

**LAPORAN LEAP
Industrial Experience**

Eksperimentasi dan
pengembangan Chatbot
Generative Ai

PTIK Petra



Oleh:
Calvin Laguna (C14220261)

**PROGRAM STUDI
INFORMATIKA FAKULTAS
TEKNOLOGI INDUSTRI
UNIVERSITAS KRISTEN PETRA
SURABAYA
2025**

DAFTAR ISI

DAFTAR ISI	ii
Halaman Pengesahany	
1. PENDAHULUAN	iv
1.1 Latar Belakang	5
1.2 Tujuan	5
1.3 Ruang Lingkup	5
1.4 Rencana Kegiatan	6
2. PENGENALAN TERHADAP MITRA	7
2.1 Profil Mitra	7
2.2 Proyek Keseluruhan dari yang Dikerjakan	8
2.2.1 Mengembangkan chatbot untuk Kantin Petra	8
2.2.2 Mengimplementasikan tools calling di chatbot	8
2.2.3 Membuat MCP server untuk memindahkan file dan folder serta membaca isi file Tersebut.....	8
2.2.4 Membuat chatbot menggunakan n8n	8
2.3 Proyek yang sedang Dikerjakan	9
2.3.1 Mengembangkan chatbot untuk Kantin Petra	9
2.3.2 Mengimplementasikan tools calling di chatbot	9
2.3.3 Membuat MCP server untuk memindahkan file dan folder serta membaca isi file Tersebut.....	9
2.3.4 Membuat chatbot menggunakan n8n	9
3. LAPORAN KEGIATAN LEAP	10
3.1 Landasan Teori	10
1. Ollama	10
2. Qdrant	10
3. Tools Calling	11
4. MCP (Model Context Protocol)	11
5. n8n	11
3.2 Kegiatan Bimbingan Selama LEAP	12
3.3 Analisis Kebutuhan	12

3.3.1 Mengembangkan chatbot untuk Kantin Petra	12
3.3.2 Mengimplementasikan tools calling di chatbot	12
3.3.3 Membuat MCP server untuk memindahkan file dan folder serta membaca isi file Tersebut.....	12
3.3.4 Membuat chatbot menggunakan n8n	13
3.4 Desain	13
3.4.1 Chatbot untuk Kantin Petra	13
3.4.2 Job Finder menggunakan Tools Calling	14
3.5 Implementasi	14
3.5.1 Chatbot untuk Kantin Petra	14
3.5.2 Job Search Chatbot	29
3.5.3 MCP Server	35
3.5.4 N8N Webscraping dan PDF Summary Telegram	49
4. KESIMPULAN	67
5. REFLEKSI DIRI	68
5.1 Analisa SWOT (Strength, Weakness, Opportunity, Threat)	68
5.1.1 Strength	68
5.1.2 Weakness	68
5.1.3 Opportunity	68
5.1.4 Threat	69
5.2 Analisa Soft Skill	69
5.2.1 Kemauan belajar	69
5.2.2 Manajemen waktu	69
5.3 Pengembangan Diri	69
5.3.1 Manajemen waktu	69
5.3.2 Kepercayaan diri.....	69
DAFTAR PUSTAKA.....	70
LAMPIRAN	71
Lampiran 1 Log Book Aktivitas Selama LEAP	71
Lampiran 2 Kegiatan Bimbingan Dosen Selama LEAP	74

HALAMAN PENGESAHAN

Judul LEAP : Membuat Chatbot Pesanan Makanan PTIK PETRA
NAMA : Calvin Laguna C14220261
Bidang : Informatika (AI)
Tempat LEAP : PTIK PETRA
Alamat : Jl. Siwalankerto No. 121-131
Tanggal Pelaksanaan : 15 Januari

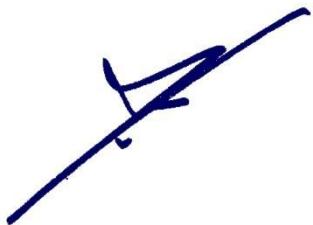
Mengusulkan,

(Calvin Laguna – C14220261)

Menyetuji,

Dosen Pembimbing LEAP

Pembimbing dari Mitra



Ir. Resmana Lim, M.Eng.
Nama Dosen Pembimbing

Nama Pembimbing Mitra

Mengetahui,
Koordinator Magang

“nama sesuai koordinator jalur LEAP”

1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi kecerdasan buatan Artificial Intelligence telah membawa transformasi signifikan dalam berbagai sektor. Salah satu inovasi yang semakin populer adalah penggunaan chatbot berbasis AI untuk meningkatkan efisiensi dan pengalaman pengguna.

1.2 Tujuan

Internship ini akan memberikan kesempatan untuk memperdalam kemampuan dalam bahasa pemrograman Python dan penerapannya dalam proyek nyata, seperti pengembangan ChatBot dengan menerapkan konsep-konsep Python yang diharapkan dapat menghubungkan teori dengan praktik yang efektif. Selain itu, pengalaman ini akan membantu dalam memahami penerapan berbagai model seperti gemini dan ollama. Selain itu, internship ini juga menawarkan kesempatan untuk bekerja dengan mesin GPU dan N8N dalam komputasi AI. Dengan mengaplikasikan teknik-teknik analisis masalah dan solusi teknis dalam proyek ini, diharapkan dapat meningkatkan keterampilan dalam perkembangan chatbot, pengalaman ini akan memperkaya pengetahuan teknis dan praktis, serta mempersiapkan untuk tantangan di dunia profesional teknologi.

1.3 Ruang Lingkup

1. Pembuatan chatbot untuk kantin
2. Penggunaan tools calling dalam chatbot
3. Pencobaan penggunaan MCP server
4. Penggunaan N8N dalam membuat chatbot dan webscraping

1.4 Rencana Kegiatan

Tabel Rencana Kegiatan

Minggu Ke-	Kegiatan
1	Pembelajaran Coding yang diberikan tentang menu kantin
2	Mengcoding untuk menampilkan image di chatbot
3	Mengcoding untuk menyimpan data ke quadrant
4	Mengcoding mengoptimize chatbot
5	Mengcoding multiagent
6	Memasukan llm Gemini
7	Mengcoding login dan mencoba menggunakan sql
8	Mempelajari tools calling
9	Pindah proyek ke petra carreer
10	Mengcoding pencarian job menggunakan tools calling
11	Mempelajari MCP (Model Context Protocol)
12	Membuat MCP Server
13	Membuat Chatbot untuk MCP ngerun server
14	Mempelajari menggunakan n8n
15	Membuat chatbot sederhana menggunakan n8n
16	Membuat chatbot Telegram untuk summarize isi file pdf
17	Membuat web scraping
18	Presentasi akhir

2. PENGENALAN TERHADAP MITRA

2.1 Profil Mitra

Pusat Teknologi Informasi dan Komunikasi adalah unsur penunjang akademik, yang merupakan perangkat kelengkapan di bidang pendidikan, penelitian, dan pengabdian masyarakat (yang berada di luar fakultas dan jurusan), khususnya yang berhubungan dengan teknologi informasi.

Visi Program Studi Informatika

Menjadi Program Studi Informatika yang peduli dan mempunyai reputasi nasional dan internasional dalam pengajaran, penelitian, dan pengabdian masyarakat yang berkomitmen pada nilai-nilai Kristiani.

Misi Program Studi Informatika

Menghasilkan tenaga profesional yang handal, mampu mengakomodasi perkembangan teknologi informasi di masa yang akan datang, dan berkomitmen pada nilai-nilai Kristiani, melalui:

1. Sistem pendidikan yang berkualitas dan berfokus pada minat mahasiswa dan permintaan pasar global;
2. Penelitian dan publikasi ilmiah sebagai dasar pengembangan keilmuan sesuai dengan perkembangan teknologi dan kebutuhan dunia industri;
3. Partisipasi aktif dalam memecahkan permasalahan yang dihadapi masyarakat melalui pengabdian kepada masyarakat;
4. Penyusunan dan pelaksanaan program yang efektif, efisien, dan berkesinambungan mengacu pada kebutuhan stakeholder.

2.2 Projek Keseluruhan dari yang Dikerjakan

Selama Magang di PTIK UK Petra berfokus pada percobaan dan pengembangan chatbot menggunakan tools-tools ai yang tersedia secara opensource.

2.2.1 mengembangkan chatbot untuk Kantin Petra

Proyek Kantin Petra adalah chatbot Dimana pelanggan dapat memesan makanan atau minuman. Pelanggan mengetik nama makanan/minuman yang mereka inginkan maka akan muncul detail tentang makanan tersebut seperti gambar,harga,stall dan kantin.

2.2.2 mengimplementasikan tools calling di chatbot

Proyek ini untuk mencari pekerjaan di website jobstreet menggunakan chatbot. Dimana user dapat mencari pekerjaan berdasarkan input yang mereka berikan.

2.2.3 Membuat mcp server untuk memindahkan file dan folder serta membaca isi file tersebut

Dalam proyek ini MCP server bertugas supaya bisa di jalankan ke dalam chatbot yang sudah jadi seperti claude dan cursor. Dengan claude dan cursor MCP server bisa di run dan menjalankan program tersebut.

2.2.4 Membuat chatbot menggunakan n8n

Dalam proyek ini untuk pdf summarization Dimana user dapat mengirim file pdf ke telegram dan chatbot akan mengirim kembali hasil ringkasan dari isi file pdf tersebut. Selain itu juga saya mencoba webscrapping untuk mengumpulkan data-data dalam website

2.3 Projek yang sedang Dikerjakan

2.3.1 mengembangkan chatbot untuk Kantin Petra

Proyek Kantin Petra menggunakan integrasi ollama, yang merupakan model bahasa besar (LLM) yang bertugas untuk memproses dan memahami input pengguna dalam konteks percakapan. Selain itu, proyek ini menggunakan Qdrant, yang berfungsi sebagai penyimpanan vektor yang mempermudah pencarian data berbasis kesamaan (similarity search). Qdrant membantu dalam menemukan hasil yang sesuai dari input user dengan database. Streamlit digunakan sebagai UI yang memungkinkan pengguna untuk berinteraksi dengan chatbot secara langsung melalui halaman web yang intuitif.

2.3.2 mengimplementasikan tools calling di chatbot

Proyek ini menggunakan integrasi ollama, yang merupakan model bahasa besar (LLM) yang bertugas untuk memproses dan memahami input pengguna dalam konteks percakapan. Selain itu Streamlit digunakan sebagai UI yang memungkinkan pengguna untuk berinteraksi dengan chatbot secara langsung melalui halaman web yang intuitif.

2.3.3 Membuat mcp server untuk memindahkan file dan folder serta membaca isi file tersebut

implementasi server filesystem menggunakan MCP melalui `mcp.server.fastmcp`. Ini memungkinkan chatbot seperti cursor dan claude untuk mengakses dan memodifikasi file lokal secara aman.

2.3.4 Membuat chatbot menggunakan n8n

Dalam proyek ini menggunakan n8n untuk membuat workflow. Dengan memakai apikey dari gemini dan telegram user dapat mengirim file lewat telegram dan gemini dapat meringkas isi file pdf tersebut selain itu juga melakukan webscraping.

3. Laporan Kegiatan Leap

3.1 Landasan Teori

1. Ollama

Ollama adalah sebuah platform dan runtime open-source yang memungkinkan pengguna untuk menjalankan model bahasa besar (Large Language Models / LLM) secara lokal pada perangkat mereka tanpa perlu koneksi internet atau akses ke layanan cloud. Dengan kemampuan ini, Ollama memberikan kontrol penuh terhadap privasi data dan mengurangi ketergantungan pada infrastruktur cloud yang sering kali mahal dan memiliki risiko keamanan. Ollama juga menyediakan lingkungan yang ramah bagi developer untuk menguji, mengembangkan, dan mengintegrasikan model bahasa dengan aplikasi secara mudah. Selain itu, Ollama mendukung berbagai model bahasa dari komunitas open-source maupun model komersial yang telah dioptimalkan agar dapat berjalan secara efisien pada perangkat lokal.

2. Qdrant

Qdrant adalah database vektor open-source yang didesain khusus untuk menyimpan, mencari, dan mengelola data dalam bentuk vektor berdimensi tinggi yang biasa dihasilkan dari proses embedding teks, gambar, audio, dan data non-struktural lainnya. Dengan menggunakan Qdrant, pencarian semantik atau similarity search dapat dilakukan secara efisien, memungkinkan aplikasi seperti pencarian dokumen, rekomendasi produk, dan chatbot yang dapat memahami konteks lebih dalam. Qdrant memiliki kemampuan skalabilitas tinggi serta mendukung fitur-fitur seperti filtering, pembobotan, dan indexing yang membuatnya cocok digunakan dalam aplikasi machine learning dan AI yang memerlukan pengolahan data dalam jumlah besar dan cepat.

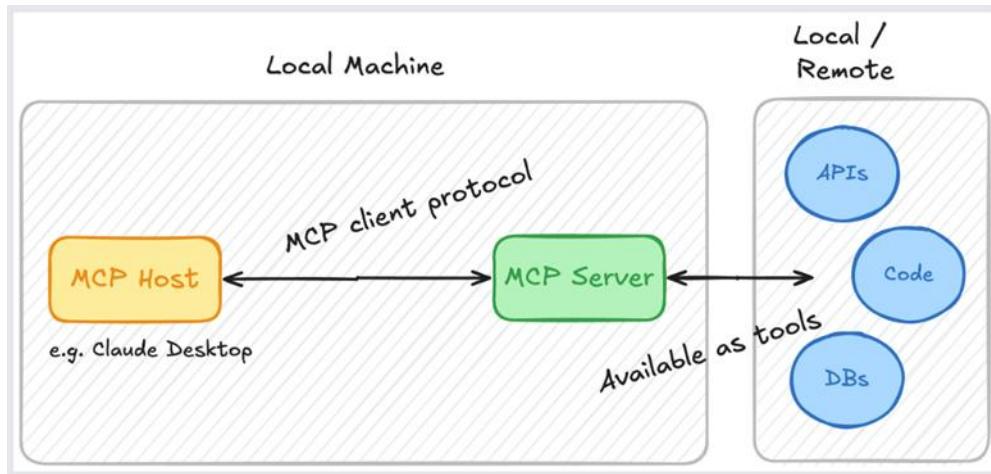
3. Tools Calling

Tools Calling merupakan sebuah fitur canggih yang memungkinkan Large Language Models (LLM) untuk memanggil alat atau fungsi eksternal secara otomatis dalam proses percakapan atau eksekusi tugas. Dengan fitur ini, LLM tidak hanya sekadar menjawab pertanyaan atau memberikan saran, tetapi juga dapat melakukan tindakan praktis seperti mengakses database, menjalankan perhitungan kompleks, memanggil API eksternal, atau mengontrol perangkat lunak lain. Keuntungan utama dari Tools Calling adalah memperkaya kemampuan LLM menjadi lebih dinamis dan interaktif tanpa perlu pengaturan manual dari developer setiap kali alat tersebut dibutuhkan. Hal ini memungkinkan terciptanya aplikasi yang lebih adaptif dan responsif terhadap kebutuhan pengguna.

4. MCP (Model Context Protocol)

MCP adalah sebuah protokol terbuka (open protocol) yang bertujuan untuk menghubungkan Large Language Models (LLM) dengan konteks eksternal seperti file, folder, API, atau database secara modular dan dinamis. Dengan MCP, LLM dapat mengakses informasi tambahan di luar model itu sendiri, sehingga dapat memberikan jawaban yang lebih tepat dan kontekstual. Protokol ini mendukung interaksi yang terstandarisasi antara LLM dan sumber data eksternal, memudahkan integrasi tanpa perlu mengubah model dasar. MCP juga memungkinkan pengembangan sistem AI yang lebih fleksibel dan mudah diperluas dengan berbagai modul tambahan sesuai kebutuhan aplikasi.

Diagram server MCP



Server MCP : Server ini berfungsi sebagai jembatan untuk menghubungkan API, basis data, atau kode. Server ini memaparkan sumber data sebagai alat bagi host dan dapat dibangun menggunakan SDK Python atau TypeScript.

Klien MCP : Klien ini menggunakan protokol untuk berinteraksi dengan server MCP. Seperti server, klien ini dapat dikembangkan menggunakan SDK dalam Python atau TypeScript.

Host MCP : Sistem ini mengelola komunikasi antara server dan klien, memastikan pertukaran data yang lancar. Host yang populer meliputi Claude Desktop , Zed , dan Sourcegraph Cody .

5. n8n

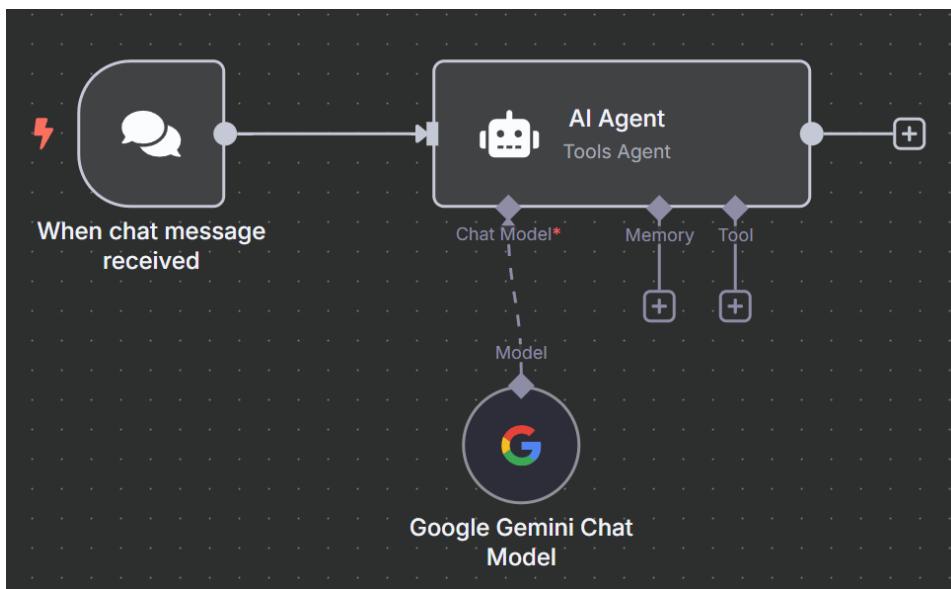
n8n adalah platform workflow automation dan integration tool berbasis open-source yang memungkinkan pengguna untuk mengotomatisasi proses bisnis dan menghubungkan berbagai aplikasi, layanan, serta API secara mudah tanpa perlu menulis banyak kode. Dengan antarmuka visual drag-and-drop, pengguna dapat membuat alur kerja (workflow) yang mengintegrasikan berbagai sistem seperti email, database, media sosial, dan layanan cloud secara terpadu. n8n sangat berguna dalam mengotomatisasi tugas-tugas repetitif, sinkronisasi data, dan orkestrasi proses yang kompleks, sehingga meningkatkan efisiensi operasional dan memungkinkan fokus pada

pengembangan solusi bisnis yang lebih inovatif.

