Integrating the Pre-trained AI Model

1. Overview

The Negotiation Chatbot API uses a pre-trained AI model to handle negotiation conversations between a customer and a supplier. This API leverages FastAPI to provide endpoints for negotiating prices and uses a pre-trained model to simulate realistic negotiation behavior.

2. Components Used

- **FastAPI**: A modern, fast (high-performance), web framework for building APIs with Python 3.7+.
- **ngrok**: A tool to expose a local server to the internet.
- **Pre-trained AI Model**: For this example, we use a simulated negotiation model. In a real-world scenario, you would use a model like ChatGPT or similar.

3. Integration Details

1. Setting Up FastAPI

The API is built using FastAPI, which simplifies creating and managing HTTP endpoints. The FastAPI class is instantiated to set up the API, and route handlers are defined to manage different endpoints.

2. Implementing Negotiation Logic

The core of the negotiation logic is implemented in the /negotiate/ endpoint. Here's how it works:

```
@app.post("/negotiate/")

async def negotiate(customer_offer: float):

    # Define supplier's initial offer and price range

    supplier_initial_offer = 100.0

    minimum_price = 80.0



    # Implement basic negotiation logic

    if customer_offer >= supplier_initial_offer:

        return {"message": "Offer accepted!"}

    elif customer_offer >= minimum_price:
```

```
        counter_offer = (customer_offer + supplier_initial_offer) / 2

        return {"message": f"Counteroffer: {counter_offer}"}

    else:

        return {"message": "Offer rejected!"}
```

Explanation:

- **Initial Setup**: We define the supplier's initial offer and the minimum acceptable price.
- **Customer Offer**: The customer's offer is compared to the supplier's offer.
  - If the offer meets or exceeds the supplier's offer, it is accepted.
  - If the offer is above the minimum price but below the supplier's offer, a counteroffer is proposed.
  - If the offer is below the minimum acceptable price, it is rejected.

3. Using a Pre-trained AI Model

the negotiation logic is simplified and does not use a pre-trained AI model. However, integrating a model like ChatGPT would involve:

- **Model Integration**:
  - **API Access**: Use OpenAI's API to send the customer's offer to the model and receive a response. You would need an API key and set up the model for negotiation responses.
  - **Endpoint Logic**: Replace the simple logic with calls to the AI model for dynamic negotiation responses based on the customer's offer.

Example with a Pre-trained Model: If integrating ChatGPT, the API call might look like this:

```
import openai



openai.api_key = 'your_openai_api_key'



@app.post("/negotiate/")

async def negotiate(customer_offer: float):

    response = openai.Completion.create(

        engine="text-davinci-003",

        prompt=f"The customer offers {customer_offer}. What should be the supplier's response?",
```

```
    max_tokens=50
  )
  return {"message": response.choices[0].text.strip()}
```

4. Running the Application

- **Ngrok**: The application is exposed to the internet using ngrok, which creates a secure tunnel to the local server.

```
public_url = ngrok.connect(8000)
print(f"Public URL: {public_url}")
```

5. Testing the API

- **Endpoints**: Test the /negotiate/ endpoint using tools like Postman or cURL.
- **Responses**: Verify that the responses are as expected based on the customer's offer.

4. Summary

This documentation explains the setup and integration of a simple negotiation logic API using FastAPI and demonstrates how to integrate a pre-trained AI model if needed. It covers API creation, negotiation logic, model integration, and testing procedures.

Additional Notes

- **Security**: Ensure that API keys and sensitive information are kept secure and not exposed in public repositories.
- **Scalability**: For production use, consider implementing more robust negotiation logic and using advanced models with fine-tuning for better results.