

# RAG DEEP DIVE















## Chat history

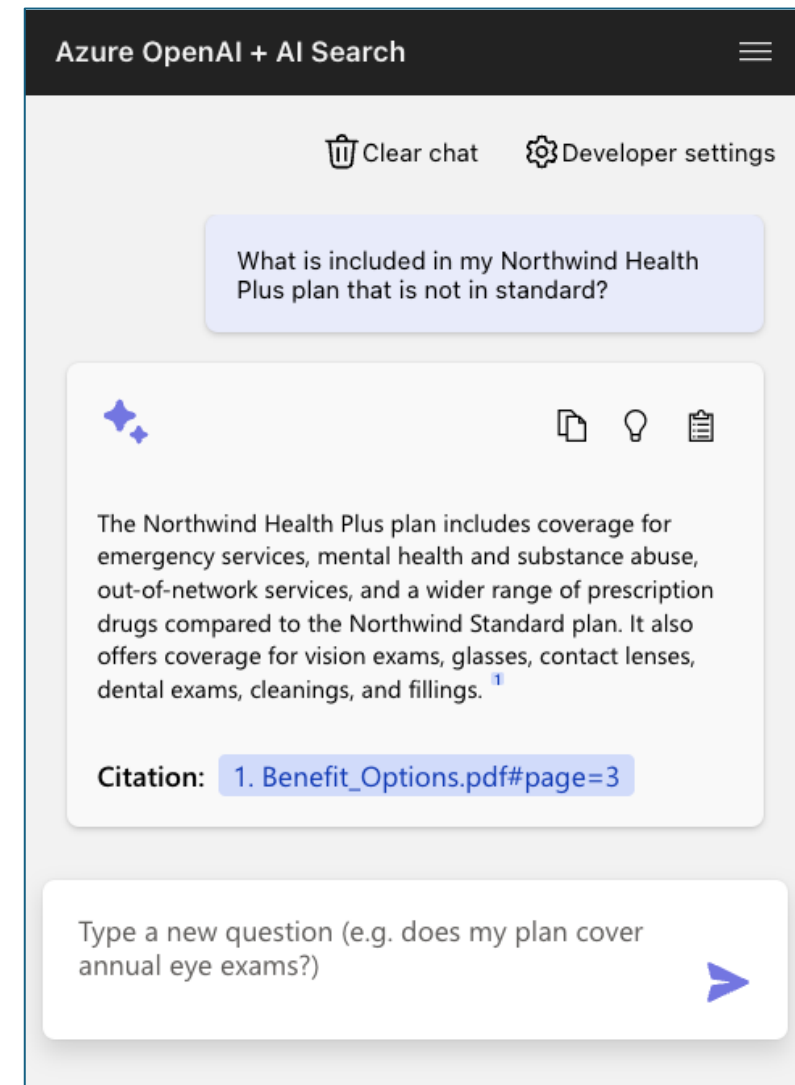
[aka.ms/ragdeepdive/chathistory/slides](https://aka.ms/ragdeepdive/chathistory/slides)