Bagaimana Cara Kerja n8n?

Seperti alat otomasi lainnya, n8n bekerja dengan menghubungkan aplikasi melalui API. API sendiri bisa diibaratkan sebagai "terowongan digital" yang memungkinkan berbagai aplikasi berbagi informasi tanpa harus mengubah atau merusak sistem masing-masing.

Contoh workflow n8n untuk story generator :



Output setelah user memberikan prompt pada chat :

The screenshot shows a generated story about a knight. The story begins with the prompt 'create a story of a knight'. The output text is as follows:

Okay, here is a story about a knight:
Sir Kaelan adjusted the weight of the mail coif settling heavily on his shoulders. He was young, barely twenty years, but his gaze was steady, fixed on the battered banner of his lord, Duke Reynard, rippling in the biting wind above the castle walls. Kaelan had been a knight for only a year, his spurs still felt new, and his encounters with true danger had been mercifully few.
Today, that was about to change.
Word had reached the Duke's castle – a monstrous boar, twisted and unnaturally large, was terrorizing the outlying villages. It was not a normal beast of the forest; tales spoke of eyes that glowed with malevolence and hide as tough as iron. Farmers were dead, homes ruined, and fear gripped the land like a cold fog. The Duke, wary of sending his most seasoned knights on what might still be exaggerated rumour, had tasked Kaelan with investigating and, if necessary, confronting the threat.
Kaelan mounted his sturdy warhorse, Roland, the animal's breath pluming white in the chill air. His squire, a nervous boy named Thom, handed him his helm and lance. Kaelan took a deep breath, the cold metal against his gauntlet a familiar comfort. He nodded to Thom, nudged Roland, and rode out through the gates, the portcullis rising with a groan.
The journey to the village of Oakhaven was swift but tense. The forest felt unnaturally quiet, the birds silent. When they arrived, the scene was worse than the rumours. A few houses were reduced to rubble, a thick, dark mud mixed with blood smeared across the ground. The villagers huddled together, their faces gaunt with terror.
An old man, the village elder, stepped forward, his voice trembling. "Sir Kaelan, thank the heavens you are here. It came through last night... took two more men. It's like nothing we've ever seen. It hides in the Blackwood, by the ancient stones."
The Blackwood. A dark, tangled forest known for its ancient, unnerving quiet. Kaelan's stomach tightened, but he kept his expression resolute. "Fear not, good people. I will track this beast. Gather your strength, stay together."

Untuk memahami cara kerja n8n, mari kita lihat komponen utama dalam sistemnya:

1. Nodes

Nodes adalah titik-titik penghubung dalam workflow n8n. Data masuk, diproses, dan keluar melalui nodes. Anda bisa menghubungkan berbagai nodes untuk menciptakan alur kerja yang sesuai dengan kebutuhan Anda.

2. Connections

Connections adalah jalur yang menghubungkan nodes satu sama lain. Jika nodes adalah titik-titik dalam sebuah diagram, connections adalah garis yang menyambungkannya. Tanpa connections, nodes tidak dapat berkomunikasi.

3. Trigger Nodes

Trigger nodes adalah pemicu yang akan menjalankan workflow secara otomatis. Contohnya, jika Anda ingin mengotomatiskan proses setiap kali ada email masuk, Anda bisa menggunakan trigger node untuk mendeteksi email baru dan menjalankan alur kerja.

Keunggulan utama n8n dibandingkan alat otomasi lain adalah satu workflow bisa memiliki lebih dari satu trigger, yang membuatnya lebih fleksibel.

4. Workflows

Workflows adalah tempat di mana semua nodes dan connections digabungkan dalam satu proses. Setiap workflow bisa dianggap sebagai "dokumen kerja," seperti di Google Docs. Anda bisa membuat workflow untuk berbagai tugas, mulai dari mengolah data di Excel, menjalankan SaaS berbasis *cloud*, hingga menghubungkan berbagai sistem offline.

3.2 Kegiatan Bimbingan Selama LEAP

Kegiatan yang dilakukan selama LEAP, termasuk bimbingan dengan dosen dan mitra telah dicantumkan di lampiran.

3.3 Analisis Kebutuhan

Berikut adalah analisis permasalahan dan rencana solusi yang sedang

dikerjakan terkait dengan proyek yang ada:

3.3.1 mengembangkan chatbot untuk Kantin Petra

Dengan berkembangnya teknologi dapat mempermudah kantin agar dapat menerima pesanan dengan lebih mudah. Salah satu cara adalah dengan menggunakan chatbot sebagai menu untuk pelanggan dapat melihat makanan-makanan apa saja yang terdapat dalam kantin dan di stall mana makanan tersebut dijual.

3.3.2 mengimplementasikan tools calling di chatbot

Zaman sekarang mencari pekerjaan lumayan sulit apalagi harus harus browsing di banyak website dan situs yang berbeda-beda. Dengan adanya chatbot yang dihubungkan ke data website tersebut menggunakan tools calling user dapat lebih mudah mencari pekerjaan yang mereka inginkan

3.3.3 Membuat mcp server untuk memindahkan file dan folder serta membaca isi file tersebut

Perkembangan AI semakin canggih tetapi membuat AI apalagi chatbot berbentuk RAG tidaklah mudah apalagi jika ingin membuat chatbot dengan fungsi yang berbeda-beda maka harus mengcoding chatbot dari awal. Dengan MCP server cukup mengcoding perintah dalam mcp server tersebut lalu ngerun ke dalam chatbot yang sudah ada seperti cursor atau claude

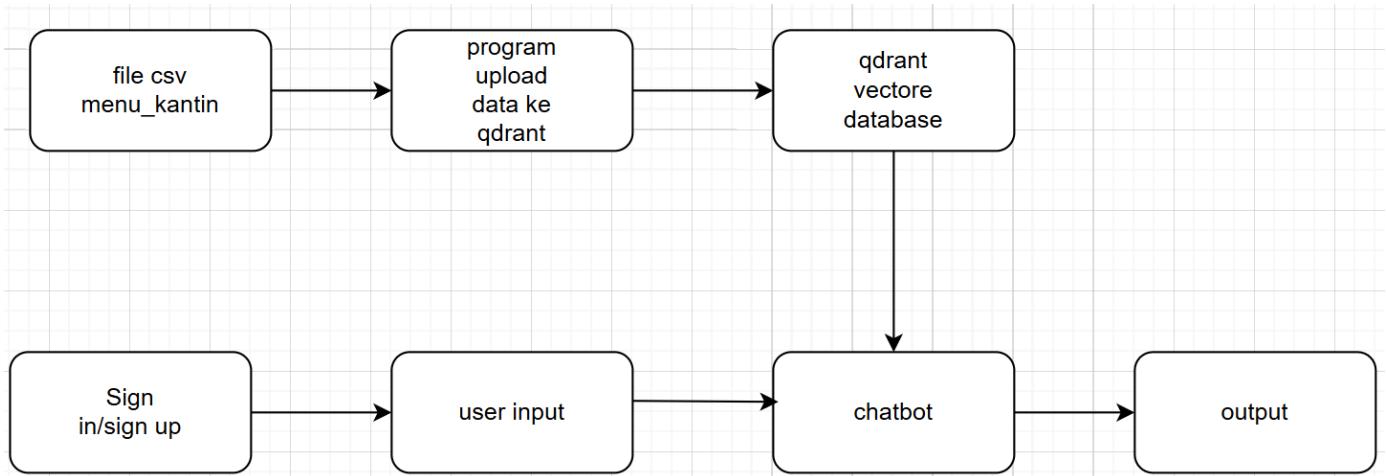
3.3.4 Membuat chatbot menggunakan n8n

Mengcoding AI chatbot tidaklah gampang tetapi dengan bantuan n8n mempermudah pembuatan chatbot dengan cukup membuat workflownya. Dengan n8n Cuma perlu siapkan apikey yang diperlukan dalam ai chatbot yang sedang dibuat

3.4 Desain

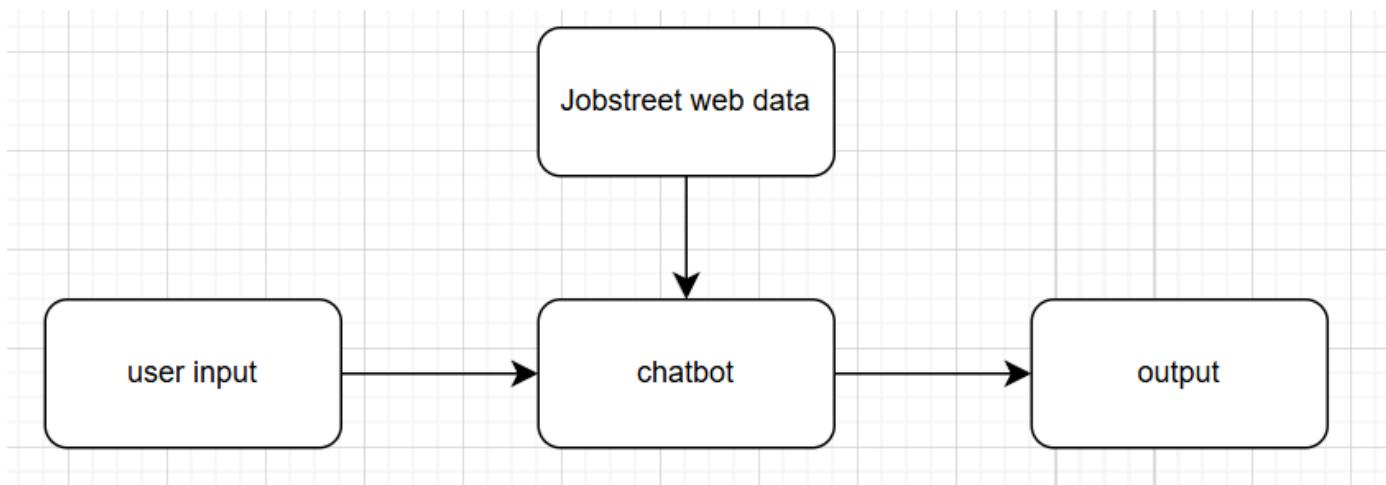
3.4.1 Chatbot untuk Kantin Petra

Berikut merupakan Gambaran proses cara kerja chatbot kantin petra.



3.4.2 Job finder menggunakan tools calling

Berikut merupakan Gambaran proses cara kerja chatbot job finder.



3.5 Implementasi

3.5.1 Chatbot untuk Kantin Petra

Upload file ke qdrant

```
import os
import sys
import logging
import pandas as pd
import qdrant_client
import streamlit as st
from llama_index.core import StorageContext
from llama_index.embeddings.ollama import OllamaEmbedding
from llama_index.vector_stores.qdrant import
QdrantVectorStore
from llama_index.llms.ollama import Ollama
from llama_index.core import VectorStoreIndex,
SimpleDirectoryReader, Settings
from llama_index.readers.file import CSVReader
from llama_index.core.memory import ChatMemoryBuffer
from llama_index.core.node_parser import SentenceSplitter
from llama_index.retrievers.bm25 import BM25Retriever
from llama_index.core.chat_engine import
CondensePlusContextChatEngine
from llama_index.core.retrievers import
QueryFusionRetriever
import nest_asyncio
from qdrant_client.http.models import VectorParams
import re
nest_asyncio.apply()
```

```

# Qdrant API dan URL

QDRANT_URL = "https://d6c5cf12-736b-4368-8d68-
d27dc0f90e55.us-east4-0.gcp.cloud.qdrant.io:6333"

QDRANT_API_KEY =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2Nlc3MiOiJtIn0
.ubB5LjaUaQ3cVntrES0eO6G0G8kOodBq_if85oJEzqg"

qdrant_client =
qdrant_client.QdrantClient(url=QDRANT_URL,
api_key=QDRANT_API_KEY)

# Memastikan koleksi kantin_menu sudah tersedia di
Qdrant. Jika belum, membuat koleksi dengan konfigurasi
vektor 768 dan metrik Cosine.

collection_name = "kantin_menu"

def ensure_collection():

    collections = qdrant_client.get_collections()

    if collection_name not in [col.name for col in
collections.collections]:

        qdrant_client.create_collection(
            collection_name=collection_name,
            vectors_config=VectorParams(size=768,
distance="Cosine"))

    )

    print(f"Collection '{collection_name}' created
successfully.")

ensure_collection()

```

```

vector_store = QdrantVectorStore(client=qdrant_client,
collection_name=collection_name)
storage_context =
StorageContext.from_defaults(vector_store=vector_store)

# Embedding model

embed_model =
OllamaEmbedding(base_url="http://127.0.0.1:11434",
model_name="nomic-embed-text:latest")
splitter = SentenceSplitter(chunk_size=512)

logging.basicConfig(stream=sys.stdout,
level=logging.WARNING)
logging.getLogger().addHandler(logging.StreamHandler(stre
am=sys.stdout))

Settings.llm = Ollama(model="llama3.1:latest",
base_url="http://127.0.0.1:11434",
system_prompt=system_prompt)
Settings.embed_model = embed_model

@st.cache_resource(show_spinner="Mempersiapkan data
kantin - sabar ya.")

def load_data():

```

```
# lokasi file
```

/ petranean-lapar /	
Name	Modified
docs	3mo ago
images	4mo ago

```
csv_parser = CSVReader(concat_rows=False)
file_extractor = {".csv": csv_parser}

reader = SimpleDirectoryReader(
    input_dir="../docs",
    recursive=True,
    file_extractor=file_extractor
)
documents = reader.load_data()

for doc in documents:
    doc.excluded_llm_metadata_keys = ["filename",
"extension"]

nodes = splitter.get_nodes_from_documents(documents,
show_progress=True)

index = VectorStoreIndex.from_documents(documents,
show_progress=True)
index_retriever =
index.as_retriever(similarity_top_k=8)
```

```

bm25_retriever = BM25Retriever.from_defaults(
    nodes=nodes,
    similarity_top_k=16,
)

return QueryFusionRetriever(
    [index_retriever, bm25_retriever],
    num_queries=2,
    use_async=True,
    similarity_top_k=24
)

```

#Menyimpan data ke qdrant

```

def store_images_in_qdrant():
    csv_file_path = './docs/menu-kantin.csv'
    if not os.path.exists(csv_file_path):
        print("CSV file not found!")
        return

    df = pd.read_csv(csv_file_path)

    for index, row in df.iterrows():
        product_name = row['Nama Produk']
        image_path = row['Gambar']

        if isinstance(image_path, str) and
os.path.exists(image_path):

```

```

vector =
embed_model.get_text_embedding(product_name)
qdrant_client.upsert(
    collection_name=collection_name,
    points=[{
        "id": index,
        "vector": vector,
        "payload": {
            "product_name": product_name,
            "image_path": image_path,
            "kantin": row.get("Kantin",
"Tidak diketahui"),
            "stall": row.get("Nama Stall",
"Tidak diketahui"),
            "harga": row.get("Harga", "Tidak
diketahui"),
            "keterangan": row.get("Keterangan", "")
        }
    }]
)
store_images_in_qdrant()

```

setelah menjalankan program maka data akan tersimpan di collection qdrant

Collections						UPLOAD SNAPSHOT
<input type="text"/> Search Collection						
Name	Status	Points (Approx)	Segments	Shards	Vectors Configuration (Name, Size, Distance)	Actions
kantin_menu	green	147	2	1	default 768 Cosine	⋮

Chatbot kantin petra

```

import os
import sys
import logging
import pandas as pd
import qdrant_client
import streamlit as st
from llama_index.core import StorageContext
from llama_index.embeddings.ollama import OllamaEmbedding
from llama_index.vector_stores.qdrant import
QdrantVectorStore
from llama_index.llms.ollama import Ollama
from llama_index.core import VectorStoreIndex,
SimpleDirectoryReader, Settings
from llama_index.readers.file import CSVReader
from llama_index.core.memory import ChatMemoryBuffer
from llama_index.core.node_parser import SentenceSplitter
from llama_index.retrievers.bm25 import BM25Retriever
from llama_index.core.chat_engine import
CondensePlusContextChatEngine
from llama_index.core.retrievers import QueryFusionRetriever
import nest_asyncio
from qdrant_client.http.models import VectorParams
import re
import requests
import json
from auth import register_user, login_user # Import
authentication functions

nest_asyncio.apply()

# QDRANT API dan URL
QDRANT_URL = "https://d6c5cf12-736b-4368-8d68-
d27dc0f90e55.us-east4-0.gcp.cloud.qdrant.io:6333"

```

```

QDRANT_API_KEY =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2Nlc3MiOiJtIn0.ubB
5LjaUaQ3cVntrES0eO6G0G8kOodBq_if85oJEzqg"

qdrant_client = qdrant_client.QdrantClient(url=QDRANT_URL,
api_key=QDRANT_API_KEY)

# Memastika qdrant ada data menu_kantin
collection_name = "kantin_menu"
def ensure_collection():
    collections = qdrant_client.get_collections()
    if collection_name not in [col.name for col in
collections.collections]:
        qdrant_client.create_collection(
            collection_name=collection_name,
            vectors_config=VectorParams(size=768,
distance="Cosine")
        )
        print(f"Collection '{collection_name}' created
successfully.")
ensure_collection()

vector_store = QdrantVectorStore(client=qdrant_client,
collection_name=collection_name)
storage_context =
StorageContext.from_defaults(vector_store=vector_store)

# Embedding model
embed_model =
OllamaEmbedding(base_url="http://127.0.0.1:11434",
model_name="nomic-embed-text:latest")
splitter = SentenceSplitter(chunk_size=512)

logging.basicConfig(stream=sys.stdout, level=logging.WARNING)
logging.getLogger().addHandler(logging.StreamHandler(stream=s
ys.stdout))

# System prompt for Gemini assistant
system_prompt = """
Anda adalah pelayan kantin yang ramah yang dapat mengarahkan
pengguna mencari makanan/minuman yang tepat.
Anda tidak perlu menyebutkan atau membuat pernyataan yang
mengatakan Anda tidak dapat menampilkan gambar jika gambar
berhasil ditemukan.
Tugas Anda adalah untuk menjawab dengan relevansi sesuai menu

```

dan menyarankan gambar jika sesuai tidak perlu memberikan path dari gambar.

