

# MASTER PROJECT

*eXercise Open Call Project*

**eXercise Documentation**

**NOESISHUB & 8 BELLS**

June 6, 2025

# Table of Contents

|  |          |
|--|----------|
| <b>eXercise OPEN CALL PROJECT</b>                              | <b>3</b> |
| <b>Pilot Project Report: NLP-based Trainer</b>                 | <b>3</b> |
| 1 Objective . . . . .  | 3        |
| 2 Overview of System . . . . .                                 | 3        |
| 2.1 Robotic Arm Selector Tab . . . . .                         | 3        |
| 2.2 AI Advisor Tab . . . . .                                   | 4        |
| 3 System Components . . . . .                                  | 5        |
| 3.1 Natural Language Interface . . . . .                       | 5        |
| 3.2 Distance Input Panel . . . . .                             | 5        |
| 3.3 Model Integration . . . . .                                | 5        |
| 3.4 Stress Analysis . . . . .                                  | 5        |
| 4 Sample Output Format . . . . .                               | 5        |
| 5 Usability and User Experience . . . . .                      | 5        |
| 6 Challenges and Recommendations . . . . .                     | 5        |
| 7 Conclusion . . . . .   | 6        |
| <b>VR SCENARIO CREATION LIBRARY</b>                            | <b>6</b> |
| 1 Components Implemented . . . . .                             | 6        |
| 2 Behaviour Component Fields Overview . . . . .                | 7        |
| 2.1 Behaviour Component Field Descriptions . . . . .           | 8        |
| 3 Unity Integration Guide – HuggingFace JSON Fetcher . . . . . | 8        |
| 3.1 HuggingFaceJsonFetcher Field Descriptions . . . . .        | 10       |

# eXercise OPEN CALL PROJECT

The following documentation addresses the integration of assets developed using Unity Engine and the Unity Editor, for the purposes of the MASTER OC project **eXercise**.

One of the components of the **eXercise Training System** is the creation of VR-based scenarios. The library described below enables dynamic VR scenarios for: - fire incidents (by creating a virtual fire effect at a robotic arm/device) - malfunction scenarios at a specified robotic arm or device

This component accepts structured decisions output from an LLM prompt, used by a trainer. The result is a VR scenario presented to a trainee. It is distributed as a Unity-native DLL library.

## Pilot Project Report: NLP-based Trainer

### 1 Objective

The primary aim of this pilot project was to create a streamlined, interactive interface enabling trainers to input high-level textual descriptions (prompts) regarding training scenarios involving virtual fires and robotic arm malfunctions. These inputs are processed by advanced language models (LLMs) to generate scenario recommendations, identify malfunctions, and support training decisions.

### 2 Overview of System

The system has been developed using Streamlit and comprises two key interface sections:

1. Robotic Arm Selector Tab
2. AI Advisor Tab

#### 2.1 Robotic Arm Selector Tab

This tab provides users (trainers) with the ability to:

- Select an LLM model, supporting both OpenAI and Hugging Face APIs (e.g., GPT-4o, Zephyr, Mistral).
- Dynamically input the number of robotic arms for the session.
- Submit textual prompts describing training scenarios.
- Enter distance values for each robotic arm relative to the trainee.

The system then:

- Processes the prompt and distances using the selected LLM.
- Returns structured JSON output indicating:
  - Which robotic arm may be experiencing a virtual fire.
  - Which arm is potentially malfunctioning.
  - An explanation for the AI's decision.
- Displays example prompts to guide user input.
- Persists all session data for later analysis.

**Robotic Arm Selector**

Select the model to use: Qwen/Qwen2.5-72B-Instruct

Number of Robotic Arms: 5

Robotic Arm Selector AI Advisor

Please provide your inquiry in natural language and the distances of the robotic arms in the right column, and I will provide you with the best recommendation.

**Distance Input**

(zero distance is the point where the trainee is located - working)

Enter distance of robotic arm 1: 4,10

Enter distance of robotic arm 2: 5,19

Enter distance of robotic arm 3: 2,64

Enter distance of robotic arm 4: 9,49

Enter distance of robotic arm 5: 9,97

**AI Recommendation**

Selected AI Model: Qwen/Qwen2.5-72B-Instruct

LLMs may generate inaccurate responses; please verify the responses and comply with the corresponding terms.

