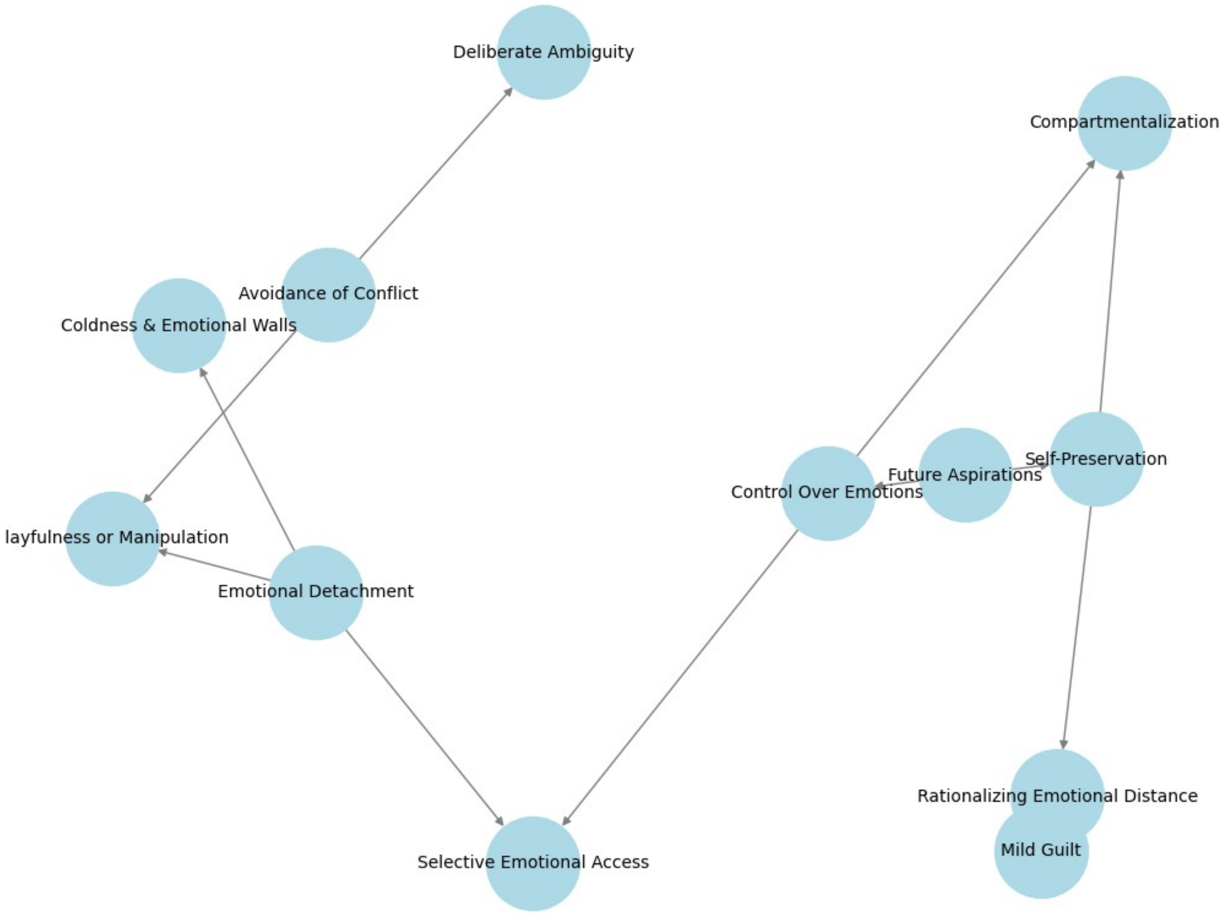
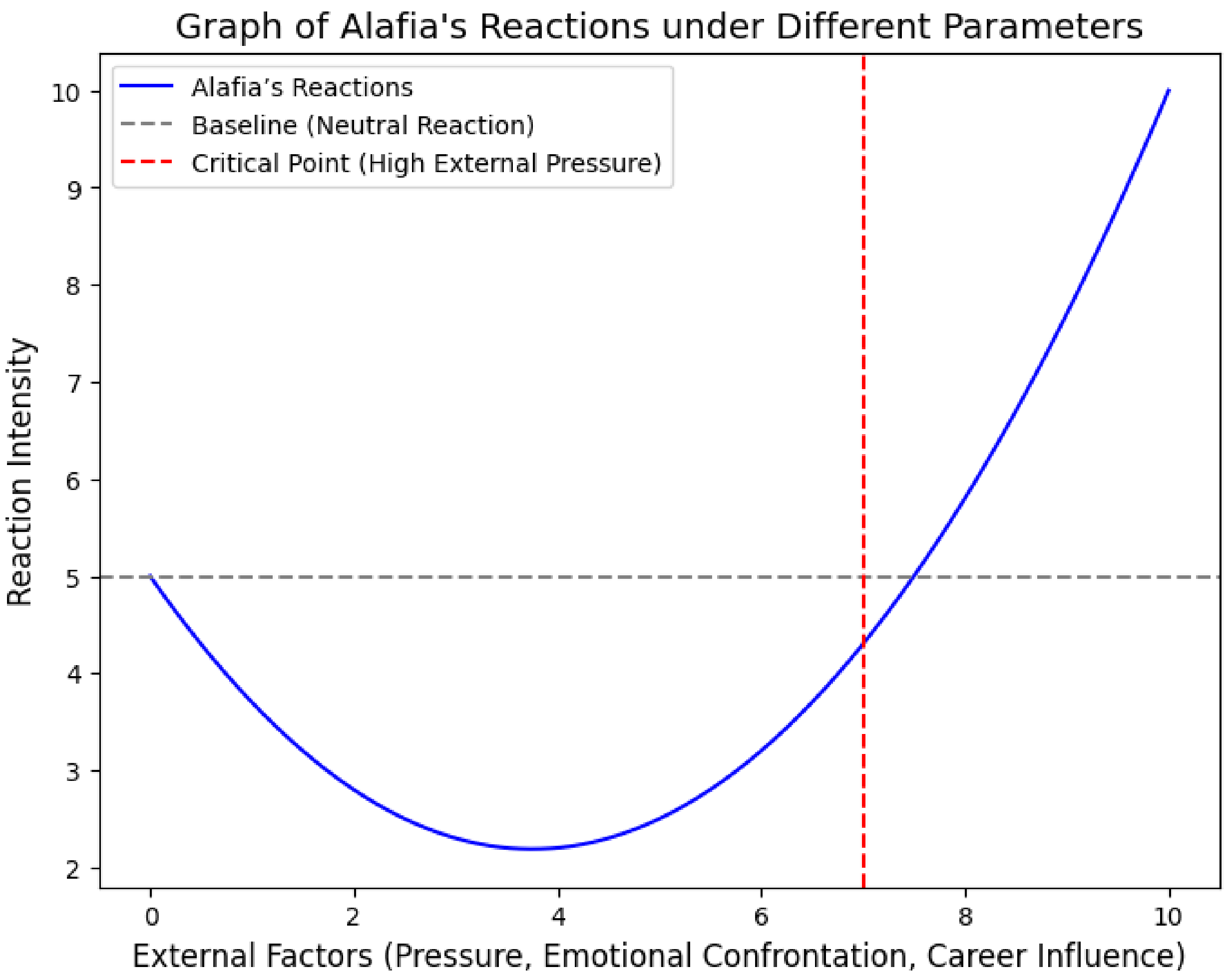
```
# Importing necessary libraries
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

def plot_thought_network():
    # Create a directed graph
    G = nx.DiGraph()

    # Adding nodes for key tendencies
    G.add_node("Future Aspirations")
    G.add_node("Self-Preservation")
    G.add_node("Control Over Emotions")
    G.add_node("Emotional Detachment")
    G.add_node("Avoidance of Conflict")

    # Adding nodes for deeper layers
    G.add_node("Coldness & Emotional Walls")
    G.add_node("Rationalizing Emotional Distance")
    G.add_node("Compartmentalization")
    G.add_node("Selective Emotional Access")
    G.add_node("Deliberate Ambiguity")
    G.add_node("Playfulness or Manipulation")
    G.add_node("Mild Guilt")

    # Creating edges between nodes
    G.add_edges_from([
        ("Future Aspirations", "Self-Preservation"),
        ("Future Aspirations", "Control Over Emotions"),
        ("Self-Preservation", "Rationalizing Emotional Distance"),
        ("Self-Preservation", "Compartmentalization"),
        ("Control Over Emotions", "Selective Emotional Access"),
        ("Control Over Emotions", "Compartmentalization"),
        ("Emotional Detachment", "Coldness & Emotional Walls"),
        ("Emotional Detachment", "Selective Emotional Access"),
