```python
# Importing necessary libraries

import numpy as np

import networkx as nx
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

import os
import subprocess
from diagrams import Diagram, Cluster, Edge
from diagrams.custom import Custom
from diagrams.onprem.client import User
from IPython.display import Image, display

# Function to visualize the multidimensional thought network with
weighted edges and manual positions
def plot_weighted_thought_network():
    # Create a directed graph
    G = nx.DiGraph()

    # Adding nodes for key tendencies
    tendencies = {
        "Future Aspirations": {"weight": 0.7},
        "Self-Preservation": {"weight": 0.9},
        "Control Over Emotions": {"weight": 0.8},
        "Emotional Detachment": {"weight": 0.9},
        "Avoidance of Conflict": {"weight": 0.8},
        "Coldness & Emotional Walls": {"weight": 0.9},
        "Rationalizing Emotional Distance": {"weight": 0.6},
        "Compartmentalization": {"weight": 0.7},
        "Selective Emotional Access": {"weight": 0.7},
        "Deliberate Ambiguity": {"weight": 0.5},
        "Playfulness or Manipulation": {"weight": 0.6},
        "Mild Guilt": {"weight": 0.4}
    }

    for node, attributes in tendencies.items():
        G.add_node(node, weight=attributes["weight"])

    # Adding weighted edges for context-specific behavior patterns
    G.add_weighted_edges_from([
        ("Future Aspirations", "Self-Preservation", 0.7),  # Strong
link
        ("Self-Preservation", "Control Over Emotions", 0.6),  #
Moderate link
        ("Self-Preservation", "Rationalizing Emotional Distance",
0.8),  # Strong link
        ("Control Over Emotions", "Selective Emotional Access", 0.6),
```

```
# Moderate
        ("Control Over Emotions", "Compartmentalization", 0.5),  #
Weaker
        ("Emotional Detachment", "Coldness & Emotional Walls", 0.9),
# Strong
        ("Emotional Detachment", "Selective Emotional Access", 0.7),
# Moderate
        ("Emotional Detachment", "Playfulness or Manipulation", 0.5),
# Moderate
        ("Avoidance of Conflict", "Deliberate Ambiguity", 0.8),  #
Strong
        ("Avoidance of Conflict", "Playfulness or Manipulation", 0.6),
# Moderate
        ("Rationalizing Emotional Distance", "Mild Guilt", 0.4),  #
Weaker link
        # Additional connections based on behavioral feedback loops
        ("Mild Guilt", "Rationalizing Emotional Distance", 0.3),  #
Mild feedback
        ("Compartmentalization", "Rationalizing Emotional Distance",
0.6),
        ("Playfulness or Manipulation", "Emotional Detachment", 0.5)
# Feedback loop
    ])

    # Manually defining node positions
    pos = {
        "Future Aspirations": (0, 2),
        "Self-Preservation": (1, 1.5),
        "Control Over Emotions": (2, 1),
        "Emotional Detachment": (2.5, 0),
        "Avoidance of Conflict": (0.5, 0),
        "Coldness & Emotional Walls": (2.5, -1),
        "Rationalizing Emotional Distance": (1, -0.5),
        "Compartmentalization": (1.5, -1),
        "Selective Emotional Access": (3, 1),
        "Deliberate Ambiguity": (0.5, -1),
        "Playfulness or Manipulation": (2, -1.5),
        "Mild Guilt": (1, -2)
    }

    # Draw the graph with manual positions
    plt.figure(figsize=(12, 10))

    # Extract edge weights for drawing the width of the edges
    edge_weights = nx.get_edge_attributes(G, 'weight')

    # Draw the nodes
    nx.draw_networkx_nodes(G, pos, node_color="lightblue",
node_size=3000)
```