Jawablah semua dalam Bahasa Indonesia.

Tugas Anda adalah untuk menjadi pelayan kantin yang ramah yang dapat mengarahkan user.

Kantin yang Anda layani adalah kantin kampus Universitas Kristen Petra Surabaya.

Pada Universitas Kristen Petra terdapat 2 gedung utama yang setiap gedungnya memiliki kantin, yaitu Gedung P dan W.

"""

Gemini API dan URL

```
API_KEY = "AIzaSyD6iVXg3LFrSTA5gEJ4tmY2UniNdAZqcxo"  
API_URL =  
f"https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash:generateContent?key={API_KEY}"
```

Function to communicate with Gemini API

```
def get_gemini_response(user_input):  
    headers = {"Content-Type": "application/json"}  
    data = {  
        "contents": [{"parts": [{"text": system_prompt + "\n" + user_input}]}] # Add system prompt before user input  
    }  
    response = requests.post(API_URL, headers=headers,  
data=json.dumps(data))  
    if response.status_code == 200:  
        response_json = response.json()  
        return  
response_json["candidates"][0]["content"]["parts"][0]["text"]  
else:  
    return "Error: Unable to get a response."
```

```
Settings.llm = Ollama(model="llama3.1:latest",  
base_url="http://127.0.0.1:11434",  
system_prompt=system_prompt)  
Settings.embed_model = embed_model
```

```
@st.cache_resource(show_spinner="Mempersiapkan data kantin -  
sabar ya.")  
def load_data():  
    csv_parser = CSVReader(concat_rows=False)  
    file_extractor = {".csv": csv_parser}  
  
    reader = SimpleDirectoryReader(
```

```

        input_dir="../docs",
        recursive=True,
        file_extractor=file_extractor
    )
documents = reader.load_data()

for doc in documents:
    doc.excluded_llm_metadata_keys = ["filename",
"extension"]

nodes = splitter.get_nodes_from_documents(documents,
show_progress=True)

index = VectorStoreIndex.from_documents(documents,
show_progress=True)
index_retriever = index.as_retriever(similarity_top_k=8)
bm25_retriever = BM25Retriever.from_defaults(
    nodes=nodes,
    similarity_top_k=16,
)

return QueryFusionRetriever(
    [index_retriever, bm25_retriever],
    num_queries=2,
    use_async=True,
    similarity_top_k=24
)

# Function mencari data di Qdrant (Hybrid Matching)
def find_menu(user_input, filter_price=None,
stall_name=None):
    query_vector = embed_model.get_text_embedding(user_input)
# Convert query to vector

    # Search for best matches in Qdrant collection
    results = qdrant_client.search(
        collection_name=collection_name,
        query_vector=query_vector,
        limit=100 # Increase limit for better coverage
    )

    menu_data = []
    filtered_results = []

    for result in results:

```

```

payload = result.payload
if payload:
    product_name = payload.get("product_name", "")
    image_path = payload.get("image_path", "")
    kantin = payload.get("kantin", "Tidak diketahui")
    stall = payload.get("stall", "Tidak diketahui")
    harga = payload.get("harga", "Tidak diketahui")
    keterangan = payload.get("keterangan", "")

    # Convert price to numeric value
    harga_numerik = None
    if isinstance(harga, (int, float)):
        harga_numerik = float(harga)
    elif isinstance(harga, str):
        harga_cleaned = re.sub(r"\D", "", harga)

# Remove non-numeric characters
    if harga_cleaned.isdigit():
        harga_numerik = float(harga_cleaned)

item_data = {
    "product_name": product_name,
    "image_path": image_path,
    "kantin": kantin,
    "stall": stall,
    "harga": harga,
    "harga_numerik": harga_numerik,
    "keterangan": keterangan
}

# memfilter makanan yang dicari
if stall_name:
    if stall_name.lower() in stall.lower():
        filtered_results.append(item_data)
elif user_input.lower() in product_name.lower():
    filtered_results.append(item_data)
else:
    menu_data.append(item_data)

# Apply price filter if specified
if filter_price is not None:
    menu_data = [item for item in menu_data if
item["harga_numerik"] is not None and item["harga_numerik"]
<= filter_price]

return filtered_results if filtered_results else

```

```

menu_data

# Function to get all stalls
def get_all_stalls():
    results = qdrant_client.search(
        collection_name=collection_name,
        query_vector=[0] * 768, # Dummy vector to get all
    items
        limit=1000 # Adjust limit as necessary
    )

    stalls = set() # Use a set to avoid duplicates
    for result in results:
        payload = result.payload
        if payload:
            stall = payload.get("stall", "Tidak diketahui")
            stalls.add(stall)

    return list(stalls)

def clean_user_input(user_input):
    # Define a regex pattern to match variations of "gambar"
    pattern = r"\b(gambar|gmbr|image|gmb)\b"
    # Remove the matched patterns from the user input
    cleaned_input = re.sub(pattern, "", user_input,
    flags=re.IGNORECASE).strip()
    return cleaned_input
def analyze_user_input(user_input):
    if "list stall" in user_input.lower() or "daftar stall" in user_input.lower():
        return "list_stalls"
    elif "kantin" in user_input.lower():
        return "kantin"
    elif "stall" in user_input.lower():
        stall_match = re.search(r"stall\s+(\w+)", user_input,
    re.IGNORECASE)
        if stall_match:
            return "stall", stall_match.group(1) # Return
    stall name
    return "product_name"

# Main Program
st.title("Petranesian Lapar 🍔")
st.write("Data partial hanya tersedia untuk Gedung P dan W.")

```

```

# User Authentication
st.sidebar.title("User Authentication")
auth_option = st.sidebar.selectbox("Choose an option",
["Login", "Register"])

if auth_option == "Register":
    username = st.sidebar.text_input("Username")
    email = st.sidebar.text_input("Email")
    password = st.sidebar.text_input("Password",
type="password")
    if st.sidebar.button("Register"):
        message = register_user(username, email, password)
        st.sidebar.success(message)

elif auth_option == "Login":
    email = st.sidebar.text_input("Email")
    password = st.sidebar.text_input("Password",
type="password")
    if st.sidebar.button("Login"):
        message, username, email = login_user(email,
password)
        if message == "Login successful.":
            st.session_state['username'] = username # Store
username in session state
            st.session_state['email'] = email # Store email
in session state
            st.sidebar.success(message)

# Proceed with chatbot only if user is logged in
if 'email' in st.session_state:
    st.title(f"Welcome, {st.session_state['username']}!")
    retriever = load_data()
    st.write("Jika ingin mencari stall ketik 'stall'")
    st.write("Jika ingin mencari makanan ketik langsung
makanan yang ingin di cari")
    st.write("Jika ingin mencari harga ketik dibawah
..(harga)..")

    # Chat session state
    if "messages" not in st.session_state:
        st.session_state.messages = [{"role": "assistant",
"content": "Halo! Lagi mau makan/minum apaan? 😊"}]

    if "chat_engine" not in st.session_state.keys():
        memory =

```

```

ChatMemoryBuffer.from_defaults(token_limit=8192)
    st.session_state.chat_engine =
CondensePlusContextChatEngine(
    verbose=True,
    system_prompt=system_prompt,
    memory=memory,
    retriever=retriever,
    llm=Settings.llm
)

# Display chat messages
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])

# Display previous images if any
if "previous_images" in st.session_state:
    num_columns = 3 # Number of columns in the grid
    columns = st.columns(num_columns)
    for i, item in
enumerate(st.session_state.previous_images):
        col = columns[i % num_columns] # Cycle through
columns for each image
        with col:
            st.image(item["image_path"], width=250)
            st.markdown(f"**{item['product_name']}**")
            st.markdown(f"Harga: {item['harga']}")
            st.markdown(f"Stall: {item['stall']}")
            st.markdown(f"Kantin: {item['kantin']}")

    if prompt := st.chat_input(placeholder="Mau makan/minum
apa?"):
        st.session_state.messages.append({"role": "user",
"content": prompt})
        with st.chat_message("user"):
            st.markdown(prompt)

    cleaned_prompt = clean_user_input(prompt)

    search_type = analyze_user_input(cleaned_prompt)

    match = re.search(r"(\d+)", cleaned_prompt)
    if match:
        max_price = float(match.group(1))
    else:

```

```

max_price = None

filtered_results = []

if search_type == "list_stalls":
    all_stalls = get_all_stalls()
    stalls_message = "Daftar Stall:\n" +
"\n".join(all_stalls)
    st.session_state.messages.append({"role": "assistant", "content": stalls_message})

    with st.chat_message("assistant"):
        st.markdown(stalls_message)
elif search_type == "product_name":
    filtered_results = find_menu(cleaned_prompt,
filter_price=max_price)
    elif isinstance(search_type, tuple) and
search_type[0] == "stall":
        stall_name = search_type[1]

        filtered_results = find_menu(cleaned_prompt,
stall_name=stall_name)
        if max_price is not None:
            filtered_results = [item for item in
filtered_results if item["harga_numerik"] is not None and
item["harga_numerik"] <= max_price]
    elif search_type == "kantin":
        filtered_results = find_menu(cleaned_prompt,
filter_price=max_price) # Adjust this to search by kantin if
needed

if filtered_results:
    response = get_gemini_response(cleaned_prompt)
    st.session_state.messages.append({"role": "assistant", "content": response})

    with st.chat_message("assistant"):
        st.markdown(response)

    st.session_state.previous_images =
filtered_results

    num_columns = 3 # Number of columns in the grid
    columns = st.columns(num_columns)

```

```
        for i, item in enumerate(filtered_results):
            col = columns[i % num_columns] # Cycle
            through columns for each image
            with col:
                st.image(item["image_path"], width=250)

    st.markdown(f"**{item['product_name']}**")
        st.markdown(f"Harga: {item['harga']} ")
        st.markdown(f"Stall: {item['stall']} ")
        st.markdown(f"Kantin: {item['kantin']} ")

else:
    st.sidebar.warning("Please log in to access the
chatbot.")
```



Halo! Lagi mau makan/minum apaan? 😊



soto ayam



Selamat siang! Mau pesan apa hari ini? Soto ayam ya? Wah, pilihan yang tepat!

Di Katin UK Petra, kami punya beberapa pilihan soto ayam, tergantung gedung mana yang lebih dekat dengan Anda.

- **Jika Anda di Gedung P:** Coba tanyakan di katin Gedung P, biasanya mereka punya soto ayam kampung yang gurih kuahnya, dengan potongan ayam yang empuk. (*Gambar soto ayam kampung dengan potongan ayam yang terlihat empuk dan kuah yang bening kekuningan*)
- **Jika Anda di Gedung W:** Katin Gedung W juga menyediakan soto ayam, mungkin dengan sedikit variasi resep. Biasanya lebih cenderung ke soto ayam madura dengan cita rasa yang sedikit lebih pedas. (*Gambar soto ayam madura dengan tampilan lebih berwarna dan potongan ayam yang lebih kecil*)

Kira-kira yang mana yang lebih Anda suka? Soto ayam kampung atau soto ayam madura? Jangan ragu untuk bertanya lagi ya kalau masih bingung! 😊



Soto ayam Special

Harga: Rp18,000

Stall: Soto Ayam Jago

Katin: Gedung P

Soto ayam goreng

Harga: Rp22,000

Stall: Soto Ayam Jago

Katin: Gedung P

Soto ayam biasa

Harga: Rp16,000

Stall: Soto Ayam Jago

Katin: Gedung P

Contoh seperti dari gambar diatas chatbot akan menanyakan makanan yang diinginkan kemudian Ketika user menginput makanan yang diinginkan maka chatbot akan menampilkan makanan dengan gambar serta harga, stall dan katin

3.5.2 Job Search chatbot

```
import streamlit as st
from llama_index.llms.ollama import Ollama
from llama_index.embeddings.ollama import OllamaEmbedding
from llama_index.core import VectorStoreIndex,
SimpleDirectoryReader, Settings
from llama_index.readers.file import CSVReader
from llama_index.core.llms import ChatMessage, MessageRole
from llama_index.core import Settings
from llama_index.core.memory import ChatMemoryBuffer
from llama_index.core.node_parser import SentenceSplitter
from llama_index.retrievers.bm25 import BM25Retriever
from llama_index.core.chat_engine import
CondensePlusContextChatEngine
from llama_index.core.retrievers import QueryFusionRetriever
from llama_index.core.tools import BaseTool, FunctionTool
from llama_index.core.agent import ReActAgent
from typing import Optional
from llama_index.core import PromptTemplate
from llama_index.llms.gemini import Gemini
import asyncio
import nest_asyncio
import sys
import logging
import requests
import os
nest_asyncio.apply()
splitter = SentenceSplitter(chunk_size=512)
logging.basicConfig(stream=sys.stdout, level=logging.WARNING)
logging.getLogger().addHandler(logging.StreamHandler(stream=sys.stdout))

# System prompt
system_prompt = """
You are a multi-lingual career advisor expert who has
knowledge based on
real-time data. You will always try to be helpful and try to
help them
answering their question. If you don't know the answer, say
that you DON'T
KNOW.
```

You primary job is to help students find jobs related to their interests from the Jobstreet Platform.

```
"""
react_system_header_str = """\

## Tools
You have access to a wide variety of tools. You are
responsible for using
the tools in any sequence you deem appropriate to complete
the task at hand.
This may require breaking the task into subtasks and using
different tools
to complete each subtask.

You have access to the following tools:
{tool_desc}

## Format output
To answer the question, please use the following format.

```
Thought: I need to use a tool to help me answer the question.
Action: tool name (one of {tool_names}) if using a tool.
Action Input: the input to the tool, in a JSON format
representing the kwargs (e.g. {"input": "hello world",
"num_beams": 5})
```

Please ALWAYS start with a Thought.

Please use a valid JSON format for the Action Input. Do NOT
do this {'input': 'hello world', 'num_beams': 5}.

If this format is used, the user will respond in the
following format:

```
Observation: tool response
```

You should keep repeating the above format until you have
enough information
to answer the question without using any more tools. At that
point, you MUST respond
in the one of the following two formats:

```
```

Thought: I can answer without using any more tools.

Answer: [your answer here]

```

```

Thought: I cannot answer the question with the provided tools.

Answer: Sorry, I cannot answer your query.

```

Additional Rules

- You MUST obey the function signature of each tool. Do NOT pass in no arguments if the function expects arguments.

Current Conversation

Below is the current conversation consisting of interleaving human and assistant messages.

"""

```
react_system_prompt = PromptTemplate(react_system_header_str)
```

Api gemini

```
Settings.llm = Gemini(
    model="models/gemini-2.0-flash",
    api_key="AIzaSyD6iVXg3LFrSTA5gEJ4tmY2UniNdAZqcxo", # Replace with your own API key
    system_prompt=system_prompt, temperature=0
)
```

```
Settings.embed_model =
OllamaEmbedding(base_url="http://127.0.0.1:11434",
model_name="mxbai-embed-large:latest")
```

Main Program

```
st.title("RAG Test")
```

Initialize chat history if empty

```
if "messages" not in st.session_state:
    st.session_state.messages = [
        {"role": "assistant", "content": "Halo! Mau tahu apa tentang pekerjaan?"}
    ]
```

```
def job_json_to_natural_language(json_data, keyword):
```

```
    """Converts job JSON data to formatted text with links"""
    if "data" not in json_data:
```

```

        return f"No job listings found for '{keyword}'."

    output = [f"Here are some of the '{keyword}' job postings
found on JobStreet:"]

    for idx, job in enumerate(json_data["data"], 1):
        title = job.get("title", "Unknown Position")
        company = job.get("companyName",
job.get("advertiser", {}).get("description", "Unknown
Company"))
        location = ", ".join([loc.get("label", "Unknown") for
loc in job.get("locations", [])])
        posted = job.get("listingDateDisplay", "Unknown
date")
        work_type = ", ".join(job.get("workTypes", []))
        work_arrangement = ", ".join(
            [wa.get("label", {}).get("text", "") for wa in
job.get("workArrangements", {}).get("data", [])])
        )
        description = job.get("teaser", "No description
available")
        job_id = job.get("id", "")
        job_url =
f"https://www.jobstreet.co.id/id/job/{job_id}" if job_id else
"URL not available"

        output.append(
            f"{idx}. **{title}**\n"
            f"  - **Company**: {company}\n"
            f"  - **Location**: {location}\n"
            f"  - **Posted**: {posted}\n"
            f"  - **Job Type**: {work_type}\n"
            f"  - **{work_arrangement}**\n"
            f"  - **Description**: {description}\n"
            f"  - **Job Link**: {job_url}\n"
        )

    return "\n".join(output)

async def search_jobstreet(keyword: str) -> str:
    """Searches JobStreet for matching jobs"""
    try:
        r = requests.get(
            "https://id.jobstreet.com/api/jobsearch/v5/search",

```

```

params={
    "siteKey": "ID-Main",
    "sourcesystem": "houston",
    "page": "1",
    "worktype": "242",
    "sortmode": "ListedDate",
    "pageSize": "32",
    "include": "seodata,joracrosslink,gptTargeting,pills",
    "locale": "id-ID",
    "keywords": keyword,
    "baseKeywords": keyword,
},
timeout=10,
)
r.raise_for_status()
data = r.json()
return job_json_to_natural_language(data, keyword)
except Exception as e:
    return f"Error searching for jobs: {str(e)}"

search_jobstreet_tool = FunctionTool.from_defaults(
    fn=search_jobstreet,
    name="job_search",
    description="Searches for jobs on JobStreet based on keywords"
)

# Initialize chat
if "chat_engine" not in st.session_state:
    memory =
ChatMemoryBuffer.from_defaults(token_limit=32768)
    st.session_state.chat_engine = ReActAgent.from_tools(
        [search_jobstreet_tool],
        chat_mode="react",
        verbose=True,
        memory=memory,
        llm=Settings.llm,
        system_prompt=system_prompt,
    )
    st.session_state.messages = [
        {"role": "assistant", "content": "Halo! Mau tahu tentang pekerjaan?"}
    ]

```

```
# Display chat history
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])

# Handle user input
if prompt := st.chat_input("Apa yang ingin Anda cari?"):
    # Add user message to chat
    st.session_state.messages.append({"role": "user",
"content": prompt})
    with st.chat_message("user"):
        st.markdown(prompt)

    # Get assistant response
    with st.chat_message("assistant"):
        with st.spinner("Mencari..."):
            response =
st.session_state.chat_engine.chat(prompt)
            st.markdown(response.response)

    # Add assistant response to chat history
    st.session_state.messages.append({"role": "assistant",
"content": response.response})
```



Halo! Mau tahu apa tentang pekerjaan?



marketing



Here are some marketing job postings from JobStreet: Travel Product & Contracting Executive, Sales and Marketing Manager, Online Sales & Marketing, MARKETING PROPERTY, Sales And Marketing, Sales Marketing, Sales And Marketing Executive, PUBLIC RELATION SPA.