Pamela Fox

Python Cloud Advocate

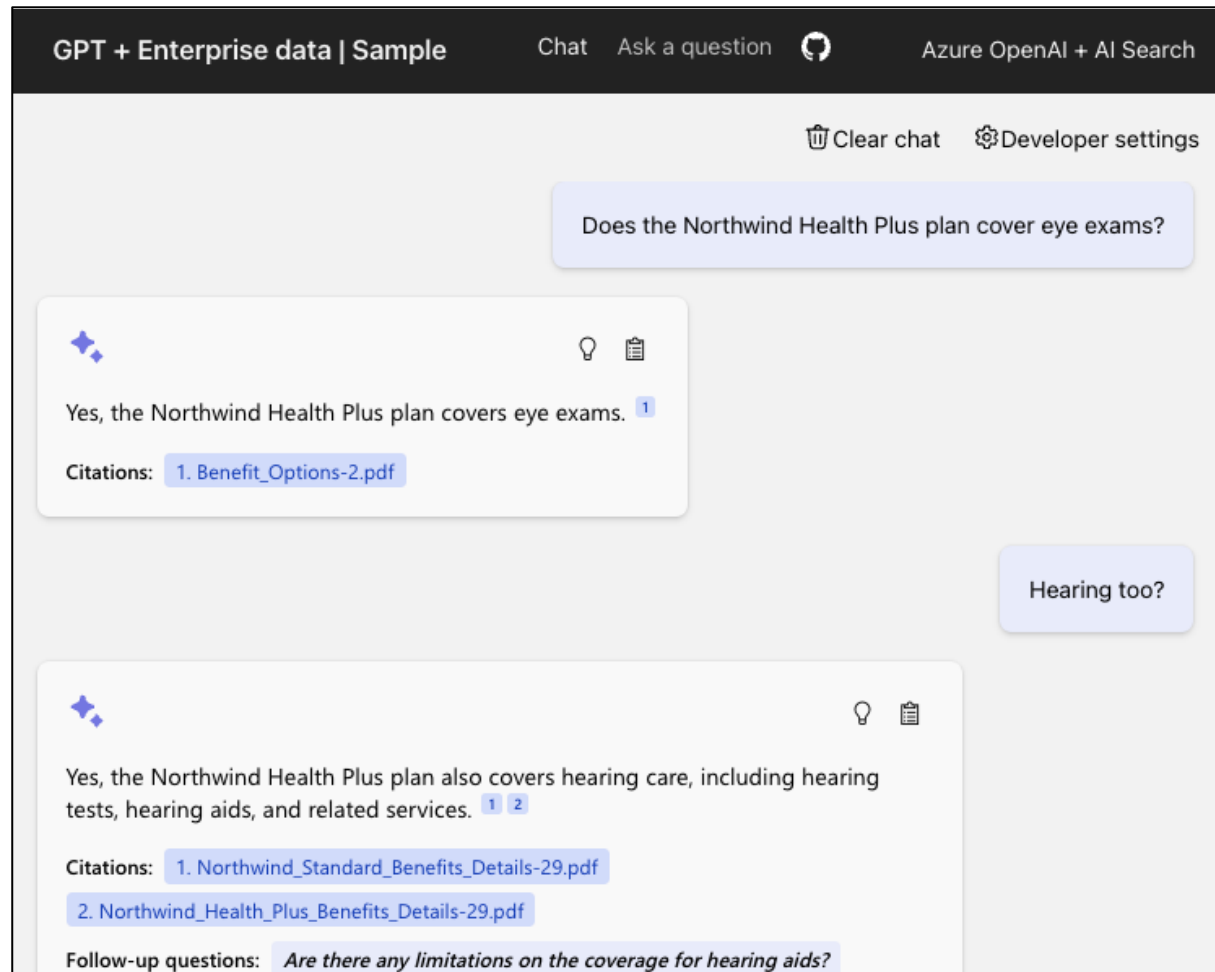
# RAG DEEP DIVE

-  1/13: The RAG solution for Azure
-  1/15: Customizing the RAG solution
-  1/21: Optimal retrieval with Azure AI Search
-  1/22: Multimedia data ingestion
-  1/27: User login and data access control
-   1/29: Storing chat history
-  2/3: Adding speech input and output
-  2/5: Private deployment
-  2/10: Evaluating RAG answer quality
-  2/12: Monitoring and tracing LLM calls
-  2/18: Extending RAG with function calling



Register for all the sessions @ [aka.ms/ragdeepdive/register](https://aka.ms/ragdeepdive/register)

# Our RAG chat solution



Azure OpenAI +  
Azure AI Search +  
Azure Container Apps / App Service

Features:

- Simple & Advanced RAG
- Conversations ("multi-turn")
- Optional vision integration
- Optional data access control

Code:

[aka.ms/ragchat](https://aka.ms/ragchat)

Demo:

[aka.ms/ragchat/demo](https://aka.ms/ragchat/demo)

# **Today we'll talk about how we handle:**

- Including message history in the RAG flow
- Conversation history in the browser
- Conversation history in Cosmos DB

**Including message history  
in the RAG flow**

# How do we handle message history?

The screenshot displays a chat application interface. At the top, a dark header bar contains the text "GPT + Enterprise data | Sample" on the left, and "Chat", "Ask a question" with a circular icon, and "Azure OpenAI + AI Search" on the right. Below the header, the chat area has a light gray background. In the top right of the chat area, there are links for "Clear chat" (with a trash icon) and "Developer settings" (with a gear icon). The chat history shows two messages from the user in light blue bubbles. The first message asks, "Does the Northwind Health Plus plan cover eye exams?". The AI response, in a white bubble, states, "Yes, the Northwind Health Plus plan covers eye exams." followed by a citation "1. Benefit\_Options-2.pdf". The second user message asks, "Hearing too?". The AI response states, "Yes, the Northwind Health Plus plan also covers hearing care, including hearing tests, hearing aids, and related services." followed by two citations: "1. Northwind\_Standard\_Benefits\_Details-29.pdf" and "2. Northwind\_Health\_Plus\_Benefits\_Details-29.pdf". At the bottom of the chat area, a "Follow-up questions:" section suggests the question, "Are there any limitations on the coverage for hearing aids?". Each AI response bubble includes a small blue star icon, a lightbulb icon, and a clipboard icon.

GPT + Enterprise data | Sample Chat Ask a question Azure OpenAI + AI Search

Clear chat Developer settings

Does the Northwind Health Plus plan cover eye exams?

Yes, the Northwind Health Plus plan covers eye exams. <sup>1</sup>

Citations: 1. Benefit\_Options-2.pdf

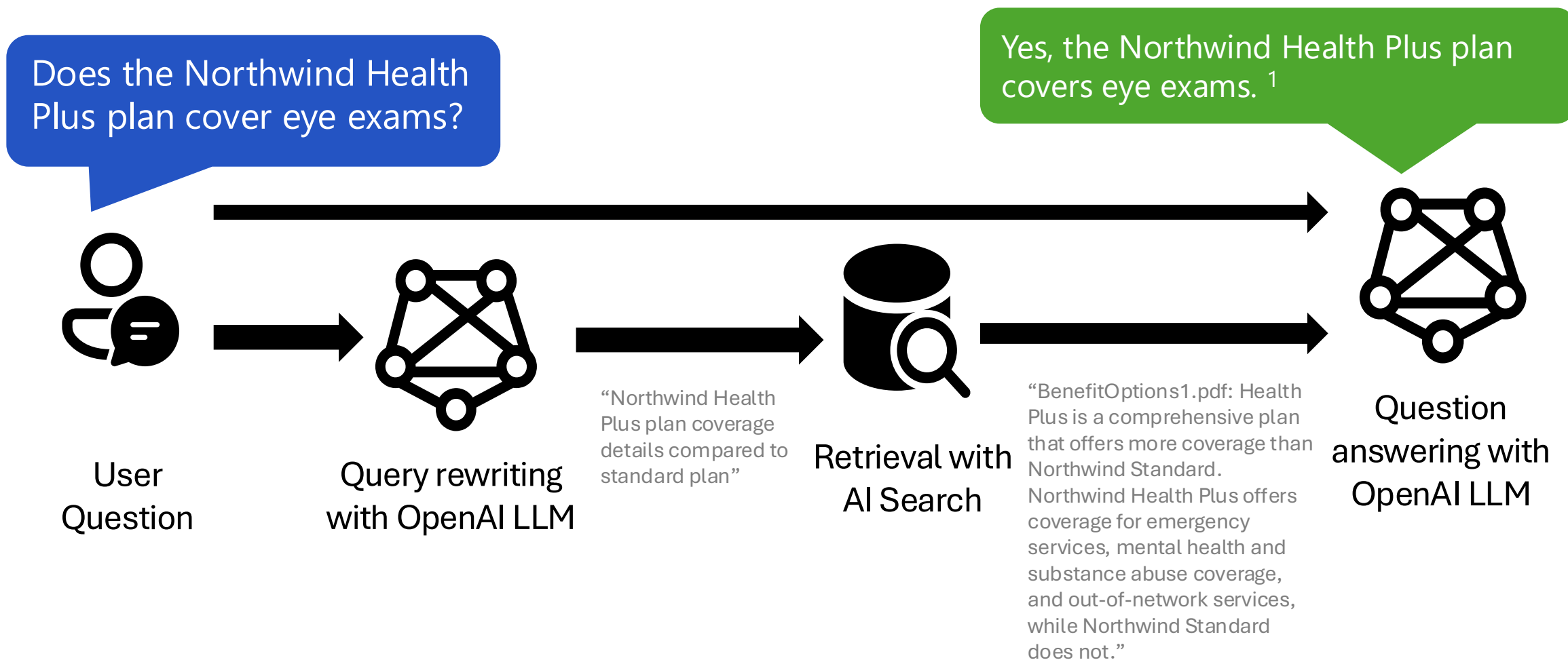
Hearing too?

