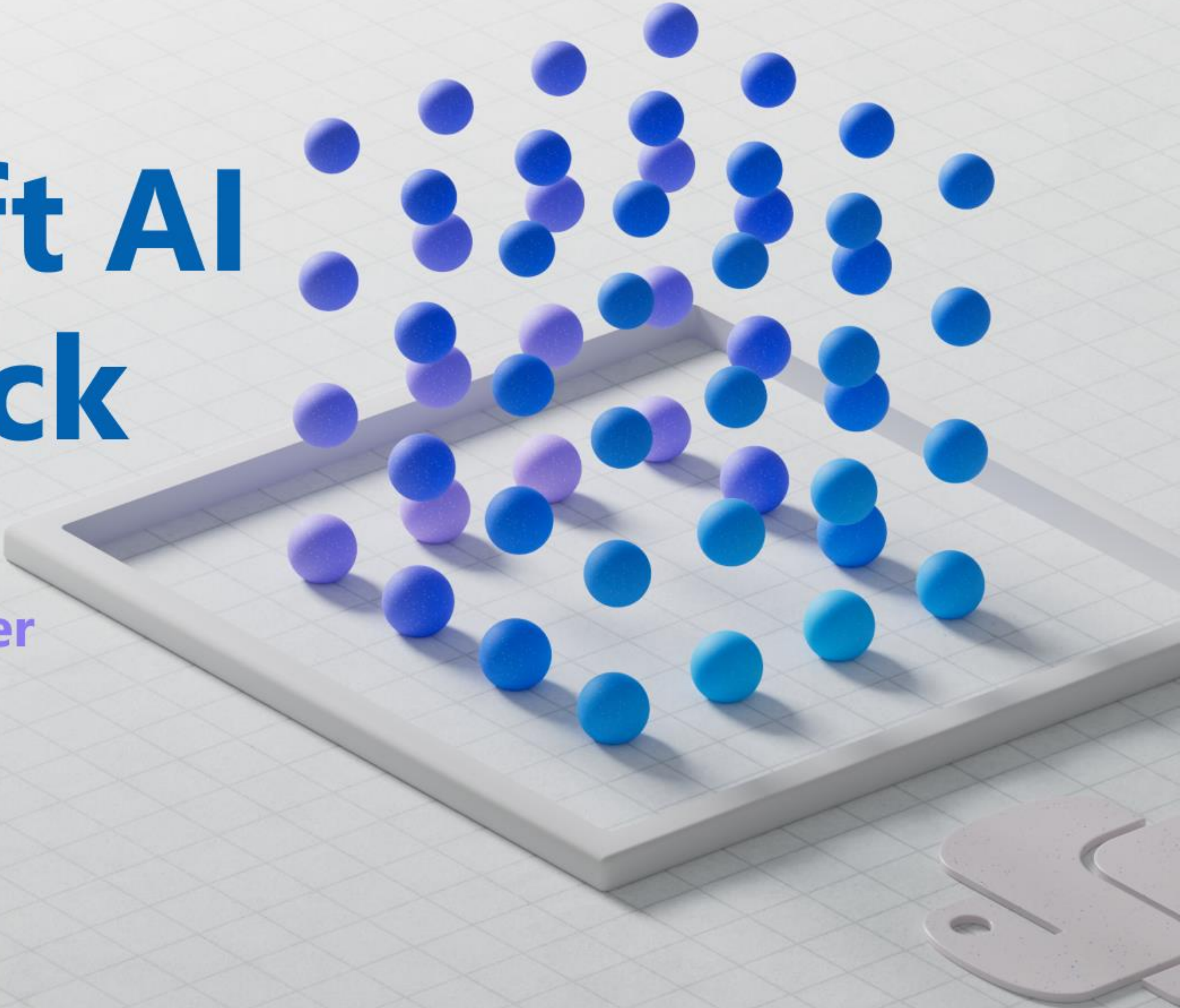
January 29th - February 12th

The Microsoft AI Chat App Hack

Build, innovate, and #HackTogether

aka.ms/hacktogether/chatapp



Microsoft

The AI Chat App Hack

January 29th - February 12th

29th ▶
Building a RAG Chat App in Python

30th ▶
Customizing your RAG Chat App

31st ▶
Azure AI Search Best Practices

1st ▶
GPT-4 with Vision

2nd
HACK
HACK
HACK

3rd
HACK
HACK

4th
HACK
HACK
HACK

5th ▶
AM: RAG Chat Web Components

PM: Access Control in RAG Chat Apps

6th ▶
Evaluating a RAG Chat App

7th ▶
RAG Chat Special Topic

8th ▶
Continuous Deployment of your Chat App

9th
HACK
HACK

10th
HACK

11th
HACK
HACK

12th
SUBMIT YOUR PROJECT

Build, innovate, and #HackTogether

Customizing your RAG chat app

Prerequisites: RAG Chat App

This talk will be all about customizing this app:

https://github.com/Azure-Samples/azure-search-openai-demo

Before you can customize it, you must deploy all the resources.

