Sentiment Analysis Module

- To design the sentiment analysis model and natural language processing (NLP) model, you can use popular NLP libraries like NLTK (Natural Language Toolkit) and spaCy. Below is an example of how to design an NLP model in Python. This example provides a basic framework for implementing the functionality you described in your documentation:

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import spacy

class SentimentAnalysis:
    def __init__(self):
        # Initialize NLTK Sentiment Intensity Analyzer
        self.sid = SentimentIntensityAnalyzer()

        # Initialize spaCy NLP model
        self.nlp = spacy.load("en_core_web_sm")

    def is_obedient(self, text, name, age, gender):
```

```
        # Sentiment analysis for obedience
        # You can define your own rules for determining obedience based on sentiment scores
        sentiment_scores = self.sid.polarity_scores(text)
        obedient_score = sentiment_scores['compound']

        if obedient_score >= 0:
            return 'Y'  # Obedient
        else:
            return 'N'  # Not obedient

    def calculate_emotion_score(self, text):
        # Calculate emotion score
        sentiment_scores = self.sid.polarity_scores(text)
        emotion_score = sentiment_scores['compound']
        return emotion_score

    def calculate_consistency_score(self, text1, text2):
        # Calculate consistency score based on two sets of text
        doc1 = self.nlp(text1)
        doc2 = self.nlp(text2)

        # You can define your own logic to calculate consistency score
```

```
        # For example, you can measure the similarity between two text samples
        similarity = doc1.similarity(doc2)
        consistency_score = (similarity + 1) / 2  # Normalize to [0, 1]

        return consistency_score

    def calculate_confidence_score(self, emotion_score, consistency_score):
        # Calculate confidence score as an average of emotion and consistency scores
        confidence_score = (emotion_score + consistency_score) / 2
        return confidence_score

# Example usage:
sentiment_analyzer = SentimentAnalysis()

name = "John"
age = 30
gender = "Male"

question1 = "I am very obedient."
question2 = "I refuse to cooperate."
question3 = "I saw the crime happen."
```

```
obedient_result = sentiment_analyzer.is_obedient(question1, name, age, gender)

emotion_score = sentiment_analyzer.calculate_emotion_score(question2)

consistency_score = sentiment_analyzer.calculate_consistency_score(question2, question3)

confidence_score = sentiment_analyzer.calculate_confidence_score(emotion_score, consistency_score)


print(f"Is {name} obedient? {obedient_result}")

print(f"Emotion Score: {emotion_score}")

print(f"Consistency Score: {consistency_score}")

print(f"Confidence Score: {confidence_score}")
```

- In this code, we create a class SentimentAnalysis with methods to calculate obedience, emotion score, consistency score, and confidence score. The actual calculations for these scores are simple examples, and you may need to define more complex rules and logic based on your specific requirements.

- You can integrate this class into your existing code and use it to analyze suspect responses during the questioning process, as described in your documentation.

You can further enhance the NLP and sentiment analysis models based on the specific needs of your project.

User