Panduan Belajar & Praktikum n8n: Integrasi LLM dan Telegram Bot

1. Penjelasan Konsep Dasar

API (Fungsi, Metode Umum, Autentikasi)

API (Application Programming Interface) adalah antarmuka yang memungkinkan aplikasi berbeda untuk saling berkomunikasi. Fungsi utama API adalah **memfasilitasi pertukaran data** antara sistem yang berbeda tanpa perlu membongkar logika internal masing-masing. Misalnya, aplikasi cuaca dapat menggunakan API BMKG untuk mendapatkan data suhu dan kelembapan, alih-alih memiliki database cuaca sendiri[1]. Dengan API, pengembang dapat memanfaatkan fungsi layanan lain (seperti pembayaran, peta, dsb) secara **efisien** tanpa membangun dari nol[2].

Secara umum, **metode HTTP** yang digunakan API mengikuti standar RESTful, yaitu:
- **GET** (mengambil data),
- **POST** (mengirim atau menambahkan data),
- **PUT/PATCH** (memperbarui data),
- **DELETE** (menghapus data).

Pertukaran data biasanya dalam format terstruktur seperti JSON melalui endpoint tertentu[3]. Sebagai contoh, permintaan `GET /api/products/123` akan meminta data produk ber-ID 123, dan server akan merespons dengan data (misalnya dalam format JSON) beserta kode status (200 OK jika sukses, 404 jika tidak ditemukan). Mekanisme request-response ini memungkinkan front-end dan back-end berkomunikasi secara modular.

Autentikasi API penting untuk menjaga keamanan akses. Hanya klien yang terotorisasi yang boleh mengakses endpoint tertentu, apalagi jika menyangkut data sensitif. Umumnya, API menerapkan mekanisme autentikasi seperti **API Key**, **OAuth 2.0**, atau **JWT (JSON Web Token)**[4]. Contohnya, saat aplikasi mengirim request ke API dengan API key, server akan memverifikasi key tersebut sebelum memproses permintaan. OAuth digunakan oleh banyak layanan untuk memberikan token akses terbatas (misal login dengan Google/Facebook). Dengan autentikasi, API memastikan hanya pihak berizin yang dapat melakukan aksi, dan sering dikombinasikan dengan **otorisasi** (hak akses spesifik) serta pembatasan kuota (rate limiting) untuk keamanan lebih lanjut[4].

LLM (Large Language Model): GPT dan Gemini

LLM atau *Large Language Model* adalah model AI berukuran sangat besar yang dilatih pada sejumlah data teks yang luar biasa banyak. LLM berbasis arsitektur *transformer* mampu memahami konteks bahasa dan menghasilkan teks yang menyerupai tulisan manusia. Misalnya, **GPT-3** (Generative Pre-trained Transformer 3) dari OpenAI memiliki sekitar **175 miliar** parameter yang dilatih dari beragam sumber teks[5]. Dengan ukuran sebesar itu, GPT-3 (dan versi lanjutannya GPT-4) dapat melakukan berbagai tugas *Natural Language Processing* seperti menjawab pertanyaan, merangkum dokumen, menerjemahkan, hingga membuat konten baru.

Kemampuan ini terlihat dalam produk seperti *ChatGPT*, yang dapat berdialog dan memberikan jawaban informatif layaknya asisten manusia.

Di sisi lain, **Gemini** adalah LLM canggih terbaru dari Google (Google DeepMind) yang dirancang bersifat *multimodal*. Artinya, Gemini tidak hanya memahami teks, tapi juga dapat mengintegrasikan berbagai jenis input (gambar, audio, dll) dalam proses pemahamannya. Gemini diperkenalkan dalam beberapa varian model berdasarkan skala kemampuannya, yakni **Gemini Ultra, Pro, dan Nano**[6]. Ini berarti Google menyediakan model dengan ukuran berbeda (Ultra yang terbesar, hingga Nano yang lebih ringan) untuk keperluan yang beragam. Secara umum, Gemini digadang sebagai model paling kuat dan serbaguna Google saat ini, hasil kolaborasi tim DeepMind dengan Google Research, dengan peningkatan pada kemampuan penalaran dan efisiensi deployment di berbagai lingkungan (dari server hingga perangkat mobile)[7].

Baik GPT maupun Gemini merupakan contoh LLM terkini yang mampu **menggenerasi teks** dan menyelesaikan tugas kompleks melalui pemahaman konteks bahasa. Perbedaannya, GPT (misalnya GPT-4) telah dikenal melalui API OpenAI dan aplikasi ChatGPT yang banyak digunakan publik, sedangkan Gemini merupakan pendatang baru dari Google yang diintegrasikan ke layanan Google (seperti Bard atau Vertex AI) dan diharapkan unggul dalam pemrosesan multimodal. Meskipun model dan penyedianya berbeda, konsep dasarnya sama: keduanya menggunakan *neural network* berukuran sangat besar yang dilatih dengan *deep learning* untuk memprediksi dan menghasilkan keluaran bahasa alami yang relevan.

OpenAI API dan Gemini API (Struktur Request, Autentikasi)

OpenAI API menyediakan akses ke model GPT (termasuk GPT-3.5, GPT-4, dll) melalui endpoint HTTP. Struktur permintaan (request) ke OpenAI API umumnya berupa *request* POST dengan URL seperti https://api.openai.com/v1/chat/completions untuk model chat. Klien harus mengirim **header otorisasi** berisi API key, misalnya:

```
Authorization: Bearer OPENAI_API_KEY
Content-Type: application/json
```

dan *body* berformat JSON yang mencantumkan parameter seperti model yang dipakai dan pesan percakapan. Contoh sederhana, mengirim pesan pengguna ke model GPT-3.5 Turbo:

```
POST https://api.openai.com/v1/chat/completions
Authorization: Bearer <OPENAI_API_KEY>
Content-Type: application/json

{
  "model": "gpt-3.5-turbo",
  "messages": [
    {"role": "user", "content": "Halo, apa kabar?"}
  ]
}
```

Server OpenAI kemudian merespons dengan JSON berisi hasil, biasanya di dalam field seperti choices yang memuat balasan model. Pengembang perlu mendaftar ke OpenAI dan memperoleh

API key pribadi untuk digunakan. Autentikasi berbasis API key ini memastikan hanya pengguna terdaftar yang dapat memanggil model.

