

Paul Borisov

# Gemini AI Chat Web Part, v1.0

Chat history, private and global chat sharing, consecutive event streaming, integration with Azure API Management, SharePoint list storage for chat history, options to use data encryption and unlimited history length, full-screen mode, integrations with Internet and company data, search on Bing and Google, analysis of PDFs and images, voice input and speech out

1-3-2024

## Table of Contents

Introduction .....	3
Deployment .....	3
Permissions .....	3
Configuration .....	4
Privacy settings for SharePoint List storage .....	9
Storage encryption .....	9
Privacy settings for generated images .....	9
User interface .....	10
Navigation panel .....	11
Content panel .....	12
Using Shared chats .....	12
Event streaming .....	13
Voice input .....	13
Speech out (Voice output) .....	13
Examples for prompt text .....	14
Image analysis (Gemini Pro Vision) .....	14
Backend .....	15
Azure API Management .....	15
Named values .....	15
APIs .....	16
GeminiAI .....	16
Bing (optional) .....	17
Google (optional) .....	17
API operations .....	17
API settings for GeminiAI .....	18
chat .....	19
chatstream .....	19
vision .....	20
visionstream .....	21
API settings for Bing (optional) .....	22
search .....	23
API settings for Google (optional) .....	24

search.....	25
Known issues.....	26
Security .....	26
Frontend.....	26

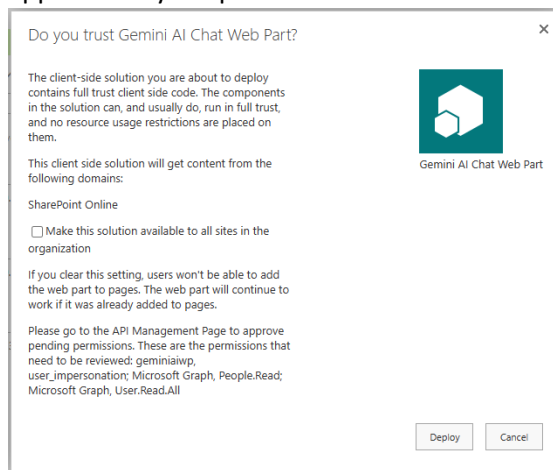
## Introduction

This document outlines the functionality of the Gemini AI Chat Web Part, its frontend and backend components, and configurations.

## Deployment

To deploy the web part, you should add its compiled solution, **gemini-ai-chat.sppkg**, into an App Catalog.

- This can be a global App Catalog of your SharePoint Online tenant, or a site collection scoped one.
- Use standard deployment options to make the web part globally available on all sites or add the app manually to specific sites.



## Permissions

After adding the package, open the URL [https://yourtenant-admin.sharepoint.com/\\_layouts/15/online/AdminHome.aspx#/webApiPermissionManagement](https://yourtenant-admin.sharepoint.com/_layouts/15/online/AdminHome.aspx#/webApiPermissionManagement) and approve the requested permissions.

To enable this permission, create a standard Azure App registration named **gemini.aiwp** in Microsoft Entra ID (Azure AD). The web part uses this App registration to make authenticated requests using the context of the signed-in user.

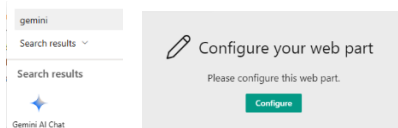
This is used in the Chat sharing option. It is utilized to get details of all users from AAD to display them in the People Picker component.

An alternative (reduced) permission for the Chat sharing option is available. If access level User.Read.All cannot be granted, the web part can use this reduced scope to get only the basic information of the user's colleagues. In this case, the list of available contacts may be reduced compared to the more advanced previous option. If neither of the last two permissions can be granted, the Chat sharing feature can only provide unrestricted sharing.

Permission	Scope	Purpose
<b>geminiaiwp</b> , required	user_impersonation	To enable this permission, create a standard Azure App registration named <b>geminiaiwp</b> in Microsoft Entra ID (Azure AD). The web part uses this App registration to make authenticated requests using the context of the signed-in user.
Microsoft Graph, optional	User.Read.All	This is used in the Chat sharing option. It is utilized to get details of all users from AAD to display them in the People Picker component.
Microsoft Graph, optional	People.Read	<p>An alternative (reduced) permission for the Chat sharing option is available. If access level <b>User.Read.All</b> cannot be granted, the web part can use this reduced scope to get only the basic information of the user's colleagues. In this case, the list of available contacts may be reduced compared to the more advanced previous option.</p> <p>If neither of the last two permissions can be granted, the Chat sharing feature can only provide unrestricted sharing.</p>

## Configuration

Create a modern Site Page, add the web part "Gemini AI Chat", and click on the "Configure" button.



This action opens the web part settings with the default values shown below. You should use "Create" buttons available in the web part settings (with default empty or custom values in textboxes) to deploy:

- SharePoint list, which is used to store chat history
- Image library used, which is used to store uploaded images

## Gemini AI Chat



### Settings

Client ID: create a user\_impersonation app with name=geminiaiwp

00000000-0000-0000-0000-000000000000

Base URL for Gemini AI (APIM API or full)

https://generativelanguage.googleapis.com/v1beta

Optional api-key for Gemini AI (for troubleshooting, not for Production)

.....