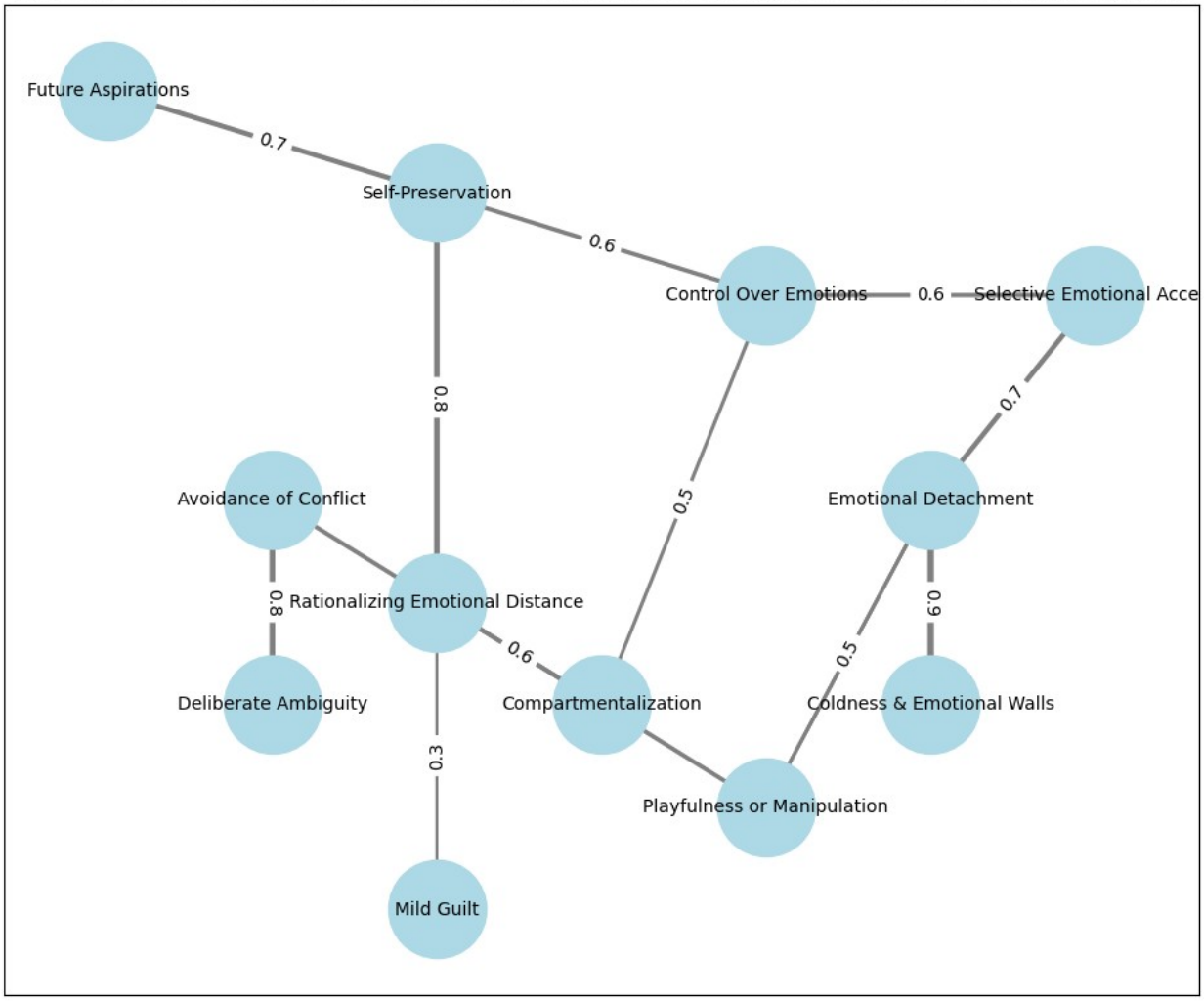
```
    # Draw the edges, with line width proportional to the edge weight
    nx.draw_networkx_edges(G, pos, width=[weight * 4 for weight in
edge_weights.values()], edge_color="gray", arrows=True)

    # Draw the labels
    nx.draw_networkx_labels(G, pos, font_size=10)

    # Display edge weights as labels on the graph
    nx.draw_networkx_edge_labels(G, pos, edge_labels={(u, v):
f'{d:.1f}' for u, v, d in G.edges(data="weight")})

    plt.title("Alafia's Multidimensional Thought Network with
Contextual Weightage (Manual Layout)", size=15)
    plt.show()


# Function to generate an expanded 3D plot for the complex surface
def plot_expanded_3d_surface():
    # Create a higher-resolution mesh grid for the external factors
(X) and reaction intensity (Y)
    external_factors = np.linspace(0, 15, 200)  # X-axis (External
Factors), expanded range and finer resolution
    reaction_intensity = np.linspace(0, 15, 200)  # Y-axis (Reaction
Intensity), expanded range and finer resolution
    X, Y = np.meshgrid(external_factors, reaction_intensity)

    # Define Z-axis: Emotional Justification Index (EJI) based on a
function of X and Y
    # Using a more dynamic relationship to simulate complex behavior
of EJI
    Z = np.sin(0.5 * X) * np.cos(0.5 * Y) + 0.3 * X - 0.2 * Y  # Z-
axis (Emotional Justification Index - EJI)

    # Create a 3D figure with a larger size
    fig = plt.figure(figsize=(12, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Plot the 3D surface with a more pronounced color spectrum based
on the Z values (EJI)
    surf = ax.plot_surface(X, Y, Z, cmap='plasma', edgecolor='none',
alpha=0.8)

    # Adding labels for the axes
    ax.set_xlabel('External Factors (X-axis)', fontsize=12)
    ax.set_ylabel('Reaction Intensity (Y-axis)', fontsize=12)
    ax.set_zlabel('Emotional Justification Index (EJI) (Z-axis)',
fontsize=12)

    # Adjusting the Z-axis limits to make the surface variations more
```

```
visible
    ax.set_zlim(np.min(Z), np.max(Z))

    # Adding a color bar to show the values of Emotional Justification
Index (EJI)
    fig.colorbar(surf, shrink=0.5, aspect=5, label='EJI (Emotional
Justification Index)', pad=0.1)

    # Set the title for the plot
    ax.set_title(' Emotional Justification Index vs. External Factors
and Reaction Intensity', fontsize=14)

    # Show the plot
    plt.show()

def plot_sequence_diagram():
    # Create a diagram
    with Diagram("Causes of Emotional Detachment", show=False,
filename="sequence_diagram"):  # Set show=False to prevent automatic
opening and specify filename
        user = User("Alafia")

        with Cluster("External Factors"):
            professional_pressure = Custom("Professional Pressure",
"https://path/to/your/icons/pressure.png")
            personal_conflict = Custom("Personal Conflict",
"https://path/to/your/icons/conflict.png")
            family_expectations = Custom("Family Expectations",
"https://path/to/your/icons/family.png")

        with Cluster("Internal Reactions"):
            self_preservation = Custom("Self-Preservation",
"https://path/to/your/icons/self_preservation.png")
            emotional_overload = Custom("Emotional Overload",
"https://path/to/your/icons/overload.png")
            rationalization = Custom("Rationalization",
"https://path/to/your/icons/rationalization.png")

        detachment = Custom("Increased Emotional Detachment",
"https://path/to/your/icons/detachment.png")

        user >> Edge(label="Triggers") >> professional_pressure
        user >> Edge(label="Triggers") >> personal_conflict
        user >> Edge(label="Triggers") >> family_expectations

        professional_pressure >> Edge(label="Influence") >>
self_preservation
        personal_conflict >> Edge(label="Influence") >>
emotional_overload
        family_expectations >> Edge(label="Influence") >>
```