See this doc for tips on deploying for free:

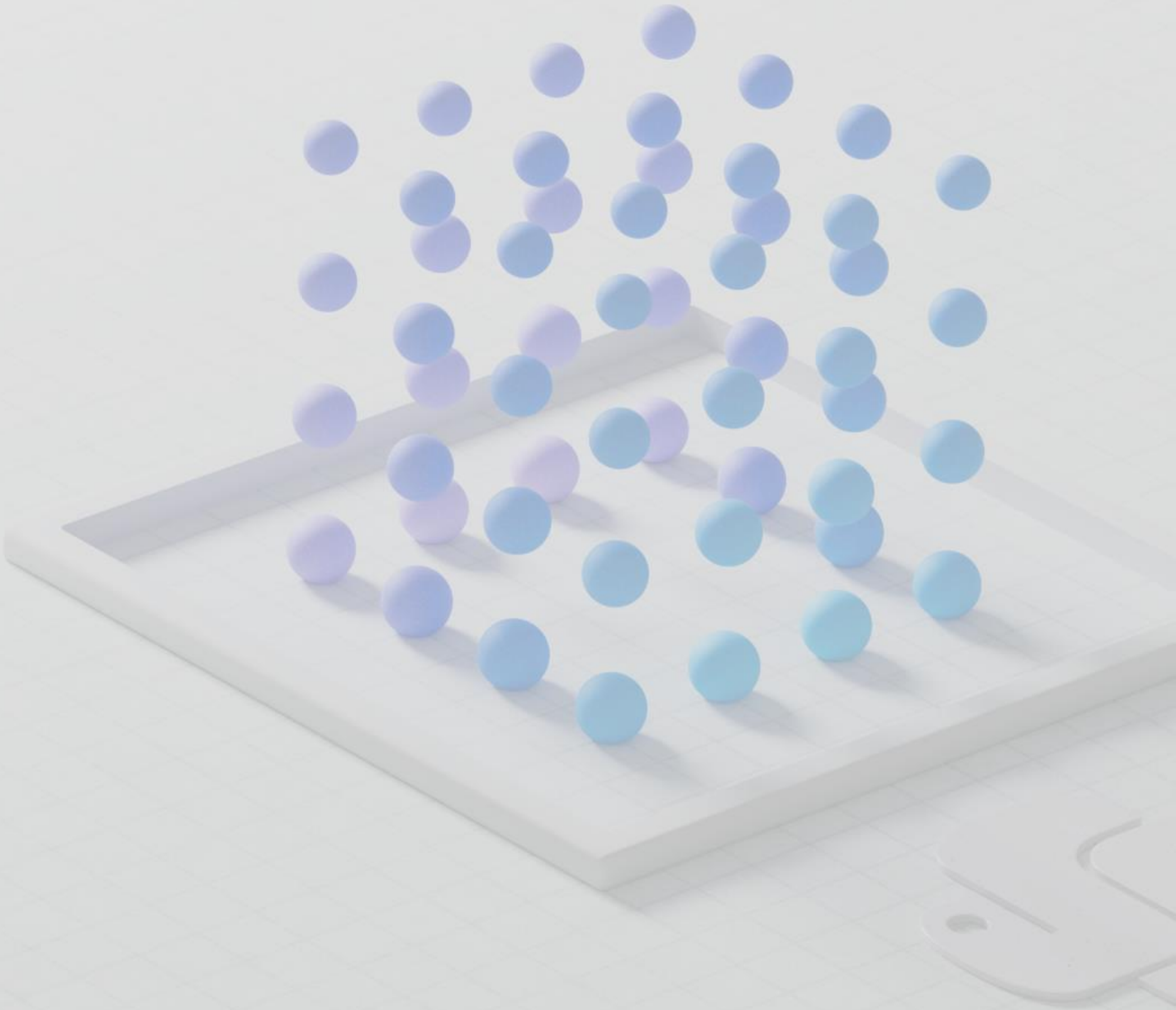
🔗 aka.ms/ragchat/free

(Keep watching either way, you can always rewatch after you deploy!)