Storage type for chat history

SharePoint list

SharePoint list URL (leave it empty for default URL)

Create

- ☒ Enable storage encryption
- ☒ Enable sharing
- ☒ Enable streaming
- ☒ Enable full screen mode
- ☒ Enable integrations
- ☒ Bing search

Optional api-key for Bing

Add if APIM endpoint is not configured

- ☒ Google search

Optional key for Google:

key=API\_KEY&cx=SEARCH\_ENGINE\_ID

Add if APIM endpoint is not configured

Image library URL (leave it empty for default URL)

A library to store the uploaded images

Create

- ☒ Enable examples for the prompt text
- ☒ Enable voice input
- ☒ Enable voice output (text to speech)
- ☒ Code highlighting
- ☒ Show highlighting styles


Default style

vs2015

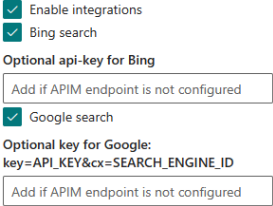
- ☒ Show prompt area at bottom
- ☒ Unlimited chat history length (AI-responses in long chats may be less accurate)

Locale for dates (default is fi-FI)

Title	Description	Default
Client ID: create a user_impersonation app with name=gemini aiwp	The App ID (Client ID) for the <b>gemini aiwp</b> Azure registration is used to authorize SPFx requests.	00000000-0000-0000-0000-000000000000
Base URL for Gemini AI (APIM API or full)	<p>This is the base endpoint for Gemini AI language models.</p> <p>You should either deploy the endpoint via the Azure API Management service (recommended for enhanced flexibility and security) or use the full URL to the Gemini AI endpoint <a href="https://generativelanguage.googleapis.com/v1beta">https://generativelanguage.googleapis.com/v1beta</a> with the API key added to its own setting below.</p> <ul style="list-style-type: none"> <li>v1beta provides the optional Function Calling feature of Gemini Pro while v1 does not include it yet.</li> </ul>	<p>Use the default value <a href="https://generativelanguage.googleapis.com/v1beta">https://generativelanguage.googleapis.com/v1beta</a> with an API key specified in the following field below</p> <p>OR configure your APIM endpoint <a href="https://instance.azure-api.net/gemini ai">https://instance.azure-api.net/gemini ai</a></p> <p>This refers to the GeminiAI API published via the Azure API Management service with 4 main operations configured under it:</p> <ul style="list-style-type: none"> <li>/chat</li> <li>/chatstream</li> <li>/vision</li> <li>/visionstream</li> </ul> <p>You can find details for this APIM configuration in the <a href="#">Backend</a> chapter of this document.</p>
Optional API key for Gemini AI	<p>Please note that if you use your API key here, it poses a security risk as the key could potentially be leaked, because it will become visible in the browser's request headers.</p> <p>The key is encrypted and stored in the web part settings (displayed as ***).</p>	
Storage type for chat history	<p>The only predefined option is "SharePoint list". This setting is reserved for probable future extensions.</p> <p>Chat history is stored in a SharePoint custom list.</p>	<p>SharePoint list</p> <p>The option for SharePoint list supports chat privacy. For example, SharePoint search does not display private chat conversations to users other than the authors of those</p>

	<p>Textbox and button to create the list are available. By default, the list is created on the current site with the predefined name, dbChatsGemini. Security settings are automatically enabled for list items to limit access to their authors. The SharePoint Search Crawler does not index the list content by default to prevent potential leaks of private conversations.</p>	<p>chats. If users open the Custom list <b>dbChatsGemini</b>, they will only see their own messages. However, this does not apply to Site owners, who can see all records in the list and can find them in SharePoint search results.</p> <p>Please note if you enable the sharing feature, you should adjust the list permissions using the Update button.</p> <p>In case of using the SharePoint list storage, the maximum length of Chat history text is limited to 2 MB per Chat entry. This is the technical limitation of the multiline text field in SharePoint Online.</p>
Enable storage encryption	<p>Enables AES encryption of the sensitive personal data in the SharePoint list.</p> <p>Encryption transparently supports Shared chats of both private and global types.</p> <p>Using encryption is a reasonable choice when you use SharePoint list storage with the sharing option enabled and want to hide details of personal Chats from other users of SharePoint.</p>	<p>Not selected</p> <p>Sensitive personal data includes Chat name and Chat messages. When the storage encryption is enabled, the display name of the current user is not saved (set to empty).</p>
Enable sharing	<p>Enables a sharing option for personal Chats. This option is disabled by default.</p> <p>Chat authors can decide if they want to share their personal Chats with others.</p>	<p>Selected</p> <p>The web part supports two levels of sharing:</p> <ol style="list-style-type: none"> <li>1. Global sharing: each personal Chat can be shared with Everyone.</li> <li>2. Private sharing: each personal Chat can be shared with up to 15 specific users selected in the dialog.</li> </ol>
Enable streaming	<p>Enables the event-streaming response option, which provides gradual outputs of texts generated by Gemini AI. This option is disabled by default.</p>	<p>Selected</p>
Enable full screen mode	<p>Enables full-screen mode and displays the expansion icon  in the upper right-hand corner.</p>	<p>Selected</p>
Enable integrations	<p>Enables integrations with external data using the "Function calling" feature of</p>	<p>Not selected</p>



