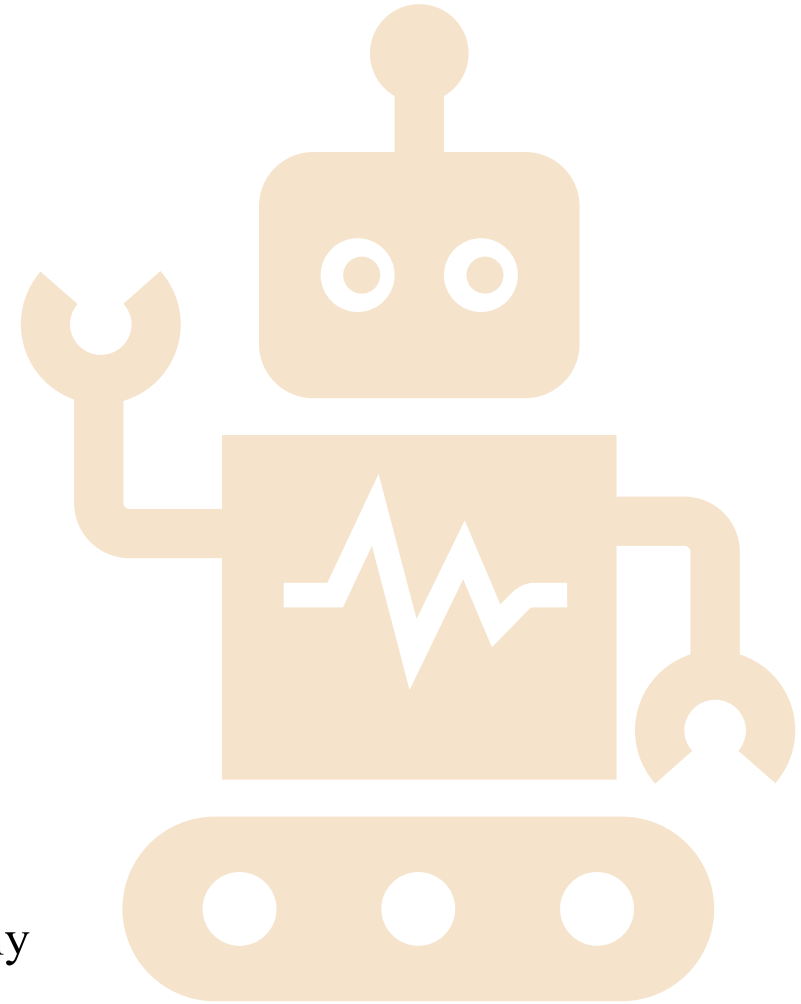
Building AI Chatbots: From PDFs to Intelligent Assistants

- Maryam Ataei
- Maryam.Ataei@kamk.fi
- Project Specialist at Kajaani University of Applied Science

- Case study: Chatbot for a Finnish Healthcare and Educational Company



HealthHub Finland EDIH KAMK aika

What Are Large Language Models (LLMs)?

- LLMs are AI systems trained to understand and generate human language.
- They've read huge amounts of text (books, websites, articles) to learn patterns of language.
- When you ask a question, they predict the next best words, like an extremely smart autocomplete.
- Modern LLMs can speak many languages, reason about problems, and generate detailed answers.
- Examples: GPT-4, Gemini, Claude, Mistral.

What Can Large Language Models Do?

- Summarize and explain long documents 📄
- Translate between languages 🌎
- Write or debug code 💻
- Create and reason over structured data 📊
- Answer questions conversationally 💬
- Build intelligent assistants 🩺

Case Study :

- **Who are they?**

  - A Finnish healthcare company offering **online health courses**
  - Focus on supporting patients after diagnosis (e.g., sleep apnea, COPD, women's health)
  - Provides **expert knowledge, practical tips, and peer support**
  - Courses available online, flexible, and accessible for one year

Why Do We Need Document-Based Chatbots?

AI Hallucinations

While AI models like ChatGPT are incredibly powerful, they have a critical flaw: they can confidently provide incorrect information. This "hallucination" problem means AI might:

- ❑Invent statistics that don't exist
- ❑Create procedures that were never approved
- ❑Mix information from different sources incorrectly
- ❑Sound completely confident while being completely wrong

Traditional Search Limitations

Keyword-based search tools only match exact words, not meaning:

- ❑Time-consuming searches across hundreds of documents
- ❑Missing relevant information due to terminology differences
- ❑Inefficient even when you know exactly what you need
- ❑No understanding of context or synonyms

The Solution: Document-Based AI

- ❑Answers ONLY from your verified documents
- ❑Understands meaning, not just keywords
- ❑Provides source citations for trust
- ❑Says "I don't know" when information isn't available

The RAG Pipeline
(Retrieval-Augmented Generation)

Your Documents

📄 PDFs

📊 JSONs

📋 Texts

→

RAG Pipeline

🔪 Chunking

🔢 Embeddings

🗄️ Vector Store

🔍 Retrieval

🤖 Generation

→

Smart Answers

💬 "What is sleep apnea? "

↓

🧮 "Sleep apnea
is a condition where... " +
[Source: page 42]

Step 1 - Loading Documents

What We Can Process:

- **PDFs**: Medical guides, research papers, manuals
- **JSON**: Structured data, web content
- **Text**: Any text-based content

```python
from PyPDF2 import PdfReader

def extract_pdf_text(pdf_file):
    """Extract all text from a PDF"""
    reader = PdfReader(pdf_file)
    text = ""

    for page in reader.pages:
        text += page.extract_text()

    return text

# Example usage
text = extract_pdf_text("Manual.pdf")
print(f"Extracted {len(text)} characters")
```

Step 2 - Smart Chunking

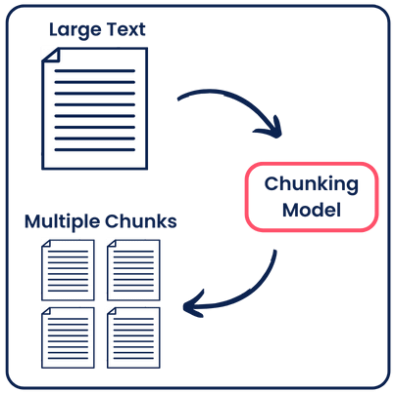
Breaking Documents Into Smaller Pieces

Why Chunk?

