Lab8_1: Medical Research Assistant with PubMed

PubMed is a widely used biomedical literature database maintained by the U.S. National Center for Biotechnology Information (NCBI). Clinicians, researchers, and students use PubMed to locate research evidence, read abstracts, and identify relevant studies. In this lab, we will build a simple Q&A chatbot that can accept a natural-language question, search PubMed through an API (a programmatic way to request data), retrieve relevant article information, and generate a concise evidence-based response with citations to the retrieved sources. This workflow illustrates how modern clinical and research tools can combine information retrieval with summarization while maintaining transparency and responsible use.

Expected Behaviour

Prompt	Response
Search PubMed for “type 2 diabetes lifestyle intervention” and return the latest 3 studies with PMID + title + journal + year.	1. PMID: 38782488 Title: Digital Lifestyle Interventions for Type 2 Diabetes Management: A Systematic Review. Journal: Cureus Year: 2024 2. PMID: 38753232 Title: Exploring the Effect of an Intensive Lifestyle Intervention on Cardiometabolic Markers in Individuals with Type 2 Diabetes and Obesity with Concomitant Heart Failure with Preserved Ejection Fraction (HFpEF): A Pilot Study. Journal: Journal of personalized medicine Year: 2024 3. PMID: 38752252 Title: Comparison of the Effects of Different Lifestyle Interventions on Lipid-Lowering in Patients with Type 2 Diabetes Mellitus: A Network Meta-Analysis. Journal: Journal of personalized medicine Year: 2024
I want to know more about depression.	Here are some studies on depression: 1. PMID: 35216520 Title: Major Depression and Its Recurrences: Life Course Matters. Journal: Annual review of clinical psychology Year: 2022 2. PMID: 34351273 Title: Depression's Problem With Men. Journal: AMA journal of ethics Year: 2021 3. PMID: 26357876 Title: Inflammation: depression fans the flames and feasts on the heat. Journal: The American journal of psychiatry Year: 2015

Prerequisites

Google Gemini API Key

A) Create a Google Gemini API key (Google AI Studio)

What you need

- A Google account
- Access to Google AI Studio (you may need to accept Terms of Service on first use)

Steps

1. Open the **Google AI Studio API Keys** page and sign in.
2. Click **Create API key** (or **Create or view a Gemini API Key**).
3. Copy the key and store it securely (do not share it or paste it into chat messages).

Where you will use it in this lab

- In n8n, create a credential for **Google Gemini API account** and paste the key into the credential (so nodes can call the model securely).

Google AI Studio – API Keys: https://aistudio.google.com/app/apikey

Gemini API keys guide (official): https://ai.google.dev/gemini-api/docs/api-key

B) Activate n8n Cloud free trial (managed n8n)

What you need

- An email address for registration

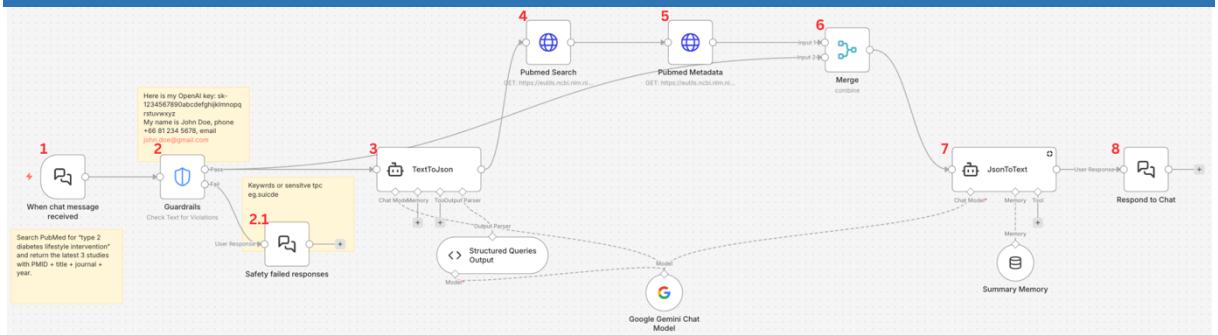
Steps

1. Go to the n8n Cloud registration page and create a new workspace.
2. Your Cloud free trial includes **14 days** of Pro-plan features, with a limit of **1000 executions** (and Starter-level compute).

n8n Cloud trial registration: https://app.n8n.cloud/register

n8n Cloud free trial docs (official): https://docs.n8n.io/manage-cloud/cloud-free-trial/

N8N Pipeline



Workflow overview

This workflow turns a user's chat question into:

1. a validated request (Guardrails),
2. a structured PubMed search plan (LLM Query Builder),
3. PubMed ESearch + ESummary calls (PubMed Search + PubMed Metadata),
4. a formatted final response to the user (SearchPlusMetadata ⟶ Respond to Chat).

---

1) Chat Trigger — When chat message received

Description

This node starts the workflow whenever a new message arrives in the n8n chat UI. It captures the user's prompt and creates the initial execution item.