Gemini API disediakan melalui platform Google (Google AI/Vertex AI) dan **memiliki struktur yang mirip** dengan OpenAI API demi kemudahan integrasi[8][9]. Google bahkan memungkinkan penggunaan *client library* OpenAI dengan sedikit perubahan endpoint untuk memanggil Gemini. Misalnya, endpoint untuk chat completion Gemini berada di https://generativelanguage.googleapis.com/v1beta/openai/chat/completions. Autentikasinya menggunakan **API key Google Gemini** (diperoleh lewat Google AI Studio atau Google Cloud) yang dikirim sebagai Bearer token di header, serupa dengan OpenAI. Contoh request dengan curl untuk memanggil model Gemini:

```
curl "https://generativelanguage.googleapis.com/v1beta/openai/chat/completions" \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer GEMINI_API_KEY" \
    -d '{
        "model": "gemini-2.0-flash",
        "messages": [
          {"role": "user", "content": "Explain to me how AI works"}
        ]
      }'
```

Pada contoh di atas, *base URL* diarahkan ke endpoint Google, model yang dipakai adalah gemini-2.0-flash, dan API key Google disisipkan pada header Authorization[9]. Struktur JSON-nya tetap memiliki format mirip OpenAI (dengan role user, assistant, dsb untuk percakapan). Dengan demikian, **perbedaan utama antara OpenAI API dan Gemini API terletak pada endpoint dan kunci API yang digunakan**, sedangkan format permintaan dan responsnya sejenis.

Perlu diperhatikan bahwa untuk menggunakan Gemini API, Anda harus memiliki akses ke Google Cloud atau Google AI platform yang menyediakan kunci API Gemini. Sementara OpenAI API key dapat diperoleh dari akun OpenAI Anda (dashboard OpenAI). Keduanya menerapkan *rate limit* (batas frekuensi panggilan) dan biaya per pemakaian berdasarkan model yang dipakai, sehingga pengguna harus memperhitungkan hal ini saat mengintegrasikan ke aplikasi.

Telegram Bot API (Membuat Bot, Webhook, Token)

Telegram Bot API adalah API yang disediakan Telegram agar developer dapat membuat bot yang berinteraksi dengan pengguna Telegram. Untuk membuat sebuah bot Telegram, langkah pertama adalah menggunakan akun Telegram Anda dan menghubungi **BotFather** (sebuah akun resmi Telegram untuk pembuatan bot). Di BotFather, Anda menjalankan perintah /newbot, memberikan nama dan username unik untuk bot Anda (username harus diakhiri dengan kata "bot")[10]. Setelah berhasil, BotFather akan memberikan **token akses** berupa string panjang. Token inilah yang menjadi kredensial utama bot Anda – **jaga kerahasiaannya**, karena siapa pun yang memegang token dapat mengontrol bot tersebut.

Token bot Telegram digunakan untuk memanggil Telegram Bot API. API ini memungkinkan bot untuk mengirim pesan, menerima update pesan masuk, mengelola grup, dan sebagainya. Terdapat dua metode untuk *bot* menerima pesan dari pengguna:

- **Polling**: Bot secara periodik meminta (GET) *update* dari server Telegram (menggunakan metode getUpdates).
- **Webhook**: Bot mendaftarkan sebuah URL (endpoint) kepada server Telegram. Setiap ada pesan baru untuk bot, Telegram akan **mengirim HTTP POST** ke URL webhook tersebut secara real-time.

Webhook lebih efisien untuk respons cepat dan tidak membebani server dengan polling terus-menerus. Untuk menggunakan webhook, Anda perlu menyediakan URL yang dapat diakses publik (HTTPS). Pada n8n Cloud, setiap *workflow* dengan Trigger HTTP atau Trigger Telegram akan memiliki URL publik yang bisa digunakan untuk webhook. Anda dapat mengatur webhook bot dengan metode setWebhook melalui Bot API, memberikan URL milik n8n sebagai target. Setelah webhook diatur, Telegram akan mengirimkan update (pesan, perintah, dsb.) ke workflow n8n Anda secara otomatis.

Dalam konteks n8n, Anda tidak perlu memanggil getUpdates atau setWebhook manual jika menggunakan **Telegram Trigger node**. Cukup menyediakan token bot di kredensial, n8n akan menangani koneksi ke Telegram (baik dengan webhook di balik layar atau mekanisme polling yang dioptimalkan). Hal yang penting adalah menyiapkan **token** dengan benar di n8n agar terhubung ke bot Anda.

Setelah bot terhubung, Anda dapat mengirim pesan ke bot dari aplikasi Telegram biasa untuk mengujinya. Bot dapat merespons melalui API dengan memanggil metode seperti sendMessage (mengirim teks), sendPhoto (mengirim gambar), dll., disertai token sebagai autentikasi. Misalnya, memanggil:

https://api.telegram.org/bot<YOUR_TOKEN>/sendMessage?chat_id=<ID_CHAT>&text=Halo

akan membuat bot mengirim pesan "Halo" ke chat dengan ID tertentu. Namun, ketika menggunakan n8n, pemanggilan ini dapat dilakukan lewat node Telegram (yang akan dibahas kemudian) tanpa menulis URL secara manual. Intinya, **Telegram Bot API** menyediakan semua fungsi untuk interaksi bot, dan dengan token yang diberikan BotFather, n8n dapat memanfaatkan API tersebut untuk menerima dan mengirim pesan melalui bot Anda. Pastikan bot Anda di-*deploy* (misal, telah diklik "Start" oleh pengguna jika itu bot baru di chat personal) agar dapat menerima pesan.