Yes, the Northwind Health Plus plan also covers hearing care, including hearing tests, hearing aids, and related services. <sup>1</sup> <sup>2</sup>

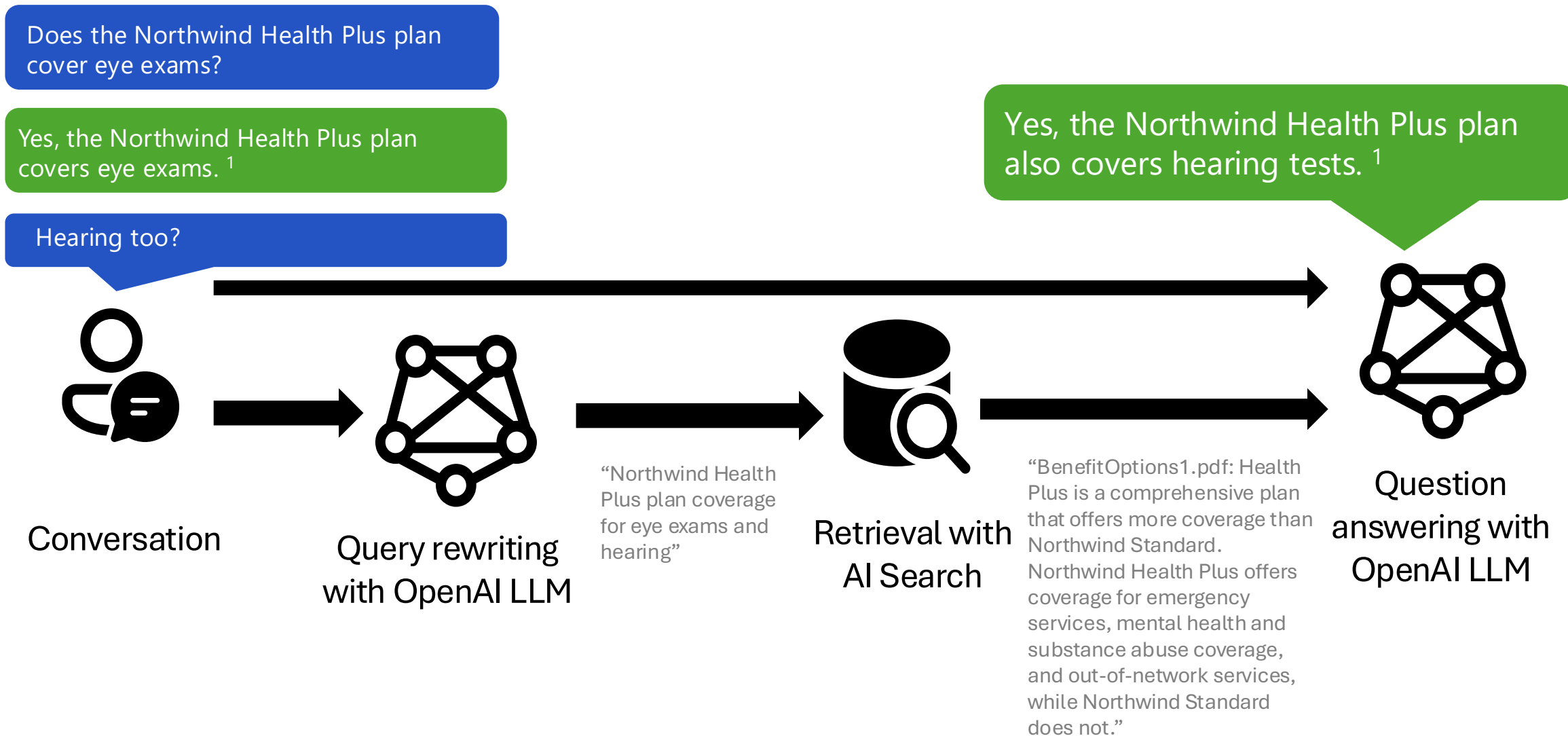
Citations: 1. Northwind\_Standard\_Benefits\_Details-29.pdf  
2. Northwind\_Health\_Plus\_Benefits\_Details-29.pdf

Follow-up questions: Are there any limitations on the coverage for hearing aids?

# RAG chat flow: Single message



# RAG chat flow: Multiple messages





# Truncating the message history

Why do we need to truncate message history?

LLMs have context window limits, between 4K and 128K tokens:

<https://learn.microsoft.com/azure/ai-services/openai/concepts/models>

Does the Northwind Health Plus plan cover eye exams?

Yes, the Northwind Health Plus plan covers eye exams. <sup>1</sup>

Hearing too?