Today's topics

- Local development tips (🔥 reload)
- Customizing the frontend (UI)
- Customizing the backend
- Ingesting custom data

Local development

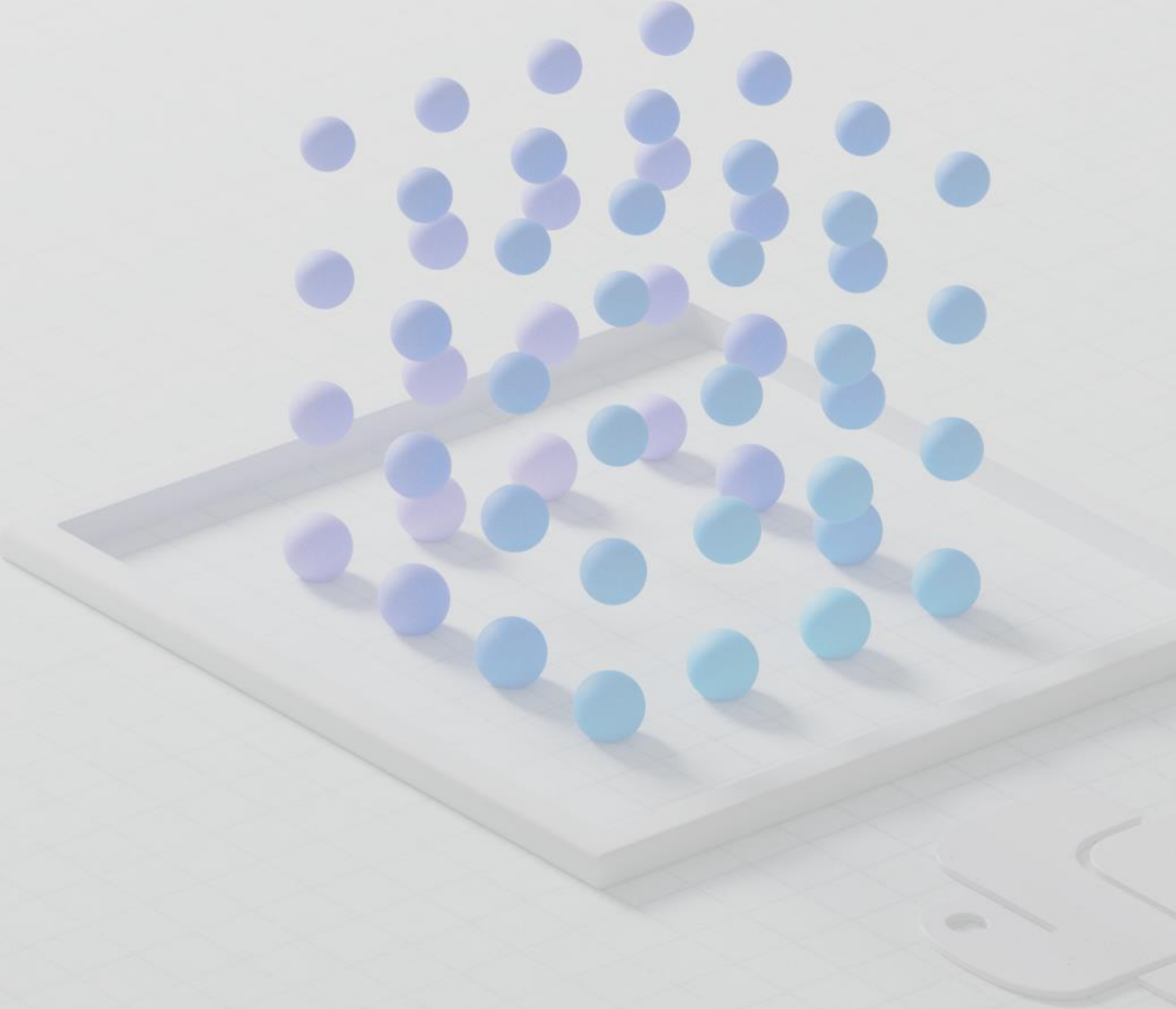


Starting the local server

```
cd app
./start.sh
```

1. Packages the React frontend typescript files using vite
2. Copies the JS into app/backend folder
3. Starts the Python app in reload mode (which only watches Python files!)