- Language models (like GPT) can't read huge files at once.
- Smaller chunks help the AI find just the parts that are most relevant to a question.
- If you don't chunk, you risk missing information or overloading the model.

Why Overlap is Crucial:

Overlap ensures no critical information is split across chunks, maintaining context for the embedding step




```python
def create_chunks(text, chunk_size=500, overlap=100):
    """Split text into overlapping chunks"""
    chunks = []
    for i in range(0, len(text), chunk_size - overlap):
        chunk = text[i:i + chunk_size]
        chunks.append(chunk)
    return chunks

chunks = create_chunks(text)

for i, chunk in enumerate(chunks):
    print(f"Chunk {i+1}:\n{chunk}\n---")
```

```python
from langchain.text_splitter import CharacterTextSplitter

def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks
```

Step 3 - The Magic of Embeddings

Converting Chunks to Numbers

The Problem:

- Computers don't understand "meaning"
- They only understand numbers
- How do we bridge this gap?

Think of it like GPS coordinates:

- “Lappeenranta” → (61°03’ N, 28°11’ E)
- “Helsinki” → (60°10’ N, 24°56’ E)

Cities close together have similar coordinates!

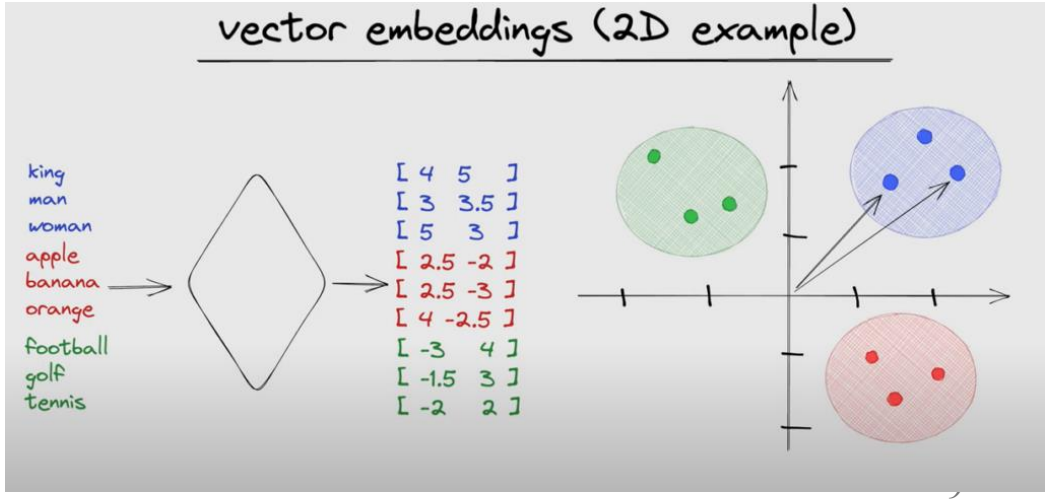
- “Rome” → (41°53’ N, 12°28’ E) (very different!)

Similarly:

- "Sleep apnea symptoms" → [0.2, 0.8, -0.3, ...]
- "Signs of sleep apnea" → [0.3, 0.7, -0.2, ...]
- "Nutrition of salmon soup" → [-0.9, 0.1, 0.5, ...] (very different!)

The Solution: Embeddings

"Embeddings are the bridge between language and math." Embedding = Converting text meaning into a list of numbers



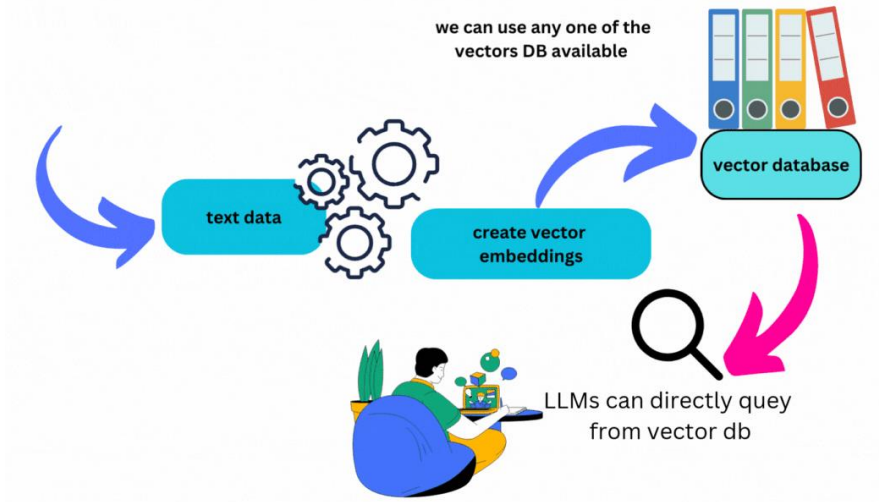
Not All Embeddings Are the Same!

When building a chatbot that works with documents in different languages, the choice of embedding model matters a lot.

- Some models, like *all-MiniLM-L6-v2*, work best for English and may miss important context in Finnish or other languages.
- Others, such as *paraphrase-multilingual-MiniLM-L12-v2*, are trained to handle many languages (including Finnish), making them much better for Finnish documents.
- OpenAI embedding models (like *text-embedding-ada-002*) are strong for many languages, but may have privacy/cost concerns compared to local models.

If your sources are in Finnish, multilingual embeddings will find more accurate and relevant information than English-focused models. Always test a few models to see which gives the best results for your real-world documents!

Step 4 - Vector Database



Building a Super-Fast Search Engine

What is FAISS?