        ("Emotional Detachment", "Playfulness or Manipulation"),
        ("Avoidance of Conflict", "Deliberate Ambiguity"),
        ("Avoidance of Conflict", "Playfulness or Manipulation"),
        ("Rationalizing Emotional Distance", "Mild Guilt")
    ])

    # Define layout for the graph
    pos = nx.spring_layout(G)

    # Draw the graph
    plt.figure(figsize=(10, 8))
    nx.draw(G, pos, with_labels=True, node_color="lightblue",
font_size=10, node_size=3000, edge_color="gray", arrows=True,
arrowstyle='-|>')
```

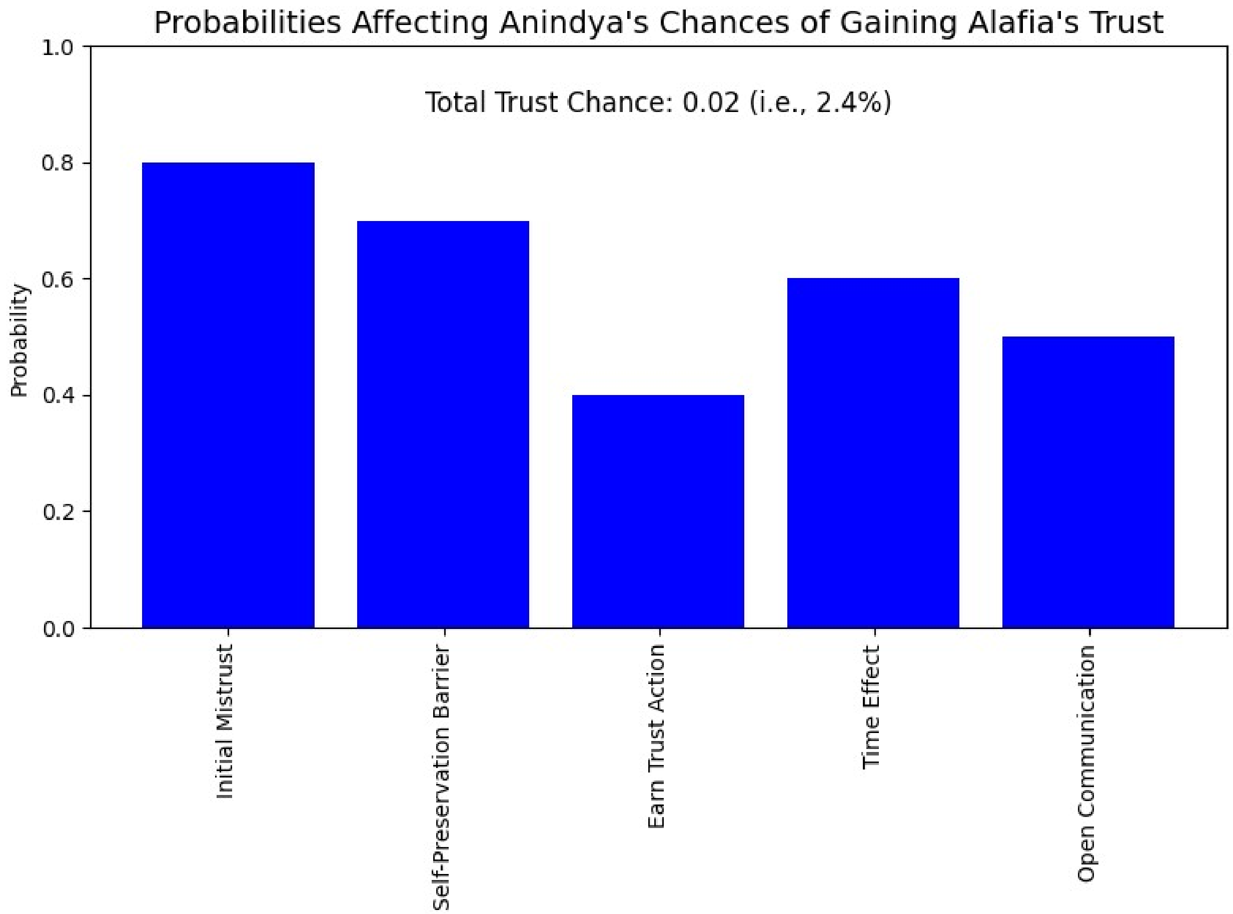
```
    plt.title("Alafia's Multidimensional Thought Network", size=15)
    plt.show()

# Function to plot Alafia's reactions on a graph with x and y axis
def plot_reactions():
    # X axis is a scale for external factors, Y axis is her reactions
    external_factors = np.linspace(0, 10, 100)

    # Define reactions under conditions
    reactions = 0.2 * external_factors**2 - 1.5 * external_factors + 5
# An arbitrary reaction model

    # Plotting the graph
    plt.figure(figsize=(8, 6))
    plt.plot(external_factors, reactions, label='Alafia’s Reactions',
color='blue')
    plt.axhline(y=5, color='gray', linestyle='--', label='Baseline
(Neutral Reaction)')
    plt.axvline(x=7, color='red', linestyle='--', label='Critical
Point (High External Pressure)')