Customizing the frontend



Hot reloading for frontend files

In a new terminal:

```
cd app/frontend
npm run dev
```

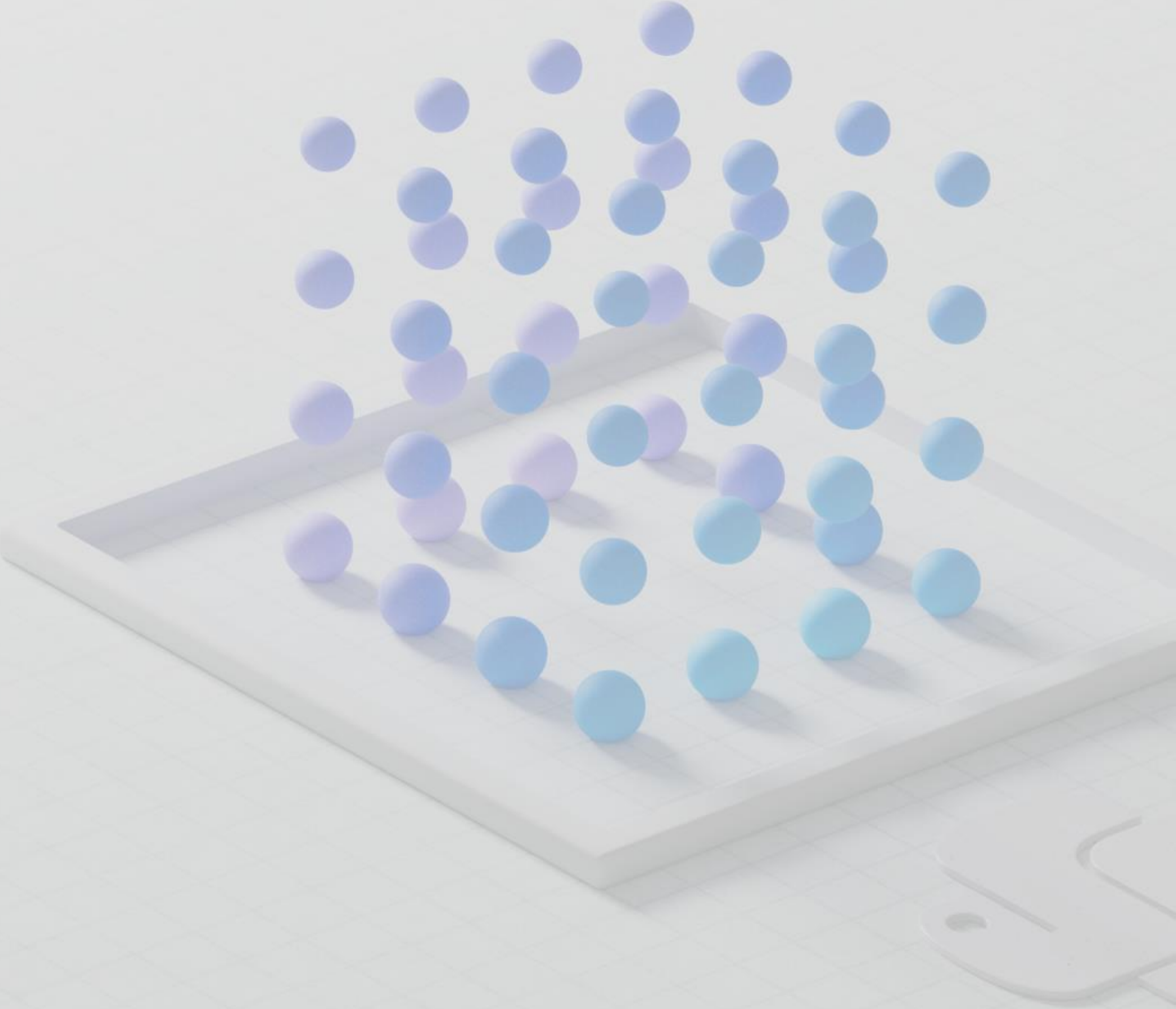
Then navigate to the URL shown in the output.

The backend will still proxy to the previously started Python server.

Frontend components

Change this file:	To customize:
app/frontend/index.html	title, metadata, script tag
app/frontend/public/favicon.ico	browser tab icon
app/frontend/src/pages/layout/Layout.tsx	Navigation bar, colors
app/frontend/src/pages/chat/Chat.tsx	“Chat” tab and default settings
app/frontend/src/pages/ask/Ask.tsx	“Ask” tab and default settings

Customizing the backend



Backend components

Change this file:	To customize:
app/backend/app.py	additional routes, app configuration
app/backend/approaches/chatreadretrieveread.py	“Chat” tab, RAG prompt and flow
app/backend/approaches/chatreadretrievereadvision.py	“Chat” tab, RAG flow when using vision
app/backend/approaches/retrievethenread.py	“Ask” tab, RAG prompt and flow
app/backend/approaches/retrievethenreadvision.py	“Ask” tab, RAG flow when using vision

