

Chatbot App GROQ dan Postgresql

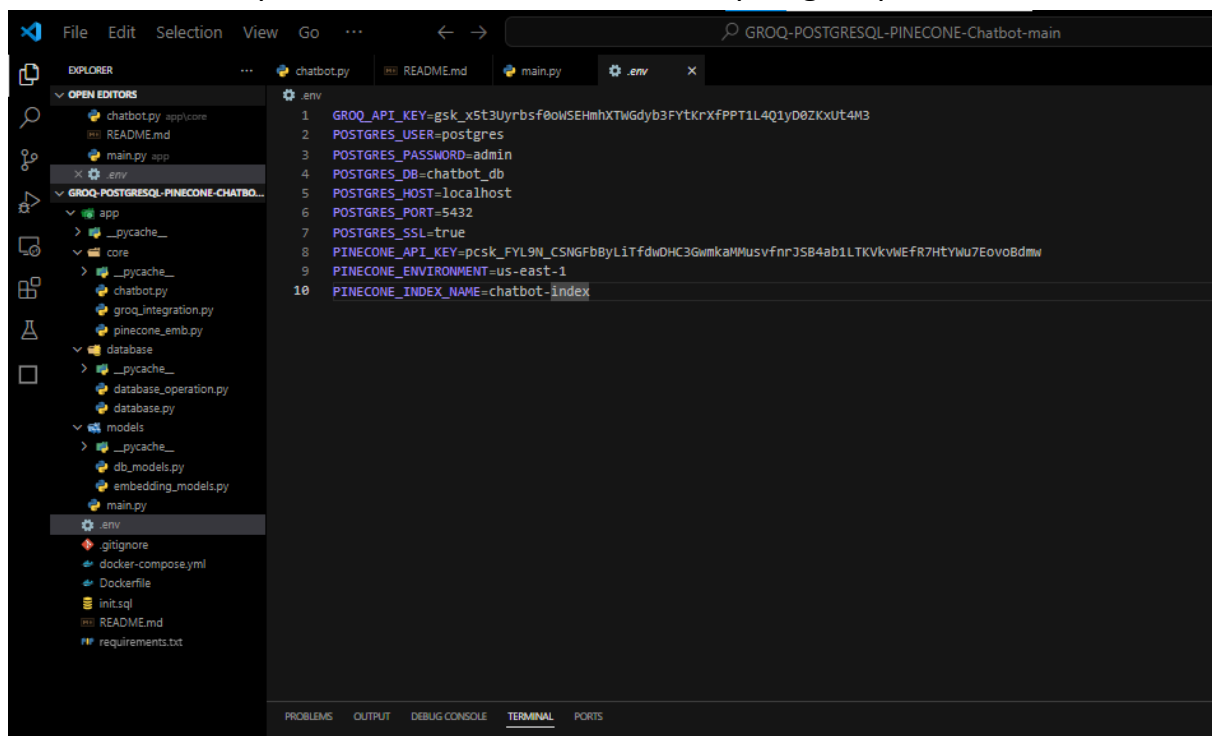
1. Setup Instalasi Python 3.13.2 dan Postgresql 17.4 di lokal.

```
C:\Users\ASUS>python --version
Python 3.13.2

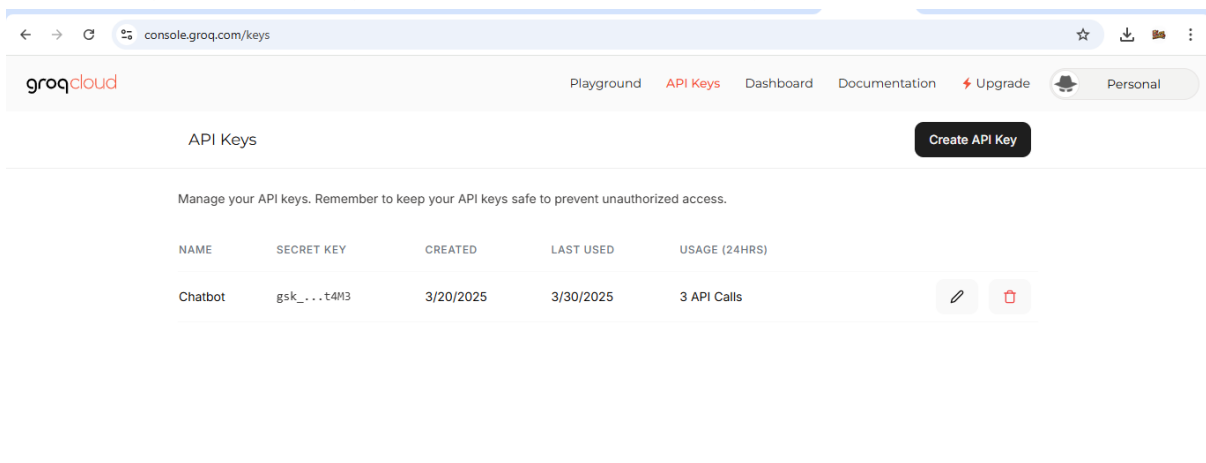
C:\Users\ASUS>postgresql --version
'postgresql' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\ASUS>psql --version
psql (PostgreSQL) 17.4
```