- Facebook AI Similarity Search
- "Like Google, but for meaning“
- Finds similar ideas, not just keywords

"We use a vector database, so searching for meaning is as quick as searching for words."

```python
from langchain.vectorstores import FAISS

# Create vector database
vectorstore = FAISS.from_texts(
    texts=chunks,           # Your text chunks
    embedding=embeddings    # The AI that converts to numbers
)

# Search for similar content
question = "What causes sleep apnea?"
similar_chunks = vectorstore.similarity_search(
    question,
    k=3  # Get top 3 most similar chunks
)
```

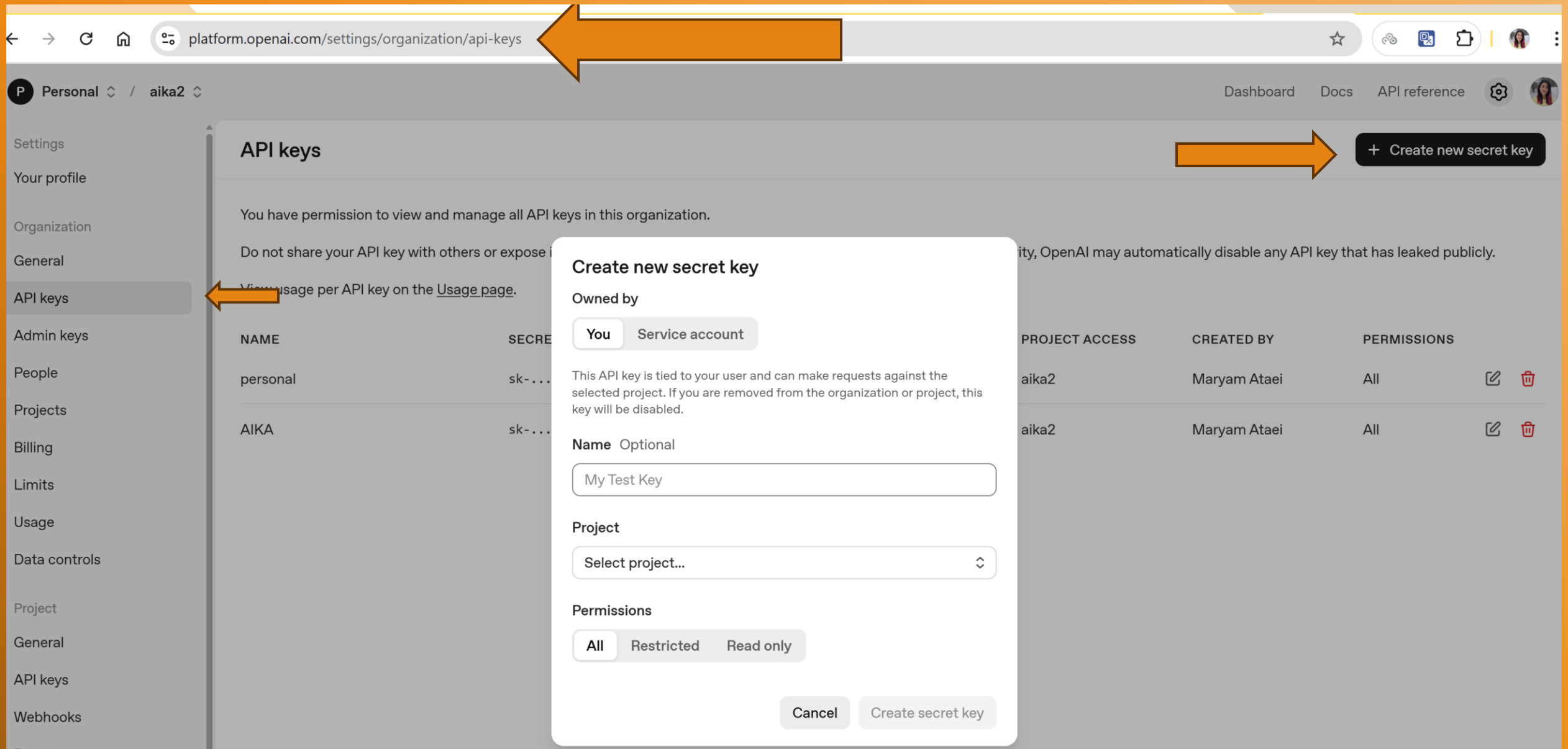
What Are APIs (Application Programming Interface)?

