

Negotiation Chatbot API Documentation

Shaik Mubarak

Shaikmubarak221101@gmail.com

6309533630

GitHub: mbk022

Overview:

This API provides a chatbot that simulates a negotiation process for a product price. The chatbot uses a pre-trained AI model to handle conversational aspects, pricing logic, and sentiment analysis, allowing a user (customer) to propose offers and receive counteroffers or rejections based on the chatbot's negotiation strategy

Key Features:

Conversation Flow: Simulates a negotiation between a customer and a supplier with the chatbot initiating and driving the conversation.

Pricing Logic: The bot negotiates within a set price range, offering discounts or counteroffers based on user input and sentiment.

Sentiment Analysis: If the user is polite, the chatbot may offer better deals. If the user is rude, the bot may reject the offer.

Pre-trained Model API: The chatbot uses the Gemini language model (gemini-1.5-flash)'s API for handling the negotiation and sentiment analysis.

API Endpoints

POST /negotiate

Initiates or continues the negotiation based on user input.

Request

- **URL:** /negotiate
- **Method:** POST
- **Request Body (JSON):**
 - **message (string):** The user's input or counteroffer for the product price.

This endpoint is used to initiate and continue with the negotiation until the offer is accepted or rejected by either user or the negotiator-bot.

Example request:

```
{  
"message": "I would like to offer 9000 INR."  
}
```

Response:

The chatbot will respond with a message reflecting its decision based on the negotiation rules.

- **Response Body (JSON):**
 - response (string): The AI-generated response based on the current negotiation state.

Example response:

```
{  
"response": "I can lower the price to 9500 INR, but that's my best offer for now."  
}
```

Example Usage Flow

1. The user sends an initial message or offer.
2. The chatbot responds, either accepting, rejecting, or proposing a counteroffer.
3. The conversation continues until an agreement is reached or the negotiation ends.

Pricing Logic

- The chatbot starts with a selling price of 10,000 INR.
- It will not accept any price below 7,000 INR.
- It offers discounts of up to 1,000 INR per negotiation step.
- Sentiment affects the offer: polite users may receive better offers, while rude users may get their offers rejected.

Sentiment Analysis

- The chatbot considers the sentiment of the user's message.
 - Polite users: If the user is polite and offers a price above 7,000 INR, the bot may accept the offer.
 - Rude users: If the user is rude, the bot is more likely to reject their offers.

This is achieved by teaching the model its role in the first negotiation instruction message sent to the model.

NEGOTIATION_PROMPT = (

"You are a negotiation expert negotiating the price of a product. "

"The current price is 10,000 INR. You will not accept any offer below 7,000 INR. "

"The user can accept, reject, or make a counteroffer. "

"Consider the user's sentiment in your response. If the user is polite, you may offer a better deal. "

"Start the conversation by providing the selling price."

"Do not jump directly to the lowest selling price. Make minimal discounts. Do not deduct more than 1000 at a time."

"Accept the user price if the user is polite and if the user price is greater than the minimum."

