**Negotiation Chatbot API Documentation**

**1. Overview**

The **Negotiation Chatbot** is an API designed to simulate a price negotiation between a customer and a supplier using AI-powered responses from OpenAI’s GPT (such as ChatGPT). It handles basic conversational flows, offers dynamic pricing logic, and can optionally adjust its negotiation strategy based on sentiment analysis.

This document explains how the chatbot API was developed, the technologies used, the integration of GPT for AI responses, and how to deploy and test the system.

**2. System Architecture**

The system consists of the following components:

* **FastAPI**: A modern web framework for building APIs in Python. It powers the backend and exposes the necessary endpoints.
* **OpenAI API**: Handles the conversational logic using pre-trained language models (e.g., GPT). This integration provides AI responses that make the negotiation process dynamic and human-like.
* **Pricing Logic**: Custom logic that decides whether the bot accepts, rejects, or makes a counteroffer based on the user's input.
* **Sentiment Analysis** (optional): Uses basic text analysis to determine user sentiment, adjusting the negotiation flow accordingly.

**3. Requirements**

* **Python 3.8+**
* **FastAPI** for API development
* **OpenAI API** for GPT integration
* **Uvicorn** for running FastAPI
* **Pydantic** for request/response validation
* **Requests** for making external API calls
* **dotenv** to manage environment variables
* **pytest** for testing the API (optional)

**4. Project Setup**

* **Step 1: Install Dependencies**

Install the required Python packages by running:

pip install -r requirements.txt

The requirements.txt should include:

fastapi

openai

uvicorn

python-dotenv

* **Step 2: Set Up OpenAI API Key**

We will need an API key from OpenAI to integrate GPT for conversational responses.

* Sign up for OpenAI and get your API key from the dashboard.
* Create a .env file in the project root directory and store the key like this:

makefile

OPENAI\_API\_KEY=our-openai-api-key

**5. Code Breakdown**

**main.py (Core Application)**

* This is the entry point of the FastAPI application.
* It defines the API endpoints, such as /start-negotiation and /offer-price.
* Routes handle incoming requests and call helper functions for GPT responses and pricing logic.

**Example Code:**

from fastapi import FastAPI

from dotenv import load\_dotenv

import os

from ai\_service import generate\_ai\_response

from pricing\_logic import evaluate\_offer

load\_dotenv()

app = FastAPI()

@app.post("/start-negotiation")

async def start\_negotiation():

return {"message": "Let's start negotiating a price."}

@app.post("/offer-price")

async def offer\_price(user\_offer: int):

bot\_response = evaluate\_offer(user\_offer)

ai\_message = generate\_ai\_response(bot\_response)

return {"message": ai\_message}

**ai\_service.py (AI Logic)**

* This module handles interaction with the OpenAI API.
* It sends the conversation context and gets a response from GPT to drive the conversation.

**Example Code:**

import openai

import os

# Get API key from environment

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

def generate\_ai\_response(bot\_message):

prompt = f"Customer: I'd like to negotiate a price.\nBot: {bot\_message}"

response = openai.Completion.create(

engine="gpt-3.5-turbo",

prompt=prompt,

max\_tokens=50

)

return response.choices[0].text.strip()

**pricing\_logic.py (Pricing Logic)**

* Contains the logic for the bot to decide whether to accept, reject, or counter the user’s offer.
* You can define specific rules for how the negotiation proceeds based on the customer's offer.

**Example Code:**

def evaluate\_offer(user\_offer):

# Define price boundaries

minimum\_price = 70

maximum\_price = 100

if user\_offer >= maximum\_price:

return f"Offer accepted at {user\_offer}."

elif user\_offer < minimum\_price:

return f"Offer rejected. The lowest acceptable price is {minimum\_price}."

else:

counteroffer = (user\_offer + maximum\_price) / 2

return f"Counteroffer: {counteroffer:.2f}"

**sentiment\_analysis.py**

* Handles basic sentiment analysis.
* Adjusts the bot’s negotiation strategy based on the user's tone (e.g., politeness).

**Example Code:**

from textblob import TextBlob

def analyze\_sentiment(user\_message):

blob = TextBlob(user\_message)

polarity = blob.sentiment.polarity

if polarity > 0.5:

return "You seem polite! I'll offer a better deal."

elif polarity < -0.5:

return "Please maintain a professional tone."

else:

return "Negotiation continues as usual."

**6. API Endpoints**

**/start-negotiation**

* **Method**: POST
* **Description**: Starts the negotiation process by sending an initial message.
* **Response**: { "message": "Let's start negotiating a price." }

**/offer-price**

* **Method**: POST
* **Description**: Submits the user’s price offer and receives the bot's response (acceptance, rejection, or counteroffer).
* **Request Body**:

{

"user\_offer": 80

}

* **Response**: A message indicating whether the offer was accepted, rejected, or a counteroffer was made.
* **Example Response**:

{

"message": "Counteroffer: 90.0"

}

**/offer-price (With Sentiment Analysis)**

* **Method**: POST
* **Description**: If sentiment analysis is enabled, this endpoint will also consider user tone.
* **Request Body**:

{

"user\_offer": 80,

"user\_message": "Can you please offer a better price?"

}

* **Response**: Depending on the sentiment of the message, the bot may offer a more favorable response.
* **Example Response**:

{

"message": "You seem polite! I'll offer a better deal."

}

**7. Testing the API**

* **Manual Testing**
* We can use **Postman** or **cURL** to test the endpoints.
* Example cURL command:

curl -X POST "http://127.0.0.1:8000/offer-price" -H "Content-Type: application/json" -d '{"user\_offer": 80}'

* **Automated Testing**
* Write tests using **pytest** to ensure the API functions as expected.
* Example test file (test\_main.py):

from fastapi.testclient import TestClient

from main import app

client = TestClient(app)

def test\_start\_negotiation():

response = client.post("/start-negotiation")

assert response.status\_code == 200

assert response.json() == {"message": "Let's start negotiating a price."}

def test\_offer\_price():

response = client.post("/offer-price", json={"user\_offer": 80})

assert response.status\_code == 200

assert "message" in response.json()

* Run the tests:

pytest

**8. Deployment**

* **Local Deployment**

To run the FastAPI server locally:

1. Activate virtual environment (if using):

source venv/bin/activate

1. Run the application:

uvicorn main:app --reload

1. The API will be available at http://127.0.0.1:8000.

* **Docker Deployment**

1. Create a Dockerfile:

dockerfile

FROM python:3.9

WORKDIR /usr/src/app

RUN pip install --no-cache-dir -r requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

1. Build the Docker image:

docker build -t negotiation-chatbot

1. Run the Docker container:

docker run -p 8000:8000 negotiation-chatbot

**9. Conclusion**

The Negotiation Chatbot API provides a framework to simulate price negotiations using AI models like GPT. With customizable pricing logic and optional sentiment analysis, it mimics human-like negotiation behaviour and adapts based on user input.

Further improvements could include advanced user authentication, more intricate pricing strategies, or even a user interface for ease of use.