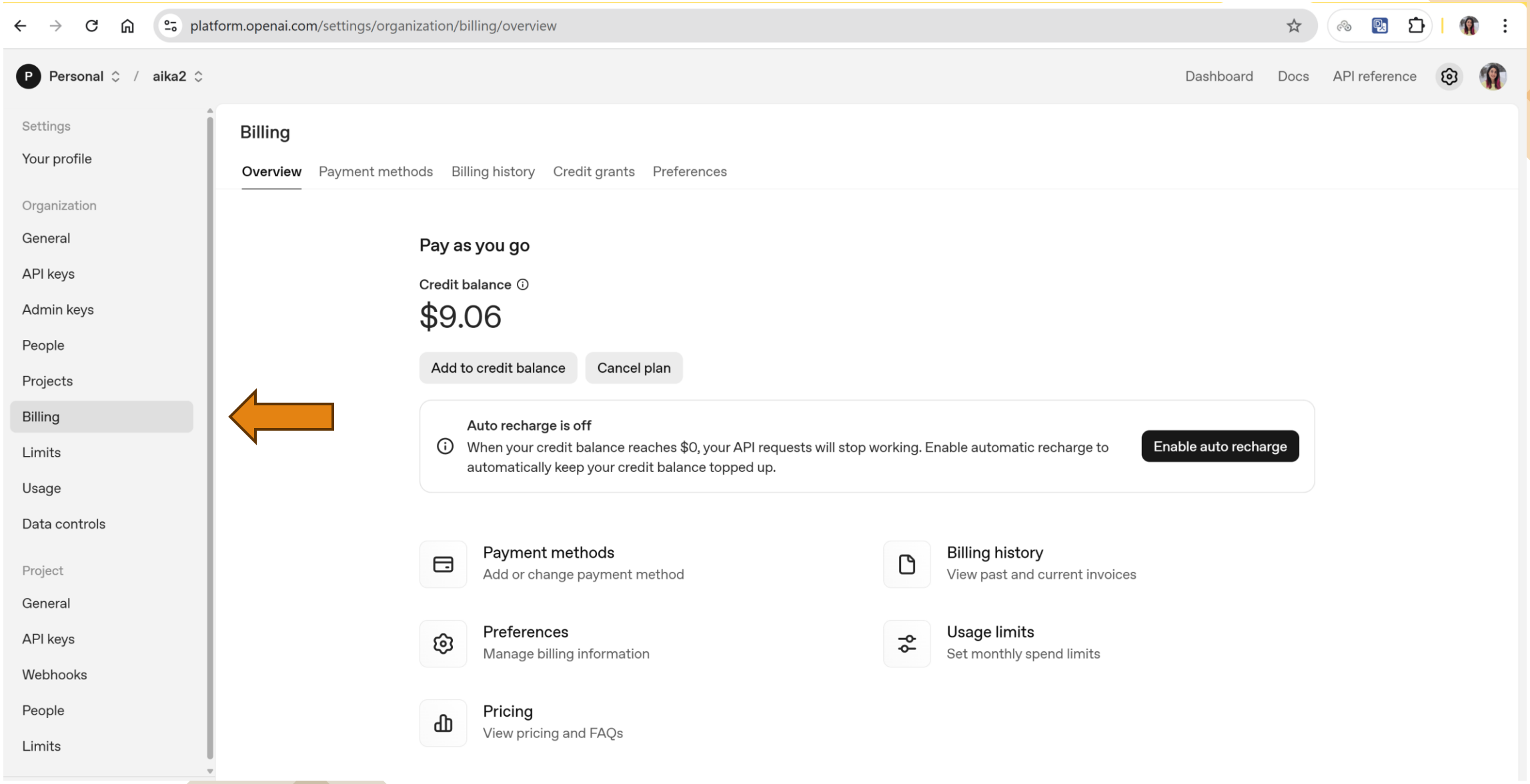
❖ This is your Gateway to AI Power

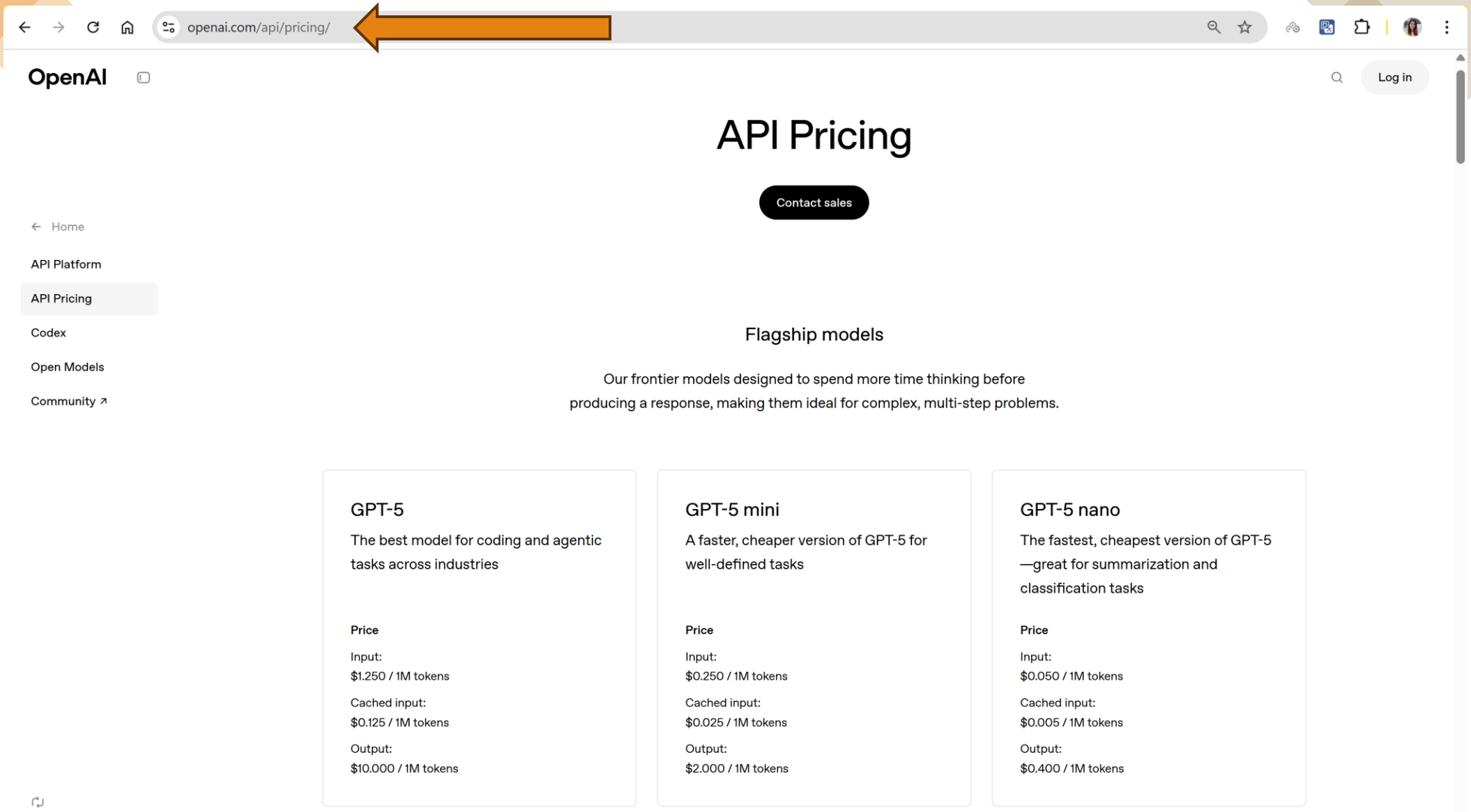
- ❑ Getting Started: Sign up at openai.com (or other AI providers like Hugging Face, Anthropic)
- ❑ Get API key ("Your secret password to access AI models")
- ❑ Pay per use (usually < $0.01 per question)

```
# Without API (impossible):
# ❌ Run GPT-4 on your laptop (needs $100k+ GPU!)
# You can only run very light models such as Mistral 7B, …
according to your laptop power.



# With API (easy):
# ✅ Send question to OpenAI servers
response = openai.ChatCompletion.create(
    model="gpt-4.1",
    messages=[{"role": "user", "content": "Hello!"}]
)
# ✅ Get answer back in seconds!
```

Once we have the most relevant document chunks, we need a powerful language model to generate a final answer. This is where the OpenAI API comes in.