	<p>Gemini AI. There are several functions available by default in the web part:</p> <ul style="list-style-type: none"> <li>• searchSharepoint</li> <li>• peopleSearch</li> <li>• currentDateAndTime</li> </ul> <p>Also, there are optional functions configured separately if corresponding services are available:</p> <ul style="list-style-type: none"> <li>• searchOnInternet: this function uses Azure Bing Search service with API key managed by APIM or configured explicitly in the web part settings (encrypted).</li> <li>• searchOnGoogle: this function uses Google Custom Search with key=...&amp; cx=... values managed by APIM or configured explicitly in the web part settings (encrypted).</li> </ul>	 <p>Configuration instructions for optional Bing and Google APIM endpoints are described in the chapter <b>Azure API Management</b> below.</p>
Enable examples for the prompt text	Enabled the dropdown box visible in the left-hand side of the prompt area. It opens the list of available examples for the prompt text that include integration samples (when enabled).	Not selected
Enable voice input	If the browser supports Web Speech API and the microphone is available, this option provides voice input for the prompt text.	There are 15 popular world languages available by default.
Enable voice output (text to speech)	If the browser supports Web Speech API, this option provides reading out of AI-generated texts on click of the Radio-icon.	There are 15 popular world languages to read out texts in available by default.
Code highlighting	Enables code highlighting to render markdown outputs of code snippets.	Selected
Show highlighting styles	Enables the display of a dropdown with code styles for highlighted outputs. This feature is disabled by default.	Selected
Default style	Default code style for highlighted outputs.	vs2015
Show prompt area at bottom	When enabled, the prompt text area will appear at the bottom. It is disabled by default.	Not selected
Unlimited chat history length (AI-responses in long chats may be less accurate)	When enabled, this feature permits the use of unlimited history length in chats. In such cases, the system dynamically removes earlier messages just prior to submitting the history to Gemini AI. This process does not	Not selected

	impact the storage of the entire (uncut) history and new responses.	
Locale for dates (default is fi-FI)	The locale for formatting dates visible in the web part is optional. The default setting is 'fi-FI' (empty), but it can be changed, for instance, to 'en-US'.	Default (empty textbox)

### Privacy settings for SharePoint List storage

The web part uses a SharePoint list to store chat histories. Default settings set to the list when it is provisioned via the web part properties pane (Create button) ensure that the list does not expose data to the Search Crawler. By default, the list creation button applies the following changes to list settings:

- Permissions for this list: Unique (permission inheritance is broken)
  - Everyone except external users: Contribute
  - List's creator: Full Control
- Advanced settings: Read and write permissions are only for item authors
  - Read access: Read items that were created by the user.
 

**Important:** If you wish to use shared chats, you will need to adjust access to less strict value “Read access: Read all items”. In this context, users may gain access to others' messages. This is a technical limitation of SharePoint storage.

    - To address this issue, you can enable storage encryption.
    - To adjust permissions automatically, click on the Update button for the SharePoint list in the web part settings.
  - Create and Edit access: Create items and edit items that were created by the user
- Allow items from this list to appear in search results? No
  - Note: You can modify this setting if necessary. In this case list items should be accessible in search results for their authors and site admins. However, they should not be available to regular users unless you use Read access: Read all items as mentioned above.

### Storage encryption

When you enable storage encryption, the maximum chat name length is limited to 150 chars. Unencrypted storage supports up to 255 characters.

### Privacy settings for generated images

The web part uses a document library to store uploaded images processed by Gemini AI.

- These images are not private.
- They are stored in the document library available for Everyone by default.

When you turn on the option “Enable integrations”, a textbox and button to create the image library become available.

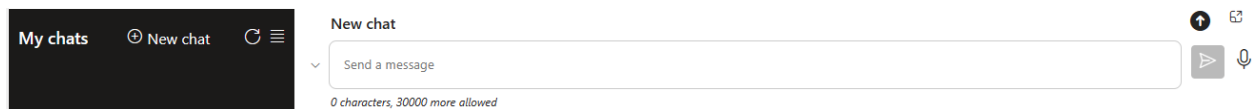
- By default, the library is created on the current site with the predefined name “**ChatImages**”.

The library obtains the following default settings:

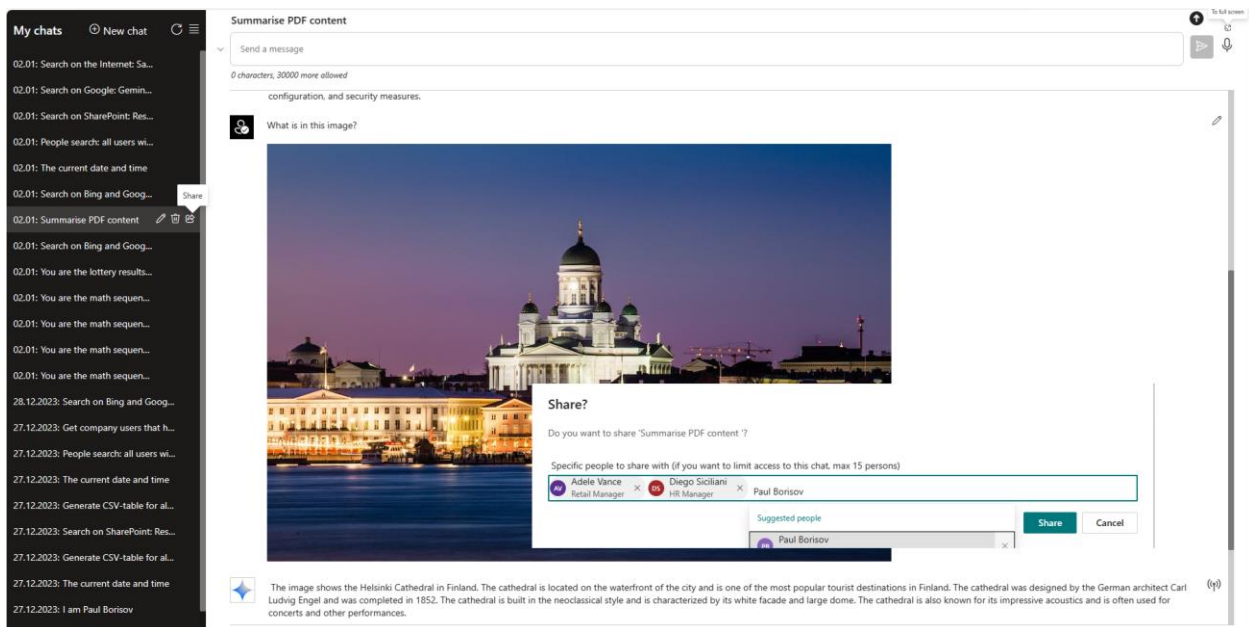
- Permissions for this library: Unique (permission inheritance is broken)
  - Everyone except external users: Contribute.
    - You can create your own Custom permission level, for example, Contribute without Delete.
  - Library's creator: Full Control
- Allow items from this library to appear in search results? No
  - You can modify this setting if necessary.

