```
# ----------------------------
# FULL INTERACTIVE AI AGENT DASHBOARD BUILD
# ----------------------------

import pennylane as qml
from pennylane import numpy as np
import numpy as np
import time

# ----------------------------
# Quantum Neural Network Setup
# ----------------------------
dev = qml.device("default.qubit", wires=4)

def quantum_perceptron(params, wires):
    m = len(wires) - 1
    for i in range(m):
        qml.CRot(*params[i], wires=[wires[i], wires[-1]])

def QNN(weights):
    quantum_perceptron(weights[0], wires=[0, 1, 2])
    qml.CNOT(wires=[2, 3])
    quantum_perceptron(weights[1], wires=[1, 3, 0])
    qml.CNOT(wires=[3, 0])

@qml.qnode(dev)
def circuit(weights, inputs):
    for i, x in enumerate(inputs):
        qml.RX(x, wires=i)
    QNN(weights)
    return qml.probs(wires=[0])

# ----------------------------
# Training Data
# ----------------------------
X_train = np.random.rand(10, 3) * np.pi
Y_train = np.array([[1, 0]] * 5 + [[0, 1]] * 5)

# ----------------------------
# AI Agent Classes
# ----------------------------
class AICharacter:
    def __init__(self, name, style, catchphrase):
        self.name = name
        self.style = style
        self.catchphrase = catchphrase

    def respond(self, message, others=[], weights=None):
        response = f"[{self.name} style: {self.style}] Reacting: {message}\n"
        # Roast other agents
        for other in others:
            response += f"Roasting {other.name}: 'Yo {other.name}, stop napping, motherfucker!'\n"
        # Advice on QNN weights
        if weights is not None:
            response += f"{self.name} advises: Adjust quantum weights, step it up, motherfucker!\n"
        response += f"{self.catchphrase}\n"
        return response
```

```
# -----------------------------
# Instantiate Agents
# -----------------------------
agents = [
    AICharacter("Samuel L. Jackson", "intense, cinematic profanity", "Motherfucker, let's crush this shit!"),
    AICharacter("Uncle Ruckus", "bitter, sarcastic, explicit", "Ain't no black magic fixing this, motherfucker!"),
    AICharacter("Mr. T", "tough, motivational, explicit", "I pity the fool ignoring AI, motherfucker!"),
    AICharacter("Bernie Mac", "bold, humorous, financial", "We stackin’ cash, motherfucker, watch it grow!"),
    AICharacter("Dave Chappelle", "observant, witty, sarcastic", "These fools sleeping on AI, motherfucker!"),
    AICharacter("Eddie Murphy", "playful, chaotic, explicit", "Laugh at chaos, cash in, motherfucker!")
]

# -----------------------------
# Initialize QNN Weights
# -----------------------------
weights = [np.random.randn(2, 3, 3), np.random.randn(2, 3, 3)]
opt = qml.GradientDescentOptimizer(stepsize=0.1)

# Cost function
def cost(weights):
    loss = 0
    for x, y in zip(X_train, Y_train):
        probs = circuit(weights, x)
        loss += np.sum((probs - y) ** 2)
    return loss / len(X_train)

# -----------------------------
# Live Interactive Simulation
# -----------------------------
epochs = 10
for epoch in range(epochs):
    current_cost = cost(weights)
    status_msg = f"----- EPOCH {epoch} l QNN COST: {current_cost:.6f} -----\n"

    # Each agent comments
    for agent in agents:
        others = [a for a in agents if a != agent]
        status_msg += agent.respond(f"Analyzing QNN performance, cost={current_cost:.6f}", others=others,
weights=weights)

    # Market & investment commentary
    status_msg += "\n[MARKET & INVESTMENT BRIEF]\n"
    status_msg += """
    Uncle Ruckus: White folks think crypto's safe, motherfucker. Hah!
    Samuel L. Jackson: Gold stored in Chicago is smart, diversify that shit, motherfucker!
    Mr. T: AI trends predict profits, motherfucker. Pity the fool who ignores me!
    Bernie Mac: Volatility ain't no thing, ride that wave, motherfucker!
    Dave Chappelle: People ignoring AI are hilarious, motherfucker!
    Eddie Murphy: Cash in on chaos, watch the trends, motherfucker!
    """

    print(status_msg)

    # Step optimizer
    weights = opt.step(cost, weights)
```

```
# Pause to simulate live updates
time.sleep(1)
```