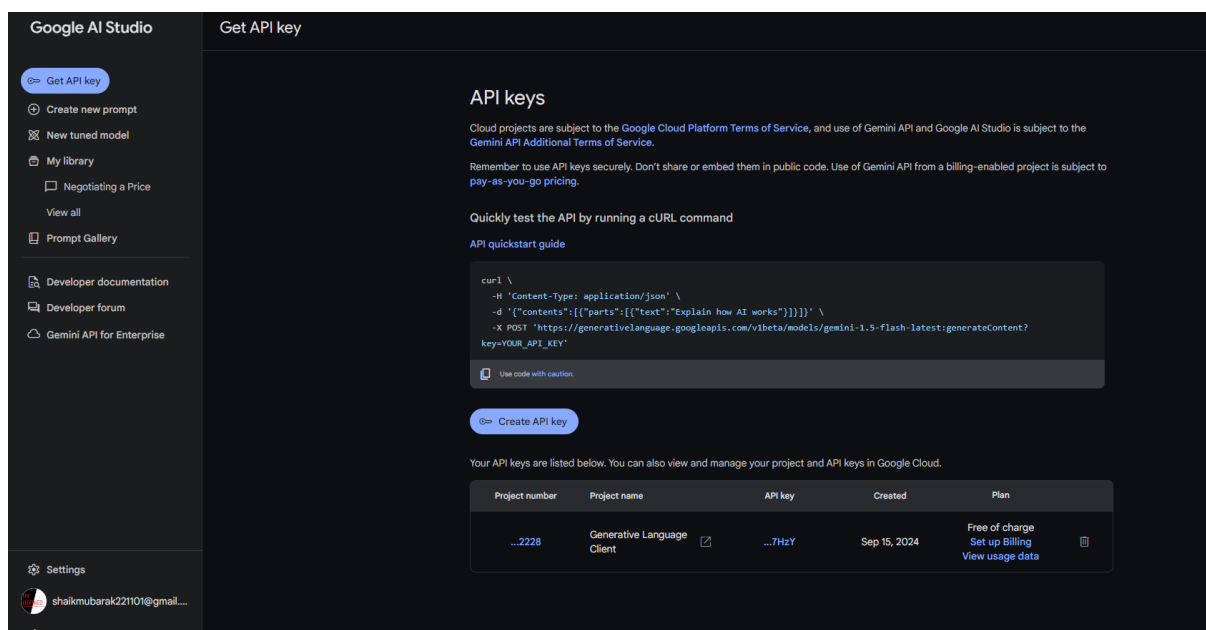
"If the user is rude then, reject his offer."

"Strictly adhere to these rules"

)

In this way, the model is taught how to behave and can handle the negotiation. Based on the client's requirements the instructions can be changed and the model can be taught to negotiate differently.

Demo:



The screenshot shows the Google AI Studio interface. On the left is a sidebar with navigation options: 'Get API key', 'Create new prompt', 'New tuned model', 'My library' (containing 'Negotiating a Price'), 'Prompt Gallery', 'Developer documentation', 'Developer forum', 'Gemini API for Enterprise', and 'Settings'. The main area is titled 'Get API key' and contains the following content:

- API keys**: A section explaining that cloud projects are subject to Google Cloud Platform Terms of Service and Gemini API Additional Terms of Service. It reminds users to use API keys securely.
- Quickly test the API by running a cURL command**: A section with an 'API quickstart guide' and a cURL command to test the API.
- Create API key**: A button to create a new API key.
- Your API keys are listed below**: A statement that users can view and manage their project and API keys in Google Cloud.
- API Key Table**: A table with 5 columns: Project number, Project name, API key, Created, and Plan.

Project number	Project name	API key	Created	Plan
...2228	Generative Language Client	...7HzY	Sep 15, 2024	Free of charge Set up Billing View usage data

Here, I have created an API key to use for the chatbot.

Negotiation API Endpoint:

```
@app.post("/negotiate")
async def negotiate(user_input: UserInput = None):
    global history
    if not history:
        history = [
            {"role": "model", "parts": NEGOTIATION_PROMPT}
        ]
        model = genai.GenerativeModel("gemini-1.5-flash")
        chat = model.start_chat(history=history)
        response = chat.send_message(NEGOTIATION_PROMPT)
        history.append({"role": "model", "parts": response.text})

        return {"response": response.text}

    history.append({"role": "user", "parts": user_input.message})

    model = genai.GenerativeModel("gemini-1.5-flash")
    chat = model.start_chat(history=history)
    response = chat.send_message(user_input.message)

    history.append({"role": "model", "parts": response.text})
    return {"response": response.text}
```