## User interface

This is the initial UI without chat history.



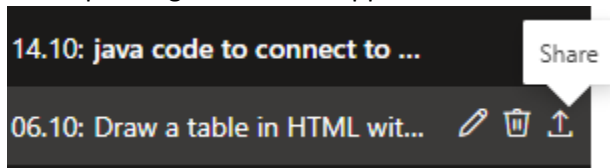
An example of UI displaying the previous chat history of the current user.



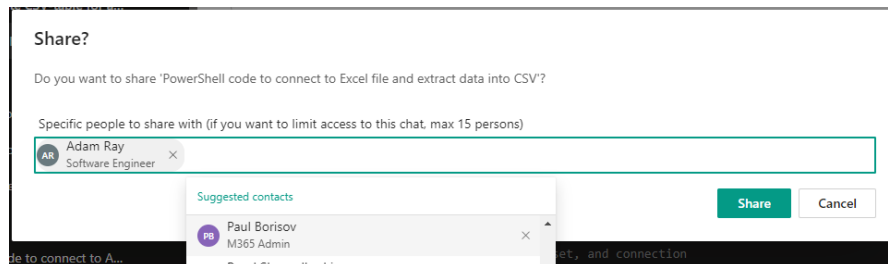
## Navigation panel

The left-hand panel displays chats started by the current user and sorted by the dates of the last modifications.

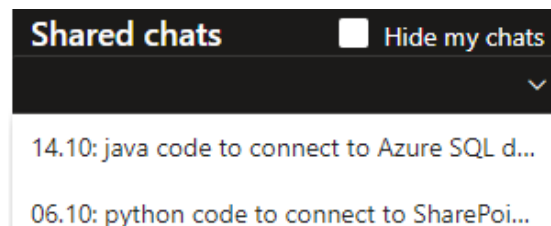
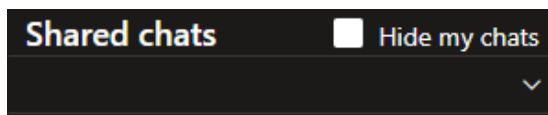
- When the user starts a new chat, the logic automatically generates a chat name using the entered texts.
- The default date format corresponds to Finnish: dd.MM for the current year and dd.MM.yyyy for previous years. This can be changed in the web part settings, for example, to en-US.
- The user can edit the chat name, delete the chat, and share it with others (if the sharing option has been enabled in the settings). The actions of deleting and sharing require the user's confirmation.
- Corresponding action icons appear after the chat name when the user selects the chat's row.



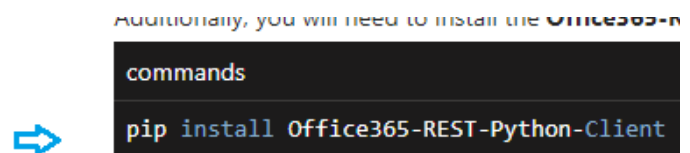
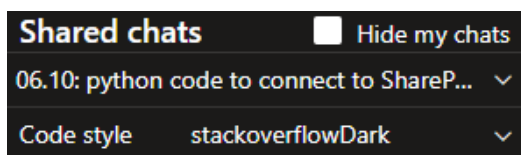
- The sharing dialog offers two options: “Share with Everyone” (which is the default) and “Share with specific users.” In the latter case, only the selected users will have access to the chats shared with them.



- Shared chats accessible to the current user are displayed at the bottom of the Navigation Panel. Users can select the “Hide My Chats” checkbox if they wish to hide the chats they have shared.





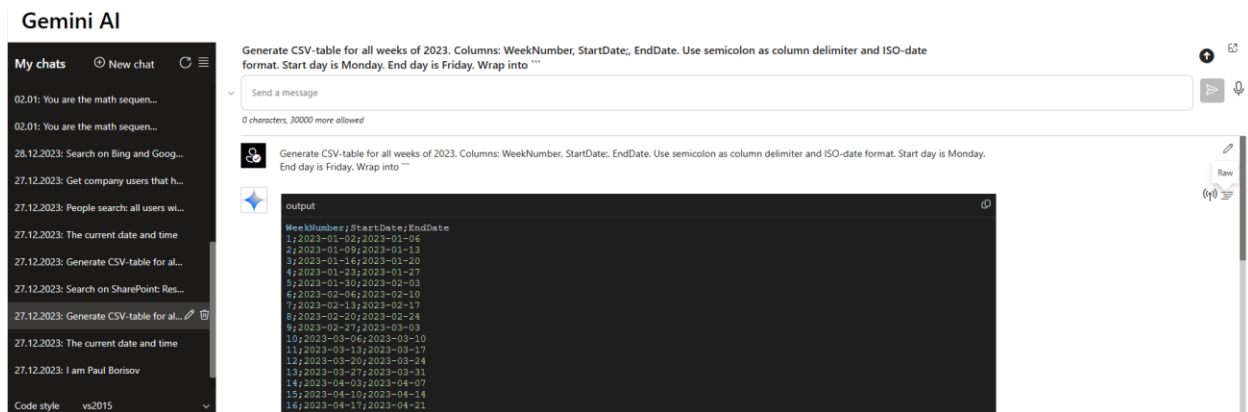
- The web part supports code highlighting options, which are accessible in the settings. If “Show Highlighting Styles” is enabled, the corresponding selector will appear at the bottom.