✗ Discard oldest messages if they don't fit

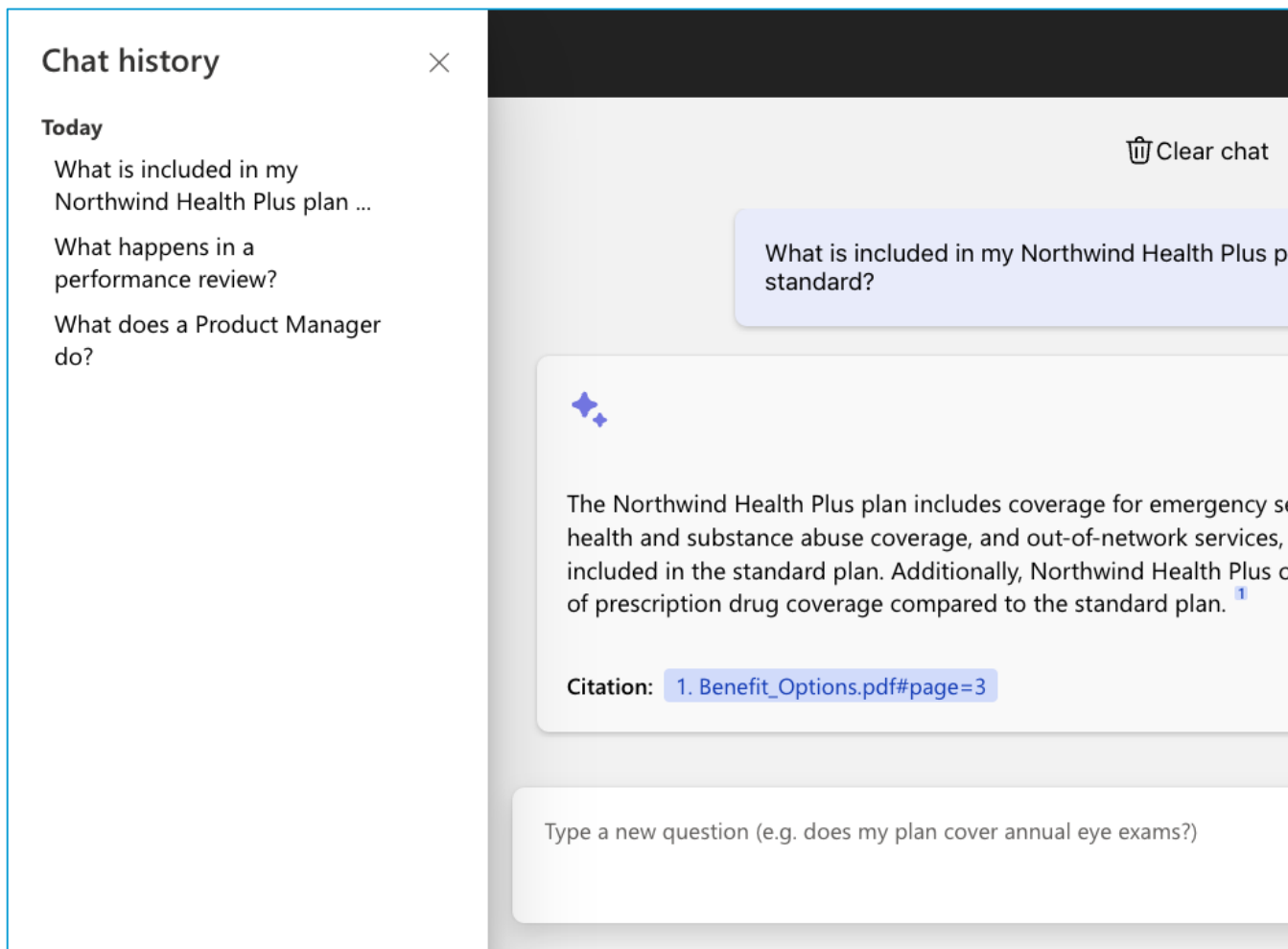
✓ Send previous message that fits in LLM context window

✓ Always send most recent message

<https://blog.pamelafox.org/2024/06/truncating-conversation-history-for.html>

# **Storing conversation history in the browser**

# RAG chat app with browser history



Azure OpenAI +  
Azure AI Search +  
**IndexedDB**

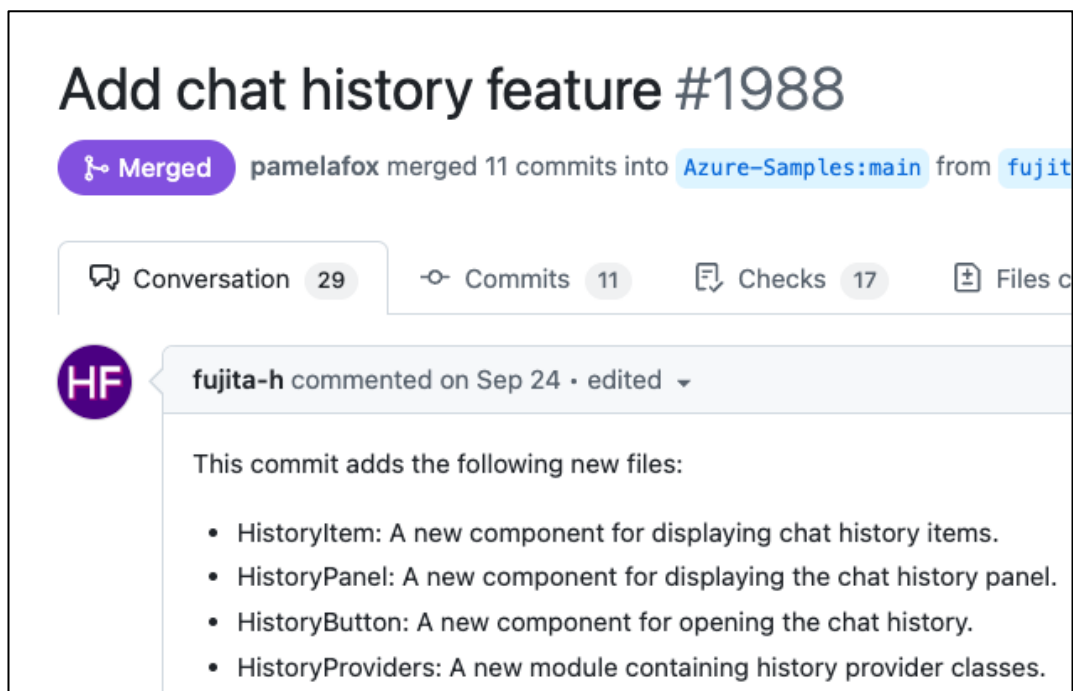
Code:  
[aka.ms/ragchat](https://aka.ms/ragchat)

Demo:  
[aka.ms/ragchat/demo](https://aka.ms/ragchat/demo)

Docs:  
[aka.ms/ragchat/browserhistory/docs](https://aka.ms/ragchat/browserhistory/docs)

# Thanks to @fujita-h for the feature!

@fujita-h designed the history to be extendible, to allow for multiple backends. ❤️