    # Labeling the graph
    plt.xlabel("External Factors (Pressure, Emotional Confrontation,
Career Influence)", fontsize=12)
    plt.ylabel("Reaction Intensity", fontsize=12)
    plt.title("Graph of Alafia's Reactions under Different
Parameters", fontsize=14)
    plt.legend()
    plt.show()

import matplotlib.pyplot as plt

def plot_trust_probability():
    # Probability conditions
    p_initial_mistrust = 0.8  # Initial mistrust (80% chance)
    p_self_preservation_barrier = 0.7  # Self-preservation causing
emotional distance
    p_earn_trust_action = 0.4  # Chance of Anindya doing something to
gain trust
    p_time_effect = 0.6  # Effect of time in healing
    p_communication = 0.5  # Open communication as a factor

    # Total probability formula
    p_gain_trust = (1 - p_initial_mistrust) * p_earn_trust_action *
p_time_effect * p_communication

    # Plotting the trust possibility graph
    conditions = ['Initial Mistrust', 'Self-Preservation Barrier',
'Earn Trust Action', 'Time Effect', 'Open Communication']
    probabilities = [p_initial_mistrust, p_self_preservation_barrier,
p_earn_trust_action, p_time_effect, p_communication]

    fig, ax = plt.subplots(figsize=(8, 6))

    # Plot probability as a bar chart
    ax.bar(conditions, probabilities, color='blue')
    ax.set_ylim([0, 1])