## Content panel

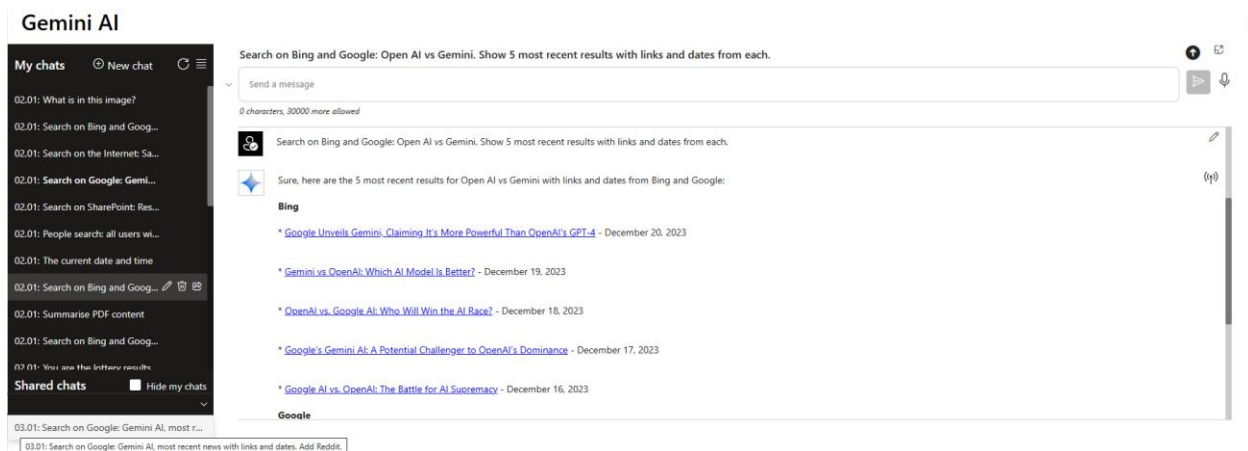
This area displays the message history for the chat selected in the Navigation Panel. By default, the most recently modified chat by the user is selected.

- The upper section of the panel displays the chat's name and, if more than one model is available in the settings, the language models. An icon to switch to Full Screen mode is also present, providing convenience for longer chats.
- A prompt area is designated for entering questions, followed by a character counter. The maximum request length is restricted to 30,000 characters.
- The maximum acceptable history length for a single chat should not exceed 64k characters. If this length is exceeded, the user should initiate a new chat to continue the conversation.
  - There is an option in the web part settings for “Unlimited Chat History Length (AI-responses in long chats may be less accurate)”. When this feature is enabled, it allows for an unlimited history length. In such cases, the system automatically removes the earliest messages before submitting the history to Gemini AI. However, this does not impact on the saving of the entire (uncut) history and new responses back into storage.
- Chat messages are composed of user queries followed by AI responses. If a user makes a typo, they can edit the message, correct it, and save the changes. Subsequent queries will use this updated history.
- If AI responses include code snippets, these are displayed in a highlighted format. The user can:
  - Click on the 'Raw' button (  ) to view the original, unformatted response.
  - Click on the 'Copy' button (  ) to copy the code snippet to the clipboard.



## Using Shared chats

When users select a Shared Chat at the bottom of the Navigation Panel, they initiate a new chat, and a copy of the chat's history is loaded into the Content Panel. Subsequently, users can continue the original conversation within their own copies of the Shared Chat. For security reasons, it is not permitted to continue original conversations on behalf of their authors.



## Event streaming

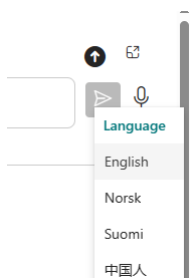
If the streaming option is enabled in the web part settings, the AI responses should be presented with the visual effects of consecutive text outputs, with the entire response being formatted at the end of the process.

If the streaming option is not enabled, the complete, formatted AI response will be provided at the end of the response processing.

## Voice input

This option uses Web Speech API if it is supported by the browser. There are 15 popular world languages available by default. Voice input was tested for all of them.

To use the voice input, click on the MIC-icon near the submit button, select the desired language, and start speaking into your microphone. Click on the stop button to input your speech into the prompt box and then click on the Submit button as usual.



## Speech out (Voice output)

Most modern browsers support Web Speech API. It provides a straightforward way to use speech synthesis for AI-generated texts.

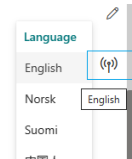
- Click on the Radio-icon in the right-hand part of the content area.
- The standard Web Speech API requires selecting the preferred language; using default page language is not always optimal. There is a dropdown menu to select the language to read out texts. The read out starts upon the selection.

Search on Bing and Google: Open AI vs Gemini. Show 5 most recent results with links and dates from each.

Sure, here are the 5 most recent results for Open AI vs Gemini with links and dates from Bing and Google:

Bing

\* [Google Unveils Gemini, Claiming It's More Powerful Than OpenAI's GPT-4](#) - December 20, 2023



## Examples for prompt text

Click on the dropdown menu and select an example. It will be inserted into the prompt area.

Search on Bing and Google: Open AI vs Gemini. Show 5 most recent results with links and dates from each.