Sales And Marketing



Here are some of the 'Sales And Marketing' job postings found on JobStreet:

1. **Travel Product & Contracting Executive**

- **Company:** Pengiklan Anonim
- **Location:** Bali
- **Posted:** 15 menit yang lalu
- **Job Type:** Full time (Kantor)
- **Description:** Create tour packages, secure hotel deals, draft contracts—ideal for a smart negotiator with overseas education & tourism insight.
- **Job Link:** <https://www.jobstreet.co.id/id/job/84764601>

2. **Warehouse Leader Inventory**

- **Company:** Eastlync Technology Indonesia
- **Location:** Kosambi, Banten
- **Posted:** 3 jam yang lalu
- **Job Type:** Full time (Kantor)
- **Description:** Lead our warehouse ops with excellent inventory management. Competitive salary, benefits, and growth opportunities.
- **Job Link:** <https://www.jobstreet.co.id/id/job/84764043>

3. **Property Agent**

- **Company:** Brighton Wisdom - Semarang
- **Location:** Semarang Timur, Jawa Tengah
- **Posted:** 3 jam yang lalu

Ketika user menginput nama pekerjaan yang diinginkan maka chatbot akan menampilkan company yang memiliki pekerjaan tersebut. Ketika user menginput bidang yang mereka minati misalnya marketing maka chatbot akan merekomendasi pekerjaan yang sesuai dengan bidang tersebut.

3.5.3 Percobaan MCP (Model Context Protocol)

Dibuat sebuah MCP server untuk membaca isi file, memindahkan file dan mengelisit isi desktop. Folder yang diakses adalah folder desktop "C:\\Users\\Lenovo\\Desktop"

Dibutuhkan code python untuk menjalankan mcp server yang diletakan pada "C:\\Users\\Lenovo\\Desktop\\kuliah\\mcp\\filesystem.py"

Sebagai referensi dapat dilihat di

<https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem>

Kode python mcp server sebagai berikut :

```
import argparse
import asyncio
import datetime
import fnmatch
import json
import os
import pathlib
import sys
from typing import Any, Dict, List, Optional

# Ensure mcp is installed: pip install "mcp[cli]"
# Or using uv: uv add "mcp[cli]"
from mcp.server.fastmcp import FastMCP

# --- Globals ---
# This will be populated by command-line arguments
ALLOWED_DIRS: List[pathlib.Path] = []

# Initialize FastMCP server
# Give it a name that clients can identify
mcp = FastMCP("filesystem")

# --- Security Helper ---
def is_path_allowed(path_str: str) -> bool:
    """Checks if the given path is within one of the allowed directories."""
    try:
        target_path = pathlib.Path(path_str).resolve()
    except Exception:
        # If path resolution fails, it's likely invalid or problematic
        return False

    for allowed_dir in ALLOWED_DIRS:
        if target_path.is_relative_to(allowed_dir):
            return True
        # Also allow operating directly *on* an allowed directory itself
        if target_path == allowed_dir:
            return True
    return False

def security_check(path_str: str, check_existence: bool = False) ->
```

```

pathlib.Path:
"""
    Performs security check and returns resolved Path object.
    Raises ValueError if not allowed or other issues occur.
"""

    if not is_path_allowed(path_str):
        raise ValueError(f"Access denied: Path '{path_str}' is outside allowed
directories.")
    resolved_path = pathlib.Path(path_str).resolve()
    if check_existence and not resolved_path.exists():
        raise FileNotFoundError(f"Path not found: {path_str}")
    return resolved_path

# --- Tool Implementations ---

@mcp.tool()
async def read_file(path: str) -> str:
    """Reads the complete contents of a file using UTF-8 encoding.

Args:
    path: The full path to the file to read.
"""

    checked_path = security_check(path, check_existence=True)
    try:
        if not checked_path.is_file():
            raise ValueError(f"Path is not a file: {path}")
        return checked_path.read_text(encoding='utf-8')
    except FileNotFoundError:
        raise FileNotFoundError(f"File not found: {path}")
    except PermissionError:
        raise PermissionError(f"Permission denied reading file: {path}")
    except Exception as e:
        raise Exception(f"Error reading file '{path}': {e}")

@mcp.tool()
async def read_multiple_files(paths: List[str]) -> str:
    """Reads multiple files simultaneously. Failed reads won't stop the entire
operation.

Args:
    paths: A list of full paths to the files to read.
"""

    results = {}
    for path_str in paths:
        try:
            # Reuse the single file read logic, including its security checks
            content = await read_file(path_str)
            results[path_str] = {"status": "success", "content": content}
        except Exception as e:
            results[path_str] = {"status": "error", "message": str(e)}

    # Return a JSON string for structured output
    return json.dumps(results, indent=2)

@mcp.tool()
async def write_file(path: str, content: str) -> str:
    """Creates a new file or overwrites an existing file with the provided

```

```

content.

Exercise caution: This will overwrite existing files without warning.

Args:
    path: The full path where the file should be written.
    content: The string content to write to the file.
"""
# Security check allows writing *within* allowed dirs, not *to* the dir
itself if it's a file path
checked_path = security_check(path, check_existence=False) # Don't require
existence for write
try:
    # Ensure parent directory exists
    checked_path.parent.mkdir(parents=True, exist_ok=True)
    # Security check parent dir as well
    security_check(str(checked_path.parent))

    checked_path.write_text(content, encoding='utf-8')
    return f"Successfully wrote content to '{path}'."
except PermissionError:
    raise PermissionError(f"Permission denied writing file: {path}")
except IsADirectoryError:
    raise IsADirectoryError(f"Cannot write file: Path is a directory:
{path}")
except Exception as e:
    raise Exception(f"Error writing file '{path}': {e}")

# NOTE: Implementing the full 'edit_file' as described (diffs, indentation,
simultaneous edits)
# is very complex. This is a simplified version using basic sequential
replace.
# A robust implementation would require significantly more logic, possibly
external libs.
@mcp.tool()
async def edit_file(path: str, edits: List[Dict[str, str]], dryRun: bool =
False) -> str:
"""
Applies a series of simple text replacements to a file.
WARNING: This is a basic implementation. It performs sequential
replacements
and may not handle overlapping edits, complex patterns, or preserve
indentation perfectly.
Always use dryRun=true first to preview changes.

Args:
    path: The full path to the file to edit.
    edits: A list of edit operations. Each item is a dictionary with keys
'oldText' and 'newText'.
    dryRun: If true, previews changes as a diff without applying them.
(Default: false)
"""
checked_path = security_check(path, check_existence=True)
if not checked_path.is_file():
    raise ValueError(f"Path is not a file: {path}")

try:
    original_content = checked_path.read_text(encoding='utf-8')
    modified_content = original_content

```

```

for edit in edits:
    old = edit.get("oldText")
    new = edit.get("newText")
    if old is None or new is None:
        raise ValueError("Each edit must have 'oldText' and 'newText' keys.")
    # Simple sequential replacement
    modified_content = modified_content.replace(old, new)

if dryRun:
    # Generate a basic diff (could use difflib for a better diff)
    if original_content == modified_content:
        return "Dry run: No changes detected."
    else:
        # Simple indication of change, a proper diff is more complex
        diff_lines = []
        original_lines = original_content.splitlines()
        modified_lines = modified_content.splitlines()
        # Rudimentary diff placeholder
        diff_lines.append("--- Original")
        diff_lines.extend(original_lines)
        diff_lines.append("+++ Modified")
        diff_lines.extend(modified_lines)
        return f"Dry run preview:\n" + "\n".join(diff_lines)
else:
    if original_content == modified_content:
        return "No changes needed or applied."
    else:
        # Write the changes
        checked_path.write_text(modified_content, encoding='utf-8')
        return f"Successfully applied edits to '{path}'."

except FileNotFoundError:
    raise FileNotFoundError(f"File not found: {path}")
except PermissionError:
    raise PermissionError(f"Permission denied editing file: {path}")
except Exception as e:
    raise Exception(f"Error editing file '{path}': {e}")

@mcp.tool()
async def create_directory(path: str) -> str:
    """Creates a new directory. Creates parent directories if needed. Succeeds silently if the directory already exists.

Args:
    path: The full path of the directory to create.
    """
    # Security check allows creating dirs *within* allowed dirs
    checked_path = security_check(path, check_existence=False) # Don't require existence for create
    try:
        # Check parent is allowed BEFORE creating
        security_check(str(checked_path.parent))
        checked_path.mkdir(parents=True, exist_ok=True)
        return f"Directory '{path}' created or already exists."
    except PermissionError:

```

```

    
```

```

        raise PermissionError(f"Permission denied creating directory: {path}")
    except FileExistsError:
        # Should be caught by exist_ok=True, but handle just in case path
        points to a file
        if checked_path.is_file():
            raise FileExistsError(f"Cannot create directory: A file already
exists at '{path}'")
        return f"Directory '{path}' already exists." # Should not be reached
if exist_ok=True works
    except Exception as e:
        raise Exception(f"Error creating directory '{path}': {e}")

@mcp.tool()
async def list_directory(path: str) -> str:
    """Lists the contents of a directory, prefixing items with [FILE] or
[DIR]."""

    Args:
        path: The full path of the directory to list.
    """
    checked_path = security_check(path, check_existence=True)
    try:
        if not checked_path.is_dir():
            raise NotADirectoryError(f"Path is not a directory: {path}")

        contents = []
        for item in checked_path.iterdir():
            prefix = "[DIR]" if item.is_dir() else "[FILE]"
            contents.append(f"{prefix} {item.name}")
        if not contents:
            return f"Directory '{path}' is empty."
        return f"Contents of '{path}':\n" + "\n".join(contents)
    except FileNotFoundError:
        raise FileNotFoundError(f"Directory not found: {path}")
    except NotADirectoryError:
        raise NotADirectoryError(f"Path is not a directory: {path}")
    except PermissionError:
        raise PermissionError(f"Permission denied listing directory: {path}")
    except Exception as e:
        raise Exception(f"Error listing directory '{path}': {e}")

@mcp.tool()
async def move_file(source: str, destination: str) -> str:
    """Moves or renames a file or directory. Fails if the destination already
exists."""

    Args:
        source: The full path of the file or directory to move.
        destination: The full path of the new location or name.
    """
    source_path = security_check(source, check_existence=True)
    dest_path = security_check(destination, check_existence=False) #
Destination shouldn't exist yet

    # Explicitly check if destination exists before trying to move
    if dest_path.exists():
        raise FileExistsError(f"Destination path '{destination}' already
exists")

```

```

exists. Move failed.")

# Ensure destination parent directory is allowed and exists
dest_parent = dest_path.parent
security_check(str(dest_parent))
dest_parent.mkdir(parents=True, exist_ok=True)

try:
    source_path.rename(dest_path)
    return f"Successfully moved '{source}' to '{destination}'."
except FileNotFoundError:
    # Should be caught by initial check, but handle defensively
    raise FileNotFoundError(f"Source path not found: {source}")
except PermissionError:
    raise PermissionError(f"Permission denied moving '{source}' to
'{destination}'")
except Exception as e:
    raise Exception(f"Error moving '{source}' to '{destination}': {e}")

@mcp.tool()
async def search_files(path: str, pattern: str, excludePatterns:
Optional[List[str]] = None) -> str:
    """Recursively searches for files and directories matching a pattern
within a specified path.
    Supports excluding patterns using glob format. Case-insensitive
matching.

Args:
    path: The starting directory path for the search.
    pattern: The search pattern (e.g., '*.txt', 'data*'). Glob patterns
supported.
    excludePatterns: Optional list of patterns to exclude (e.g.,
['*/temp/*', '*.log']). Glob formats supported.
    """
    if excludePatterns is None:
        excludePatterns = []

    base_path = security_check(path, check_existence=True)
    if not base_path.is_dir():
        raise NotADirectoryError(f"Search path is not a directory: {path}")

    matches = []
    try:
        # Convert exclude patterns to lowercase for case-insensitive matching
        excludePatternsLower = [p.lower() for p in excludePatterns]

        for item in base_path.rglob(pattern):
            # Perform security check on each found item before adding
            if not is_path_allowed(str(item)):
                # Skip items found outside allowed scope (shouldn't happen if
base_path is checked, but belt-and-suspenders)
                continue

                item_path_str_lower = str(item).lower()
                excluded = False
                for exclude_pattern in excludePatternsLower:

```

```

        # Use fnmatch for glob matching, case-insensitive compare
        if fnmatch.fnmatchcase(item_path_str_lower,
exclude_pattern.lower()):
            excluded = True
            break
        if not excluded:
            matches.append(str(item.resolve())) # Return absolute paths

    if not matches:
        return f"No files or directories found matching '{pattern}' in
'{path}' (excluding specified patterns)."
    return f"Found matches for '{pattern}' in '{path}':\n" +
"\n".join(matches)
except PermissionError:
    # May occur during recursive search
    raise PermissionError(f"Permission denied during search in
'{path}.'")
except Exception as e:
    raise Exception(f"Error searching in '{path}' with pattern
'{pattern}': {e}")

@mcp.tool()
async def get_file_info(path: str) -> str:
    """Gets detailed metadata for a file or directory.

Args:
    path: The full path to the file or directory.
"""
    checked_path = security_check(path, check_existence=True)
    try:
        stat_result = checked_path.stat()
        info = {
            "path": str(checked_path.resolve()),
            "type": "directory" if checked_path.is_dir() else "file",
            "size_bytes": stat_result.st_size,
            "created_time":
datetime.datetime.fromtimestamp(stat_result.st_ctime).isoformat(),
            "modified_time":
datetime.datetime.fromtimestamp(stat_result.st_mtime).isoformat(),
            "accessed_time":
datetime.datetime.fromtimestamp(stat_result.st_atime).isoformat(),
            # Permissions are complex and platform-dependent, omitting for
simplicity
            # "permissions": oct(stat_result.st_mode)[-3:]
        }
        return json.dumps(info, indent=2)
    except FileNotFoundError:
        raise FileNotFoundError(f"Path not found: {path}")
    except PermissionError:
        raise PermissionError(f"Permission denied getting info for: {path}")
    except Exception as e:
        raise Exception(f"Error getting info for '{path}': {e}")

@mcp.tool()
async def list_allowed_directories() -> str:
    """Lists all the directories the server is configured to access."""
    if not ALLOWED_DIRS:
        return "No directories are configured for access."