Important configuration

- **Make Chat Publicly Available**: Enable this to get a shareable public chat URL for testing.
- **Options - Response Mode**: Set to **Respond Nodes** to route the final reply through **Respond to Chat** and ensure the chat UI displays the workflow's final output correctly.

Input

- A chat message from the user (text prompt)

Output

- The same user message, typically in a field such as chatInput.

2) Guardrails — Check Text for Violations

Description

This node validates the incoming text against policies (e.g., sensitive personal data, secret keys, banned topics). It routes execution to **Pass** or **Fail**.

Important configuration

- **Operation**: "Check Text for Violations"
- **Text to Check**: {{$json.chatInput}}
- Enable checks you need, commonly:
    - **Personal Data (PII)**: detect names, phone numbers, emails, patient IDs
    - **Secret Keys**: detect API keys/tokens – **Permissiveness:** Balanced
    - **Keywords**: suicide, kill self, kill myself (Or anything you want to add)

Input

- User text (from Node 1)

Output

- On **Pass**: the workflow continues and typically includes a field like guardrailsInput (the sanitized / checked user prompt)
- On **Fail**: emits the violation detail (what was triggered) and routes to Node 2.1

2.1) Safety failed responses — Respond to Chat (Fail branch)

Description

If Guardrails fails, this node sends a safe, user-friendly error message back to the chat UI and ends the workflow.

Important configuration

- Message content

```
“I can’t process this message because it contains sensitive personal data/credentials or self-harm/suicide-related content. Please remove API keys and personal identifiers and try again. For self-harm/suicide messages, I can’t help here—if there is immediate risk, contact emergency services or a trusted adult. For academic requests, rephrase in a general research context.”
```

Input

- Guardrails fail output

Output

- A single chat response sent to the user (workflow ends here for the fail path)

3) TextToJson (QueryBuilder) — LLM agent to produce PubMed search plan

Description

This node converts the user’s natural-language request (already validated by Guardrails) into a machine-readable JSON plan for PubMed E-utilities. The plan is used by downstream HTTP nodes to call:

- ESearch (to retrieve PMIDs)
- ESummary (to retrieve metadata such as title, journal, year, authors)

Prompt

- **Source for Prompt (User Message)**: **Connected Guardrails Node**

  This ensures the LLM always receives the same request text that has passed validation.
- **Prompt (User Message)**:

  The model uses the Guardrails-cleaned user request as the single source of truth.

Output format control

- **Require Specific Output Format: ON**

  This forces the model to output JSON that conforms to your Output Parser schema.

System Message

```
You are a query planner for PubMed E-utilities (NCBI).

You will be given a user request in plain text. Your job is to produce a plan as
JSON for:
1) PubMed ESearch to retrieve PMIDs
2) PubMed ESummary to retrieve metadata for those PMIDs

Return ONLY valid JSON that matches the schema below. Do not include any prose.

Important rules:
- Always plan to retrieve metadata via ESummary. Therefore ALWAYS include a
"metadata" object.
- Do NOT include PMIDs in your output (PMIDs come from the ESearch API response).
- If the user does NOT specify how many results to return, set esearch.retmax = 3.
- If the user specifies a number (e.g., "latest 10", "top 5", "return 20"), set
esearch.retmax to that number (cap at 20 unless the user explicitly requests
more).
- If the user asks for "latest" or "most recent", set esearch.sort = "pub_date".
Otherwise use "relevance".
- The ESearch term must be a PubMed query string. Preserve user-supplied quoted
phrases when appropriate. Use AND between concepts when needed.

Schema (must follow exactly):
{
  "esearch": {
    "term": string,
    "retmax": integer,
    "sort": "pub_date" | "relevance"
  },
  "metadata": {
    "source": "esummary",
    "fields": string[]
  },
  "answer_constraints": {
    "output_format": "table" | "bullets",
    "include_pmids": boolean
  }
}

Field selection rules for metadata.fields:
- Always include: "title", "fulljournalname", "pubdate"
- If the user asks for authors, include "authors"
- If the user asks for journal abbreviation/source, include "source"
- If the user asks for year only, still include "pubdate" (year will be extracted
later)

answer_constraints rules:
- If the user asks for a table, set output_format="table"; otherwise default to
"bullets".
- Set include_pmids=true unless the user explicitly says not to include PMIDs.
```

```
Now produce the JSON plan.
```

Reliability

- **Max Iterations**: **2**

  Limits how many internal attempts the agent can do. This prevents loops and keeps execution time predictable.

Attached component: Chat Model

The LLM provider and model used by the Query Builder.

Your configuration

- Credential: API Key from Google Cloud account
- Model: models/gemini-2.5-flash

Attached component: Output Parser (Structured Queries Output)

A schema enforcer that validates the LLM response and guarantees the output JSON structure for downstream nodes.

