Hathr External API

Hostnames

oAuth authentication: **hathr.auth-fips.us-gov-west-1.amazoncognito.com**
API: **api.hathr.ai**

Authentication

All requests must include an oAuth 2.0 bearer token.
A client ID and client secret will be provided. These should be used to obtain the bearer token.
When requesting the bearer token, the scope 'hathr/llm' must be used.

Methods

chat

Endpoint: `/v1/chat` (POST)

Makes a chat request to Hathr AI.
Conversation history should be sent in the messages array, oldest first.

Request Body

```
{
  "messages": [
    {
      "role": "user|assistant",
      "text": "string"
    }
  ],
  "temperature": "float",
  "topP": "float"
}
```

- `messages`: Array of message objects (Required)

    - `role`: String - Either “user” or “assistant”
    - `text`: String - The message content
- `temperature`: Float - Controls randomness (Optional, default: 0.2, range: 0-2.0)
- `topP`: Float - Controls diversity (Optional, default: 1.0, range: 0-1.0)

Response

```
{
 "status": 200,
 "message": "Success",
 "data": {
   "usage": {
     // Token usage statistics
   },
   "message": "string" // Response text
 }
}
```

Error Responses

- `400` - Invalid request format or parameters

upload_url

Endpoint: `/v1/document/upload` (POST)

Returns a pre-signed S3 URL for uploading documents which can be used in future chat requests.

Request Body

```
{
 "filename": "string",
 "type": "string"
}
```

- `filename`: The name of the file to be uploaded
- `type`: The MIME type of the file to be uploaded, for example application/pdf

Response

```
{
 "status": 200,
 "message": "Document embedding URL created successfully",
 "response": {
   "signedUrl": "string"
 }
}
```

document/complete

Endpoint: `/v1/document/complete` (GET)

Checks the status of document processing.

Response

```
{
  "code": "integer",
  "message": "string",
  "response": {
    // Processing status details
  }
}
```

- `code`: 200 for complete, 202 for processing
- `message`: Status description

document/list

Endpoint: `/v1/document/list` (GET)

Retrieves a list of all available documents.

Response

```
{

  "code": "integer",

  "message": "string",

  "response": {

    "documents": [
      {
        "id": "string",
        "name": "string"
      }
    ]
  }
}
```

Each document entry includes an `id` that can be used to delete the document.

document/delete

Endpoint: `/v1/document/{documentId}` (DELETE)

Deletes the specified document and any embeddings associated with it.

Path Parameters

- `documentId`: String - Identifier returned by `/v1/document/list`

Response

```
{
  "code": 200,
  "message": "Document and embeddings associated with it have been deleted",
  "response": null
}
```

document/chat

Endpoint: `/v1/document/chat` (POST)

Makes a chat request to Hathr AI using specific documents as context.

Request Body

```
{
  "documents": ["string"],
  "messages": [
    {
      "role": "user|assistant",
      "text": "string"
    }
  ]
}
```

- `documents`: Array of document names to use as context
- `messages`: Array of message objects
  - `role`: String - Either “user” or “assistant”
  - `text`: String - The message content’

Response

```
{
  "status": 200,
  "message": "Success",
  "data": {
    "usage": {
      // Token usage statistics
    },
    "message": "string" // Response text
  }
}
```