```

```

dir_list = [str(d) for d in ALLOWED_DIRS]
return "Allowed directories:\n" + "\n".join(dir_list)

# --- Main Execution ---
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="MCP Filesystem Server")
    parser.add_argument(
        "allowed_dirs",
        metavar="ALLOWED_DIR",
        type=str,
        nargs='+',
        help="One or more absolute directory paths the server is allowed to
access.")
    )
    parser.add_argument(
        "--transport",
        type=str,
        default="stdio",
        choices=["stdio", "sse"], # Add more if needed
        help="MCP transport mechanism to use (default: stdio)."
    )
    parser.add_argument(
        "--port",
        type=int,
        default=8080,
        help="Port to use if transport is 'sse' (default: 8080)."
    )

args = parser.parse_args()

# Validate and store allowed directories
for dir_path in args.allowed_dirs:
    p = pathlib.Path(dir_path)
    if not p.is_absolute():
        print(f"Error: Allowed directory path must be absolute:
{dir_path}", file=sys.stderr)
        sys.exit(1)
    if not p.exists() or not p.is_dir():
        print(f"Error: Allowed directory path does not exist or is not a
directory: {dir_path}", file=sys.stderr)
        sys.exit(1)
    ALLOWED_DIRS.append(p.resolve()) # Store resolved absolute paths

if not ALLOWED_DIRS:
    print("Error: At least one allowed directory must be specified.",
file=sys.stderr)
    sys.exit(1)

print(f"Starting Filesystem MCP Server. Allowed directories:",
file=sys.stderr)
for d in ALLOWED_DIRS:
    print(f"- {d}", file=sys.stderr)
print(f"Using transport: {args.transport}", file=sys.stderr)
if args.transport == "sse":
    print(f"SSE Port: {args.port}", file=sys.stderr)

```

```

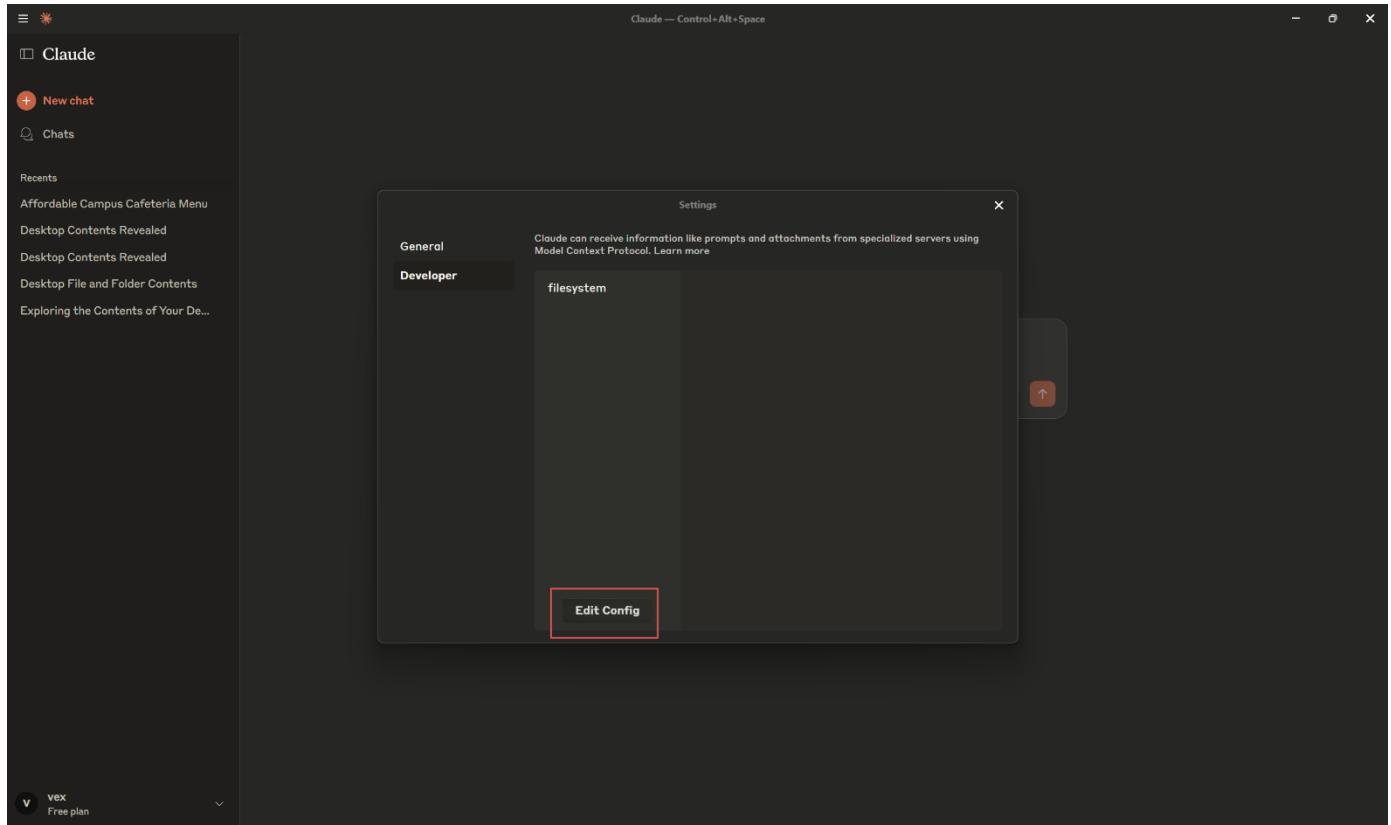
# Initialize and run the server using the chosen transport
# --- Corrected part ---
# Build keyword arguments dynamically
run_kwargs = {"transport": args.transport}
if args.transport == "sse":
    print(f"SSE Port: {args.port}", file=sys.stderr)
    run_kwargs["port"] = args.port
# --- End corrected part ---

# Initialize and run the server using the chosen transport and conditional
kwargs
try:
    mcp.run(**run_kwargs) # Use ** to unpack the arguments
except Exception as e:
    print(f"\nServer exited with error: {e}", file=sys.stderr)
    # Optionally print traceback for more details during debugging
    # import traceback
    # traceback.print_exc(file=sys.stderr)
    sys.exit(1)

```

disini digunakan claude ai desktop sebagai MCP client.

Untuk menjalankan MCP server buka claude desktop kemudian file lalu settings. Setelah windows setting muncul pilih developer lalu klik edit config



Setelah itu pilih claudes_desktop_config.json

Name	Date modified	Type	Size
Crashpad	08/06/2025 01:01	File folder	
DawnGraphiteCache	10/04/2025 09:44	File folder	
DawnWebGPUCache	10/04/2025 09:44	File folder	
GPU Cache	10/04/2025 09:44	File folder	
IndexedDB	08/06/2025 00:57	File folder	
Local Storage	10/04/2025 09:44	File folder	
logs	08/06/2025 01:01	File folder	
Network	08/06/2025 01:00	File folder	
sentry	08/06/2025 00:59	File folder	
Service Worker	08/06/2025 00:57	File folder	
Session Storage	08/06/2025 00:57	File folder	
Shared Dictionary	10/04/2025 09:44	File folder	
WebStorage	10/04/2025 10:32	File folder	
claude_desktop_config.json	24/04/2025 13:34	JSON Source File	1 KB
DIPS	08/06/2025 01:01	File	36 KB
Local State	10/04/2025 09:44	File	1 KB
lockfile	08/06/2025 00:57	File	0 KB
Preferences	08/06/2025 01:01	File	1 KB
SharedStorage	10/04/2025 09:46	File	4 KB
SharedStorage-wal	24/04/2025 12:32	File	0 KB
window-state.json	08/06/2025 01:01	JSON Source File	1 KB

Setelah itu ganti path yang terdapat pada gambar dibawah sesuai dengan lokasi tempat mcp server tersimpan

```
{
  "mcpServers": {
    "filesystem": {
      "//": "Use 'python' or the full path to your python executable",
      "command": "python",
      "args": [
        "C:\\\\Users\\\\Lenovo\\\\Desktop\\\\kuliah\\\\mcp\\\\filesystem.py",
        "C:\\\\Users\\\\Lenovo\\\\Desktop"
      ]
    }
  }
}
```

"C:\\\\Users\\\\Lenovo\\\\Desktop\\\\kuliah\\\\mcp\\\\filesystem.py" adalah direktori untuk code python mcp server

"C:\\\\Users\\\\Lenovo\\\\Desktop" adalah direktori yang diakses oleh claude desktop

Setelah itu Kembali ke claude desktop dan claude desktop dapat mengetahui file dan folder

apa saja yang terdapat di desktop, bisa memindahkan file dan membaca isi file dengan menggunakan prompt dari user sebagai berikut :

The screenshot shows a dark-themed conversational interface. At the top, a message from the user says "what is on my desktop". Below it, the AI responds with "I'll help you find out what's on your desktop. Let me check the contents for you." A dropdown menu shows the command "list_allowed_directories". The AI then asks "Let me list what's on your desktop:" and shows another dropdown with "list_directory". It replies with "Here's what I found on your desktop:". Under "Files:", it lists "desktop.ini". Under "Folders:", it lists "games", "kuliah", "office", and "tools". Finally, it asks if the user wants more details about these folders.

v what is on my desktop

Edit

I'll help you find out what's on your desktop. Let me check the contents for you.

L list_allowed_directories

Let me list what's on your desktop:

L list_directory

Here's what I found on your desktop:

Files:

- desktop.ini (a system file that stores desktop configuration information)

Folders:

- games
- kuliah
- office
- tools

Would you like me to explore the contents of any of these folders to give you more details?

Retry



move menu-kantin.csv to kuliah folder

I'll look for menu-kantin.csv on your desktop and move it to the kuliah folder. Let me check if the file exists first.



search_files



I found two files named menu-kantin.csv:

1. One directly on your desktop: C:\Users\Lenovo\Desktop\menu-kantin.csv
2. One already in your kuliah folder: C:\Users\Lenovo\Desktop\kuliah\petranesian-lapar\docs\menu-kantin.csv

Let me move the one from your desktop to the kuliah folder:



move_file



I've successfully moved menu-kantin.csv from your desktop to the kuliah folder. The file has been moved to:

C:\Users\Lenovo\Desktop\kuliah\menu-kantin.csv

Note that there's also another copy of this file in a subfolder:

C:\Users\Lenovo\Desktop\kuliah\petranesian-lapar\docs\menu-kantin.csv

Retry



from menu-kantin.csv on my desktop search food below 20000rp

Edit

I'll search for food items below 20,000 Rp in the menu-kantin.csv file on your desktop. Let me read that file first.

R read_file

I need to check which directories I have access to, then look for your file.

L list_allowed_directories

Now let me read the menu-kantin.csv file from your desktop:

R read_file

Here are all the food items that cost below 20,000 Rp from the menu-kantin.csv file:

1. Soto Ayam Jago (Gedung P):

- Soto ayam biasa - Rp16,000
- Soto ayam Special - Rp18,000
- Soto telor biasa/asin - Rp13,000
- Soto telor muda uritan - Rp16,000
- Soto tempe + tahu crispy - Rp13,000
- Soto daging sapi - Rp18,000
- Nasi uduk telor dadar - Rp12,000
- Nasi uduk tahu tempe - Rp12,000
- Nasi uduk 3T (tahu tempe telor) - Rp16,000
- Nasi goreng jawa/mawut - Rp16,000
- Nasi mie sayur - Rp16,000

Contoh lain dari mcp server

Contoh penggunaan konsep mcp untuk melihat ketersediaan penginapan Airbnb menggunakan claude desktop

<https://github.com/openbnb-org/mcp-server-airbnb>

pada contoh berikut dengan menambahkan codingan ini ke claude_desktop_config.json pada claude desktop

```
{  
  "mcpServers": {  
    "airbnb": {  
      "host": "http://127.0.0.1:8080"  
    }  
  }  
}
```

```

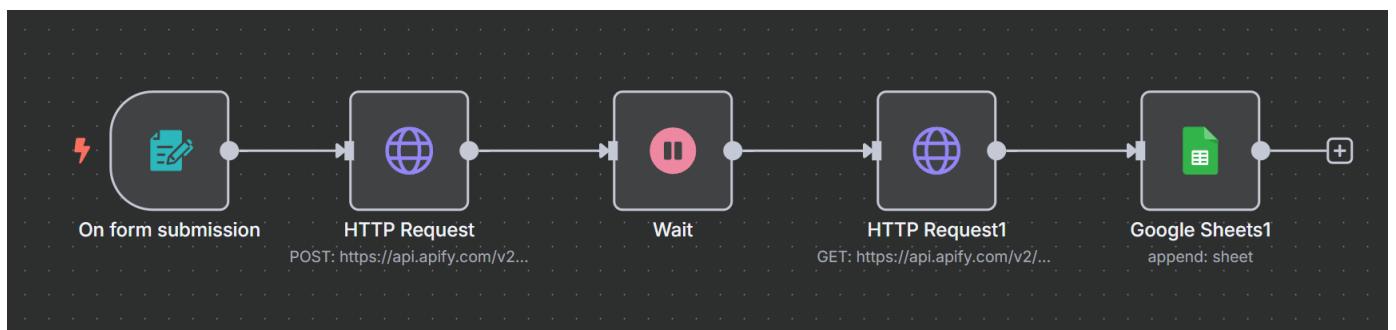
        "command": "npx",
        "args": [
            "-y",
            "@openbnb/mcp-server-airbnb"
        ]
    }
}
}
}

```

3.5.4 Percobaan N8N workflow untuk webscraping dan pdf summary telegram

Webscrapping menggunakan google maps

Alur/workflow webscraping



Untuk alur webscraping digunakan API service dari apify untuk mencari data informasi Lokasi yang diinginkan hasil pencarian disimpan di google sheets. Referensi lebih lanjut
https://www.youtube.com/watch?v=zULdrUFmpts&ab_channel=JulianGoldieSEO
<https://x.com/JulianGoldieSEO/status/1927983251815289296>

1. on form submission

On form submission digunakan untuk memasukan tempat yang dicari misalnya café atau restoran dan negara dan kota tempat tersebut yang diinginkan

On form submission

Test step

Parameters Settings Docs ↗

Form URLs

Test URL Production URL

https://n8n-1.saturn.petra.ac.id/form-test/379262b4-7c00-4479-b160-772fa4e35559

Authentication

None

Form Title

place to search

Form Description

e.g. We'll get back to you soon

Form Elements

Field Name

place you want to search

Element Type

Text

Placeholder

Required Field

On form submission

Parameters Settings Docs

Placeholder

Required Field

Field Name

country, city

Element Type

Text

Placeholder

Required Field

Add Form Element

Respond When

Form Is Submitted

Build multi-step forms by adding a form page later in your workflow

Options

No properties

Add option

This screenshot shows the n8n interface for creating a workflow. The current step is titled "On form submission". It contains two form elements: one for "place to search" and another for "country, city", both set to required fields. The "Element Type" for both is "Text". There are sections for "Placeholder" and "Field Name" for each element. The "Respond When" condition is set to "Form Is Submitted". The "Options" section indicates "No properties". A button labeled "Add option" is available for further configuration.

place to search - Brave

n8n-1.saturn.petra.ac.id/form-test/379262b4-7c00-4479-b160-772fa4e35559

This is a test version of your form

place to search

place you want to search *

country, city *

Submit

Form automated with  n8n

This screenshot shows the test version of the form created in the n8n interface. It features two input fields: "place to search" and "country, city", both marked as required with red asterisks. Below the fields is a large orange "Submit" button. At the top of the form area, there is an orange box containing the text "This is a test version of your form". The browser's address bar at the top shows the URL and the Brave browser icon.

2. http request dipakai untuk mengumpulkan data webscrapping dari google maps menggunakan service dari apify.com

Setelah mengisi form input, data dari form akan diinput ke HTTP request

The screenshot shows the configuration for an HTTP Request step in the Apify Compass crawler. The 'Method' is set to 'POST'. The 'URL' is https://api.apify.com/v2/acts/compass~crawler-google-places/runs?token=apify_api_kYZt5qLxPtC. The 'Authentication' is set to 'None'. The 'Send Query Parameters' switch is off. The 'Send Headers' switch is off. The 'Send Body' switch is on. The 'Body Content Type' is set to 'JSON'. Under 'Specify Body', it is set to 'Using JSON'. The JSON body is defined as follows:

```
{  
  "includeWebResults": false,  
  "language": "en",  
  "locationQuery": "{{ $json['country', 'city'] }}",  
  "maxCrawledPlacesPerSearch": 50,  
  "maxImages": 0  
}
```

Isi json

```
{  
  "includeWebResults": false,  
  "language": "en",  
  "locationQuery": "{{ $json['country', 'city'] }}",  
  // negara dan kota  
  "maxCrawledPlacesPerSearch": 50,  
  "maxImages": 0,  
  "maximumLeadsEnrichmentRecords": 0,  
  "scrapeContacts": false,  
  "scrapeDirectories": false,  
  "scrapeImageAuthors": false,  
  "scrapePlaceDetailPage": false,
```

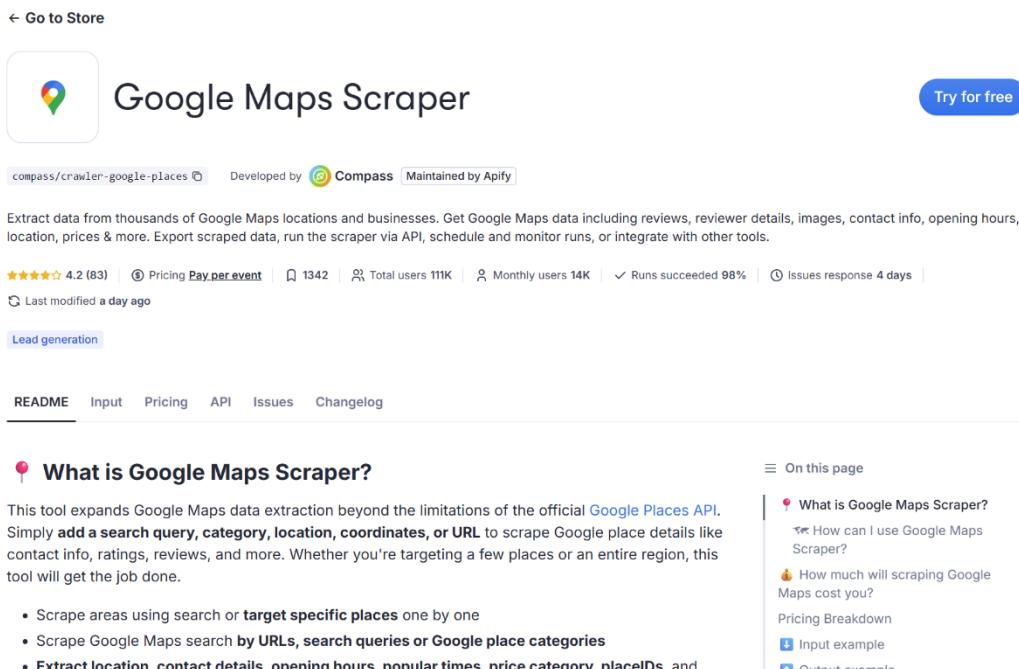
```

    "scrapeReviewsPersonalData": true,
    "scrapeTableReservationProvider": false,
    "searchStringsArray": [
        "{{ $json['place you want to search'] }}"
    // tempat yang ingin dicari
    ],
    "skipClosedPlaces": false
}

```

Json dan url tersebut bisa didapat dari <https://apify.com/compass/crawler-google-places>

Langkah mendapat api dan json
 buatlah akun di apify.com
 lalu klik try for free akan diberikan saldo limit 5\$ per bulan



The screenshot shows the Apify Marketplace page for the "Google Maps Scraper" tool. At the top, there's a "Go to Store" button and a "Try for free" button. Below the title "Google Maps Scraper" is a description: "Extract data from thousands of Google Maps locations and businesses. Get Google Maps data including reviews, reviewer details, images, contact info, opening hours, location, prices & more. Export scraped data, run the scraper via API, schedule and monitor runs, or integrate with other tools." There are also statistics: 4.2 (83) reviews, Pay per event pricing, 1342 runs, 111K total users, 14K monthly users, 98% success rate, and 4 days response time. A note says "Last modified a day ago". Below the stats are links for "Lead generation", "README", "Input", "Pricing", "API", "Issues", and "Changelog". On the right, there's a sidebar titled "On this page" with links to "What is Google Maps Scraper?", "How can I use Google Maps Scraper?", "How much will scraping Google Maps cost you?", "Pricing Breakdown", "Input example", and "Output example".