Changing the prompt

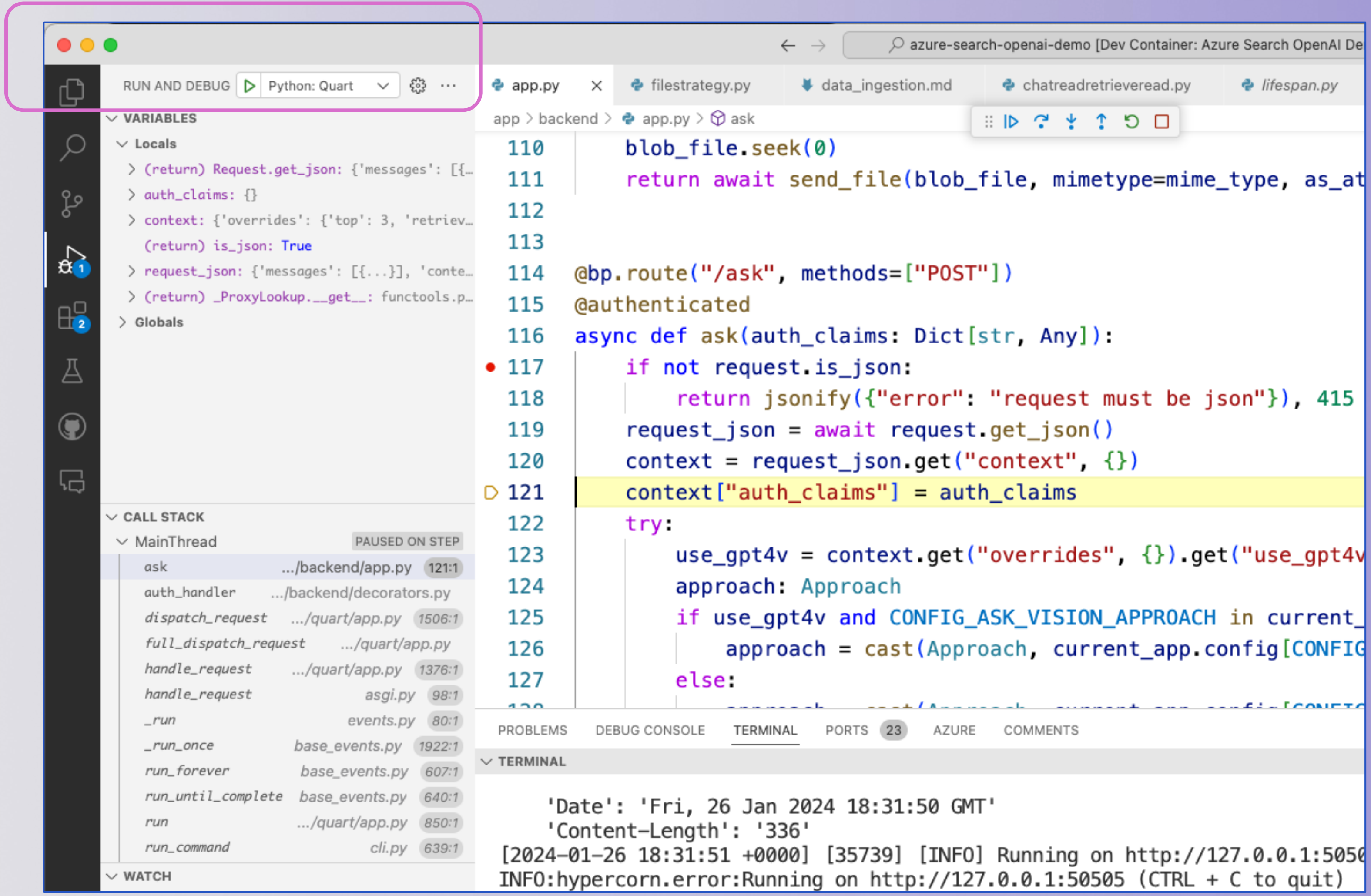
Assistant helps the company employees with their healthcare plan questions, and questions about the employee handbook. Be brief in your answers.

Answer ONLY with the facts listed in the list of sources below. If there isn't enough information below, say you don't know. Do not generate answers that don't use the sources below. If asking a clarifying question to the user would help, ask the question.

For tabular information return it as an html table. Do not return markdown format. If the question is not in English, answer in the language used in the question.

Each source has a name followed by colon and the actual information, always include the source name for each fact you use in the response. Use square brackets to reference the source, for example [info1.txt]. Don't combine sources, list each source separately, for example [info1.txt][info2.pdf].

VS code debugger for backend



VARIABLES

Locals

```
(return) Request.get_json: {'messages': [{…
auth_claims: {}
context: {'overrides': {'top': 3, 'retriev…
(return) is_json: True
request_json: {'messages': [{...}], 'conte…
(return) _ProxyLookup.__get__: functools.p…
```

Globals

```
    blob_file.seek(0)
    return await send_file(blob_file, mimetype=mime_type, as_at


@bp.route("/ask", methods=["POST"])
@authenticated
async def ask(auth_claims: Dict[str, Any]):
    if not request.is_json:
        return jsonify({"error": "request must be json"}), 415
    request_json = await request.get_json()
    context = request_json.get("context", {})
    context["auth_claims"] = auth_claims
    try:
        use_gpt4v = context.get("overrides", {}).get("use_gpt4v
        approach: Approach
        if use_gpt4v and CONFIG_ASK_VISION_APPROACH in current_
            approach = cast(Approach, current_app.config[CONFIG
        else:
```