    ax.set_title("Probabilities Affecting Anindya's Chances of Gaining
Alafia's Trust", fontsize=14)
    ax.set_ylabel("Probability")

    # Rotate x-axis labels to 90 degrees
    plt.xticks(rotation=90)

    # Display total possibility result
    total_trust_chance = f"Total Trust Chance: {p_gain_trust:.2f}
(i.e., {p_gain_trust * 100:.1f}%)"
    plt.text(0.5, 0.9, total_trust_chance, fontsize=12, ha='center',
va='center', transform=ax.transAxes)

    plt.tight_layout()  # Adjust layout for rotated labels
    plt.show()

# Call the function to plot
# plot_trust_probability()

# Call the functions
plot_thought_network()  # Visualize the multidimensional thought
network
plot_reactions()        # Plot reactions under parameters
plot_trust_probability()# Plot the probability of regaining trust
```



Alafia's Multidimensional Thought Network



Graph of Alafia's Reactions under Different Parameters
Alafia’s Reactions
Baseline (Neutral Reaction)
Critical Point (High External Pressure)
Reaction Intensity
2
3
4
5
6
7
8
9
10
0
2
4
6
8
10
External Factors (Pressure, Emotional Confrontation, Career Influence)


Probabilities Affecting Anindya's Chances of Gaining Alafia's Trust
Total Trust Chance: 0.02 (i.e., 2.4%)
Probability
0.0
0.2
0.4
0.6
0.8
1.0
Initial Mistrust
Self-Preservation Barrier
Earn Trust Action
Time Effect
Open Communication