platform.openai.com/settings/organization/api-keys
Personal / aika2
Dashboard
Docs
API reference
Settings
Your profile
Organization
General
API keys
Admin keys
People
Projects
Billing
Limits
Usage
Data controls
Project
General
API keys
Webhooks
People
Limits
API keys
+ Create new secret key
You have permission to view and manage all API keys in this organization.
Do not share your API key with others or expose i
ity, OpenAI may automatically disable any API key that has leaked publicly.
usage per API key on the Usage page.
NAME
SECRE
PROJECT ACCESS
CREATED BY
PERMISSIONS
personal
sk-...
aika2
Maryam Ataei
All
AIKA
sk-...
aika2
Maryam Ataei
All
Create new secret key
Owned by
You
Service account
This API key is tied to your user and can make requests against the selected project. If you are removed from the organization or project, this key will be disabled.
Name Optional
My Test Key
Project
Select project...
Permissions
All
Restricted
Read only
Cancel
Create secret key


platform.openai.com/settings/organization/billing/overview
Personal / aika2
Dashboard
Docs
API reference
Settings
Your profile
Organization
General
API keys
Admin keys
People
Projects
Billing
Limits
Usage
Data controls
Project
General
API keys
Webhooks
People
Limits
Billing
Overview
Payment methods
Billing history
Credit grants
Preferences
Pay as you go
Credit balance
$9.06
Add to credit balance
Cancel plan
Auto recharge is off
When your credit balance reaches $0, your API requests will stop working. Enable automatic recharge to automatically keep your credit balance topped up.
Enable auto recharge
Payment methods
Add or change payment method
Billing history
View past and current invoices
Preferences
Manage billing information
Usage limits
Set monthly spend limits
Pricing
View pricing and FAQs


openai.com/api/pricing/
OpenAI
Log in
API Pricing
Contact sales
Home
API Platform
API Pricing
Codex
Open Models
Community
Flagship models
Our frontier models designed to spend more time thinking before producing a response, making them ideal for complex, multi-step problems.
GPT-5
The best model for coding and agentic tasks across industries
Price
Input:
$1.250 / 1M tokens
Cached input:
$0.125 / 1M tokens
Output:
$10.000 / 1M tokens
GPT-5 mini
A faster, cheaper version of GPT-5 for well-defined tasks
Price
Input:
$0.250 / 1M tokens
Cached input:
$0.025 / 1M tokens
Output:
$2.000 / 1M tokens
GPT-5 nano
The fastest, cheapest version of GPT-5 —great for summarization and classification tasks
Price
Input:
$0.050 / 1M tokens
Cached input:
$0.005 / 1M tokens
Output:
$0.400 / 1M tokens

Step 5 - Generating Smart Answers with RAG

Bringing It All Together: Retrieval + Generation = RAG

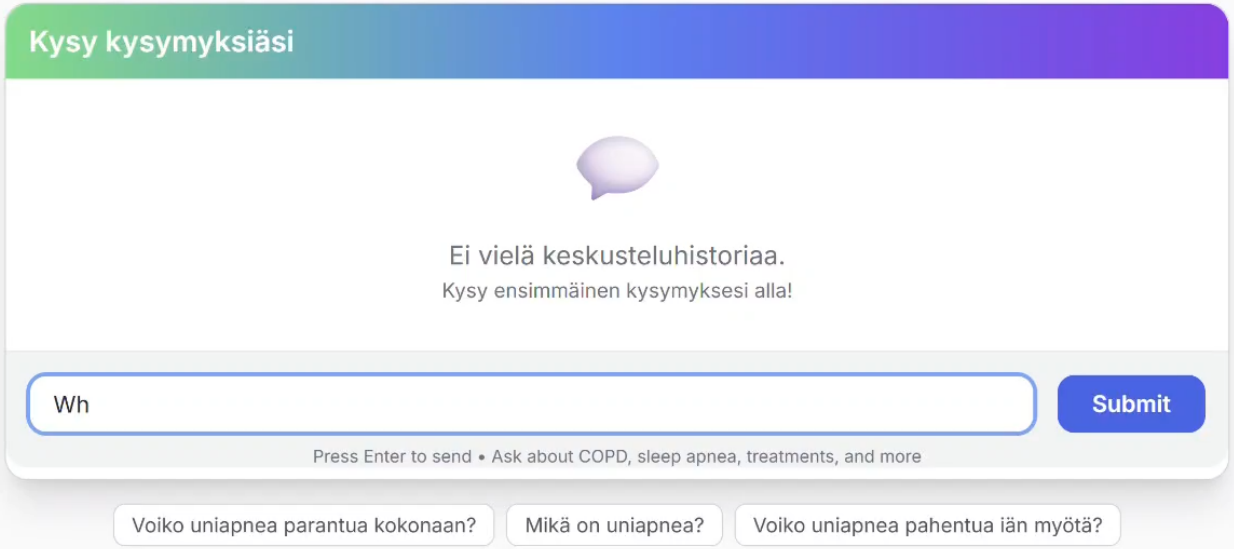
- The RAG Process: [User Question]
  ↓
- [RETRIEVAL: Find relevant document chunks]
  ↓
- [GENERATION: Send both to OpenAI API with instructions]
  ↓
- ✓ [Get answer, limited to those sources]