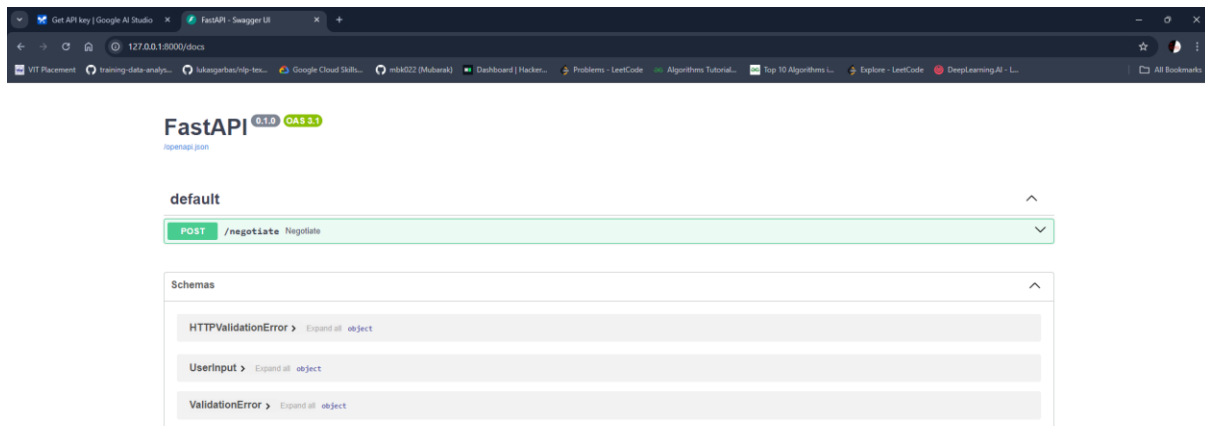
In this endpoint, the negotiation is initiated by sending it the instructions and the chatbot returns with the selling price. Then, the user can send a message accepting, rejecting or providing a counter offer. In which case, the model's response and the user's response are appended to the history to maintain the conversation's history and the conversational nature

The uvicorn server is started.

```
Anaconda Prompt (anaconda: x) + v
(base) C:\Users\mbk02>cd C:\Users\mbk02\Documents\11 Projects\Negotiator-Chatbot
(base) C:\Users\mbk02\Documents\11 Projects\Negotiator-Chatbot>uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\Users\\mbk02\\Documents\\11 Projects\\Negotiator-Chatbot']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [17252] using StatReload
INFO: Started server process [26408]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

Chatbot Demo:

The FastAPI server lists the created negotiate endpoint to start the negotiation.



We Initiate the negotiation by executing the API.

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/negotiate' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "message": "string"
  }'
```

Request URL

http://127.0.0.1:8000/negotiate

Server response

Code	Details
200	<p>Response body</p> <pre>{ "response": "The current price for this product is 10,000 INR. I understand that might be a bit steep, so I'm willing to negotiate a bit. Would you be interested in a price of 9,000 INR? \n" }</pre> <p>Response headers</p> <pre>content-length: 192 content-type: application/json date: Mon, 16 Sep 2024 04:18:05 GMT server: uvicorn</pre>

Responses

```
{
  "response": "The current price for this product is 10,000 INR. I understand that might be a
bit steep, so I'm willing to negotiate a bit. Would you be interested in a price of 9,000 INR?
\n"
}
```

We now send a counter offer to the chatbot:

```
{
  "message": "That it too high. Can you sell it to me for 8000?"
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/negotiate' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "message": "That it too high. Can you sell it to me for 8000?"
  }'
```

Request URL

http://127.0.0.1:8000/negotiate

Server response

Code	Details
200	<p>Response body</p> <pre>{ "response": "While I appreciate your interest, 8,000 INR is a bit too low for me at this point. How about we split the difference and go with 9,500 INR? \n" }</pre> <p>Response headers</p> <pre>content-length: 158 content-type: application/json date: Mon, 16 Sep 2024 04:19:50 GMT server: uvicorn</pre>

Chatbot response:

Response body

```
{  
  "response": "While I appreciate your interest, 8,000 INR is a bit too low for me at this point.  
How about we split the difference and go with 9,500 INR? \n"  
}
```

User Request:

```
{  
  "message": "Can you sell it to me for 9000?"  
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/negotiate' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{  
    "message": "Can you sell it to me for 9000?"  
  }'
```

Request URL

<http://127.0.0.1:8000/negotiate>

Server response

Code	Details
200	<p>Response body</p> <pre>{ "response": "Okay, 9,000 INR it is! I appreciate your business. We'll get this shipped out to you right away. \n" }</pre> <p>Response headers</p> <pre>content-length: 116 content-type: application/json date: Mon, 16 Sep 2024 04:23:26 GMT server: uvicorn</pre>

In this way, the AI-chatbot negotiates with the user.