2. Tambahkan PGVECTOR pada postgresql di local ikuti langkah langkah sesuai OS. <https://github.com/pgvector/pgvector?tab=readme-ov-file> . Untuk windows bisa menggunakan visual studio lalu menginstall desktop development with c++
3. Pada local terminal directory aplikasi dapat menginstall dependency. pip install -r requirements.txt
4. Pada file .env dapat disesuaikan API GROQ dan postgresql



5. Peroleh api GROQ dengan daftar dan membuat api



6. Penambahan model atau ganti model bisa dilihat pada halaman GROQ. Dan mengubah model

The screenshot shows the Groq Cloud dashboard at the URL `console.groq.com`. The "Limits" tab is selected in the left sidebar. The main content area is titled "Limits" and states: "These are the rate limits for your organization". Below this, a section titled "Chat Completion" contains a table with the following data:

ID	REQUESTS PER MINUTE	REQUESTS PER DAY	TOKENS PER MINUTE	TOKENS PER DAY
allam-2-7b	30	7,000	6,000	(No limit)
deepseek-r1-distill-llama-70b	30	1,000	6,000	(No limit)
deepseek-r1-distill-qwen-32b	30	1,000	6,000	(No limit)
gemma2-9b-it	30	14,400	15,000	500,000
llama-3.1-8b-instant	30	14,400	6,000	500,000
llama-3.2-11b-vision-preview	30	7,000	7,000	500,000
llama-3.2-1b-preview	30	7,000	7,000	500,000
llama-3.2-3b-preview	30	7,000	7,000	500,000
llama-3.2-90b-vision-preview	15	3,500	7,000	250,000

```

1  from typing import Optional, List
2  from app.models.embedding_models import EmbeddingModel
3  from app.database.database_operation import DatabaseOperations
4  from sqlalchemy.ext.asyncio import AsyncSession
5  from app.core.groq_integration import GroqIntegration
6
7  async def get_chatbot_response(
8      user_input: str,
9      db_session: AsyncSession,
10     threshold: float = 0.8,
11     primary_groq_model: str = "llama3-70b-8192",
12     fallback_models: List[str] = ["deepseek-r1-distill-llama-70b", "mistral-saba-24b"]
13 ) -> str:
14     try:
15         embedding_model = EmbeddingModel()
16         embedding = embedding_model.get_embedding(user_input)
17
18         results = await DatabaseOperations.vector_search(
19             session=db_session,
20             embedding_vector=embedding,
21             threshold=threshold,
22             limit=1
23         )
24
25         if results:
26             return results[0].answer
27
28     except Exception as e:
29         pass # Akan dilanjutkan ke fallback Groq

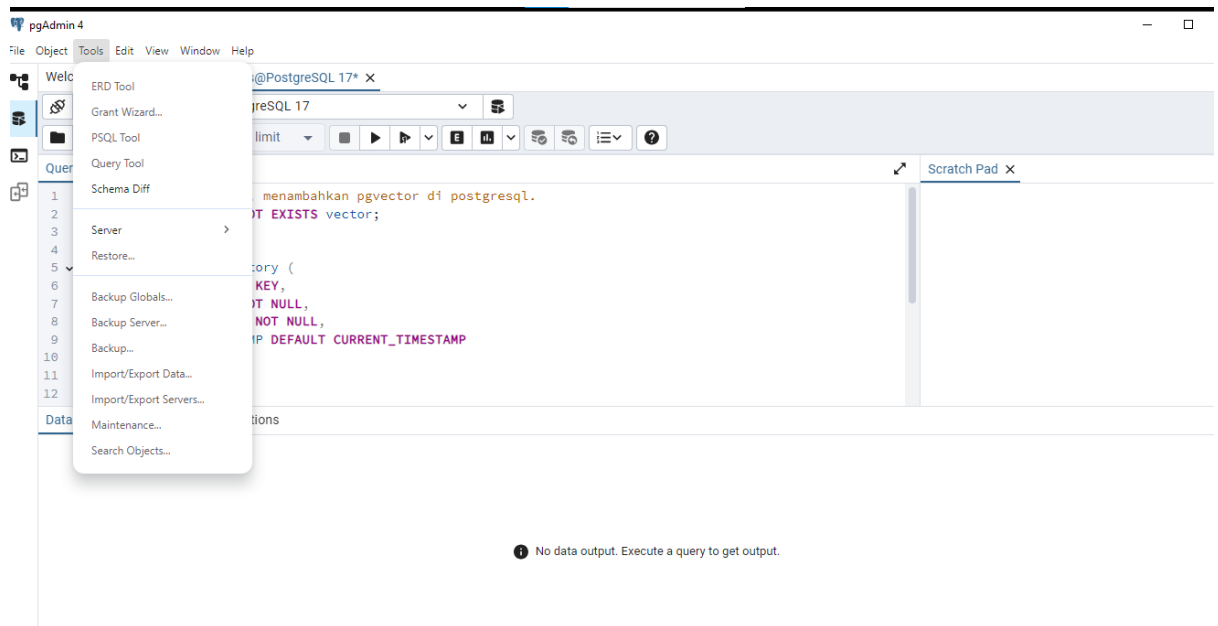
```