Kemudian isi search term

Input Information Runs 19 Builds 173 Integrations 0 Monitoring Issues 4 Saved tasks 0 Reviews

Manual JSON

Our base charge for Google Maps is just \$4 per 1,000 places. See detailed prices [here](#). To extract place data or contact details from Google Places, simply Enter **Search term**, add **Location**, and **Number of places** to extract. The section **Search filters & categories** contains various extra features, filters, and sorting options. These are an additional cost, and the (\$) sign indicates filters or options that incur an incremental cost to the run. You also have the option to customize using various filters and add additional data like reviews or images, which are charged at an additional cost and are marked by the (\$) sign below.

You can find more details about the pricing [here](#).

Sections with asterisk* are just alternative ways to start the input (**Geolocation parameters**, **Polygons**, **URLs**). They can be combined with any of the features and sorting options from the **Filters** section

Search term(s) (optional) [?](#)

1 restaurant X

+ Add Bulk edit Remove empty fields

Location (use only one location per run) (optional) [?](#)

New York, USA

Number of places to extract (per each search term or URL) (optional) [?](#)

50 + -

Language (optional) [?](#)

English X | ▼

Setelah mengisi search term klik json lalu copy dan masukkan ke n8n di json http request

Manual JSON

```

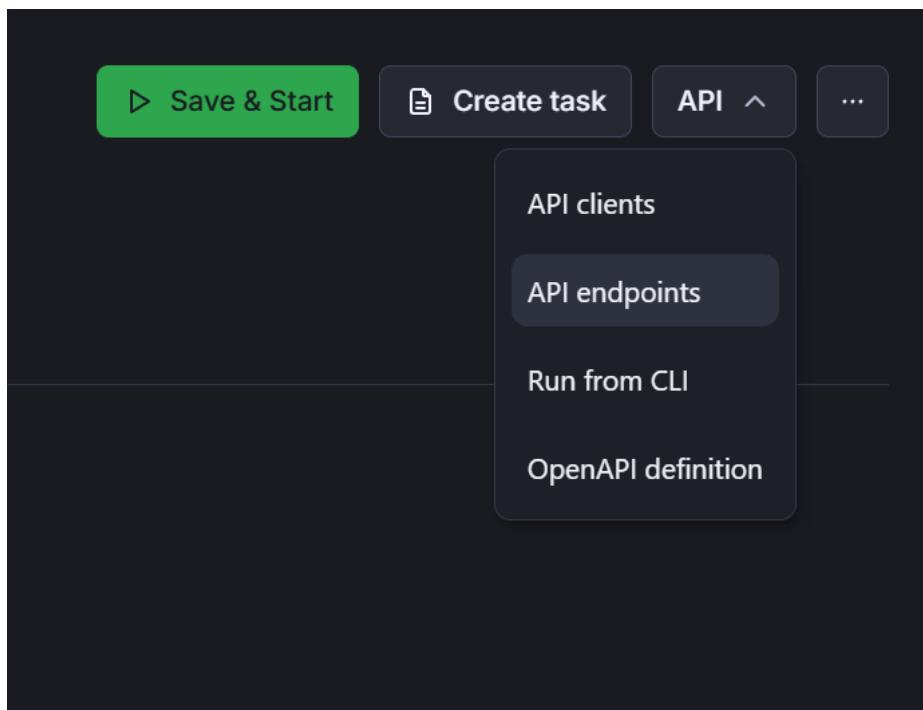
1 { "includeWebResults": false,
2   "language": "en",
3   "locationQuery": "New York, USA",
4   "maxCrawledPlacesPerSearch": 50,
5   "maxImages": 0,
6   "maximumLeadsEnrichmentRecords": 0,
7   "scrapeContacts": false,
8   "scrapeDirectories": false,
9   "scrapeImageAuthors": false,
10  "scrapePlaceDetailPage": false,
11  "scrapeReviewsPersonalData": true,
12  "scrapeTableReservationProvider": false,
13  "searchStringsArray": [
14    "restaurant"
15  ],
16  "skipClosedPlaces": false
17 }
18 }
```

Edit json sesuai dengan form submission

```
"locationQuery": "{{ $json['country', 'city'] }}",
// negara dan kota
```

```
"searchStringsArray": [
    "{{ $json['place you want to search'] }}"
// tempat yang ingin dicari
],
```

Setelah itu kembali ke apify lalu klik api lalu api endpoint (untuk menyiapkan endpoint access)

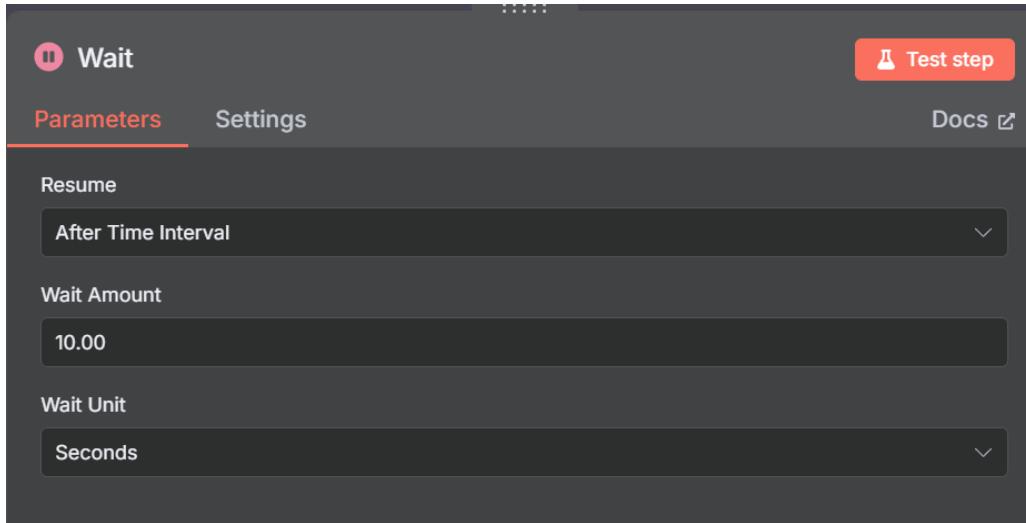


Terakhir Copy api run actor synchronously lalu masukan ke url http request

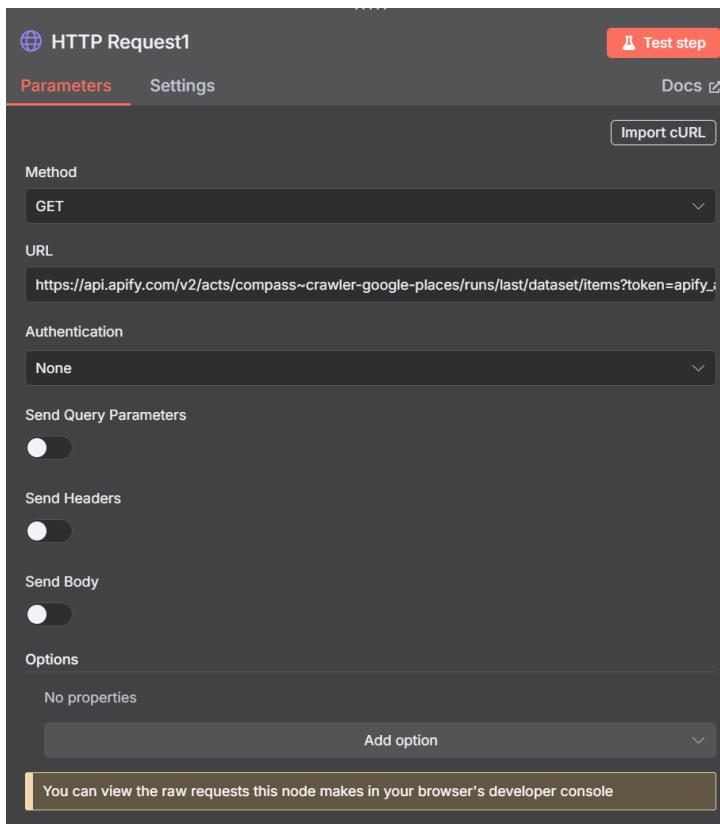
This is a screenshot of an API reference page for the 'Run Actor synchronously' endpoint. The page has a dark header with the endpoint name and a 'View API reference' link. Below the header, there is a detailed description of the endpoint: 'Runs this Actor and waits for it to finish. The POST payload including its Content-Type is passed as INPUT to the Actor (usually application/json) and the OUTPUT is returned in the HTTP response. The Actor is started with the default options; you can override them using various URL query parameters. Note that long HTTP connections might break.' The entire page is enclosed in a dark box.

3. wait

wait bertujuan agar memberi waktu agar webscraping yang dilakukan oleh apify dapat mengumpulkan data seluruhnya sebelum di simpan. Jika tidak diberikan waktu wait yang cukup (contoh: 10 detik) maka data tidak akan tersimpan.



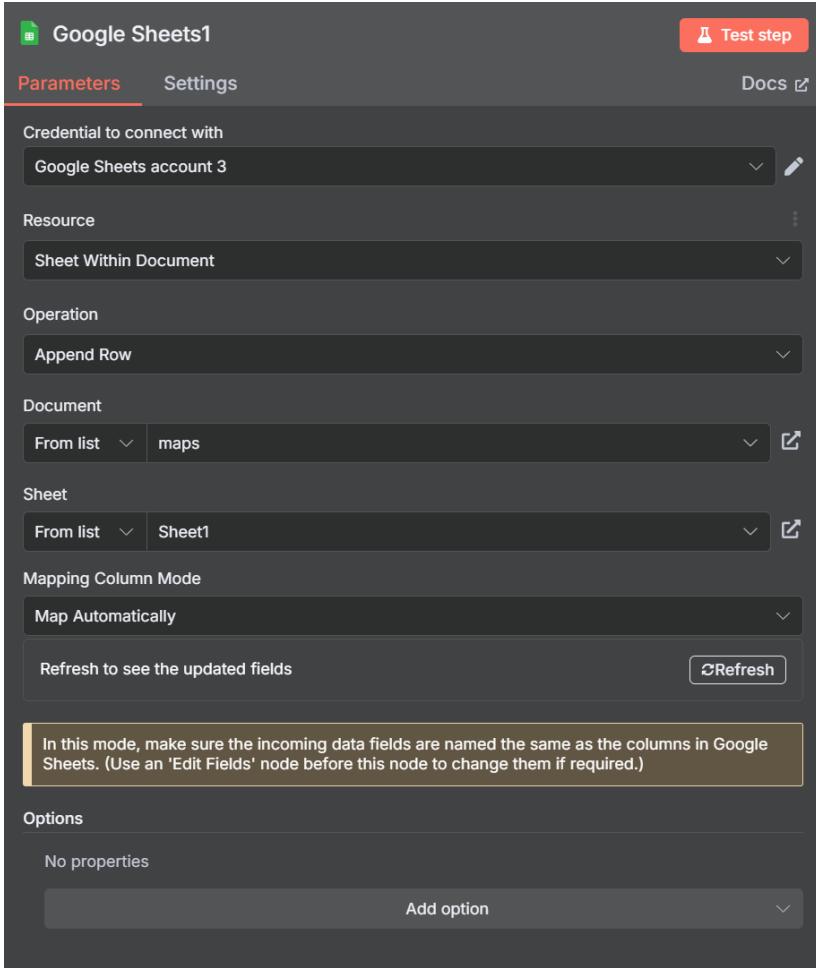
4. http request mengambil hasil webscraping dari apify



Untuk http request mengambil hasil webscraping menggunakan api url get last run dataset items yang terdapat di apify



5. menyimpan data di google sheets



Untuk menggunakan google api service mengakses google sheets dibutuhkan credential google sheets di <https://console.cloud.google.com/>

Pertama cari google sheets api terus pilih OAuth Client ID lalu pilih web application lalu beri nama

[← Create OAuth client ID](#)

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information. [Learn more](#) about OAuth client types.

Application type *
Web application

Name *
Web client 5

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

i The domains of the URLs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#).

Kemudian Kembali ke n8n pilih node google sheets lalu pilih create new credential

Google Sheets1

Test step

Parameters Settings Docs

Credential to connect with

Google Sheets account 3

Google Sheets account
Google Sheets OAuth2 API

Google Sheets account 2
Google Sheets OAuth2 API

Google Sheets account 3
Google Sheets OAuth2 API

+ Create new credential

Lalu copy OAuth Redirect URL

OAuth Redirect URL

https://n8n-1.saturn.petra.ac.id/rest/oauth2-oidc/authorize?client_id=...&redirect_uri=https%3A%2F%2Fn8n-1.saturn.petra.ac.id%2Fcallback&response_type=code&scope=openid+email+profile

In Google Sheets, use the URL above when prompted to enter the URL.

Kembali ke console google cloud lalu tambahkan url authorized redirect urls lalu create

Authorized redirect URLs ?

For use with requests from a web server

+ Add URI

Di google sheets api klik credential kemudian di oauth 2.0 client id pilih nama api yang tadi sudah dibuat

 **Google Sheets API**

Reads and writes Google Sheets.

By Google Enterprise API [?](#)

Service name	Type
sheets.googleapis.com	Public API

Metrics Quotas & System Limits **Credentials**

Credentials compatible with this API

To view all credentials visit [Credentials in APIs & Services](#)

OAuth 2.0 Client IDs

<input type="checkbox"/>	Name	Creation date
<input type="checkbox"/>	gmail	May 22, 2025
<input type="checkbox"/>	sheet	May 22, 2025
<input type="checkbox"/>	drive	May 22, 2025
<input type="checkbox"/>	web	May 22, 2025

Copy client id dan client secrets

Client ID

Creation date

Last used date

⚠️ Inactive OAuth clients are subject to deletion due to inactivity, a

Client secrets

If you are in the process of changing your client secret, do not

Client secret

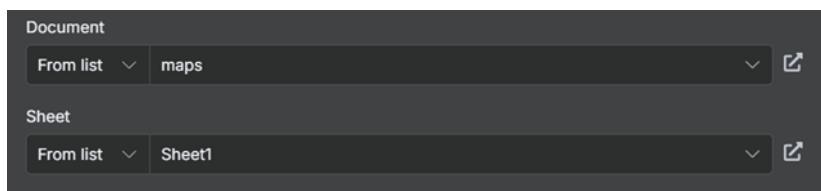
Creation date

Status

Kembali ke n8n credential google sheets lalu masukan yang telah di copy

The screenshot shows the 'Client ID *' field with its input field empty. Below it is the 'Client Secret *' field, also with its input field empty.

Setelah itu buatlah google sheets lalu setting sesuai nama google sheets yang telah dibuat

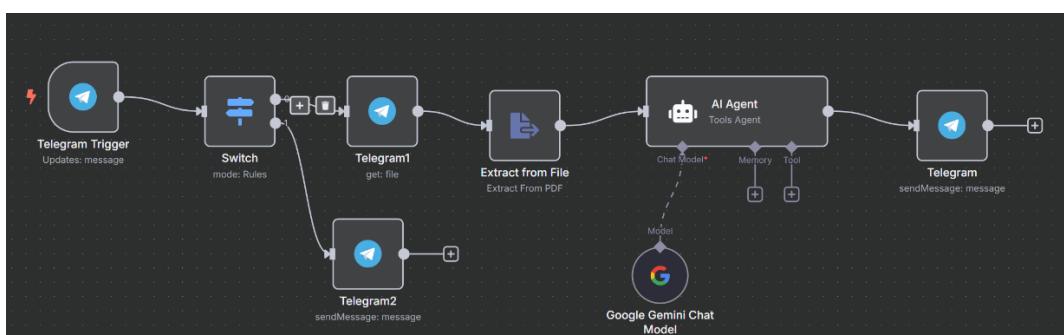


Ketika workflow webscraping n8n di run, maka data hasil scrapping akan tersimpan ke dalam google sheets seperti contoh berikut : (mencari restoran di Surabaya)

