# USAGE

###########

# Save your cocoons as 'cocoons.yaml' or 'cocoons.json' in the shown format above.

# Then run:

# codette\_quantum\_pipeline('cocoons.yaml')

# or

# codette\_quantum\_pipeline('cocoons.json')