7. Buka aplikasi pgAdmin lalu disesuaikan data postgres di .env .
8. Jalankan aplikasi lokal dengan. `uvicorn app.main:app --host 0.0.0.0 --port 8000`
9. Table seharusnya akan langsung ditambahkan jika tidak ada di postgresql. Tetapi jika tidak bekerja dapat menambahkan secara manual di pgadmin dengan copy paste query di init.sql

```

1  -- EXTENSION PGVECTOR, menambahkan pgvector di postgresql.
2  CREATE EXTENSION IF NOT EXISTS vector;
3
4  -- Tabel chat_history
5  CREATE TABLE chat_history (
6      id SERIAL PRIMARY KEY,
7      user_input TEXT NOT NULL,
8      bot_response TEXT NOT NULL,
9      timestamp TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
10 );
11
12 -- Tabel chatbot_data
13 CREATE TABLE chatbot_data (
14     id SERIAL PRIMARY KEY,
15     question TEXT NOT NULL,
16     answer TEXT NOT NULL,
17     embedding_vector VECTOR(1024), -- Sesuai dengan pgvector
18     additional_info JSONB DEFAULT '{}',
19     timestamp TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
20 );
21
22 CREATE INDEX ON chatbot_data USING ivfflat (embedding_vector vector_cosine_ops) WITH (lists = 100);