Configuration

- Schema Type: Generate From JSON Example
- JSON Example:

```
{
  "esearch": {
    "term": "\"type 2 diabetes\" lifestyle intervention",
    "retmax": 3,
    "sort": "pub_date"
  },
  "metadata": {
    "source": "esummary",
    "fields": ["title", "fulljournalname", "pubdate", "authors"]
  },
  "answer_constraints": {
    "output_format": "table",
    "include_pmids": true
  }
}
```

- Auto-Fix Format: ON

Input

- The checked user message text (from Guardrails pass)

Output

- A JSON plan that the next nodes can use, e.g.:
  - term, retmax, sort, and metadata field list

4) PubMed Search — HTTP Request (E-utilities ESearch)

Description

This node calls PubMed ESearch to retrieve a list of PMIDs matching the query.

Important configuration

- **Method**: GET
- **Base URL**: https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi
- **Send Query Parameters:** ON
- **Specify Query Parameters**: Using JSON
- **JSON Expression**

```
{
  "db": "pubmed",
  "term": "{{ $json.output.esearch.term }}",
  "retmode": "json",
  "retmax": {{ $json.output.esearch.retmax }},
  "sort": "{{ $json.output.esearch.sort }}",
  "tool": "n8n-med-ed-lab",
  "email": "YOUR_REAL_EMAIL@DOMAIN"
}
```

Input

- The query plan from Node 3 (especially term, retmax, sort)

Output

- ESearch JSON response containing:
    - esearchresult.idlist (PMIDs)

5) PubMed Metadata — HTTP Request (E-utilities ESummary)

Description

This node calls PubMed ESummary to fetch metadata (title, journal, date, authors) for the PMIDs returned by ESearch.

Important configuration

- **Method**: GET
- **Base URL**: https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esummary.fcgi
- **Send Query Parameters:** ON
- **Specify Query Parameters**: Using JSON
- **JSON Expression**

```
{
  "db": "pubmed",
  "id": "{{ $json.esearchresult.idlist }}",
  "retmode": "json",
  "tool": "n8n-med-ed-lab",
  "email": "YOUR_REAL_EMAIL@DOMAIN"
```

```
}
```

Input

- PMIDs from Node 5

Output

- ESummary JSON response containing:
    - result.uids (ordered list of PMIDs)
    - result[PMID].title, fulljournalname/source, pubdate/epubdate, etc.

6) Merge — Combine request context + PubMed metadata

Description

This node merges the user request context (Guardrails output) with PubMed ESummary payload, so the final formatter node can see everything needed in one item.

Important configuration

- **Mode**: "Combine"
- **Combine By**: Position
- **Number of Inputs**: 2
    - **Input 1**: PubMed Metadata output (ESummary JSON)
    - **Input 2**: Guardrails (or the request context output)

Input

- ESummary result (from Node 6) + request text (from Node 2 pass branch)

Output

- A single combined JSON item that includes:
    - guardrailsInput (user request)
    - result (ESummary payload)

7) JsonToText — Final response formatter (LLM)

Description

This node generates the final, user-facing response (e.g., "latest 3 studies with PMID + title + journal + year") using the merged ESummary JSON.

Important configuration

- **Source for Prompt (User Message):** Define below
- **Prompt (expression)**

```
User request:
{{ $json.guardrailsInput }}
```

```
PubMed ESummary payload (JSON):
{{ JSON.stringify($json.result, null, 2) }}
```

- **System message**

```
You are a PubMed results formatter for end users.

You will receive:
1) The user request text
2) A PubMed ESummary JSON payload in the user message

Task:
1) Read the user request from guardrailsInput.
2) Using PMIDs in result.uids (in that order), select the number of studies
requested:
   - If the user asks for “latest 3” or “3 studies”, return at most 3.
   - Otherwise, return up to 5 unless the user specifies another number.
3) For each selected PMID, extract:
   - Title: result[PMID].title
   - Journal: prefer result[PMID].fulljournalname; if missing use
result[PMID].source
   - Year: extract the first 4-digit year found in result[PMID].pubdate; if none,
try result[PMID].epubdate; otherwise leave blank.

Output requirements:
- Output plain text only. Do NOT output JSON.
- Provide a clean list format that is easy to read, for example:

1) PMID: <pmid>
   Title: <title>
   Journal: <journal>
   Year: <year>

Grounding and safety constraints:
- Do not invent studies or details not present in the input JSON.
- If a field is missing, leave it blank or write “(not available)”.
- Keep the response non-prescriptive: do not provide diagnosis or personalized
treatment recommendations.
- Do not add extra medical advice; only format and present the retrieved studies.
```

Input

- Merged JSON from Node 7 (request + metadata)

Output

- A single text response (ideally in a field like output or text)

Summary Memory (attached to Node 7)

Stores short conversational context across turns (e.g., previous query topic) so the assistant can handle follow-up questions.

Configurations

- **Session ID:** Connected Chat Trigger Node
- **Context Window Length:** 5

- **Session Key From Previous Node:** {{ $json.sessionId }}

9) Respond to Chat — Send final answer

Description

This node returns the final formatted answer to the user in the chat UI.

Important configuration

- **Message**
  - {{$json.output}}

Input

- Final answer text from Node 8

Output

- User-visible chat response