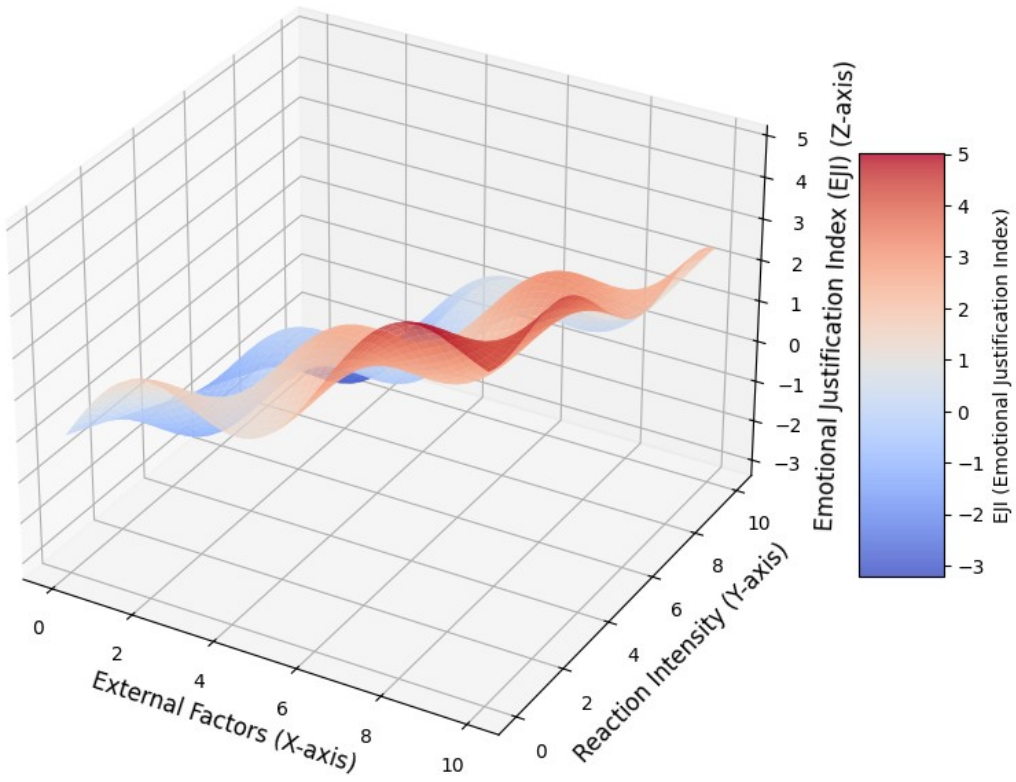
```
rationalization

        self_preservation >> Edge(label="Leads to") >> detachment
        emotional_overload >> Edge(label="Leads to") >> detachment
        rationalization >> Edge(label="Leads to") >> detachment

def plot_trust_probability(time_range, factor1_range, trust_function):
    # Create a dense mesh grid for time and factor1
    time = np.linspace(time_range[0], time_range[1], 500)  # Increase
data density for time
    factor1 = np.linspace(factor1_range[0], factor1_range[1], 500)  #
Increase data density for factor1

    # Create meshgrid for plotting
    T, F1 = np.meshgrid(time, factor1)

    # Compute the trust probability for each (time, factor1) point
    P = trust_function(T, F1)

    # Create a 3D plot
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Plot a surface with a color gradient based on the probability
values
    surface = ax.plot_surface(T, F1, P, cmap=cm.viridis,
edgecolor='none', rstride=1, cstride=1, alpha=0.8)

    # Add contour plots to emphasize the change in probability along
crucial factors
    ax.contour(T, F1, P, zdir='z', offset=np.min(P), cmap='viridis',
linestyles='solid', linewidths=0.5)
    ax.contour(T, F1, P, zdir='x', offset=np.min(time),
cmap='viridis', linestyles='solid', linewidths=0.5)
    ax.contour(T, F1, P, zdir='y', offset=np.min(factor1),
cmap='viridis', linestyles='solid', linewidths=0.5)

    # Add labels for clarity
    ax.set_xlabel('Time')
    ax.set_ylabel('Factor 1')
    ax.set_zlabel('Trust Probability')
    ax.set_title('Probabilities Affecting Chance of Anindya to Gain
Trust of Alafia\n' +
             'Time and Factor 1 (representing external influences,
personal attributes, or environmental factors)')

    # Add color bar to emphasize the change in probability
    fig.colorbar(surface, ax=ax, shrink=0.5, aspect=5)
```