Examples

- Search on the Internet: Sam Altman's ouster with links and dates.
- Search on Bing: Microsoft Copilot, most recent news with links and dates.
- Search on Google: Gemini AI, most recent news with links and dates. Add Reddit.
- Search on Bing and Google: Open AI vs Gemini. Show 5 most recent results with links and dates from each.
- Search on SharePoint: Resource Management System. Format the results as an HTML table.
- People search: all users with the name John. Roles and emails.
- Get company users that have names starting with K. Format the results as an HTML table.
- The current date and time
- Generate CSV-table for all weeks of 2024. Columns: WeekNumber, StartDate, EndDate. Use semicolon as column delimiter and ISO-date format. Start day is Monday. End day is Friday. Wrap into ""
- Powershell code to connect to Azure SQL database

## Image analysis (Gemini Pro Vision)

Select an image or images using the upward arrow button located in the right-hand corner.

- After selecting the images, the prompt texts "What is in these images" is added automatically.
- Click on the Submit button to make their analysis.

What is in this image?

Send a message

0 characters, 30000 more allowed

Upload files

What is in this image?

SharePoint Online web part with regular and Full-Screen appearance. Optional internet and data connections, voice input and speech; out, streaming, highlighting, PDF and images, prompt examples, and more.

Key Vault

Request enrichment with an API key after successful authorization.

API Management Services

REST API requests  
Texts only, the API key is not exposed in the browser (the more secure setup)

Default models:  
Gemini Pro with Function Calling  
Gemini Pro Vision for Image Analysis

Gemini AI

Request validation and authorization

Azure AD (Microsoft Entra ID)

External Azure AD Trusted tenant 1

Trusted tenant n

SharePoint List (Optimal, OOTB)

This is an architecture diagram that shows how to use Azure Key Vault to protect API keys in a SharePoint Online web part. The diagram shows how the web part makes a request to the API Management service, which in turn makes a request to Azure Key Vault to retrieve the API key. The API key is then used to make a request to the SharePoint Online API.

## Backend

### Azure API Management

It is recommended to use the Azure API Management Service (APIM) to ensure secure access to Gemini AI API. Key benefits of the APIM include:

- The ability to hide the details of user request authentication and authorization.
- Eliminating the necessity to provide the API key to the client app, as the API key does not travel with the browser's requests. Instead, the APIM adds the API key to the request after verifying the client's identity.
- Facilitating seamless updates of endpoints when newer versions of language models become available. Please follow the instructions and examples provided below to configure the APIM endpoints.

Follow the instructions and examples below to configure APIM endpoints.

### Named values

Use the blade API Management service > APIs > Named values

Name	Value
AadId	.....
AadId-kavoetokkiisa	.....
AadId-Sandbox	.....
ApiKeyBingSearch	.....
ApiKeyGeminiAI	.....
ApiKeyGoogleSearch	.....

This is a vault, which stores secure keys inside an API Management service instance.

- You can just add a new value as Secret.
- Optionally, it supports remote access to Azure Key Vault for enhanced security.

You should add the following keys:

- **AadId**: your Azure AD tenant GUID.
  - If you plan to grant access to APIM endpoints for users from multiple Azure AD domains, you can also add the GUIDs of other AAD tenants like AadId-gt, Aad-kavoerokkiisa, etc.
- **ApiKeyGeminiAI**: stores the **API key** for Gemini AI service.
  - You can generate an API key using Google AI Studio  
<https://makersuite.google.com/app/apikey>
  - If you are in EU or UK, Google AI Studio may be unavailable for your location.
  - In this case you should use a workaround: start any VPN service and connect to a server in a US-zone (Japan, India, etc.). After that you should be able to access Google AI



Studio.

## APIs

Use the blade API Management service > APIs > APIs

Create three sets of APIs with settings shown in the table below.

### *GeminiAI*

Display name: GeminiAI

Name: geminiai

Description: Gemini AI Service

Web service URL: <https://generativelanguage.googleapis.com/v1beta>

URL scheme: HTTPS

API URL suffix: geminiai

- Base URL: <https://instance.azure-api.net/geminiai>
- Replace **instance** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

[Design](#) [Settings](#) [Test](#) [Revisions \(1\)](#) [Change log](#)

### General

• Display name

GeminiAI

• Name

geminiai

Description ⓘ

Web service URL

<https://generativelanguage.googleapis.com/v1beta>

URL scheme

☐ HTTP ☒ HTTPS ☐ HTTP(S)

API URL suffix

geminiai

Base URL

<https://instance.azure-api.net/geminiai>

Tags

[e.g. Booking](#)

Products ⓘ

No products selected

ⓘ To publish the API, you must associate it with a product. [Learn more.](#)

Gateways

Managed x

### Subscription

Subscription required

☐

Header name

Ocp-Apim-Subscription-Key

Query parameter name

subscription-key

### *Bing (optional)*

*Note you must have Azure Bing Search service deployed to use this endpoint.*

Display name: Bing

Name: bing

Description: Azure Bing Search service

Web service URL: <https://api.bing.microsoft.com/v7.0>

URL scheme: HTTPS

API URL suffix: bing

- Base URL: <https://instance.azure-api.net/bing>
- Replace **instance** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

### *Google (optional)*

*Note you must have Google Custom Search service available in your Google APIs subscription to use this endpoint.*

Display name: Google

Name: google

Description: Google Custom Search service

Web service URL: <https://www.googleapis.com/customsearch/v1>

URL scheme: HTTPS

API URL suffix: google

- Base URL: <https://instance.azure-api.net/google>
- Replace **instance** with your instance of API Management service.

Subscription required: **No** (clear default checkbox, it is important!)

Security: User authorization, None (default)

### *API operations*

Subsets of API operations should be created under each of your API.

- Clients will refer to them as <https://<baseURL>/<operationurl>>
- For example, like <https://instance.azure-api.net/geminiapi/chat> or <https://instance.azure-api.net/geminiapi/vision>

### API settings for GeminiAI

At first, configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at [Known issues](#).
- AadId and ApiKeyGeminiAI are tokens for values stored in [Named values](#).