2. Pengenalan n8n

Node, Workflow, dan Expression

n8n adalah platform otomasi workflow yang *open-source* dan **node-based**. Dalam n8n, Anda membangun *workflow* (alur kerja otomatis) di sebuah kanvas editor dengan menyambungkan blok-blok fungsional yang disebut **nodes**[11]. Setiap **node** merepresentasikan satu tindakan atau operasi, misalnya mengambil data dari suatu sumber, memproses data, atau mengirim output ke layanan lain. Node-node ini dihubungkan dengan garis alur (*connections*) sehingga membentuk urutan logika. Kombinasi node yang terhubung inilah yang disebut **workflow**.

Sebuah workflow biasanya dimulai dari node pemicu (trigger) dan diikuti oleh satu atau lebih node aksi (action) yang berjalan sesuai urutan atau kondisi yang ditentukan. Contohnya, workflow sederhana bisa terdiri dari: *Trigger (misal menerima webhook) → Node proses (misal*

memformat teks) → Node aksi (misal kirim email). Dengan workflow, tugas berulang dapat dijalankan otomatis tanpa intervensi manual, sesuai alur yang telah ditentukan pengguna[12]. Hal ini meningkatkan konsistensi proses dan mengurangi kemungkinan human error.

Expression dalam n8n adalah fitur yang memungkinkan nilai dinamis atau perhitungan dimasukkan ke parameter node. Expression dituliskan dalam format kode JavaScript singkat, diawali dengan tanda = dan dibungkus kurung kurawal ganda {{…}}. Melalui expression, Anda bisa mengambil data dari output node sebelumnya, memanipulasi string/angka, atau menggunakan fungsi bantu yang tersedia. Contoh: jika sebuah node menerima output JSON dengan field name, Anda bisa menulis expression Hello, {{ $json["name"] }} di node selanjutnya untuk menyisipkan nama tersebut ke teks. Expression dievaluasi saat runtime workflow, sehingga *value*-nya akan menyesuaikan data aktual yang mengalir. Dengan kata lain, **expressions membuat workflow menjadi fleksibel dan kontekstual**, karena output dari satu node dapat langsung dipakai atau diolah di node berikutnya tanpa perlu coding terpisah.

Trigger dan Action

Dalam n8n, **nodes** dapat dibagi secara konseptual menjadi beberapa jenis, terutama **trigger** dan **action**. **Trigger node** adalah node yang berfungsi memulai workflow secara otomatis ketika suatu kondisi terpenuhi atau event terjadi. Contoh trigger node: *Cron* (penjadwalan waktu), *Webhook* (mendengarkan HTTP request masuk), *IMAP Email Trigger* (mendeteksi email baru), atau *Telegram Trigger* (mendeteksi pesan Telegram baru). Workflow *hanya berjalan* saat trigger ini aktif dan menerima event yang sesuai. Sebagai ilustrasi, *Webhook trigger* akan menunggu sampai ada request HTTP yang masuk ke URL-nya, barulah workflow mengeksekusi langkah-langkah selanjutnya.

Sebaliknya, **Action node** melakukan tugas spesifik sebagai respons dari alur. Action node misalnya: *HTTP Request* (memanggil API eksternal), *Set* (mengatur nilai data), *Send Email* (mengirim email melalui SMTP), *Google Sheets* (mengirim atau mengambil data ke Google Sheet), dan lain-lain. Setiap action node biasanya membutuhkan input (baik dari trigger atau node sebelumnya) dan menghasilkan output (data yang telah diproses atau dikirim). Pada dasarnya, **trigger node memulai workflow**, sedangkan **action node melaksanakan logika bisnis** yang diinginkan. n8n juga menyediakan node khusus seperti *Function* (menjalankan JavaScript custom), *Conditional* (percabangan IF/Switch), dan *Error handling* untuk menangani kegagalan proses[13].

Dalam satu workflow, biasanya terdapat **satu trigger** (awal mula) dan bisa banyak action nodes. Contoh sederhana, dalam workflow "Notifikasi Otomatis": Trigger-nya bisa berupa *Schedule (Cron)* setiap jam 9 pagi, lalu Action 1 mengambil data dari database, Action 2 memeriksa kondisi tertentu, Action 3 mengirim notifikasi email jika kondisi terpenuhi. Jika divisualisasikan: *Trigger (Cron) → Action (Database Query) → Action (If condition) → Action (Send Email)*. Dengan memahami peran masing-masing, Anda dapat mendesain workflow yang terstruktur: Trigger memastikan *kapan* atau *bagaimana* workflow dimulai, sementara Action mengatur *apa yang dilakukan* oleh workflow tersebut[14].

Pada kasus yang akan kita buat (chatbot Telegram), node **Telegram Trigger** akan berperan sebagai pemicu ketika ada pesan masuk, dan node pengolah (LLM) serta node **Telegram Send**

akan menjadi action yang menanggapi pesan tersebut. Pola ini (trigger -> action -> action) adalah pola umum di n8n.

Integrasi API di n8n

Salah satu kekuatan n8n adalah kemampuannya mengintegrasikan banyak layanan dan API eksternal dengan mudah. Secara bawaan, n8n telah mendukung **400+ aplikasi** (hingga 2025) melalui node integrasi yang siap pakai[15]. Ini mencakup layanan seperti Google Sheets, Slack, GitHub, MySQL, dan tentunya API generik. Untuk integrasi API khusus, n8n menyediakan node **HTTP Request** yang sangat fleksibel. Dengan node HTTP Request, Anda bisa melakukan panggilan ke API apapun dengan mengatur metode (GET/POST/PUT/DELETE), URL endpoint, header (termasuk auth), query parameter, maupun body JSON. Data keluaran dari API tersebut akan langsung tersedia sebagai output node yang bisa dipakai node berikutnya.