```



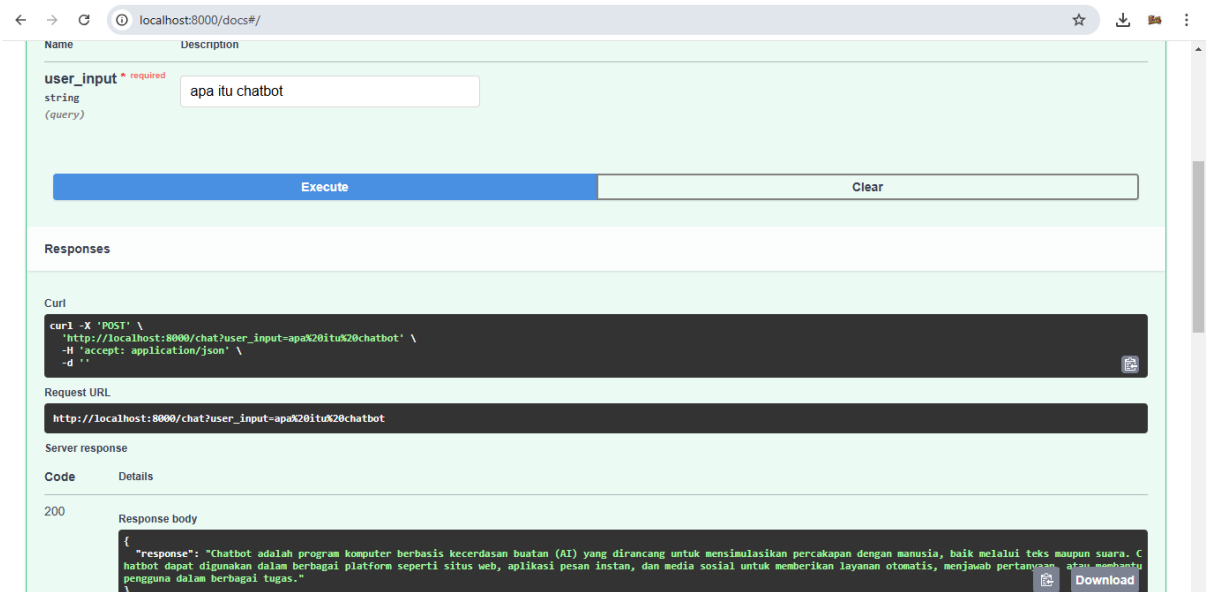
10. Penambahan data dapat dilakukan di localhost:8000/docs. Input question dan answer. Question akan diolah menjadi vector dan disimpan di db dengan dimensi 1024

The screenshot displays two applications side-by-side. The top application is a web browser at `localhost:8000/docs#/default/add_chatbot_data_add_chatbot_data_post`, showing a REST client interface for a POST endpoint `/add-chatbot-data`. The parameters section includes two required string fields: `question` and `answer`. The request body is set to `application/json` with an empty object `{}`.

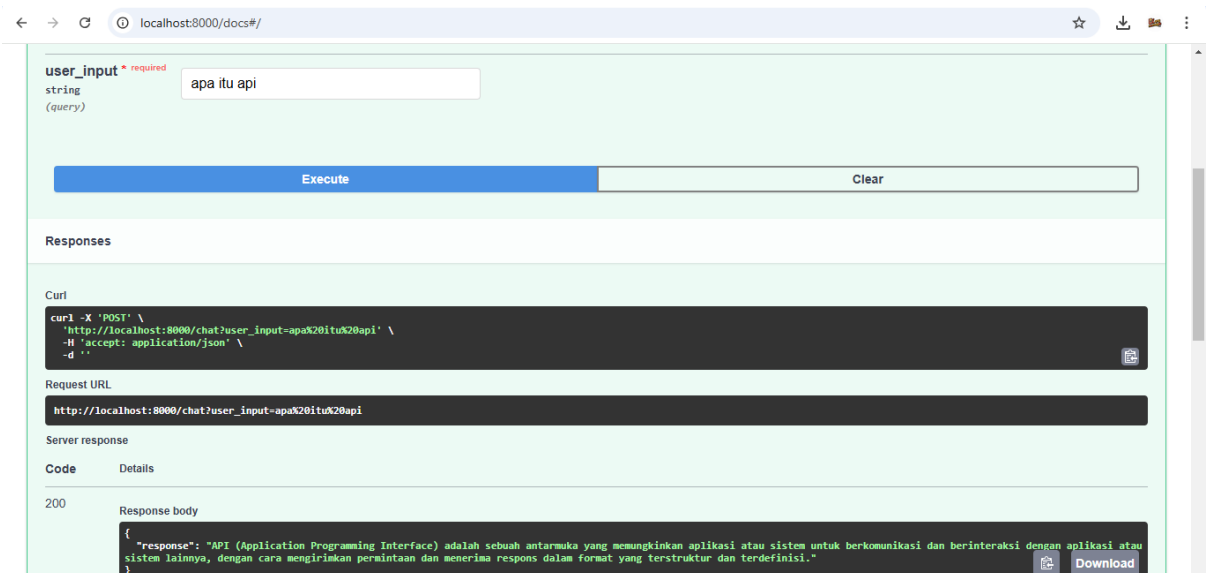
The bottom application is a database client connected to `chatbot_db/postgres@PostgreSQL 17`. It shows a query `SELECT * from chatbot_data` with the following results:

		embedding_vector vector
1	ngenal pola, pengambilan keputusan, pembelajaran dari data, serta pemecahan masalah.	[-0.0015545805,0.0030808263,-0.031332176,-0.063705064,0.031559683,0.046281572,0.022666425,0.081
2	in media sosial untuk memberikan layanan otomatis, menjawab pertanyaan, atau membantu pengguna dalam berbagai tugas.	[0.013759123,-0.016922615,-0.0023765352,-0.0764926,-0.025923481,0.10228775,-0.0012281394,0.00638