**Add chat history feature #1988**

Merged pamelafox merged 11 commits into Azure-Samples:main from fujita-h

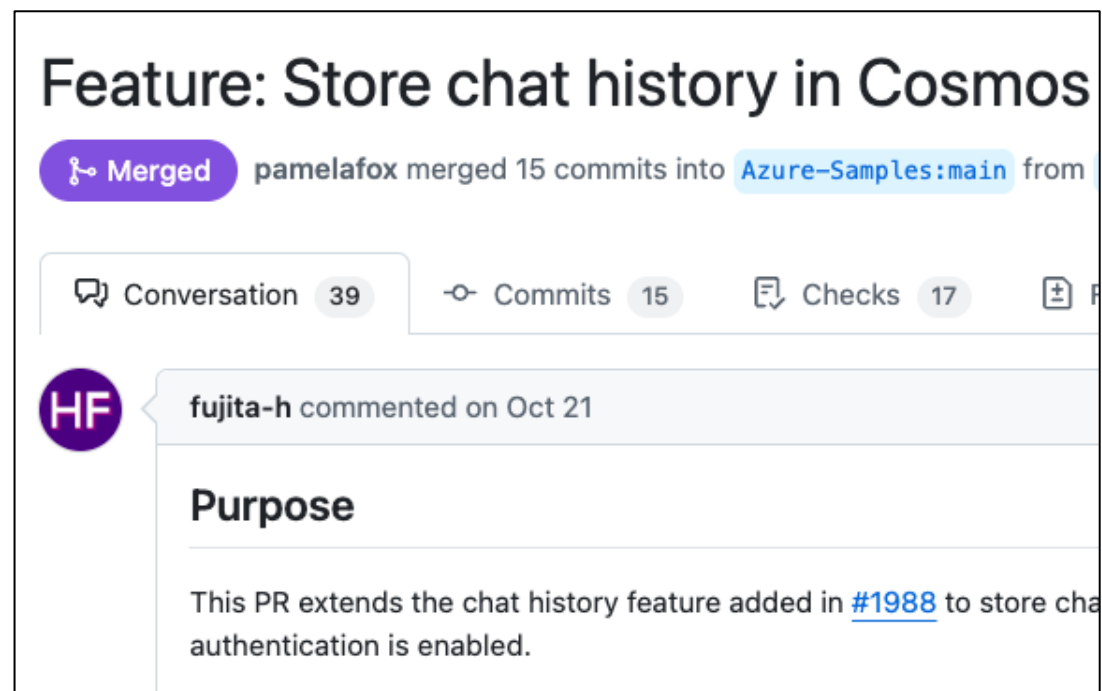
Conversation 29 Commits 11 Checks 17 Files changed 1

**HF** fujita-h commented on Sep 24 • edited

This commit adds the following new files:

- HistoryItem: A new component for displaying chat history items.
- HistoryPanel: A new component for displaying the chat history panel.
- HistoryButton: A new component for opening the chat history.
- HistoryProviders: A new module containing history provider classes.

[github.com/Azure-Samples/azure-search-openai-demo/pull/1988](https://github.com/Azure-Samples/azure-search-openai-demo/pull/1988)