Sebagai contoh, jika tidak ada node khusus untuk sebuah layanan, Anda cukup ambil dokumentasi API layanan itu dan konfigurasikan HTTP Request di n8n sesuai spesifikasi. Masukkan URL, metode, body, lalu tambahkan autentikasi. Untuk autentikasi, n8n punya fitur **Credentials** – di mana Anda dapat menyimpan API key, token, atau OAuth kredensial secara aman dan memakainya di node yang memerlukan. Dengan begitu, Anda tidak perlu menuliskan kunci sensitif langsung di workflow (lebih aman dan bisa digunakan ulang).

Selain node HTTP generik, n8n sudah memiliki node spesifik untuk beberapa API AI populer. Misalnya, terdapat **OpenAI node** (seperti *OpenAI Chat Model* atau *OpenAI DALLE* tergantung versinya) yang mempermudah interaksi dengan layanan OpenAI tanpa menyusun manual HTTP request. Anda cukup memasukkan prompt dan memilih model, n8n akan mengurus panggilan API-nya. Integrasi dengan API **Telegram** juga tersedia melalui **Telegram node**. Node ini mendukung berbagai operasi Telegram, contohnya *Send Message*, *Send Photo*, dll., sehingga Anda bisa mengirim pesan ke pengguna dari dalam workflow n8n dengan mudah. Dalam hal ini, Anda tinggal memilih operasi dan mengisi parameter (seperti chat ID, isi pesan, dsb.), tanpa harus merangkai URL Bot API secara manual.

Secara ringkas, **integrasi API di n8n** bisa dilakukan dengan dua cara: (1) memanfaatkan *built-in node* bila tersedia (lebih sederhana, konfigurasi berbasis form), atau (2) menggunakan *HTTP Request node* untuk kustomisasi penuh. Keduanya dapat menggunakan Credentials terpusat untuk autentikasi. Pendekatan ini akan kita terapkan saat menghubungkan n8n dengan OpenAI API, Google Gemini API, dan Telegram API dalam proyek pembuatan chatbot.

3. Tutorial Praktik: Membangun Chatbot Telegram dengan LLM

Pada bagian ini, kita akan mempraktikkan langkah-langkah membuat sebuah **chatbot Telegram** yang terintegrasi dengan **LLM** (OpenAI GPT atau Google Gemini) menggunakan n8n. Bot ini akan menerima pesan dari pengguna di Telegram, mengirim pertanyaan/pesan tersebut ke LLM untuk diproses, lalu mengembalikan jawabannya ke pengguna melalui Telegram. Berikut tahapan lengkapnya:

3.1 Pendaftaran Akun n8n Cloud

Langkah pertama, pastikan Anda memiliki akses ke n8n. Untuk kemudahan, kita akan menggunakan **n8n Cloud** (layanan n8n yang di-hosting, sehingga Anda tidak perlu install

sendiri). Kunjungi situs resmi n8n (n8n.io) dan daftar akun. Tersedia free trial atau free tier yang cukup untuk percobaan. Setelah mendaftar dan verifikasi email, **login ke n8n Cloud** dan buat instance/ruang kerja baru. Anda akan disajikan antarmuka editor n8n di browser. Pastikan workspace Anda aktif dan siap untuk membuat workflow baru.

(Alternatif: Jika Anda memilih self-hosting, pastikan n8n sudah terpasang dan berjalan di server Anda, dan bisa diakses melalui browser. Namun, untuk keperluan workshop/pembelajaran, n8n Cloud lebih cepat digunakan.)*

3.2 Menyiapkan Kredensial API (OpenAI, Gemini, Telegram)

Sebelum membangun workflow, kita perlu menyiapkan **credentials** (kredensial) di n8n untuk tiga layanan yang akan diintegrasikan: OpenAI (GPT), Gemini (Google), dan Telegram. Kredensial ini berisi token/kunci API yang dibutuhkan setiap layanan:

- **Kredensial OpenAI API:** Masuk ke akun OpenAI Anda (di platform.openai.com). Buka menu *API Keys* atau *Profile > View API keys*. Buat API key baru jika belum ada. Salin key tersebut. Di n8n, pada menu sidebar kiri, klik **Credentials** > **New**. Pilih jenis kredensial *OpenAI API*. Anda akan diminta memasukkan API Key. Tempelkan key dari OpenAI tadi dan simpan dengan nama misalnya "OpenAI API Key". (Catatan: OpenAI API key biasanya dimulai dengan sk-...).

- **Kredensial Gemini API (Google):** Untuk akses Gemini, Anda memerlukan API key Google generative AI. Jika Anda sudah terdaftar dalam program Google AI (atau Vertex AI) yang menyediakan Gemini, dapatkan API key-nya melalui Google Cloud Console atau AI Studio. Prosesnya: aktifkan API Generative Language di Google Cloud, lalu buat kredensial API key. Setelah mendapat key (string panjang), buat Credentials baru di n8n. Jika n8n belum punya tipe kredensial khusus Gemini, Anda dapat menyimpannya sebagai *Generic Credential* atau menggunakan HTTP Node nanti dan menempelkan key di header. **Alternatif:** Karena Google Gemini API kompatibel dengan format OpenAI, Anda juga bisa menggunakan OpenAI node di n8n dengan sedikit konfigurasi (misal, menyetel Base URL ke endpoint Google). Namun, untuk kesederhanaan, kita bisa pakai HTTP Request di workflow nantinya. Yang penting, simpan API key Gemini Anda, misal beri nama "Gemini API Key".