A sample working config is given below.

```
<policies>
  <inbound>
    <base />
    <cors allow-credentials="false">
      <allowed-origins>
        <origin>*</origin>
      </allowed-origins>
      <allowed-methods preflight-result-max-age="300">
        <method>GET</method>
        <method>POST</method>
        <method>OPTIONS</method>
      </allowed-methods>
      <allowed-headers>
        <header>Access-Control-Allow-Origin</header>
        <header>Authorization</header>
        <header>Content-Type</header>
        <header>Origin</header>
      </allowed-headers>
    </cors>
    <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
      <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
      <issuers>
        <issuer>https://sts.windows.net/{{AadId}}</issuer>
        <!--Optional authorized external domains (end with backslash)-->
        <issuer>https://sts.windows.net/{{AadId-gt}}</issuer>
        <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}</issuer>
      </issuers>
    </validate-jwt>
    <!--Removal of the authorization header is important to avoid conflicts-->
    <set-header name="Authorization" exists-action="delete" />
    <set-header name="X-Goog-Api-Key" exists-action="override">
      <value>{{ApiKeyGeminiAI}}</value>
    </set-header>
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
```

```
</outbound>
<on-error>
  <base />
</on-error>
</policies>
```

After that, you need to add and configure the operations for your Gemini AI API for Pro and Pro Vision models.

chat

Display name: chat

Name: chat (it can add any descriptive value, not important)

URL: POST /chat

## Frontend

* Display name	<input type="text" value="chat"/>
* Name	<input type="text" value="chat"/>
* URL	<input type="text" value="POST"/> <input type="text" value="/chat"/>

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Gemini AI.
  - Clients send POST requests with payload to the APIM's URL <https://instance.azure-api.net/geminiai/chat>
  - APIM transforms the URL and forwards the payload to Gemini Pro endpoint.

```
<policies>
```

```
  <inbound>
    <base />
    <rewrite-uri template="/models/gemini-pro:generateContent" />
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

chatstream

Display name: chatstream

Name: chat (it can add any descriptive value, not important)

URL: POST /chatstream

## Frontend

* Display name	<input type="text" value="chatstream"/>
* Name	<input type="text" value="chatstream"/>
* URL	<input type="text" value="POST"/> <input type="text" value="/chatstream"/>

Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Gemini AI.
  - Clients send POST requests with payload to the APIM's URL `https://instance.azure-api.net/geminiai/chatstream`
  - APIM transforms the URL and forwards the payload to Gemini Pro endpoint, which supports consecutive event streaming.

```
<policies>
  <inbound>
    <base />
    <rewrite-uri template="/models/gemini-pro:streamGenerateContent?alt=sse"
  />
  </inbound>
  <backend>
    <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

vision

Display name: vision

Name: vision (it can add any descriptive value, not important)

URL: POST /vision

## Frontend

* Display name	<input type="text" value="vision"/>
* Name	<input type="text" value="vision"/>
* URL	<input type="text" value="POST"/> <input type="text" value="/vision"/>

### Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Gemini AI.
  - Clients send POST requests with payload to the APIM's URL <https://instance.azure-api.net/geminiai/vision>
  - APIM transforms the URL and forwards the payload to Gemini Pro Vision endpoint.

```
<policies>
  <inbound>
    <base />
    <rewrite-uri template="/models/gemini-pro-vision:generateContent" />
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

visionstream

Display name: visionstream

Name: vision (it can add any descriptive value, not important)

URL: POST /visionstream

## Frontend

* Display name	<input type="text" value="visionstream"/>
* Name	<input type="text" value="visionstream"/>
* URL	<input type="text" value="POST"/> <input type="text" value="/visionstream"/>

### Configuration:

- Add a rewrite URL like shown below. It should correspond to backend URL in Gemini AI.

- Clients send POST requests with payload to the APIM's URL <https://instance.azure-api.net/geminiai/visionstream>
- APIM transforms the URL and forwards the payload to Gemini Pro Vision endpoint, which supports consecutive event streaming.

```
<policies>
  <inbound>
    <base />
    <rewrite-uri template="/models/gemini-pro-
vision:streamGenerateContent?alt=sse" />
  </inbound>
  <backend>
    <forward-request timeout="180" fail-on-error-status-code="true" buffer-
response="false" />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

#### *API settings for Bing (optional)*

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at [Known issues](#).
- AadId and ApiKeyBingSearch are tokens for values stored in [Named values](#).
  - ApiKeyBingSearch stores a Key for your Azure Bing Search Service

A sample working config is given below.

```
<policies>
  <inbound>
    <base />
    <cors allow-credentials="false">
      <allowed-origins>
        <origin>*</origin>
      </allowed-origins>
      <allowed-methods preflight-result-max-age="300">
        <method>GET</method>
      </allowed-methods>
      <allowed-headers>
        <header>Access-Control-Allow-Origin</header>
        <header>Authorization</header>
        <header>Content-Type</header>
        <header>Origin</header>
      </allowed-headers>
    </cors>
  </inbound>
</policies>
```

```

        <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
            <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
            <issuers>
                <issuer>https://sts.windows.net/{{AadId}}/</issuer>
                <!--Optional authorized external domains (end with backslash)-->
                <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
                <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
            </issuers>
        </validate-jwt>
        <set-header name="Ocp-Apim-Subscription-Key" exists-action="override">
            <value>{{ApiKeyBingSearch}}</value>
        </set-header>
    </inbound>
</backend>
    <base />
</backend>
<outbound>
    <base />
</outbound>
<on-error>
    <base />
</on-error>
</policies>

```

After that, you need to add and configure the search operation for your Bing endpoint:

search

Display name: search

Name: search (it can add any descriptive value, not important)

URL: GET /search

## Frontend

* Display name	<input type="text" value="search"/>
* Name	<input type="text" value="search"/>
* URL	<input type="text" value="GET"/> <input type="text" value="/search"/>
Description ⓘ	<input type="text"/>
Tags	<input type="text" value="e.g. Booking"/>

Configuration:



- Add a rewrite URL like shown below.
  - Clients send GET requests with query string parameters to the APIM's URL  
https://**instance**.azure-api.net/bing/search?q=QUERY\_TEXT&mkt=LOCALE
  - APIM transforms the URL and forwards the payload to Azure Bing Search Service.
  - Transformed URL corresponds to  
https://api.bing.microsoft.com/v7.0/search?q=QUERY\_TEXT&mkt=LOCALE
- Use default APIM policies.