CALL STACK

MainThread PAUSED ON STEP

```
ask                     .../backend/app.py        121:1
auth_handler            .../backend/decorators.py
dispatch_request        .../quart/app.py          1506:1
full_dispatch_request   .../quart/app.py
handle_request          .../quart/app.py          1376:1
handle_request          asgi.py                   98:1
_run                    events.py                 80:1
_run_once               base_events.py            1922:1
run_forever             base_events.py            607:1
run_until_complete      base_events.py            640:1
run                     .../quart/app.py          850:1
run_command             cli.py                    639:1
```

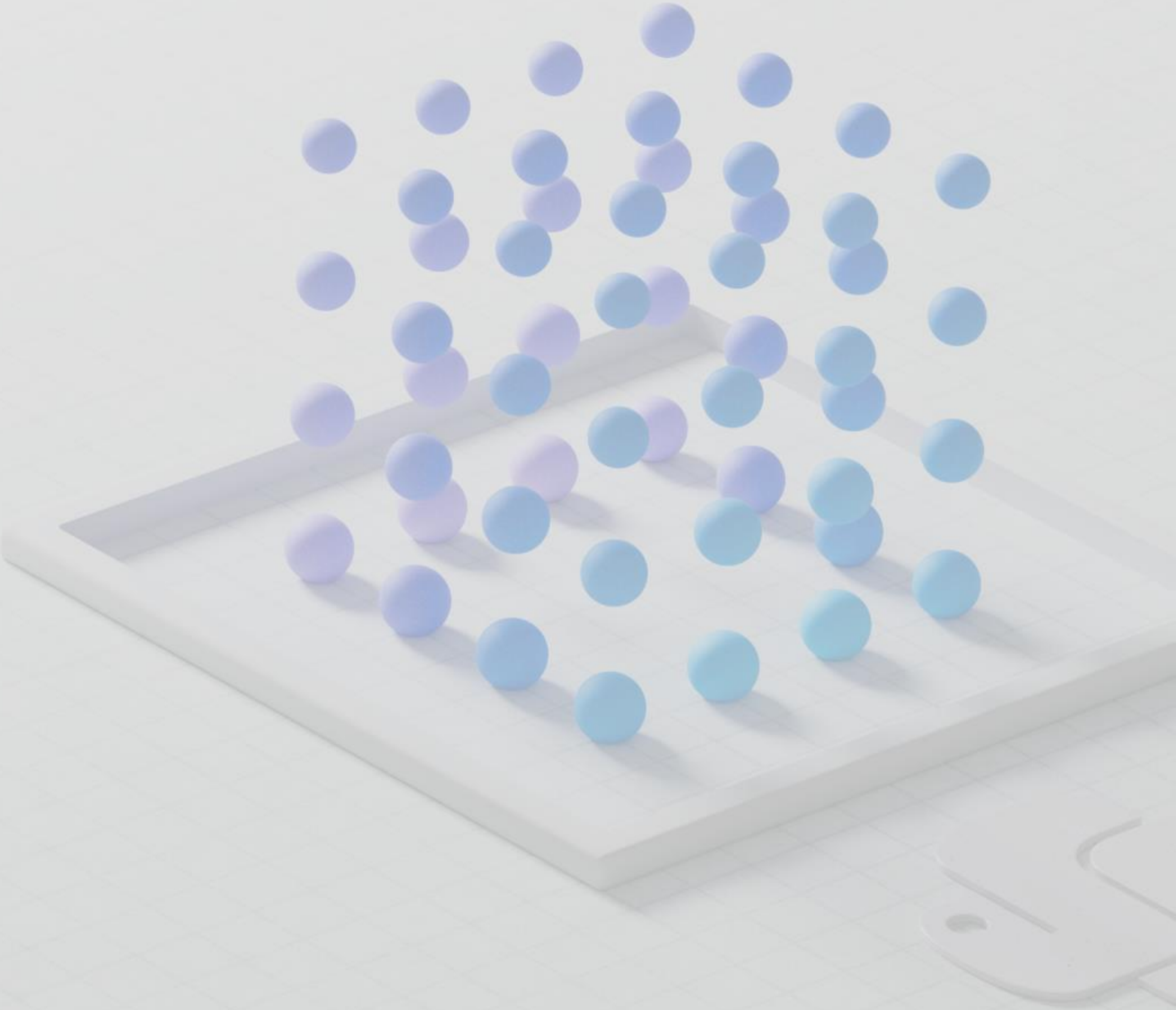
WATCH

TERMINAL

```
    'Date': 'Fri, 26 Jan 2024 18:31:50 GMT'
    'Content-Length': '336'
[2024-01-26 18:31:51 +0000] [35739] [INFO] Running on http://127.0.0.1:5050
INFO:hypercorn.error:Running on http://127.0.0.1:50505 (CTRL + C to quit)
```

Config in launch.json

Ingesting custom data



Removing the current data

In scripts/prepdocs.sh or scripts/prepdocs.ps1:

```
./scripts/prepdocs.py --removeall
```

You can also delete the index entirely in the Portal, if you need to add more fields.