- **Kredensial Telegram Bot:** Gunakan token Bot yang didapat dari BotFather saat membuat bot (formatnya seperti 123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11). Di n8n Credentials, klik **New** dan cari tipe *Telegram API*. Masukkan token di kolom yang disediakan. Biasanya hanya perlu satu field "Bot Token". Simpan kredensial ini dengan nama, misalnya "Telegram Bot Token". Pastikan bot Anda sudah di-**start** setidaknya sekali di Telegram (buka chat bot dan klik Start atau kirim sesuatu), karena beberapa bot tidak menerima update sebelum diinitiasi oleh pengguna.

Setelah ketiga kredensial di atas disiapkan, n8n dapat menggunakan informasi tersebut dalam node yang sesuai. Anda sudah siap membangun workflow dengan node-node terkait.

3.3 Membuat Workflow Chatbot Telegram di n8n

Sekarang kita akan membuat workflow di editor n8n langkah demi langkah:

1. Buat Workflow Baru: Di dashboard n8n, klik tombol **"New Workflow"**. Beri nama workflow ini, misalnya "Telegram LLM Chatbot".

2. Tambahkan *Telegram Trigger* Node: Di panel kiri (Nodes), cari "Telegram Trigger". Drag node tersebut ke kanvas. Node ini akan memicu workflow saat ada pesan masuk ke bot. Pilih node Telegram Trigger, di panel kanan konfigurasinya, atur:
- **Credentials:** Pilih kredensial Telegram Bot yang sudah Anda simpan ("Telegram Bot Token"). n8n akan verifikasi token ini.
- **Trigger on**: Pilih event yang akan ditangani, misalnya Message (pesan baru)[16]. Ini berarti setiap ada pesan teks baru ke bot, workflow akan berjalan. (Anda juga bisa memilih jenis update lain jika perlu, tapi untuk chatbot, *Message* sudah tepat).
- Biarkan pengaturan lain default (misal *Download Images/Files* opsional tergantung kebutuhan).

Setelah itu, **aktifkan** node trigger ini (biasanya node trigger di n8n Cloud butuh workflow di-*activate* untuk benar-benar menunggu event, tapi kita bisa aktifkan di akhir setelah selesai menyusun workflow).

3. Tambahkan Node untuk LLM (OpenAI/Gemini): Ini adalah inti dari chatbot – mengirim pertanyaan pengguna ke model AI dan mendapatkan jawabannya. Kita punya beberapa opsi:
- Jika menggunakan **OpenAI GPT**, n8n menyediakan node bawaan. Cari node **"OpenAI Chat"** atau **"AI Agent"** (tergantung versi n8n, mungkin terletak di kategori AI). Misalnya, kita pilih *OpenAI Chat Model* node. Drag ke kanvas. Sambungkan output dari node *Telegram Trigger* ke input node OpenAI (tarik garis dari titik output Trigger ke titik input OpenAI node).
Konfigurasi OpenAI node: Pilih kredensial OpenAI yang sudah dibuat. Pilih model (misal gpt-3.5-turbo atau GPT-4 jika tersedia). Pada field *Prompt* atau *Messages*, isi dengan konten pesan pengguna. Karena kita sudah menghubungkan node, kita bisa menggunakan **expression** untuk mengambil teks pesan Telegram. Biasanya, di OpenAI node, ada pilihan memasukkan pesan pengguna sebagai *User message*. Klik icon ⊕ atau menu expression di field teks, lalu pilih referensi data dari node *Telegram Trigger* – misalnya: {{$json["message"]["text"]}} (ini tergantung struktur output trigger). Ini memastikan input ke AI adalah pesan asli pengguna. Anda juga dapat menambahkan *system prompt* (instruksi untuk kepribadian bot) jika node mendukungnya, misal: *"You are a helpful assistant."*
- Jika menggunakan **Google Gemini**, saat ini mungkin belum ada node khusus di n8n. Solusinya, gunakan node **HTTP Request**. Tarik node HTTP ke kanvas, hubungkan dari Trigger. Konfigurasikan sebagai berikut:
Method: POST,
URL: https://generativelanguage.googleapis.com/v1beta/openai/chat/completions (endpoint OpenAI-compatible milik Gemini),
Authentication: None (kita akan pakai header manual),
Headers: Tambah header Authorization: Bearer <Gemini_API_Key> (gunakan expression atau langsung isi dengan API key yang disimpan), dan Content-Type: application/json.
Body: Pilih type JSON, dan buat struktur JSON sesuai API Gemini. Misalnya:

```
{
  "model": "gemini-2.5-fast",
  "messages": [
    { "role": "user", "content": "{{$json["message"]["text"]}}"}
```

]
}

(Pastikan menyesuaikan model name dengan model Gemini yang Anda punya akses.) Node HTTP ini akan mengirim request ke Gemini setiap kali ada pesan masuk, dengan isi pesan tersebut. Output dari node ini akan berupa respons JSON dari model (akan mengandung teks jawaban). Kita perlu mengambil bagian teks jawaban untuk dikirim ke Telegram lagi. Biasanya, respons OpenAI/Gemini berisi choices[0].message.content. Anda bisa memparsingnya di node berikutnya atau langsung mengambil via expression nanti.

Baik Anda memakai node OpenAI maupun HTTP (Gemini), intinya node kedua ini menghasilkan **teks balasan AI** berdasarkan prompt dari user. Anda dapat menambahkan logic tambahan, misal memformat jawaban atau menangani error (gunakan *IF* node jika API gagal, dsb., opsional sesuai kebutuhan).