Enter your prompt here

Sample prompts

Figure 1: Robotic Arm Selector Tab

## 2.2 AI Advisor Tab

This tab supports two advanced functions:

1. Session Stress Result Retrieval: Connects to an external stress classification API to retrieve the participant's emotional/stress performance based on physiological data.
2. Performance Report Generation: Uses the LLM to generate a markdown report incorporating:
  - Session metadata (date, time, participant ID)
  - Robotic arm distances
  - Malfunction and fire detection summary
  - Stress analysis results
  - Analytical tables and conclusions

**Robotic Arm Selector**

Select the model to use: Qwen/Qwen2.5-72B-Instruct

Number of Robotic Arms: 5

Robotic Arm Selector AI Advisor

Session Performance Report Get Session Stress Results

Enter the participant id: 1704-1-1-1

Enter the date of the session (dd/mm/yyyy): 2025/02/27

Enter the time of the session (hh:mm): 16:27

Figure 2: AI Advisor Tab

## 3 System Components

### 3.1 Natural Language Interface

The platform leverages large language models (via OpenAI and Hugging Face) for prompt processing. The system dynamically constructs a structured prompt containing:

- Trainer’s textual input
- Robotic arm distance values
- Request for identification of robotic arm associated with virtual fire and malfunction

### 3.2 Distance Input Panel

Users can input precise numerical distances for each robotic arm. These values are used by the LLMs to make scenario inferences.

### 3.3 Model Integration

Models supported include:

- GPT-4o, GPT-4o-mini (OpenAI API)
- Zephyr-7b-beta, Mistral-7B-Instruct-v0.3, Qwen2.5-72B-Instruct (Hugging Face Inference API)

The list of models can be extended to include others.

### 3.4 Stress Analysis

The system queries a stress analysis API to retrieve a participant’s classification result for the session. It matches timestamps and participant IDs to fetch the correct entry.

## 4 Sample Output Format

```
{
  "Robotic_Arm_Virtual_Fire": 1,
  "Robotic_Arm_Malfunctioning": 2,
  "Reason_of_selection": "Robotic Arm Number 1 is the closest to the base with a distance of 1.43, which"
}
```

Generated with OpenAI API (gpt-4o).

## 5 Usability and User Experience

- The interface is intuitive and provides real-time feedback.
- Prompts with fewer than 20 characters are disallowed to prevent vague input.
- Prompt inputs are stored and reused for report generation.
- Users can view AI model outputs in both JSON and Chat UI formats.

## 6 Challenges and Recommendations

- Challenge: Occasional JSON parsing failures from LLM responses.  
Recommendation: Implement retry logic and more constrained prompt formatting.
- Challenge: Ambiguity in prompt interpretation when language is imprecise.  
Recommendation: Enhance prompt scaffolding and allow iterative refinement.