🔗 Data ingestion guide: Removing documents aka.ms/ragchat/remove-data

Adding new data: PDFs

1. Add PDFs to data/ folder
2. Rerun ./scripts/prepdocs.ps1 or ./scripts/prepdocs.sh
3. Script will only process files that don't yet have a corresponding .md5 file

🔗 Data ingestion guide: Adding documents   aka.ms/ragchat/add-data

Microsoft

Adding new data: non-PDFs

Options:

- Convert data to PDFs (e.g. webpages to PDFs)
- Use Document Intelligence to convert
- Use integrated vectorization (cloud-based)
- Write your own parser / use Llamaindex

Microsoft

Converting webpages to PDF

1. Create a list of URLs that need indexing
2. Use Playwright to visit each page & save PDF

   🔗 Converting HTML pages to PDFs with Playwright   aka.ms/html2pdf

3. Add new PDFs as usual
4. Customize frontend to render webpage

```
export function getCitationFilePath(citation: string): string {
   return `https://playwright.dev/python/docs/${citation}`;
}
```

Writing a custom parser

Demo by Chris Ayers

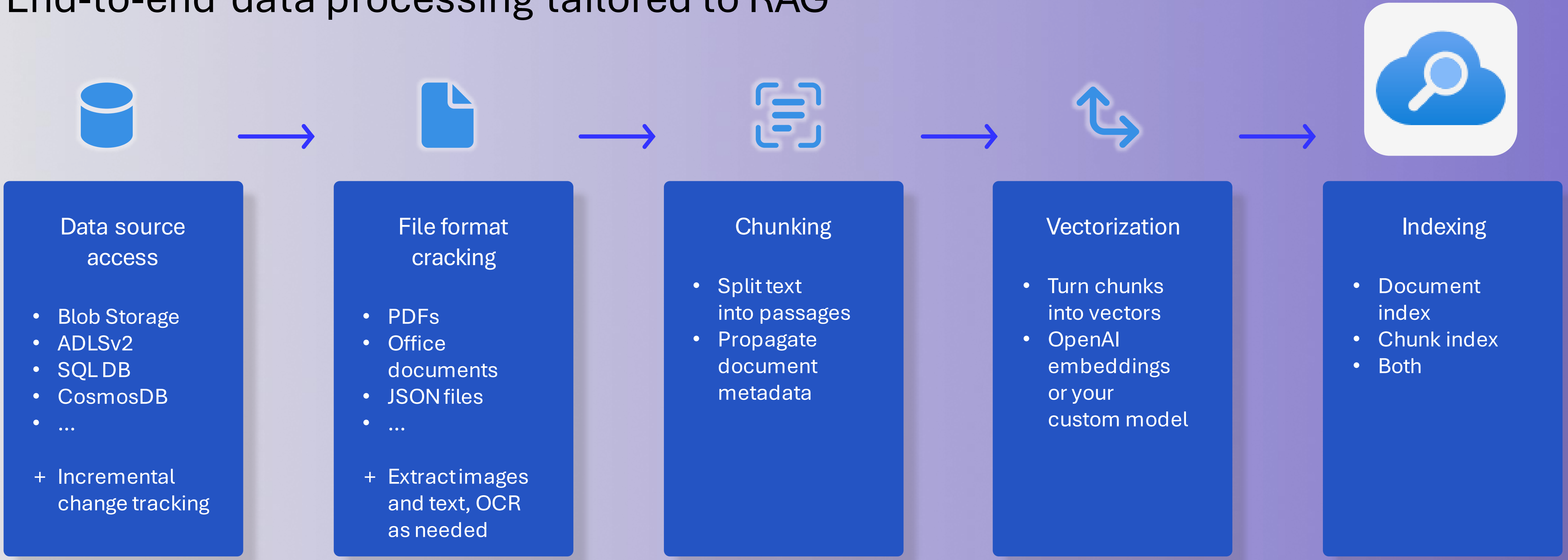
[PR: Adding a JSONParser class](aka.ms/ragchat/json) aka.ms/ragchat/json