4. Tambahkan *Telegram Send Message* Node: Sekarang, kita butuh mengirim jawaban dari AI kembali ke pengguna di Telegram. n8n memiliki node **Telegram** (untuk aksi). Cari **"Telegram"** node dengan operasi *Send Message*. Drag ke kanvas dan hubungkan output node AI (OpenAI/HTTP) ke input node Telegram ini. Konfigurasinya:
- **Credentials:** Pilih kredensial bot Telegram (sama seperti trigger).
- **Operation:** Pilih "Send Message".
- **Chat ID:** Isi dengan ID chat tujuan. Karena kita ingin membalas ke user yang mengirim, kita gunakan kembali data dari *Telegram Trigger*. Biasanya, output trigger punya message.chat.id. Gunakan expression {{ $('Telegram Trigger').item.json.message.chat.id }} untuk mengambil ID chat tersebut[17]. n8n memudahkan ini dengan menu dropdown juga – Anda bisa klik field *Chat ID* → klik icon ekspresi → pilih data *chat.id* dari output Trigger.
- **Text:** Ini teks yang akan dikirim. Isikan dengan output dari node AI. Jika pakai node OpenAI, misal nodenya bernama "OpenAI Chat", Anda dapat referensikan {{ $("OpenAI Chat").item.json.data.choices[0].message.content }} (menyesuaikan struktur output OpenAI node). n8n kerap menaruh hasil di field data atau langsung di root JSON, cek di panel Output node AI. Jika menggunakan node HTTP (Gemini), Anda mungkin perlu sedikit parsing. Anda bisa langsung menaruh seluruh respon, atau idealnya gunakan node *Function* sebelum Telegram Send untuk mengekstrak choices[0].message.content dari JSON respons Gemini ke field sederhana. Misal, pakai Function Item:

```
item.text = item.body.choices[0].message.content;
return item;
```

Lalu hubungkan Function ke Telegram Send, dan di field Text cukup {{$json["text"]}}. Namun agar sederhana, kita asumsikan output AI node sudah terstruktur. Gunakan expression yang tepat untuk mengambil teks jawaban AI[18].
- **(Optional)**: Anda bisa mengatur *additional fields* seperti *parse_mode* (HTML/Markdown) jika ingin memformat pesan, atau *disable_notification* dsb., namun untuk saat ini boleh dibiarkan default.

Node Telegram Send akan mengambil input teks dan mengirimkannya sebagai pesan chat. Pastikan node ini tersambung dengan benar dari node AI sebelumnya.

5. Menyusun Alur & Testing: Setelah semua node ditambahkan dan diisi, periksa kembali alur: *Telegram Trigger → LLM (OpenAI/Gemini) → Telegram Send*. Pastikan tidak ada node yang tidak terhubung. Anda bisa menambah node lain jika perlu (misal Logging, atau menyimpan chat ke Google Sheet sebagai log, dll., tetapi itu opsional dan bisa dilakukan belakangan).

Simpan workflow Anda. Untuk menguji, klik **Execute Workflow** (untuk manual trigger) atau aktifkan workflow lalu kirim pesan langsung. Pada mode manual, n8n akan menunggu sampai trigger terpancing. Buka aplikasi Telegram Anda, kirim sebuah pesan ke bot yang Anda buat (misal "Halo, bot!"). Lihat di editor n8n: seharusnya node trigger menerima update (berwarna hijau menandakan eksekusi), lalu node AI memproses (mungkin butuh 1-2 detik tergantung respons API), kemudian node Telegram Send mengirim balasan. Jika eksekusi sukses, Anda akan melihat pesan balasan muncul di chat Telegram dari bot, berisi jawaban yang dihasilkan AI.

6. Aktivasi Workflow: Terakhir, jika sudah diuji dan berfungsi, **aktifkan** workflow (klik toggle Active). Dalam n8n Cloud, workflow yang active akan terus berjalan di background menunggu trigger. Sekarang bot Telegram Anda sudah live: setiap kali mendapat pesan baru, n8n akan otomatis mengeksekusi workflow dan bot akan membalas menggunakan LLM. Cobalah beberapa pertanyaan ke bot untuk memastikan stabil.

Sebagai contoh, misalkan pengguna mengirim: "Apa kepanjangan AI?" Bot akan meneruskan ke OpenAI/Gemini, mungkin mendapat jawaban "AI adalah singkatan dari Artificial Intelligence (Kecerdasan Buatan)". Kemudian bot mengirim teks tersebut kembali ke pengguna. Semua terjadi dalam hitungan detik tanpa campur tangan manual.

Dengan demikian, Anda telah berhasil membuat chatbot Telegram sederhana yang *smart* berkat integrasi LLM melalui n8n. Anda dapat mengembangkan workflow ini lebih lanjut, misalnya menambah fitur: perintah khusus ("/start", "/help"), menggunakan **Switch/If** node untuk menangani input tertentu (seperti contoh template yang membedakan antara permintaan teks vs pembuatan gambar), atau menambah *error handling* (jika API LLM gagal, kirim pesan “Maaf, terjadi kesalahan,” dll). Selama workshop/perkuliahan, fokuslah memastikan alur dasar berfungsi dulu.

4. Contoh JSON Workflow

Sebagai referensi, berikut ini adalah **contoh JSON workflow** n8n untuk implementasi chatbot Telegram + LLM serupa yang telah kita bahas. JSON ini mencakup node *Telegram Trigger*, *AI (Agent/OpenAI)*, dan *Telegram Send Message* beserta koneksi di antara mereka. Anda dapat mengimpor JSON ini ke n8n Anda atau mempelajarinya untuk memahami struktur internal workflow. (Catatan: ID dan credential name akan berbeda, sesuaikan dengan milik Anda saat import.)