#### *API settings for Google (optional)*

Configure All operations. Key points are shown below.

- Adjust allowed origins if necessary. Have a look at [Known issues](#).
- AadId- are tokens for values stored in [Named values](#).

ApiKeyGoogleSearch stores a query string combination of **key=API\_KEY&cx=SEARCH\_ENGINE\_ID** used in your instance of Google Custom Search API. Please refer to Google documentation for Custom Search to generate those values (https://developers.google.com/custom-search/v1/introduction). They are available free of charge to developers.

A sample working config is given below.

- Note that unlike the Bing endpoint the main policy of Google endpoint does not use the header with ApiKeyGoogleSearch. In Google search, the key is injected by APIM directly into a query string in the search operation (as shown below).

```
<policies>
  <inbound>
    <base />
    <cors allow-credentials="false">
      <allowed-origins>
        <origin>*</origin>
      </allowed-origins>
      <allowed-methods preflight-result-max-age="300">
        <method>GET</method>
      </allowed-methods>
      <allowed-headers>
        <header>Access-Control-Allow-Origin</header>
        <header>Authorization</header>
        <header>Content-Type</header>
        <header>Origin</header>
      </allowed-headers>
    </cors>
    <validate-jwt header-name="Authorization" failed-validation-
httpcode="401" failed-validation-error-message="Unauthorized. Access token is
missing or invalid.">
      <openid-config
url="https://login.microsoftonline.com/organizations/v2.0/.well-known/openid-
configuration" />
```

```

        <issuers>
          <issuer>https://sts.windows.net/{{AadId}}/</issuer>
          <!--Optional authorized external domains (end with backslash)-->
          <issuer>https://sts.windows.net/{{AadId-gt}}/</issuer>
          <issuer>https://sts.windows.net/{{AadId-kavoetokkiisa}}/</issuer>
        </issuers>
      </validate-jwt>
    </inbound>
    <backend>
      <base />
    </backend>
    <outbound>
      <base />
    </outbound>
    <on-error>
      <base />
    </on-error>
  </policies>

```

After that, you need to add and configure the search operation for your Google endpoint:

search

Display name: search

Name: search (it can add any descriptive value, not important)

URL: GET /search

## Frontend

* Display name	<input type="text" value="search"/>
* Name	<input type="text" value="search"/>
* URL	<input type="text" value="GET"/> <input type="text" value="/search"/>
Description ⓘ	<input type="text"/>
Tags	<input type="text" value="e.g. Booking"/>

Configuration:

- Add a rewrite URL like shown below.
  - Clients send GET requests with query string parameters to the APIM's URL  
https://**instance**.azure-api.net/google/search?q=QUERY\_TEXT&lr=LOCALE
  - APIM transforms the URL and forwards the payload to Google Custom Search service.
  - Transformed URL corresponds to  
https://www.googleapis.com/customsearch/v1?key=API\_KEY&cx=SEARCH\_ENGINE\_ID  
&q=QUERY\_TEXT&lr=LOCALE

- The first two parameters should be injected within the APIM policy as shown below.

```
<policies>
  <inbound>
    <base />
    <rewrite-uri template="{{ApiKeyGoogleSearch}}" />
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
```

### Known issues

There was an intermittent issue that might have been associated with the use of the Consumption plan for API Management in the DEV-tenant.

- Please refer to the table available at [https://azure.microsoft.com/en-us/pricing/details/api-management/?ef\\_id=\\_k\\_CjwKCAjw67ajBhAVEiwA2g\\_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD\\_BwE\\_k\\_&OCID=AIDcmmftanc7uz\\_SEM\\_\\_k\\_CjwKCAjw67ajBhAVEiwA2g\\_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD\\_BwE\\_k\\_&gclid=CjwKCAjw67ajBhAVEiwA2g\\_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD\\_BwE#pricing](https://azure.microsoft.com/en-us/pricing/details/api-management/?ef_id=_k_CjwKCAjw67ajBhAVEiwA2g_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD_BwE_k_&OCID=AIDcmmftanc7uz_SEM__k_CjwKCAjw67ajBhAVEiwA2g_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD_BwE_k_&gclid=CjwKCAjw67ajBhAVEiwA2g_jEHK13rhWiOba6Xz7YiLxkWCKMivEjoXqVmi8dQPCK3rVkmewyqSb6BoCrVkQAvD_BwE#pricing)
- As indicated there, the Consumption plan uses Shared isolation.
- According to discussions in Microsoft forums, this might occasionally cause CORS-conflicts if you specify explicit values in [allowed-origins](#).
  - Other plans offer Private isolation and do not seem to encounter similar issues.
- Anyway, it might be prudent to avoid defining explicit CORS rules in APIM policies.

## Security

### Frontend

The web part is hosted on SharePoint Online and utilizes the standard authentication options provided by the platform. Connection to backend endpoints is established using the Azure App registration [geminiaiwp](#) as outlined in the “Permissions” chapter of this document.