Microsoft

Integrated vectorization

In preview

End-to-end data processing tailored to RAG



Data source access

- Blob Storage
- ADLSv2
- SQL DB
- CosmosDB
- …

+ Incremental change tracking

File format cracking

- PDFs
- Office documents
- JSON files
- …

+ Extract images and text, OCR as needed

Chunking

- Split text into passages
- Propagate document metadata

Vectorization

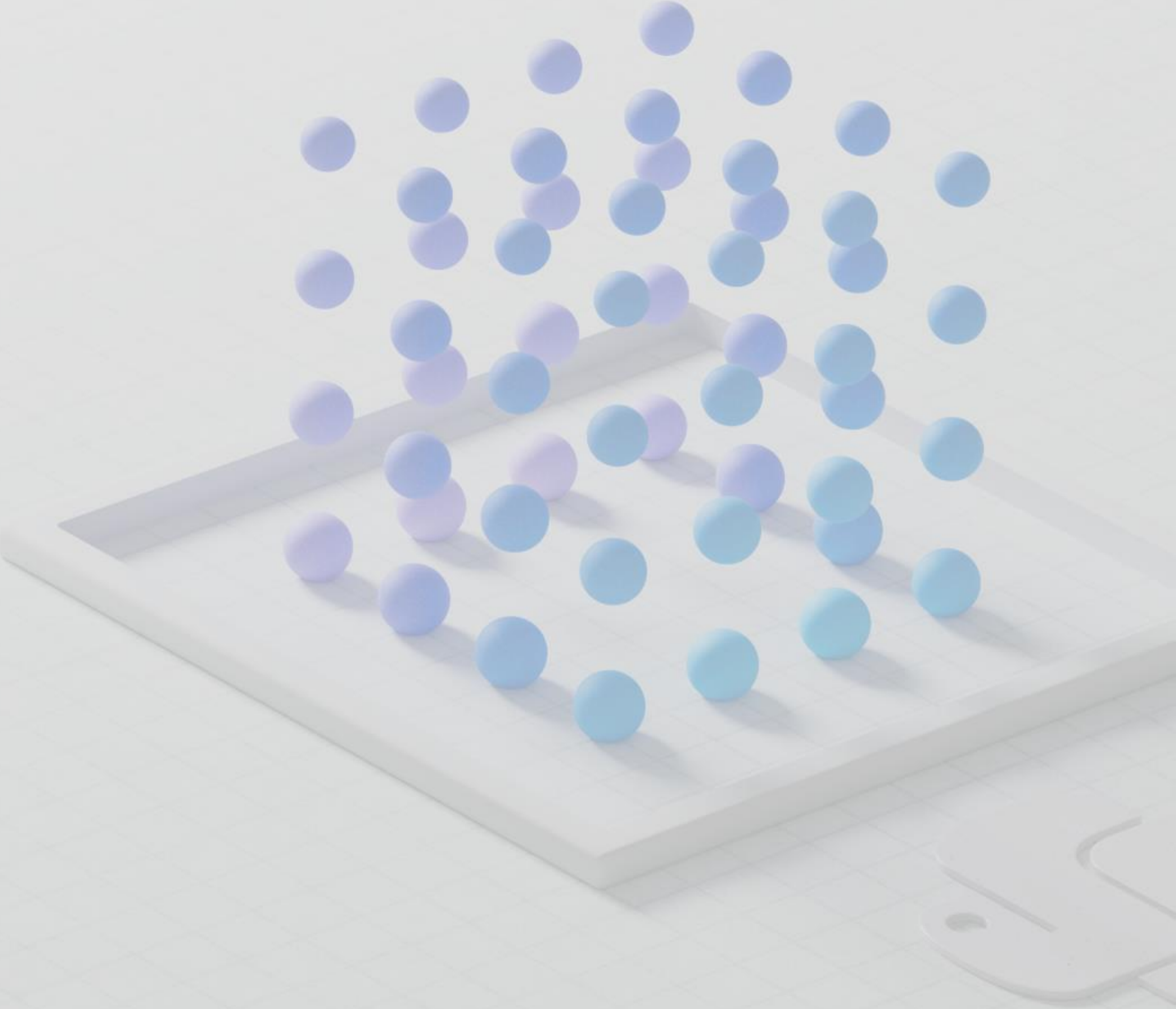
- Turn chunks into vectors
- OpenAI embeddings or your custom model

Indexing

- Document index
- Chunk index
- Both

Integrated data chunking and embedding in Azure AI Search aka.ms/integrated-vectorization

Adding new features



Don’t reinvent the wheel!

1. Check the pull requests
(doc upload UI, markdown support, speech, etc)
2. Check other RAG chat repos
(collaboration mode, conversation history, thumbs up, etc)
3. Ask in the discussion forum

But if you do come up with a new feature, please share! 🙏

Next steps

- Register for the hackathon → aka.ms/hacktogether/chatapp
- Introduce yourself in our discussion forum
- Deploy the repo with the sample data
  - See steps on low cost deployment → aka.ms/ragchat/free
- Start customizing the project!
- Post in forum if you have any issues deploying or questions about customization. 🙋🏼‍♀️ 🙋🏾‍♀️ 🙋🏻‍♀️ 🙋🏼‍♀️ 🙋🏿‍♂️
- Join tomorrow's session: Azure AI Search Best Practices