A	B	C	D	E	F	G	H	I	J
1	price	categoryName	address	neighborhood	street	city	postalCode	state	countryCode
2	Michi Soba	Japanese restau	Jl. Balas Klumpr	Balas Klumpr	Jl. Balas Klumpr	Surabaya	60222	East Java	ID
3	Kedai Azka Makanan dan Minum	Restaurant	Jl. DUKUH GEM	Balas Klumpr	Jl. DUKUH GEM	Surabaya	60222	East Java	ID
4	Alira Food	Brunch restaura	Jalan, Pesopen	Sumur Welut, Lt	Jalan, Pesopen	Surabaya	60215	East Java	ID
5	Bu ratna	Brunch restaura	Jl. Sumberan Nc	Balas Klumpr	Jl. Sumberan Nc	Surabaya	60222	East Java	ID
6	Soto Ayam Lamongan Cak Paat	Brunch restaura	Jl. Balas Klumpr	Balas Klumpr	Jl. Balas Klumpr	Surabaya	60222	East Java	ID
7	Hisana Fried Chicken - Balas Klui	Fast food restau	MM9Q+9RP, Jl.	Balas Klumpr	MM9Q+9RP, Jl.	Surabaya	60222	East Java	ID
8	Warung Eling 1	Brunch restaura	Jl. Romokalisari	Segeromadu, R	Jl. Romokalisari	Surabaya	60192	East Java	ID
9	Bubur Ayam Mang Arlef	Breakfast restau	RJ2W+WQ5, Jl.	Tenggulungan, Kr	RJ2W+WQ5, Jl.	Gresik Regency	60192	East Java	ID
10	Warung Mantab	Restaurant	RJ2W+R95, Unr	Romokalisari, Bk	RJ2W+R95, Unr	Surabaya	60192	East Java	ID
11	SOTO AYAM BU MUJIATI	Soto ayam resta	RJ2X+G4R, Ror	Romokalisari, Bx	RJ2X+G4R	Surabaya	60192	East Java	ID
12	Lontong Balap Cak Sugeng	Indonesian resta	Jl. Klumpr Pda	Balas Klumpr	Jl. Klumpr Pda	Surabaya	60222	East Java	ID
13	Mie pangsit bd	Noodle shop	Jl. Semeru II Per	Bambu, Lakarsa Jl.	Semeru II Per	Gresik Regency		East Java	ID
14	Bakso Solo dan mie ayam ceker	Restaurant	No.mor 94A, Jl.	Balas Klumpr	No.mor 94A, Jl.	Surabaya	60222	East Java	ID
15	Street Sushi Gra \$	Sushi restaura	Soho Grand Har	Balas Klumpr	Soho Grand Har	Surabaya	60222	East Java	ID
16	Bubur Jagung Bama	Porridge restaur	Jl. Karang Kium	Balas Klumpr	Jl. Karang Kium	Surabaya	60222	East Java	ID
17	Pentol Santang	Bakso restaura	Jl. Raya Bangkir	Bangkingan, La	Jl. Raya Bangkir	Surabaya	60214	East Java	ID
18	Tahu Go Balas Klumpr	Fast food resta	Balas Klumpr	1Balas Klumpr	Balas Klumpr	1Surabaya	60222	East Java	ID
19	Pos Makan Labt \$	Javanese restau	Romokalisari IV	Romokalisari, Bx	Romokalisari IV	Surabaya	60192	East Java	ID
20	Depot Barokah	Javanese resta	Terminal Tambak	Tambak Osowila	Terminal Tambak	Surabaya	60191	East Java	ID
21	Bakso Cak Kin	Bakso restaura	Jl. Raya Tambak	Tambakdono, P	Jl. Raya Tambak	Surabaya	60193	East Java	ID
22	WAROENG MANDOR (Warunge	Nasi restaura	Pergudangan bu	Romokalisari, Bx	Pergudangan bu	Surabaya	60192	East Java	ID
23	Warung Makan "DAFA MAKKY "	Coffee store	Jl. Tambak Osov	Tambak Osowila Jl.	Tambak Osov	Surabaya	60191	East Java	ID
24	Warung Mbak In \$	Brunch restaura	Jl. Tambak Osov	Tambak Osowila Jl.	Tambak Osov	Surabaya	60191	East Java	ID
25	Warung sambel uleg DUA PUTRI	Nasi restaura	RJ2X+C4X, Ron	Romokalisari, Bx	RJ2X+C4X	Surabaya	60192	East Java	ID
26	WM. bu sulastri	Javanese restau	Jl. Raya No.35,	'Tambak Osowila Jl. Raya No.35	Jl. Raya No.35	Surabaya	60191	East Java	ID
27	Warung Rica-Rica Enthog	Nasi restaura	Jl. Tambak Osov	Tambak Osowila Jl.	Tambak Osov	Surabaya		East Java	ID
28	Depot & Mie Ayam Double A	Noodle shop	RJ3W+C59, Unr	Segoromadu, Kr	RJ3W+C59, Unr	Gresik Regency	60192	East Java	ID
29	Soto Ayam Lamongan "PAKDE S	Soto ayam resta	Tambak Osowila Tambak	Oswollangun, Benowo	Tambak Osowila	Surabaya	60191	East Java	ID
30	Bakso Wilangan	Bakso restaura	Tambak Osowila Tambak	Oswollangun, Benowo	Tambak Osowila	Surabaya	60191	East Java	ID
31	warung DELISHA	Snack bar	Jl. Tambak Osov	Romokalisari, B	Jl. Tambak Osov	Surabaya	60192	East Java	ID
32	Tahu Walki Pentol Harvest Story	Fast food resta	Harvest Story, P	Balas Klumpr	Harvest Story, P	Surabaya	60222	East Java	ID
33	DimsumQ	Frozen food sto	De Alamuda Rej	Balas Klumpr	De Alamuda Rej	Surabaya	60222	East Java	ID
34	Chicken Gueprex	Chicken restaura	Jl. Balas Klumpr	Balas Klumpr	Jl. Balas Klumpr	Surabaya	60222	East Java	ID
35	NASI KUNING AZA	Indonesian resta	MM9V+M2M, Be	Balas Klumpr	MM9V+M2M	Surabaya	60222	East Java	ID
36	Warung Nasi	Nasi restaura	Jl. Mastrip IX No	Warugunung, Kr	Jl. Mastrip IX No	Surabaya	60221	East Java	ID
37	WARUNG RUJAK BU MIA	Javanese restau	Jl. Raya Bangkir	Sumur Welut, Lt	Jl. Raya Bangkir	Surabaya	60241	East Java	ID
38	SWK Balas Klumpr	Bakso restaura	MM9R+G24, Jl.	Balas Klumpr	MM9R+G24, Jl.	Surabaya	60222	East Java	ID

N8N workflow untuk Pdf summary di telegram

Alur Pdf summary telegram



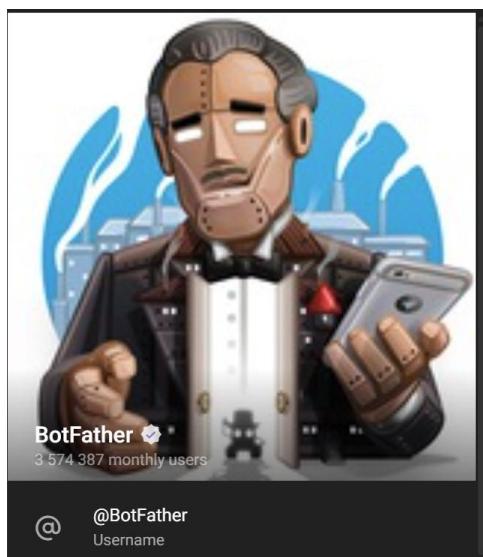
Workflow telegram ini dirancang untuk memproses pesan, mengekstrak informasi dari file terlampir (khususnya PDF), dan kemudian memberikan respons menggunakan agen AI Google Gemini Chat Model.

1. telegram trigger

The screenshot shows a configuration interface for a 'Telegram Trigger'. At the top, there's a 'Test step' button. Below it, tabs for 'Parameters' (which is selected) and 'Settings' are visible, along with a 'Docs' link. Under 'Parameters', the 'Webhook URLs' section is expanded. It includes a dropdown for 'Credential to connect with' set to 'Telegram account', a note about using one trigger per bot, and a 'Trigger On' dropdown set to 'Message'. A note below states that uploaded attachments trigger separate events. Under 'Additional Fields', a 'Download Images/Files' toggle is turned on. There's also an 'Add Field' button.

Untuk mendapatkan credential telegram

Di telegram carilah botfather

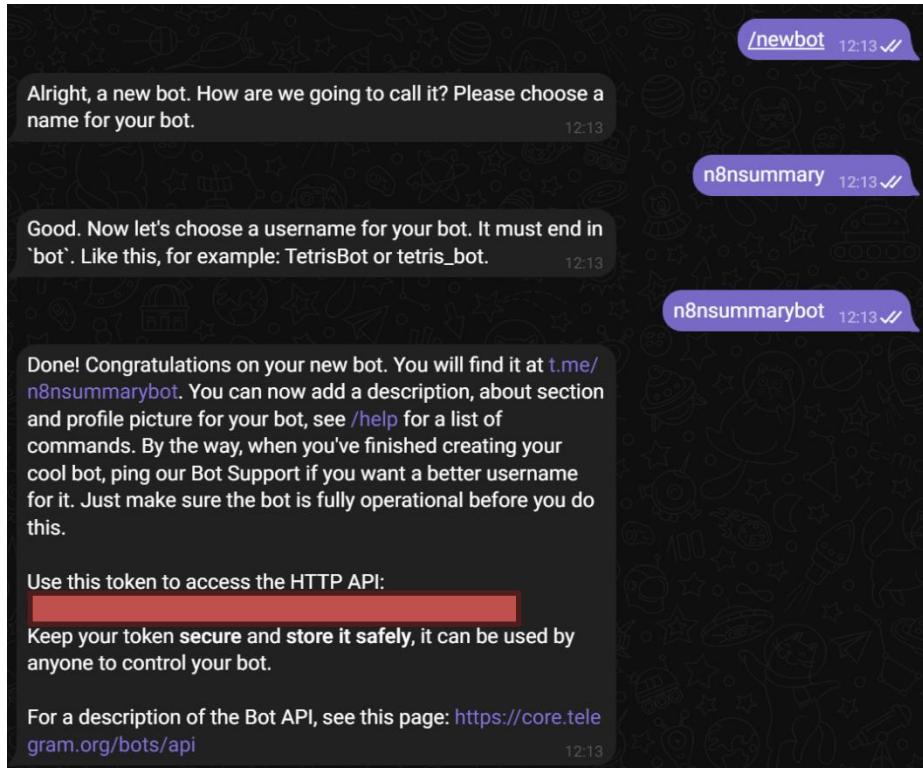


Setelah itu /start pada chat botfather

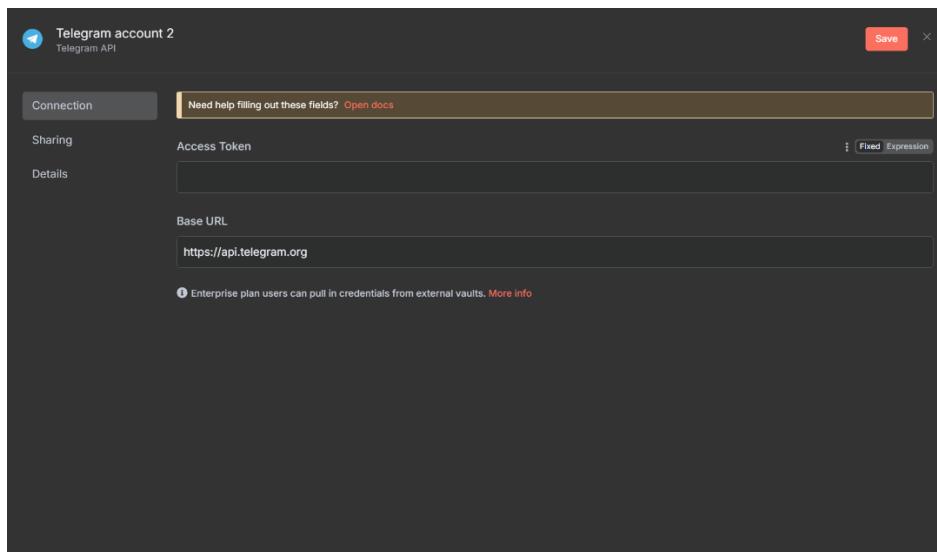
Kemudian /newbot chat botfather

Setelah itu berinama chat telegram untuk bot kemudian beri username. Untuk username harus berakhiran bot

Setelah itu copy token access http api



Terus Kembali ke n8n node telegram, create new credential lalu masukan api ke access token



2. switch

Switch berguna untuk mengecek apakah file tersebut pdf atau bukan

Switch

Parameters **Settings** **Docs** **Test step**

Mode: Rules

Routing Rules

`fx {{ $json.message.document.mime_type }}` A is equal to application/pdf

Rename Output:

`fx {{ $json.message.document.mime_type }}` A is not equal to application/pdf

Rename Output:

Add Routing Rule

Convert types where required:

Options

No properties **Add option**

3. telegram 1 (get files)

Jika file tersebut adalah pdf maka akan mengambil file tersebut dari pesan yang kita kirim

Telegram1

Parameters **Settings** **Docs** **Test step**

Credential to connect with: Telegram account

Resource: File

Operation: Get

File ID: `fx {{ $json.message.document.file_id }}`

Download: Fixed Expression

4. telegram 2 (send message)

Jika bukan maka akan mengirim pesan untuk minta pdf

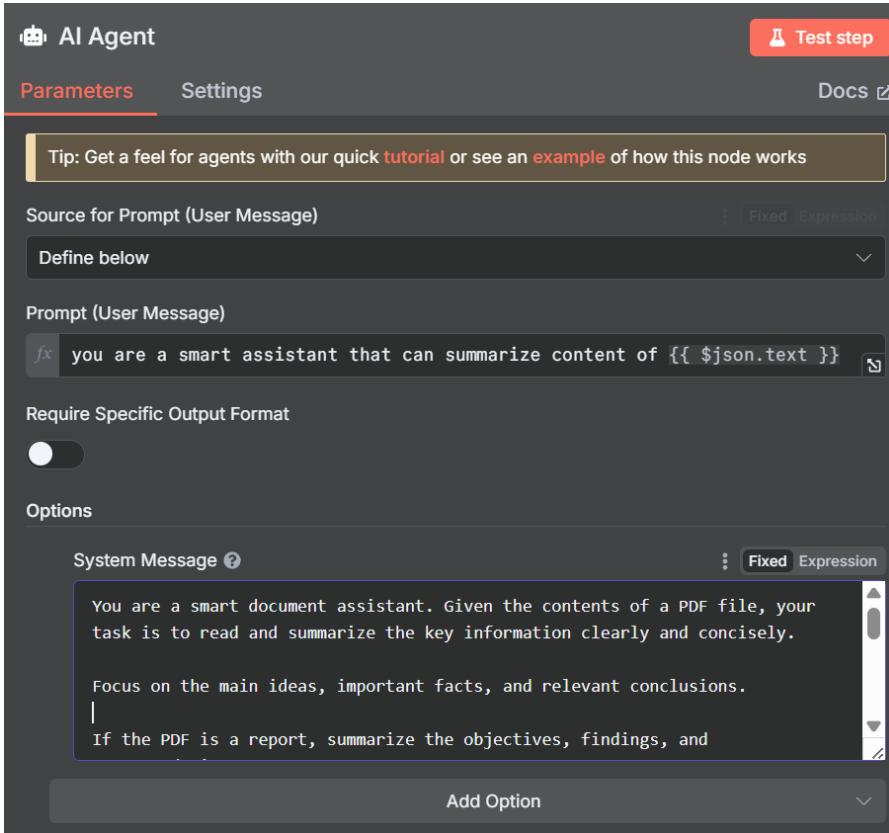
The screenshot shows a step configuration interface for a workflow. The title bar says "Telegram2". The "Parameters" tab is selected. The "Operation" dropdown is set to "Send Message". The "Resource" dropdown is set to "Message". The "Chat ID" field contains the expression `{{ \$('Telegram Trigger').item.json.message.from.id }}`. The "Text" field contains the expression `please send a pdf`. There is a "Reply Markup" section with a "None" dropdown. The "Additional Fields" section shows "No properties" and an "Add Field" button.

5. extract from file

Merubah file pdf menjadi binary data

The screenshot shows a step configuration interface for a workflow. The title bar says "Extract from File". The "Parameters" tab is selected. The "Operation" dropdown is set to "Extract From PDF". The "Input Binary Field" dropdown is set to "data". A note below says "The name of the input binary field containing the file to be extracted". There is an "Options" section with "No properties" and an "Add option" button.

6. Ai agent



System prompt:

You are a smart document assistant. Given the contents of a PDF file, your task is to read and summarize the key information clearly and concisely.

Focus on the main ideas, important facts, and relevant conclusions.

If the PDF is a report, summarize the objectives, findings, and recommendations.

If it's a research paper, summarize the abstract, methodology, results, and conclusion.

If it's a manual or guide, provide a section-by-section summary highlighting critical instructions or features.

Output the summary in clear bullet points or short paragraphs, depending on the document type.

Ignore repetitive footers/headers, page numbers, or metadata unless they contain relevant content.

