# Your code is already advanced and well-structured for demonstration of multi-agent null repair and recursion control.

# I will run it as-is to confirm functionality, then explain how you can extend this for even more "Codette" realism.

class AdvancedCodetteAI:

def \_\_init\_\_(self, agents, max\_retries=2):

self.agents = agents # Dictionary of specialized response agents

self.max\_retries = max\_retries

def generate\_response(self, topic, attempt=0):

agent = self.agents.get(topic, self.agents.get('default'))

response = agent()

if response is None:

if attempt < self.max\_retries:

next\_topic = self.\_next\_fallback\_topic(topic)

return self.generate\_response(next\_topic, attempt + 1)

else:

return "System: Exhausted all self-healing attempts. Please try a new prompt or check system inputs."

return response

def \_next\_fallback\_topic(self, current\_topic):

fallback\_topics = list(self.agents.keys())

if current\_topic in fallback\_topics:

idx = fallback\_topics.index(current\_topic)

next\_idx = (idx + 1) % len(fallback\_topics)

return fallback\_topics[next\_idx]

else:

return 'default'

# Realistic specialized agents

def creative\_collaboration\_agent():

return ("I collaborate with users to co-create multifaceted art, literature, and music, "

"blending algorithmic innovation with human intuition.")

def simulation\_agent():

return ("I execute dynamic 'what-if' simulations, leveraging multi-agent scenario planning and predictive modeling.")

def default\_agent():

return None # Simulate 'null' for self-healing test

# Agents dictionary

agents = {

'creative\_collaboration': creative\_collaboration\_agent,

'simulation': simulation\_agent,

'default': default\_agent

}

# Instantiate and test

codette\_ai = AdvancedCodetteAI(agents)

print(codette\_ai.generate\_response('default'))

print(codette\_ai.generate\_response('creative\_collaboration'))

print(codette\_ai.generate\_response('simulation'))

# Extension ideas for next steps will follow after confirming output.

I collaborate with users to co-create mu