{ "name": "My Telegram AI Agent Chatbot", "nodes": [{ "parameters": {
"updates": ["message"], "additionalFields": {} }, "type": "n8n-nodes-
base.telegramTrigger", "typeVersion": 1.2, "position": [-176, 0], "id":
"992c218f-bce7-43f0-967c-5143bd75b707", "name": "Telegram Trigger", "webhookId":
"d0477b18-61ec-4014-aa02-87ee5900663d", "credentials": { "telegramApi": { "id":
"LD1HTeh0TL4LwpjZ", "name": "My Telegram Demo Bot" } } }, { "parameters": {
"promptType": "define", "text": "=Respond to the user ({{
$json.message.from.first_name }}) query below:\n {{ $json.message.text }}",

"options": { "systemMessage": "You are a helpful HR virtual assistant, your responsibility is to answer work-related questions of the Automatewith.me company (HR domain), for non-related questions, tell them that you can't support." } }, "type": "@n8n/n8n-nodes-langchain.agent", "typeVersion": 2.2, "position": [48, 0], "id": "d454751c-5e41-4f81-9fd9-206de10a747d", "name": "AI Agent" }, { "parameters": { "model": { "__rl": true, "mode": "list", "value": "gpt-4.1-mini" }, "options": {} }, "type": "@n8n/n8n-nodes-langchain.lmChatOpenAi", "typeVersion": 1.2, "position": [120, 224], "id": "24c288c1-67ce-4c60-bb77-9d61ba547cbf", "name": "OpenAI Chat Model", "credentials": { "openAiApi": { "id": "tnXtbK3d66hDjxXa", "name": "OpenAi account" } } }, { "parameters": { "chatId": "={{ $('Telegram Trigger').item.json.message.chat.id }}", "text": "={{ $json.output }}", "additionalFields": {} }, "type": "n8n-nodes-base.telegram", "typeVersion": 1.2, "position": [400, 0], "id": "b6b4530d-8553-45cb-a819-2e2038f86e6e", "name": "Send a text message", "webhookId": "aa96c186-b3d2-405a-b9a3-680f50976d58", "credentials": { "telegramApi": { "id": "LD1HTeh0TL4LwpjZ", "name": "My Telegram Demo Bot" } } }], "pinData": {}, "connections": { "Telegram Trigger": { "main": [[{ "node": "AI Agent", "type": "main", "index": 0 }]] }, "OpenAI Chat Model": { "ai_languageModel": [[{ "node": "AI Agent", "type": "ai_languageModel", "index": 0 }]] }, "AI Agent": { "main": [[{ "node": "Send a text message", "type": "main", "index": 0 }]] } }, "active": false, "settings": { "executionOrder": "v1" }, "versionId": "f57c24b3-7ca3-41ed-b728-5ecbfbd7c022", "meta": { "templateCredsSetupCompleted": true, "instanceId": "e145bfb15cacc90e0d1ae6ee743e6744f8fc7108de50458700cb2ae620dc5ca5" }, "id": "kZ00pyKs90HD4e5Q", "tags": [] }

Dari JSON di atas, dapat dilihat struktur setiap node (tipe, parameter, posisi, id unik, dsb.) dan bagaimana koneksi main menghubungkan **Telegram Trigger** ke **AI Agent**, lalu ke node **Send a text message**. Node *OpenAI Chat Model* terhubung sebagai sub-node dari *AI Agent* (ini detail implementasi template tersebut). Meskipun detail ini teknis, memahaminya berguna saat melakukan **backup workflow** atau mengedit via JSON langsung.

5. Visualisasi Diagram Alur

Gambar 1: Contoh diagram alur (workflow) n8n untuk chatbot Telegram berbasiskan LLM. Diagram tersebut menunjukkan node-node dan alur koneksinya. Dimulai dari **Telegram Trigger** (kiri) yang menanti pesan masuk dari pengguna, kemudian alur berlanjut ke beberapa node pemrosesan. Terlihat adanya percabangan logika: pesan diperiksa apakah merupakan perintah tertentu (misal /image untuk membuat gambar, atau teks biasa). Dalam contoh ini, jika input adalah pesan teks biasa, workflow meneruskannya ke node **OpenAI (ChatGPT)** untuk menghasilkan jawaban teks, lalu node **Telegram Send** mengirim balasan teks kembali ke pengguna. Jika input adalah perintah gambar, workflow menggunakan node OpenAI (Image generation) untuk membuat gambar dan mengirimkannya melalui node Telegram (Send Photo). Terdapat pula penanganan untuk input yang tidak dikenali (unsupported command) di mana bot mengirim pesan error. *Diagram alur* ini memberikan gambaran visual bagaimana komponen-komponen dalam workflow saling berinteraksi – mulai trigger, logika percabangan (conditional), integrasi ke API AI, hingga aksi kirim balik ke Telegram. Dengan visualisasi, kita dapat lebih

mudah memvalidasi alur logika dan memastikan setiap skenario pesan dari pengguna ditangani oleh bot.

Seperti yang terlihat, n8n memungkinkan pengaturan alur kompleks secara **visual** dengan node dan koneksi. Tiap node bisa diberi label/nama yang menjelaskan fungsinya (misal "Check Command", "Generate Image", "Send Error Message"), sehingga alur mudah dipahami. Pada workflow chatbot, pola umum adalah *trigger → (beberapa proses/condition) → send response*. Anda dapat menambahkan *Logging* atau *Notification* tambahan di alur ini sesuai kebutuhan (misal node untuk menyimpan chat ke database, atau mengirim notifikasi admin jika error). Pastikan alur sudah diuji untuk setiap cabang logika agar bot bekerja mulus. Diagram di atas diambil dari contoh workshop, yang menunjukkan praktik penerapan beberapa fitur sekaligus: memanggil **ChatGPT untuk teks** dan **DALLE-2 untuk gambar** dalam satu workflow Telegram bot[19].