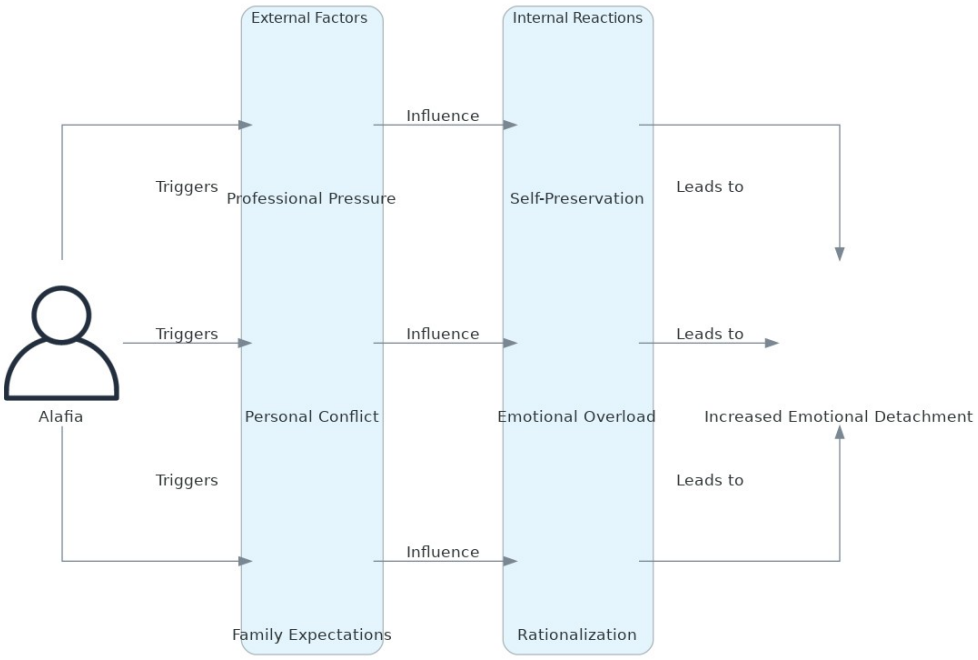
```
    # Show the plot
    plt.show()

# Example of trust function (user will replace this with the actual
function)
def trust_function(time, factor1):
    # A placeholder function for demonstration purposes
    return np.sin(time) * np.cos(factor1)

# Call the functions
plot_weighted_thought_network() #multidimensional thought network

plot_3d_surface()        # Plot reactions under parameters

plot_sequence_diagram() # Call the function to generate the diagram
display(Image(filename="sequence_diagram.png")) # Display the
generated image within the Jupyter Notebook

plot_trust_probability((0, 10), (0, 5), trust_function) # Call the
function with sample ranges
```

Alafia's Multidimensional Thought Network with Contextual Weightage (Manual Layout)



3D Surface Plot: Emotional Justification Index vs. External Factors and Reaction Intensity





Causes of Emotional Detachment

Probabilities Affecting Chance of Anindya to Gain Trust of Alafia
Time and Factor 1 (representing external influences, personal attributes, or environmental factors)