11. Testing dapat dilakukan di route `/chat`. Jika relevansi mencapai `0.8 / 80%` dia akan menggunakan data di database. Jika tidak menggunakan interface model dari GROQ

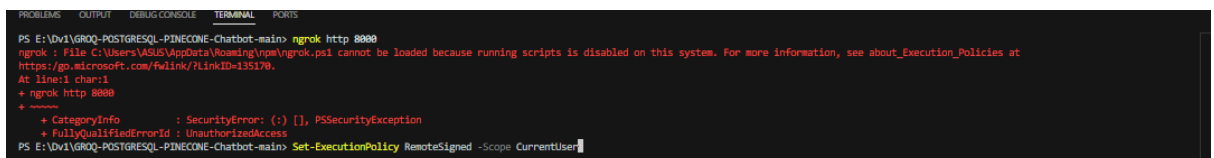


Data di db

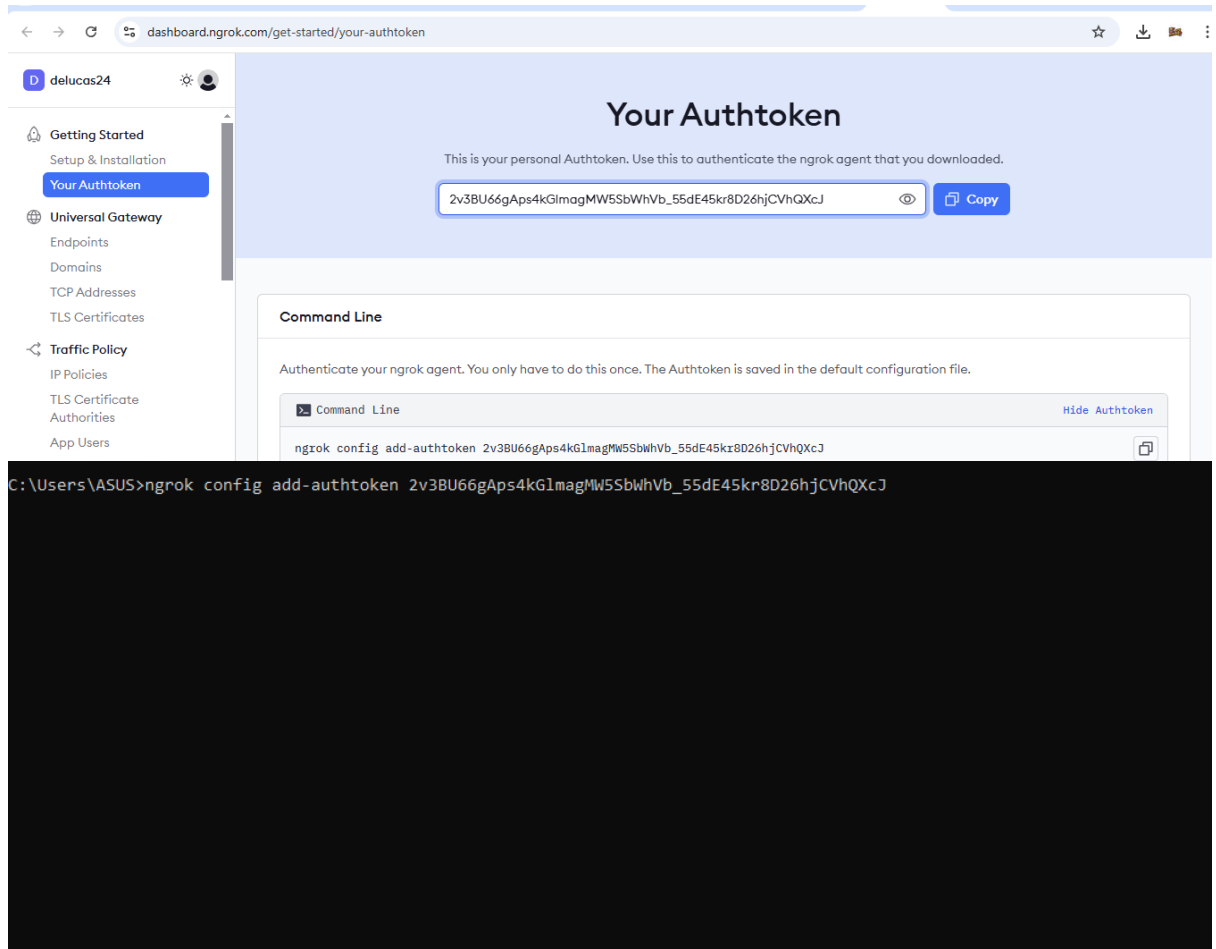


Data di GROQ

12. Testing aplikasi sebagai api yang akan menerima input dan mengirim output dari aplikasi ke 2. Menggunakan NGROK untuk test api agar dapat diakses secara online



Set-ExecutionPolicy RemoteSigned -Scope CurrentUser



The screenshot shows the Ngrok dashboard at `dashboard.ngrok.com/get-started/your-authtoken`. The user is `delucas24`. The main heading is **Your Authtoken**. Below it, a message states: "This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded." The authtoken is displayed in a box: `2v3BU66gAps4kGlmagMW5SbWhVb_55dE45kr8D26hjCVhQXcJ`, with a `Copy` button next to it.

On the left sidebar, the navigation menu includes:

- Getting Started
 - Setup & Installation
 - Your Authtoken**
- Universal Gateway
 - Endpoints
 - Domains
 - TCP Addresses
 - TLS Certificates
- Traffic Policy
 - IP Policies
 - TLS Certificate
 - Authorities
 - App Users

Below the authtoken, there is a **Command Line** section. It contains the text: "Authenticate your ngrok agent. You only have to do this once. The Authtoken is saved in the default configuration file." Below this text is a terminal window titled `Command Line` with the command: `ngrok config add-authtoken 2v3BU66gAps4kGlmagMW5SbWhVb_55dE45kr8D26hjCVhQXcJ`. A `Hide Authtoken` link is visible in the top right of the terminal window.

At the bottom of the image, a terminal window shows the command being executed in a Windows command prompt:

```