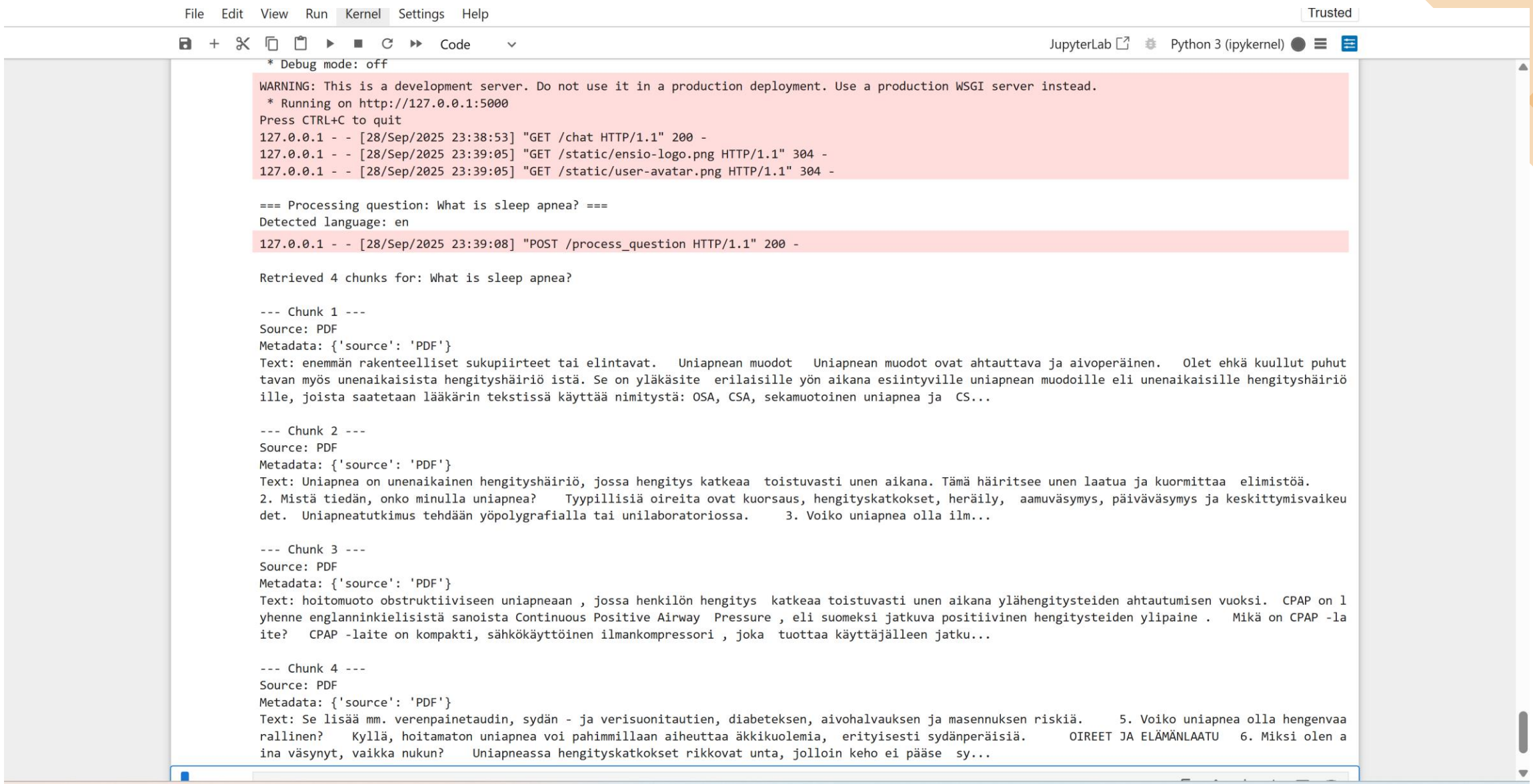
This combination of Retrieval and Generation is why it's called RAG!

```
prompt = f"""
You are a helpful medical assistant.
Use ONLY the following information to answer.
If you don't know, say "I don't have that information."

Context from documents:
{relevant_chunks}

Question: {user_question}

Answer:
"""

# Send to GPT-4
response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "system", "content": prompt}]
)
```

Without RAG: "Sleep apnea affects 25% of adults" (Made up statistic!)

With RAG: "According to page 42 of the medical guide, sleep apnea affects..."


Kysy kysymyksiäsi
Ei vielä keskusteluhistoriaa.
Kysy ensimmäinen kysymyksesi alla!
Wh
Submit
Press Enter to send • Ask about COPD, sleep apnea, treatments, and more
Voiko uniapnea parantua kokonaan?
Mikä on uniapnea?
Voiko uniapnea pahentua iän myötä?


File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [28/Sep/2025 23:38:53] "GET /chat HTTP/1.1" 200 -
127.0.0.1 - - [28/Sep/2025 23:39:05] "GET /static/ensio-logo.png HTTP/1.1" 304 -
127.0.0.1 - - [28/Sep/2025 23:39:05] "GET /static/user-avatar.png HTTP/1.1" 304 -
=== Processing question: What is sleep apnea? ===
Detected language: en
127.0.0.1 - - [28/Sep/2025 23:39:08] "POST /process_question HTTP/1.1" 200 -
Retrieved 4 chunks for: What is sleep apnea?
--- Chunk 1 ---
Source: PDF
Metadata: {'source': 'PDF'}
Text: enemmän rakenteelliset sukupiirteet tai elintavat. Uniapnean muodot Uniapnean muodot ovat ahtauttava ja aivoperäinen. Olet ehkä kuullut puhut tavan myös unenaikaisista hengityshäiriö istä. Se on yläkäsite erilaisille yön aikana esiintyville uniapnean muodoille eli unenaikaisille hengityshäiriö ille, joista saatetaan lääkärin tekstissä käyttää nimitystä: OSA, CSA, sekamuotoinen uniapnea ja CS...
--- Chunk 2 ---
Source: PDF
Metadata: {'source': 'PDF'}
Text: Uniapnea on unenaikainen hengityshäiriö, jossa hengitys katkeaa toistuvasti unen aikana. Tämä häiritsee unen laatua ja kuormittaa elimistöä. 2. Mistä tiedän, onko minulla uniapnea? Tyypillisiä oireita ovat kuorsaus, hengityskatkokset, heräily, aamuväsymys, päiväväsymys ja keskittymisvaikeu det. Uniapneatutkimus tehdään yöpolygrafialla tai unilaboratoriossa. 3. Voiko uniapnea olla ilm...
--- Chunk 3 ---
Source: PDF
Metadata: {'source': 'PDF'}
Text: hoitomuoto obstruktiiviseen uniapneaan , jossa henkilön hengitys katkeaa toistuvasti unen aikana ylähengitysteiden ahtautumisen vuoksi. CPAP on l yhenne englanninkielisistä sanoista Continuous Positive Airway Pressure , eli suomeksi jatkuva positiivinen hengitysteiden ylipaine . Mikä on CPAP -la ite? CPAP -laite on kompakti, sähkökäyttöinen ilmankompressori , joka tuottaa käyttäjälleen jatku...
--- Chunk 4 ---
Source: PDF
Metadata: {'source': 'PDF'}
Text: Se lisää mm. verenpainetaudin, sydän - ja verisuonitautien, diabeteksen, aivohalvauksen ja masennuksen riskiä. 5. Voiko uniapnea olla hengenvaa rallinen? Kyllä, hoitamaton uniapnea voi pahimmillaan aiheuttaa äkkikuolemia, erityisesti sydänperäisiä. OIREET JA ELÄMÄNLAATU 6. Miksi olen a ina väsynyt, vaikka nukun? Uniapneassa hengityskatkokset rikkovat unta, jolloin keho ei pääse sy...