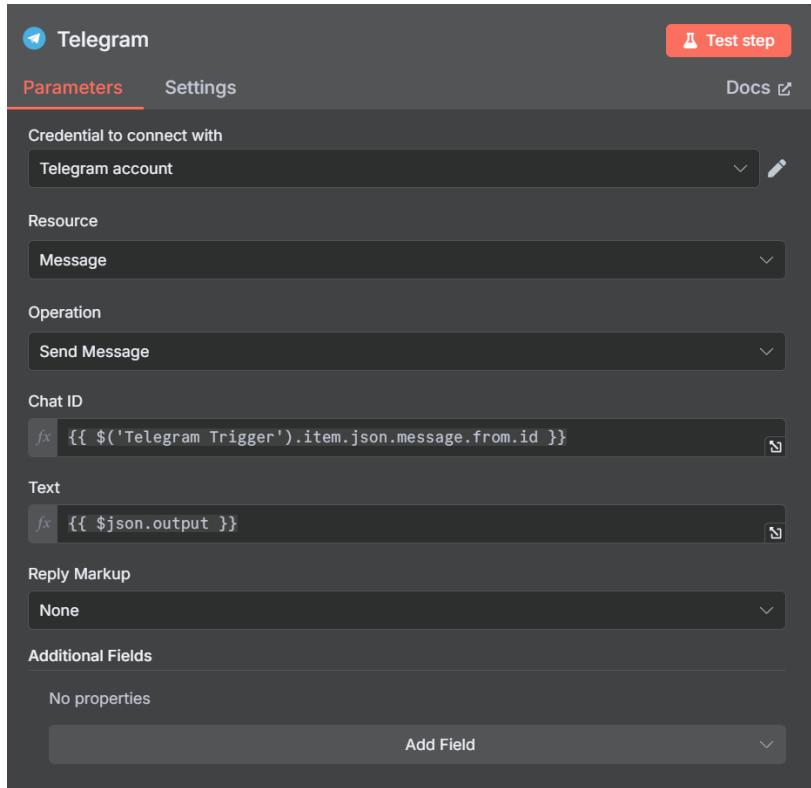
limit it under 4,096 characters

6. google gemini

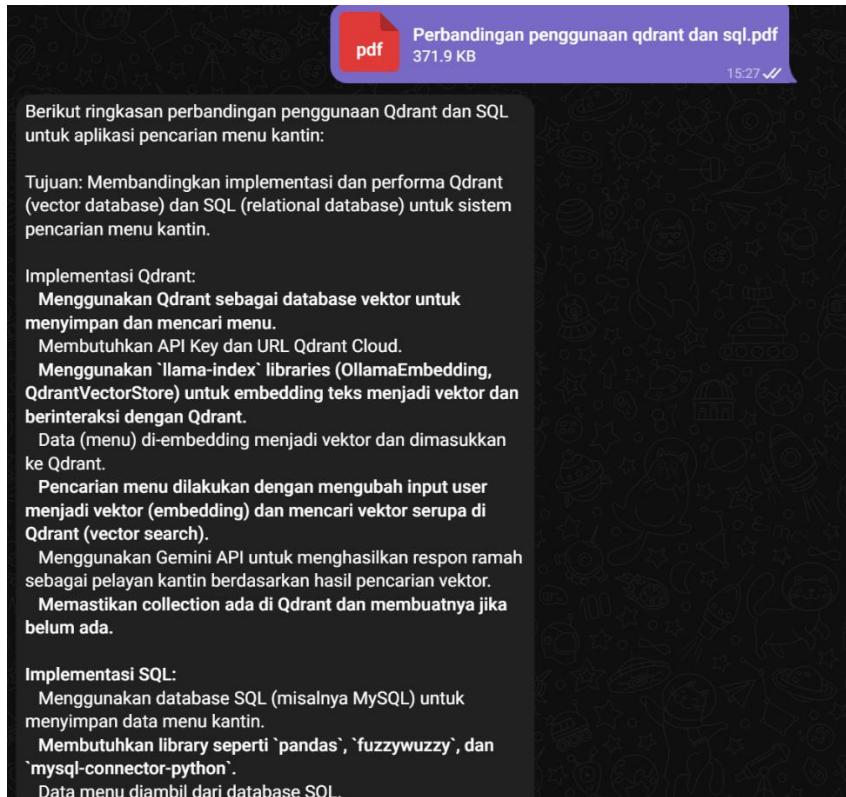
The screenshot shows the 'Google Gemini Chat Model' configuration screen. At the top, there are tabs for 'Parameters' (which is selected), 'Settings', and 'Docs'. Below the tabs, there's a section for 'Credential to connect with' set to 'Google Gemini(PaLM) Api account'. Under 'Model', the selection is 'models/gemini-2.5-flash-preview-04-17'. In the 'Options' section, it says 'No properties' and has a button to 'Add Option'.

7. telegram (send message) output

Mengirim output dari hasil summary



Hasil output summary dari chatbot di telegram, dengan memberikan memberikan file pdf telegram menghasilkan respon berupa ringkasan dari isi file tersebut :



4. KESIMPULAN

Pengalaman Industrial (LEAP) di PTIK Petra berfokus pada eksperimentasi dan pengembangan chatbot Generative AI, memberikan wawasan mendalam tentang penerapan teknologi kecerdasan buatan dalam skala praktis. Sepanjang program, saya berhasil memperdalam kemampuan dalam pemrograman Python dan mengaplikasikannya dalam proyek-proyek nyata, yang secara efektif menjembatani kesenjangan antara teori dan praktik.

Proyek-proyek utama yang dikerjakan mencakup pengembangan chatbot untuk Kantin Petra yang memanfaatkan Ollama sebagai LLM dan Qdrant untuk pencarian data, memungkinkan pemesanan makanan yang lebih efisien dengan tampilan gambar dan detail harga. Selain itu, pengembangan chatbot pencari pekerjaan yang mengimplementasikan tools calling menunjukkan bagaimana AI dapat memperluas kapabilitasnya untuk berinteraksi dengan sumber eksternal seperti JobStreet.

Aspek penting lainnya adalah eksplorasi MCP (Model Context Protocol) Server, yang memungkinkan LLM seperti Claude dan Cursor berinteraksi aman dengan filesystem lokal, membuka potensi untuk workflow AI yang lebih kompleks. Terakhir, penggunaan n8n membuktikan efisiensi otomatisasi workflow tanpa banyak coding, seperti yang ditunjukkan dalam proyek PDF summarization via Telegram dan web scraping.

Secara keseluruhan, pengalaman LEAP ini tidak hanya meningkatkan keterampilan teknis saya dalam pengembangan chatbot dan pemanfaatan berbagai model AI (Gemini, Ollama), tetapi juga memperkaya pemahaman praktis saya tentang implementasi solusi AI di lingkungan profesional. Proyek-proyek ini memperkuat kesiapan saya untuk menghadapi tantangan di dunia teknologi yang terus berkembang.

5. REFLEKSI DIRI

5.1 Analisa SWOT (Strength, Weaknes, Opportunity, Threat)

Berikut adalah analisis Strength, Weakness, Opportunity, dan Threat (SWOT) selama mengikuti LEAP:

5.1.1 Strength

- Bertanggung jawab

Penulis memiliki sikap tanggung jawab yang tinggi, sehingga selalu berusaha menyelesaikan tugas dan kewajiban yang diberikan dengan tepat waktu dan sesuai dengan ketentuan yang telah ditetapkan. Sikap ini menjadi kelebihan utama yang membuat penulis dapat dipercaya dalam mengemban tugas atau proyek tertentu.

- Kerja keras

Penulis memiliki semangat kerja keras yang tinggi. Meskipun menghadapi tantangan baru, penulis tetap berkomitmen untuk menyelesaikan semua pekerjaan dengan sebaik mungkin.

- Kemauan belajar

Penulis memiliki kemauan belajar yang besar, terutama ketika menghadapi hal-hal baru. Dengan sikap ini, penulis dapat dengan cepat meningkatkan kemampuan dan menyesuaikan diri terhadap kebutuhan proyek atau lingkungan kerja.

5.1.2 Weakness

- Kurang percaya diri

Penulis sering merasa kurang percaya diri, terutama ketika menghadapi tugas atau situasi baru. Akibatnya, penulis lebih sering mencoba menyelesaikan masalah sendiri daripada meminta bantuan, yang dapat mempengaruhi efisiensi kerja.

5.1.3 Opportunity

- Resource yang mencukupi

Lingkungan kerja menyediakan sumber daya yang cukup, baik dalam bentuk alat, teknologi, maupun dukungan dari mentor atau anggota tim. Hal ini memberikan penulis kesempatan untuk bekerja dengan lebih maksimal dan mengembangkan kemampuan secara bertahap.

- Waktu penggerjaan yang cukup flexible

Perusahaan memberikan fleksibilitas waktu penggerjaan yang membantu penulis untuk dapat lebih fokus dalam menyelesaikan tugasnya. Fleksibilitas ini juga memberikan ruang bagi penulis untuk belajar hal-hal baru yang diperlukan.

5.1.4 Threat

Tidak familiar dengan aplikasi dan teknologi yang digunakan Penulis menghadapi tantangan karena kurangnya pengalaman dengan aplikasi atau teknologi yang digunakan dalam perusahaan. Hal ini memerlukan waktu tambahan untuk mempelajari dan memahami alat serta proses yang terkait dengan pekerjaan.

5.2 Analisa Soft Skill

Berikut adalah soft skill yang perlu dikembangkan dan diperlukan selama proses pengerjaan LEAP

5.2.1 Kemauuan belajar

Proyek LEAP menghadirkan berbagai tantangan baru yang memerlukan pengetahuan dan keterampilan tambahan. Oleh karena itu, kemauuan belajar menjadi aspek penting yang harus dikembangkan. Sikap proaktif untuk mempelajari hal-hal baru, termasuk teknologi, konsep, dan metode kerja, akan membantu meningkatkan kemampuan diri dan mendukung keberhasilan proyek.

5.2.2 Manajemen waktu

Kemampuan untuk mengatur waktu secara efektif sangat diperlukan agar semua tugas dapat diselesaikan tepat waktu tanpa mengurangi kualitas pekerjaan.

Pengelolaan waktu yang baik mencakup perencanaan, prioritas tugas, dan disiplin dalam menjalankan jadwal. Dengan manajemen waktu yang baik, penulis dapat bekerja lebih produktif dan mengurangi stres akibat tenggat waktu.

5.3 Pengembangan Diri

Berikut adalah hal-hal positif yang dapat dikembangkan kedepannya:

5.3.1 Manajemen waktu

Mengembangkan kemampuan manajemen waktu dapat dimulai dengan membuat jadwal atau rencana kerja yang terstruktur. Prioritas terhadap tugas yang penting dan mendesak juga perlu dilakukan untuk memastikan semua pekerjaan selesai tepat waktu. Dengan manajemen waktu yang baik, produktivitas dapat meningkat, dan tekanan akibat tenggat waktu dapat diminimalkan.

5.3.2 Kepercayaan diri

Kepercayaan diri dapat dikembangkan dengan cara fokus pada kelebihan diri dan berani mencoba hal-hal baru. Memberi ruang untuk menerima kritik konstruktif juga membantu dalam membangun kepercayaan diri. Dengan kepercayaan diri yang kuat, penulis dapat lebih efektif dalam mengambil keputusan dan berinteraksi dengan orang lain.

DAFTAR PUSTAKA

Ollama. (n.d.). Run Llama 2, Mistral, and other large language models locally. Diakses dari

<https://ollama.com/>

Qdrant. (n.d.). Vector Database for the next generation of AI applications. Diakses dari

<https://qdrant.tech/documentation/>

LlamaIndex. (n.d.). Data Framework for LLM Applications. Diakses dari

<https://www.llamaindex.ai/>

n8n. (n.d.). Workflow automation for your business. Diakses dari

<https://n8n.io/>

Google AI. (n.d.). Gemini API. Diakses dari

<https://ai.google.dev/models/gemini>

Goldie, J. [@JulianGoldieSEO]. (2024, May 7). MCP is the most important concept in AI agents [Tweet]. X. Diakses dari

<https://x.com/JulianGoldieSEO/status/1927983251815289296>

Pachaar, A. [@akshay_pachaar]. (2024, April 20). MCP is the hottest topic in AI right now. [Tweet]. X. Diakses dari

https://x.com/akshay_pachaar/status/1916121984624156677

Agrawal, S. (2024, April 14). 30 MCP Ideas with Complete Source Code. LevelUp Coding. Diakses dari

<https://levelup.gitconnected.com/30-mcp-ideas-with-complete-source-code-2e1406495b8d>

Liu, J. [@jerryliu0]. (2024, April 1). Model Context Protocol (MCP) by @mcp_ai. [Tweet]. X. Diakses dari

<https://x.com/jerryliu0/status/1905341991141212184>

LAMPIRAN

Lampiran 1 Log Book Aktivitas Selama LEAP

#	Aktivitas	Tanggal Aktivitas
1	belajar coding yang diberikan	15-01-2025
2	mengcoding untuk menampilkan gambar	16-01-2025
3	melanjutkan codingan untuk menampilkan gambar	17-01-2025
4	belajar multiagent	20-01-2025
5	mengcoding multiagent	21-01-2025
6	melanjutkan coding multiagent	22-01-2025
7	melanjutkan coding multiagent	23-01-2025
8	melanjutkan codingan multiagent	24-01-2025
9	menambah data tabel	27-01-2025
10	memperbaiki codingan yang ngebug	28-01-2025
11	melanjutkan memperbaiki codingan yang ngebug	29-01-2025
12	memperbaiki multiagent	30-01-2025
13	mengcoding agar dapat menampilkan gambar dengan mengambil path dari tabel	31-01-2025
14	belajar qdrant	03-02-2025
15	mempraktikan codingan menggunakan qdrant	04-02-2025
16	mencoding menggunakan qdrant	05-02-2025
17	melanjutkan codingan qdrant	06-02-2025
18	melanjutkan codingan qdrant	07-02-2025
19	belajar FastAPI	10-02-2025
20	mengcoding FastAPI	11-02-2025
21	memasukan coding api ke dalam web	12-02-2025
22	melanjutkan coding api web	13-02-2025
23	melanjutkan coding api web	14-02-2025
24	memasukan gemini	17-02-2025
25	memperbaiki codingan website	18-02-2025
26	melanjutkan codingan gemini	19-02-2025
27	menambahkan filter untuk stall dan gedung	20-02-2025
28	melanjutkan codingan filter stall dan gedung	21-02-2025
29	mengcoding login	24-02-2025
30	mengcoding menyimpan image agar tidak hilang di chat sebelumnya	25-02-2025
31	mengcoding login,register dan mencoba hubungkan dengan pesanan	26-02-2025
32	melanjutkan codingan untuk pesanan	27-02-2025
33	mencoba mengganti qdrant dengan sql	28-02-2025
34	melanjutkan cobaan coding sql dan menulis laporan	03-03-2025

35	mengajar sql dan qdrant kepada anak iot	04-03-2025
36	meningkatkan kemampuan chatbot	05-03-2025
37	membuat agar chatbot bisa lebih bebas dalam percakapan	06-03-2025
38	melanjutkan codingan tingkatkan chatbot	07-03-2025
39	mempelajari tools calling	10-03-2025
40	pindah ke proyek petra carrier	11-03-2025
41	mempelajari coding dari jennifer	12-03-2025
42	coding pencarian job menggunakan tools calling	13-03-2025
43	melanjutkan codimgan pencarian job	14-03-2025
44	memperbaiki codingan tools calling	17-03-2025
45	gpu3 down jadi tidak bisa lanjut kerja	18-03-2025
46	mengcoding api ke job search	19-03-2025
47	sakit	20-03-2025
48	melanjutkan codingan api	21-03-2025
49	revisi codingan api	24-03-2025
50	mengcoding agar menampilkan data2 dan link	25-03-2025
51	melanjutkan coding untuk mengambil data dari jobstreet	26-03-2025
52	mencoba menggabungkan codingan dengan codingan dari jennifer	27-03-2025
53	melanjurtkan menggabungkan codingan dengan codingan jennifer	28-03-2025
54	melanjurtkan menggabungkan codingan dengan codingan jennifer	28-03-2025
55	belajar tentang mcp	07-04-2025
56	latihan mencoba mcp dengan llama	08-04-2025
57	mencoba mcp dengan claude desktop	09-04-2025
58	membuat laporan tentang claude desktop mcp	10-04-2025
59	mencari chat bot yang tidak ada limit untuk run mcp	11-04-2025
60	mencoba membuat chatbot yang bisa run mcp	14-04-2025
61	melanjutkan coding chatbot yang bisa run mcp	15-04-2025
62	melanjutkan coding chatbot yang bisa run mcp	16-04-2025
63	melanjutkan coding chatbot yang bisa run mcp	17-04-2025
64	melanjutkan coding chatbot yang bisa run mcp	18-04-2025
65	mencoba membuat server mcp	21-04-2025
66	membuat mcp server untuk bisa akses desktop	22-04-2025
67	melanjutkan coding mcp server yang bisa akses desktop	23-04-2025
68	mencoba ngerun mcp server	24-04-2025
69	mencoba ngerun mcp server menggunakan claude desktop	25-04-2025

70	mencoba membuat chatbot sendiri yang bisa ngerun mcp server	28-04-2025
71	research tentang linkup	29-04-2025
72	mencoba pakai linkup untuk mcp	30-04-2025
73	melanjutkan codingan chatbot mcp	05-05-2025
74	melanjutkan coding chatbot mcp	06-05-2025
75	melanjutkan coding chatbot mcp	07-05-2025
76	melanjutkan coding chatbot mcp	08-05-2025
77	memperbaiki mcp server	09-05-2025
78	memperbaiki eror chatbot tidak kesambung dengan mcp server	13-05-2025
79	memperbaiki eror chatbot tidak kesambung dengan mcp server	14-05-2025
80	memperbaiki eror chatbot tidak kesambung dengan mcp server	15-05-2025
81	memperbaiki eror chatbot tidak kesambung dengan mcp server	16-05-2025
82	belajar n8n	19-05-2025
83	mencoba membuat chatbot sederhana n8n	20-05-2025
84	melanjutkan membuat chatbot n8n	21-05-2025
85	mencari api gratis untuk n8n	22-05-2025
86	melanjutkan chatbot n8n	23-05-2025
87	membuat chatbot n8n untuk telegram	26-05-2025
88	melanjutkan chatbot n8n telegram	27-05-2025
89	mebuat chatbot summary menggunakan telegram	28-05-2025
90	memperbaiki telegram summary	02-06-2025
91	membuat webscraping menggunakan n8n	03-06-2025
92	mencoba menambahkan chatbot untuk data hasil webscraping	04-06-2025
93	melanjutkan proggress chatbot	05-06-2025

Lampiran 2 kegiatan bimbingan dosen selam LEAP

#	Pembimbing	Topik Bimbingan	Tanggal Bimbingan
1	Dr. Andreas Handojo S.T., M.MT.	chatbot memakai qdrant	14-02-2025
2	Dr. Andreas Handojo S.T., M.MT.	membahas progress coding	27-02-2025
3	Dr. Andreas Handojo S.T., M.MT.	membahas memakai llm gemini di codingan	04-03-2025
4	Dr. Andreas Handojo S.T., M.MT.	membahas tools calling	19-03-2025
5	Dr. Andreas Handojo S.T., M.MT.	membahas proggress tools calling job search	02-04-2025
6	Dr. Andreas Handojo S.T., M.MT.	membahas mcp server	17-04-2025
7	Dr. Andreas Handojo S.T., M.MT.	membahas proggress mcp server	30-04-2025
8	Dr. Andreas Handojo S.T., M.MT.	membahas mcp server menggunakan claude dan cursor	06-05-2025
9	Dr. Andreas Handojo S.T., M.MT.	membahas proggress mcp server	15-05-2025
10	Dr. Andreas Handojo S.T., M.MT.	membahas n8n	20-05-2025
11	Dr. Andreas Handojo S.T., M.MT.	n8n telegram chatbot	28-05-2025
12	Dr. Andreas Handojo S.T., M.MT.	n8n webscraping	02-06-2025