**Feature: Store chat history in Cosmos**

Merged pamelafox merged 15 commits into Azure-Samples:main from fujita-h

Conversation 39 Commits 15 Checks 17 Files changed 1

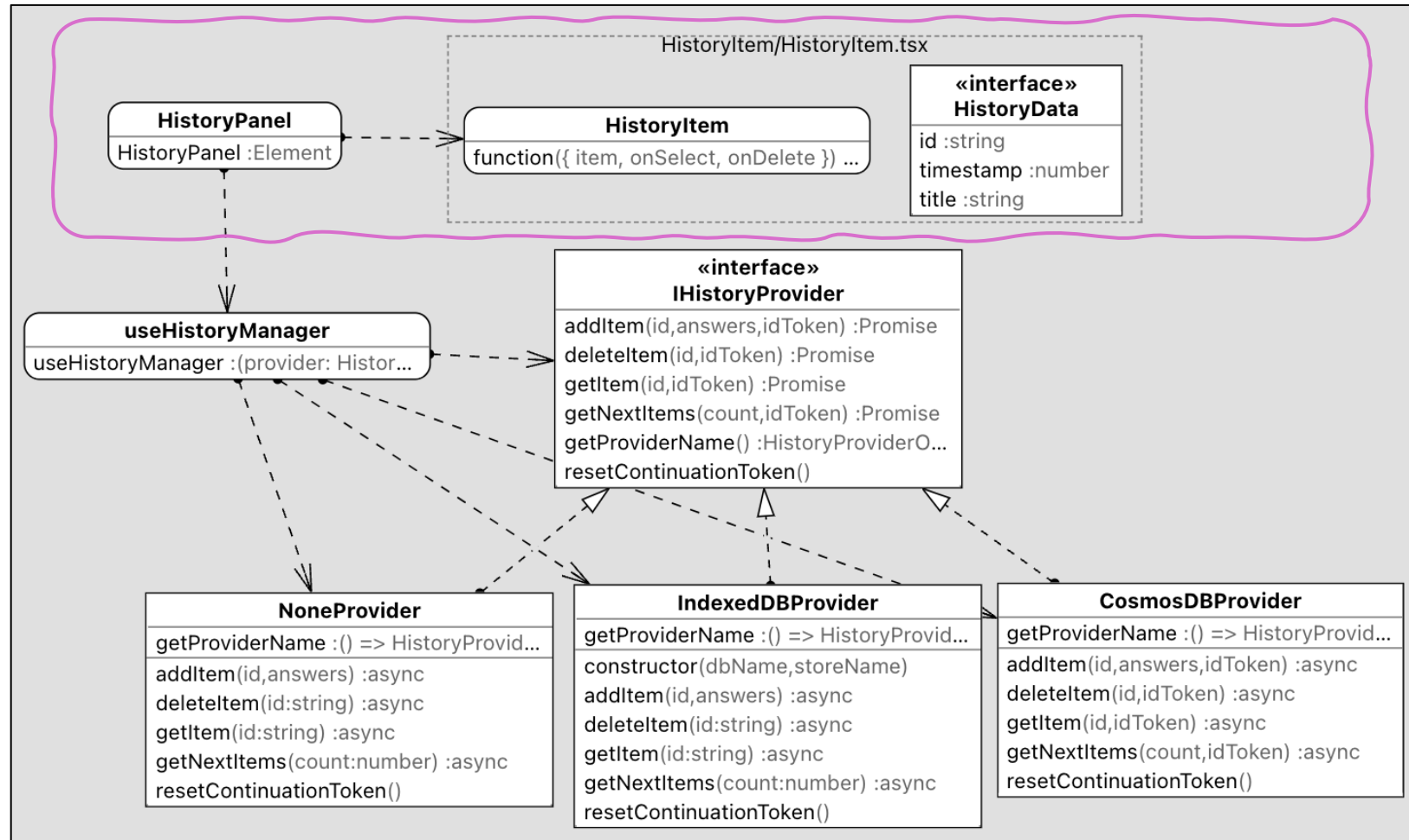
**HF** fujita-h commented on Oct 21

**Purpose**

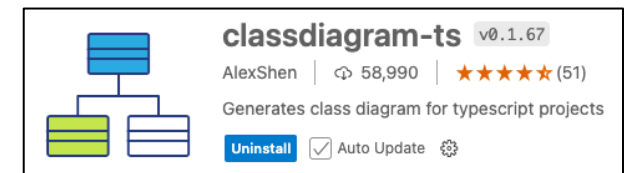
This PR extends the chat history feature added in [#1988](#) to store chat history in Cosmos DB when authentication is enabled.

[github.com/Azure-Samples/azure-search-openai-demo/pull/2063](https://github.com/Azure-Samples/azure-search-openai-demo/pull/2063)

# Frontend architecture



Visualized with:



classdiagram-ts VS Code extension

# Chat history with IndexedDB

IndexedDB is a client-side data storage mechanism usable on all modern browsers:

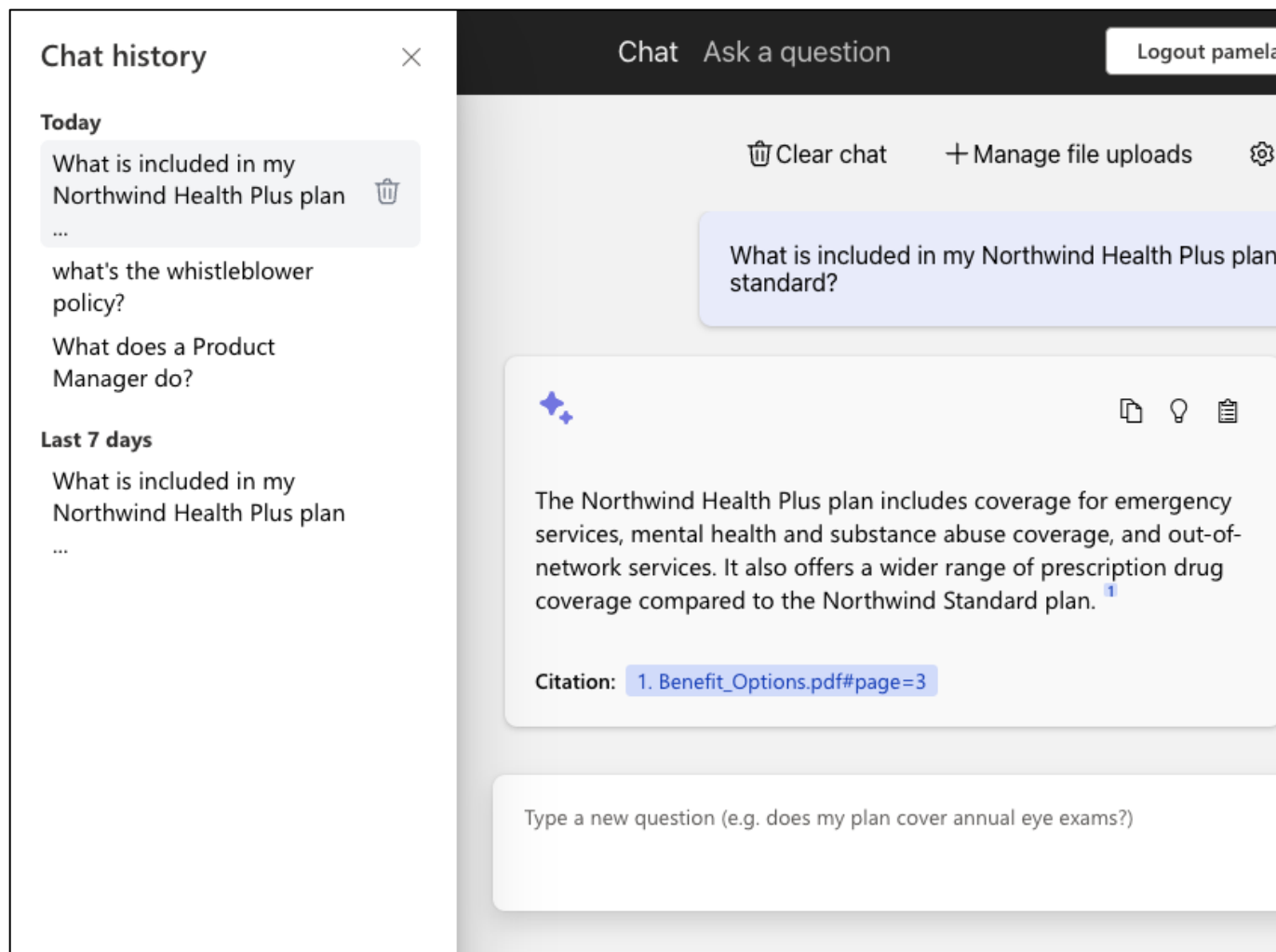
The screenshot displays a web application interface for a chat. On the left, a 'Chat history' sidebar lists previous questions: 'What does a Product Manager do?' and 'What happens in a performance review?'. The main chat area shows a text input field with a placeholder 'Type a new question (e.g. does my plan cover annual eye exams?)' and a blue arrow button. Below the chat area, the browser's developer tools are open, showing the 'Application' tab. The 'Storage' section is expanded, revealing 'IndexedDB' and a database named 'chat-database'. Within 'chat-database', the 'chat-history' object is selected, showing a table of chat history entries. The table has three columns: '#', 'Key (Key path: "id")', and 'Value'. Two entries are visible:

#	Key (Key path: "id")	Value
0	"01dd0b47-afa9-4a5c-8c9c-fbb8735dc658"	<pre>{id: '01dd0b47-afa9-4a5c-8c9c-fbb8735dc658', answers: [Array(2)], id: '01dd0b47-afa9-4a5c-8c9c-fbb8735dc658', timestamp: 1733186253846, title: 'What does a Product Manager d'}</pre>
1	"450434a5-a32d-4c84-aff5-5c2000bd4a54"	<pre>{id: '450434a5-a32d-4c84-aff5-5c2000bd4a54', answers: [Array(2)], id: '450434a5-a32d-4c84-aff5-5c2000bd4a54', timestamp: 1729897356278, title: 'What happens in a performance'}</pre>

But! If the user uses a new browser, their chat history will be gone. 🐱

# **Storing conversation history in Cosmos DB**

# RAG chat app with Cosmos DB history



Azure OpenAI +  
Azure AI Search +  
**Cosmos DB**

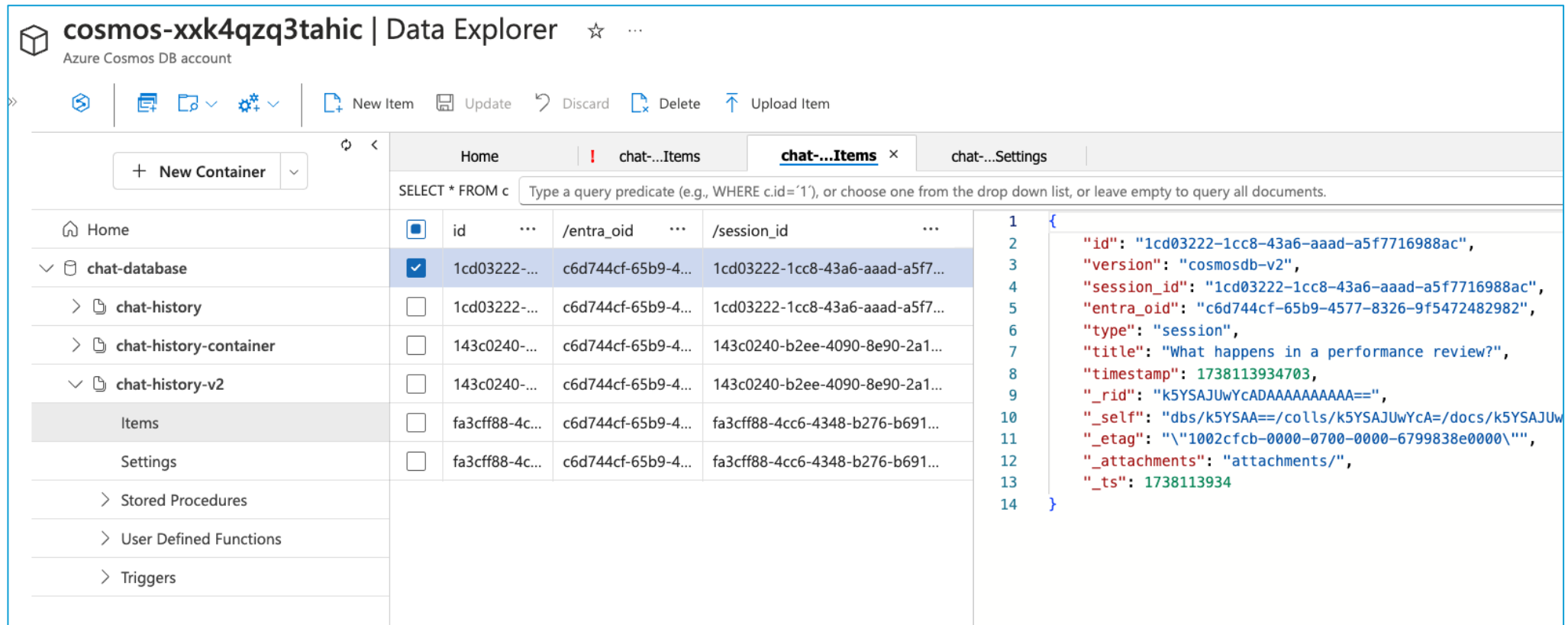
Code:  
[aka.ms/ragchat](https://aka.ms/ragchat)

Docs:  
[aka.ms/ragchat/cosmoshistory/docs](https://aka.ms/ragchat/cosmoshistory/docs)



# Chat history with Cosmos DB

Azure Cosmos DB is a highly scalable non-relational database.



The screenshot displays the Azure Cosmos DB Data Explorer interface for an account named 'cosmos-xxk4qzq3tahic'. The left sidebar shows a tree view with the following structure:

- Home
- chat-database
  - chat-history
  - chat-history-container
  - chat-history-v2
    - Items (selected)
    - Settings
  - Stored Procedures
  - User Defined Functions
  - Triggers

The main pane shows the 'chat-...Items' view with a table of documents. The table has columns: id, /entra\_oid, and /session\_id. The first document is selected:

id	/entra_oid	/session_id
1cd03222-...	c6d744cf-65b9-4...	1cd03222-1cc8-43a6-aaad-a5f7...
1cd03222-...	c6d744cf-65b9-4...	1cd03222-1cc8-43a6-aaad-a5f7...
143c0240-...	c6d744cf-65b9-4...	143c0240-b2ee-4090-8e90-2a1...
143c0240-...	c6d744cf-65b9-4...	143c0240-b2ee-4090-8e90-2a1...
fa3cff88-4c...	c6d744cf-65b9-4...	fa3cff88-4cc6-4348-b276-b691...
fa3cff88-4c...	c6d744cf-65b9-4...	fa3cff88-4cc6-4348-b276-b691...