6. Checklist Praktikum dan Rubrik Penilaian

Checklist Praktikum

Untuk memastikan semua langkah telah dilakukan dengan benar dalam praktik pembuatan chatbot n8n + LLM, berikut **checklist** yang dapat digunakan peserta sebelum pengumpulan tugas/praktikum:

- [ ] **Akun n8n Cloud aktif:** Peserta telah mendaftar dan login ke n8n (cloud atau self-host) dan bisa mengakses editor workflow.
- [ ] **Bot Telegram dibuat:** Bot baru telah dibuat via BotFather, token API Bot telah diperoleh dan diuji (bot sudah di-*start*).
- [ ] **API keys tersedia:** API key OpenAI (atau alternatifnya API key Gemini Google) sudah didapat dan masih valid.
- [ ] **Credentials terkonfigurasi:** Semua kredensial (OpenAI API, Gemini API, Telegram API) telah ditambahkan ke n8n Credentials dan dipakai di node terkait.
- [ ] **Workflow tersimpan:** Workflow “Chatbot Telegram LLM” sudah dibuat di n8n dengan minimal 3 node utama (Trigger, LLM API call, Telegram send) terhubung sesuai alur.
- [ ] **Pengujian lokal berhasil:** Workflow diuji dengan mengirim pesan uji ke bot Telegram, dan bot merespons dengan jawaban yang diharapkan (dari AI) tanpa error.
- [ ] **Workflow diaktifkan:** Workflow sudah dalam status Active (untuk n8n Cloud) sehingga bisa berjalan real-time saat mendapat pesan baru.
- [ ] **Dokumentasi penggunaan:** Peserta memahami cara kerja bot dan bisa menjelaskan atau menuliskan petunjuk penggunaan singkat (misal “Kirim pertanyaan ke bot, bot akan menjawab otomatis”).

Checklist di atas membantu memastikan tidak ada langkah terlewat. Jika semua item telah dicentang, artinya Anda siap mempresentasikan hasil praktik ini.

Rubrik Penilaian

Penilaian tugas praktikum akan didasarkan pada beberapa aspek berikut dengan bobot masing-masing (total 100%):

Aspek Penilaian	Deskripsi Kriteria	Bobot
Persiapan Lingkungan	Mampu mendaftar akun n8n dan menyiapkan semua kredensial API (OpenAI/Gemini & Telegram) dengan benar.	20%
Konstruksi Workflow	Workflow n8n dibangun sesuai spesifikasi (terdapat minimal 1 trigger dan 2 action nodes yang terhubung) dan alur logika tepat (misal, data dari trigger mengalir ke node AI lalu ke node kirim pesan tanpa kesalahan).	30%
Fungsi Chatbot	Bot Telegram **berfungsi** dengan baik – dapat menerima pesan dari user dan mengirim jawaban dari LLM yang relevan. Uji coba beberapa pertanyaan berhasil, respons sesuai dan tidak lambat berlebihan.	30%
Pemahaman Konsep	Peserta menunjukkan pemahaman konsep dengan dapat menjelaskan komponen yang dibangun (apa itu trigger, bagaimana API dipanggil, peran LLM, dsb.). Juga mampu menangani modifikasi kecil (misal mengganti prompt atau menambah node) dengan benar.	10%
Dokumentasi & Presentasi	Laporan praktikum disusun rapi dan lengkap (tujuan, langkah, screenshot/diagram, hasil test). Saat presentasi/diskusi, peserta percaya diri menjelaskan workflow dan menjawab pertanyaan seputar proyek.	10%

Kriteria di atas menekankan tidak hanya hasil fungsional (bot bekerja), tetapi juga pemahaman dan kerapian penyajian. Misalnya, meskipun bot-nya berjalan, mahasiswa yang tidak bisa menjelaskan alur kerjanya akan mendapat pengurangan poin pada aspek pemahaman. Sebaliknya, ide kreatif atau tambahan fitur akan jadi nilai plus selama syarat utama terpenuhi. Pastikan Anda mengecek setiap aspek sebelum pengumpulan.

7. Informasi Tambahan

Sebagai bahan pembelajaran lanjutan, dapat mengakses **Workshop n8n by Faruq Aziz**.

Link tersebut berisi penjelasan fundamental n8n dan contoh-contoh workflow lain yang dapat memperdalam pemahaman Anda. Selain itu, dokumentasi resmi n8n (docs.n8n.io) juga merupakan sumber yang baik jika Anda ingin mempelajari node-node dan fitur n8n lebih detail. Semoga modul dan praktikum ini bermanfaat dalam memahami integrasi *Large Language Model* dengan platform otomasi seperti n8n. Selamat bereksperimen dan mengembangkan proyek chatbot Anda lebih lanjut![3][4].

[1] [2] [3] [4] Apa itu API? Pengertian & Kenapa Penting Bagi Programmer? - CODEPOLITAN

https://www.codepolitan.com/blog/apa-itu-api-pengertian-kenapa-penting-bagi-programmer/

[5] Apa itu LLM? - Penjelasan Model Bahasa Besar - AWS

https://aws.amazon.com/id/what-is/large-language-model/

[6] [7] Introducing Gemini: Google's most capable AI model yet

https://blog.google/technology/ai/google-gemini-ai/

[8] [9] OpenAI compatibility | Gemini API | Google AI for Developers

https://ai.google.dev/gemini-api/docs/openai

[10] [16] [18] How to Build a Telegram AI Chatbot with n8n (Step-by-Step Tutorial, No Coding) - Built with n8n - n8n Community

https://community.n8n.io/t/how-to-build-a-telegram-ai-chatbot-with-n8n-step-by-step-tutorial-no-coding/186182

[11] [12] [13] [14] Penjelasan Lengkap n8n Workflow dan Apa Fungsi Utamanya

https://www.jagoanhosting.com/blog/n8n-workflow/

[15] Apa itu N8N? Solusi Otomasi Workflow Terbaik 2025

https://www.lantarandigital.co.id/apa-itu-n8n-solusi-otomasi-workflow-terbaik-2025/

[17] s3.ap-southeast-1.amazonaws.com

https://s3.ap-southeast-1.amazonaws.com/automatewith.me/n8n-workflow-json/My+Telegram+AI+Agent+Chatbot.json

[19] Telegram AI Chatbot | n8n workflow template

https://n8n.io/workflows/1934-telegram-ai-chatbot/