Turning Your Chatbot Into a Web App

Why not just a Python script?

- Running a script means you need Python installed and have to use the command line.
- Only people who can code or who have your environment can use it.
- There's no user-friendly interface. No buttons, no chat bubbles, no web access.

	Python Script	Web App
Who can use?	You/Developers	Anyone with a browser
How to use?	Command line, scripts	Type in a browser
Looks like	Text on black screen	Chat page, buttons

What Makes a Web App Work?

Flask: One Simple Way to Turn Code Into a Web App

- Flask is a lightweight Python tool for building web apps
- It’s popular because it’s easy to learn and flexible
- Handles website addresses (like /, /chat) and user interactions
- Connects your AI logic to a simple web page

How Your Chatbot Gets Its “Look”

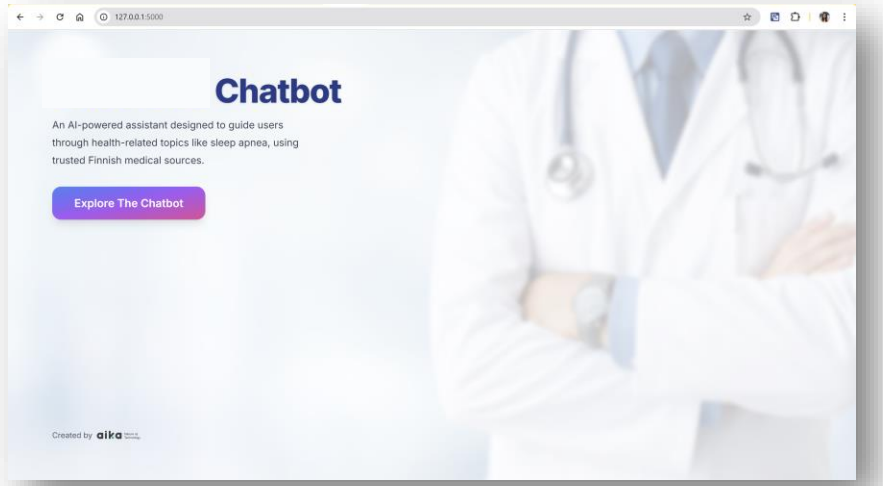
HTML Templates: Designing the Chatbot Interface

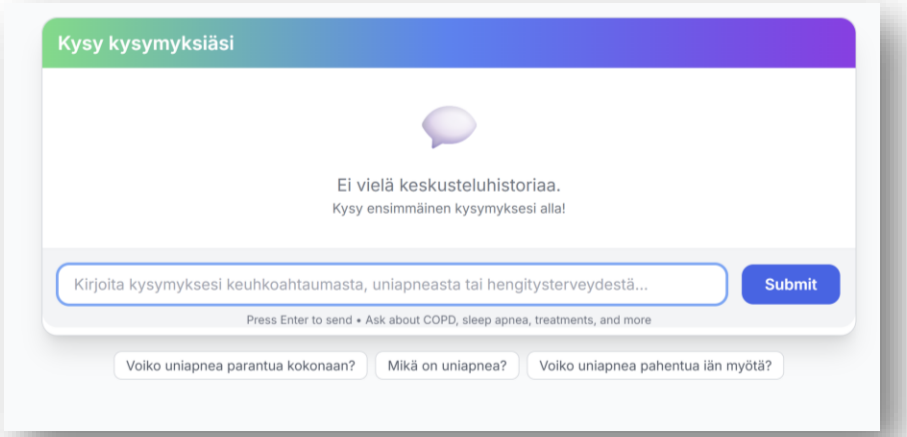
- Files in templates/ folder, e.g., index.html, chat.html
- Mix of HTML + placeholders (e.g., for showing chat messages)
- Flask fills in data from your Python code
- Lets you control how your chatbot looks and feels
- Easy to edit, customize, and style

```
<!-- chat.html -->
{% for message in chat_history %}
  <div>{{ message.content }}</div>
{% endfor %}
```

static/: Images, CSS (for styles), JavaScript (for interactivity)

Flask uses these folders so you can easily change how your app looks, without touching the python code





Project Folder Example

Keep it tidy for easier updates, teamwork, and deployment!

my-chatbot/	
├ app.py	# Main application code
├ requirements.txt	# Needed packages
├ templates/	# HTML templates
├ static/	# Images, CSS, JS
├ documents/	# PDFs and other files
└ README.md	# Usage instructions

How to Organize Your Project for GitHub and Azure

You can create a folder in your GitHub repository that has the same structure as the one you use on your own computer for your chatbot project.

Keep all the important folders and files:

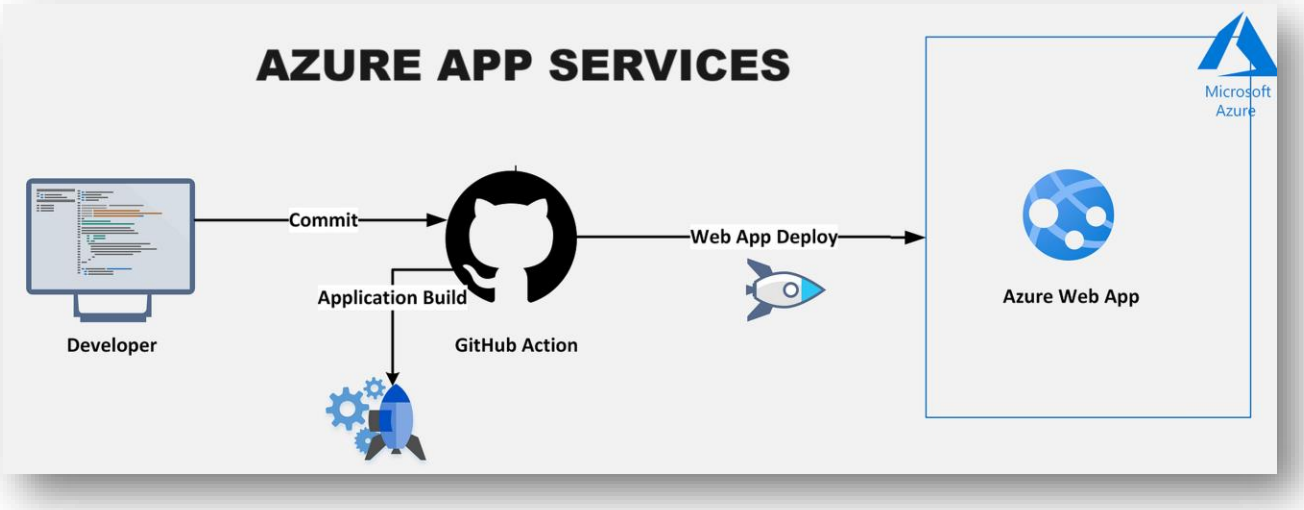
- Your main Python files (like app.py)
- The templates folder (for your HTML files)
- The static folder (for styles and images)
- The documents folder (for your PDFs or data)
- Any other folders your app needs

This is a common way to share and deploy Python web apps.

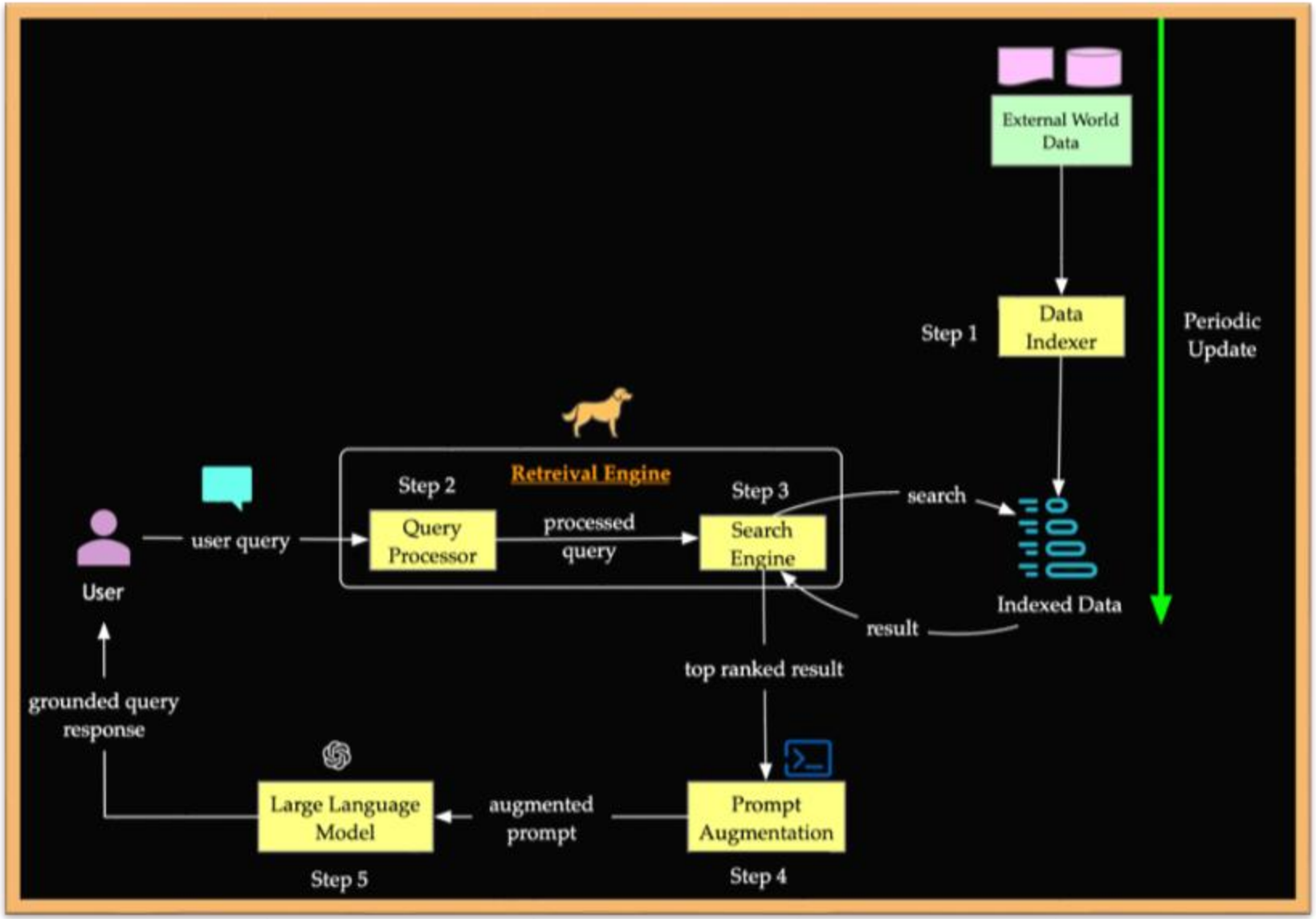
Deploying Your Chatbot as a Web App (Azure Example)

How to Put Your Chatbot Online

1. Sign up at portal.azure.com
2. Create a new “Web App” (choose Python/Flask)
3. Link your GitHub repo for easy updates
4. Set environment variables (API keys) securely in Azure dashboard
5. Azure installs dependencies and runs your Flask app
6. Your chatbot is now live! (e.g., https://yourapp.azurewebsites.net)



No server setup, automatic updates, secure secrets, and global access.


External World Data
Step 1
Data Indexer
Periodic Update
Retreival Engine
Step 2
Query Processor
processed query
Step 3
Search Engine
search
user query
User
Indexed Data
result
top ranked result
grounded query response
Large Language Model
augmented prompt
Prompt Augmentation
Step 5
Step 4

Thanks