The right pane shows the JSON document for the selected item:

```
1 {
2   "id": "1cd03222-1cc8-43a6-aaad-a5f7716988ac",
3   "version": "cosmosdb-v2",
4   "session_id": "1cd03222-1cc8-43a6-aaad-a5f7716988ac",
5   "entra_oid": "c6d744cf-65b9-4577-8326-9f5472482982",
6   "type": "session",
7   "title": "What happens in a performance review?",
8   "timestamp": 1738113934703,
9   "_rid": "k5YSAJUwYcADAAAAAAAAA==",
10  "_self": "dbs/k5YSAA==/colls/k5YSAJUwYcA=/docs/k5YSAJUwYcA=/documents/1cd03222-1cc8-43a6-aaad-a5f7716988ac",
11  "_etag": "\"1002cfc0-0000-0700-0000-6799838e0000\"",
12  "_attachments": "attachments/",
13  "_ts": 1738113934
14 }
```

<https://learn.microsoft.com/azure/cosmos-db/data-explorer>

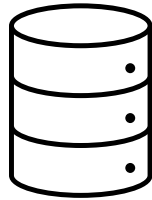
# Cosmos DB architecture

entra\_oid:123 / session\_id: abc

CosmosDB

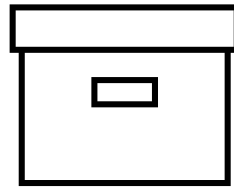


Database



chat-database

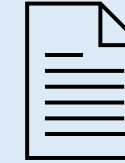
Container



chat-history



entra\_oid: "123"  
session\_id: "abc"  
type: "session"  
title: "whats a PM do?"  
timestamp: 17334282



entra\_oid: "123"  
session\_id: "abc"  
type: "message\_pair"  
question: "whats a PM do?"  
response: {content: "A PM..."}

entra\_oid: 456 / session\_id: def



entra\_oid: "456"  
session\_id: "def"  
type: "session"  
title: "company mission?"  
timestamp: 173332282

# Cosmos DB: Configuration with Bicep

```
containers: [{  
  name: 'chat-history'  
  paths: ['/entra_oid', '/session_id']  
  kind: 'MultiHash'  
  indexingPolicy: {  
    indexingMode: 'consistent'  
    automatic: true  
    includedPaths: [{  
      path: '/entra_oid/?'  
    }]  
    {  
      path: '/session_id/?'  
    }  
    {  
      path: '/timestamp/?'  
    }  
    {  
      path: '/type/?'  
    }  
  ]  
  excludedPaths: [{  
    path: '/*'  
  }]]}]
```

← Sets hierarchical partition keys

[learn.microsoft.com/azure/cosmos-db/partitioning-overview](https://learn.microsoft.com/azure/cosmos-db/partitioning-overview)

← Automatic indexing for four fields

[learn.microsoft.com/azure/cosmos-db/index-policy](https://learn.microsoft.com/azure/cosmos-db/index-policy)

# Cosmos DB: API routes

cosmosdb.py

POST

/chat\_history

```
batch_operations =  
    [("upsert", (session_item,))] +  
    [("upsert", (message_pair_item,))  
     for message_pair_item in message_pair_items  
    ]  
container.execute_item_batch(  
    batch_operations=batch_operations,  
    partition_key=[entra_oid, session_id])
```

GET

/chat\_history/sessions

```
container.query_items(  
    query="SELECT c.id, c.entra_oid, c.title,  
c.timestamp FROM c WHERE c.entra_oid = @entra_oid  
AND c.type = @type ORDER BY c.timestamp DESC",  
    parameters=[  
        dict(name="@entra_oid", value=entra_oid),  
        dict(name="@type", value="session")  
    ],  
    partition_key=[entra_oid],  
    max_item_count=count  
)
```

# Cosmos DB: API routes

cosmosdb.py

GET

/chat\_history/sessions/<session\_id>

```
res = container.query_items(  
    query="SELECT * FROM c WHERE c.session_id = @session_id  
AND c.type = @type",  
    parameters=[  
        dict(name="@session_id", value=session_id),  
        dict(name="@type", value="message_pair")],  
    partition_key=[entra_oid, session_id],  
)
```

DELETE

/chat\_history/sessions/<session\_id>

```
res = container.query_items(  
    query="SELECT * FROM c WHERE c.session_id = @id",  
    parameters=[dict(name="@id", value=session_id)],  
    partition_key=[entra_oid, session_id],  
)  
# ...(gather ids)...  
batch_operations = [  
    ("delete", (id,)) for id in ids_to_delete]  
container.execute_item_batch(  
    batch_operations=batch_operations,  
    partition_key=[entra_oid, session_id])
```

# Questions

# Chat history with access control?

What if we wanted to allow users to share a chat with another user?

Options:

1. Add a field for `allowed_entra_oids`, and query across partitions for that field.
  - 😬 Unbounded cross-partition queries are generally slow, so it'd be better to choose a different partition key or none at all.
2. Make `entra_oid` into an array.
  - 😊 We can use performant queries for finding chats.
  - 😬 That gives the other user 100% access to the chat

# Chat history with access control? Pt. 2

Store a new document type for each user that contains IDs of chats shared with them:

```
{
  "id": "00000000-0000-0000-0000-000000000000",
  "entra_oid": "aaaabbbb-0000-cccc-1111-dddd2222eeee",
  "shared_sessions": [{
    "session_id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2",
    "entra_oid": "bbbbcccc-1111-dddd-2222-eeee3333ffff"
  }],
  "type": "shared_session"
}
```

Thanks to Sidney Andrews for prototyping:












<https://gist.github.com/seesharprun/ace6b930706b4906fdb65b5d8e92124>



# Next steps

- Join upcoming streams! →
- Deploy the RAG chat app:  
[aka.ms/ragchat](https://aka.ms/ragchat)  
Post questions in the issue tracker or discussions
- Come to Pamela's Office Hours on Thursdays in Discord:  
[aka.ms/ragdeepdive/oh](https://aka.ms/ragdeepdive/oh)

## RAG DEEP DIVE

-  1/13: The RAG solution for Azure
-  1/15: Customizing the RAG solution
-  1/21: Optimal retrieval with Azure AI Search
-  1/22: Multimedia data ingestion
-  1/27: User login and data access control
-  1/29: Storing chat history
-  2/3: Adding speech input and output
-  2/5: Private deployment
-  2/10: Evaluating RAG answer quality
-  2/12: Monitoring and tracing LLM calls
-  2/18: Extending RAG with function calling

**Register for all the sessions @**

**[aka.ms/ragdeepdive/register](https://aka.ms/ragdeepdive/register)**