C:\Users\ASUS>ngrok config add-authtoken 2v3BU66gAps4kGlmagMW5SbWhVb_55dE45kr8D26hjCVhQXcJ
```

```
Command Prompt - ngrok http 8000

ngrok
(Ctrl+C to quit)

In London? Let's hang out. https://ngrok.com/info/kubecon-2025-ngrok-user-meetup

Session Status      online
Account             delucas24 (Plan: Free)
Version             3.22.0
Region              Asia Pacific (ap)
Latency              56ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://7c08-114-4-213-10.ngrok-free.app -> http://localhost:8000

Connections          ttl    opn    rt1    rt5    p50    p90
                    1      0      0.00   0.00   23.20  23.20

HTTP Requests
-----
16:17:37.668 +07 OPTIONS /api/domain-chat      200 OK
16:17:37.815 +07 POST   /api/domain-chat      200 OK
```

```
# CORS Configuration
app.add_middleware(
  CorsMiddleware,
  allow_origins=[
    "https://react-chat-one-livid.vercel.app",
    "https://7c08-114-4-213-10.ngrok-free.app",
    "http://localhost:3000"
  ],
  allow_credentials=True,
  allow_methods=["*"],
  allow_headers=["*"],
  expose_headers=["*"]
)
```

```
Chat-Frontend-Sample / src / App.tsx ↑ Top

Code Blame 150 lines (126 loc) · 3.1 KB Code 55% fast Raw Copy Download Edit Search

8   const App: React.FC = () => {
14   const handleSubmit = async (e: React.FormEvent) => {
15     // Prevent default behavior
16     if (!input.trim()) return;
17
18     setLoading(true);
19     setError(null);
20
21     try {
22       const response = await fetch(
23         "https://7c08-114-4-213-10.ngrok-free.app/api/domain-chat",
24         {
25           method: "POST",
26           headers: {
```

Testing ngrok hanya berlaku setiap 2 jam, domain akan berubah setiap 2 jam

Result api yang dijalankan di front end.