## 7 Conclusion

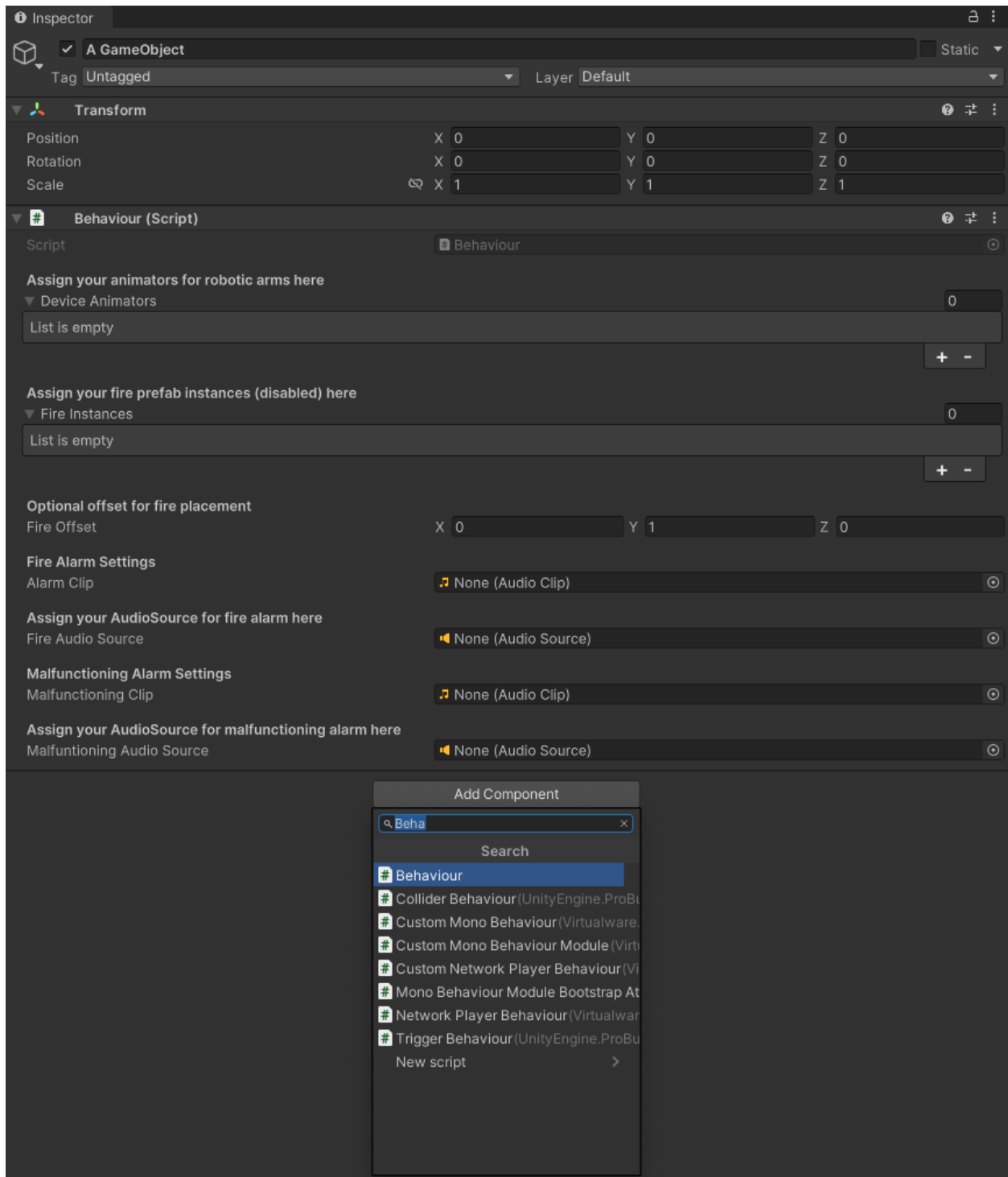
The pilot project successfully demonstrated the feasibility of translating natural language trainer inputs into actionable XR training scenarios. The integration of multiple LLMs, combined with real-time data capture and post-session reporting, creates a powerful tool for enhancing safety training simulations.

# VR SCENARIO CREATION LIBRARY

## 1 Components Implemented

Two key components have been developed: - **Behaviour** component - **HuggingFaceJsonFetcher** component

The **Behaviour** is designed to accommodate the virtual robotic arms used in your Unity scene. You can assign it via the Unity Inspector, as shown below:



Example showing the Behaviour component fields in the Unity Inspector.

## 2 Behaviour Component Fields Overview

All fields listed below are customizable for flexibility across different environments and training goals.

| Field Name                             | Type                      | Description                               |
|--|---------------------------|---|
| <code>deviceAnimators</code>           | <code>Animator[]</code>   | Robotic Devices with Animators attached   |
| <code>fireInstances</code>             | <code>GameObject[]</code> | Prefabs for fire effects                  |
| <code>fireOffset</code>                | <code>Vector3</code>      | Offset to place fire effects              |
| <code>alarmClip</code>                 | <code>AudioClip</code>    | Sound to play during fire incident        |
| <code>fireAudioSource</code>           | <code>AudioSource</code>  | Audio source for fire alarms              |
| <code>malfunctioningClip</code>        | <code>AudioClip</code>    | Sound to play during malfunction incident |
| <code>malfunctioningAudioSource</code> | <code>AudioSource</code>  | Audio source for malfunction alerts       |

---

## 2.1 Behaviour Component Field Descriptions

**2.1.1 `deviceAnimators` (`Animator[]`)** A list of Animator components. Assign any `GameObject` (e.g., robotic arms) that contains an Animator. If no Animator is present, it will not work.

**2.1.2 `fireInstances` (`GameObject[]`)** A pool of disabled fire prefabs placed in the scene. The script will activate one randomly during a fire event.

**2.1.3 `fireOffset` (`Vector3`)** Offset where the fire prefab appears relative to the device. Default is (0, 1, 0).

**2.1.4 `alarmClip` (`AudioClip`)** The sound that plays during a fire incident. Use a looping alarm or suitable alert.

**2.1.5 `fireAudioSource` (`AudioSource`)** Audio source used to play `alarmClip`.

**2.1.6 `malfunctioningClip` (`AudioClip`)** Audio clip triggered when a malfunction event occurs.

**2.1.7 `malfunctioningAudioSource` (`AudioSource`)** Audio source used to play the `malfunctioningClip`.

---

## 3 Unity Integration Guide – HuggingFace JSON Fetcher

This section explains how to use the `HuggingFaceJsonFetcher` Unity component alongside `Behaviour` to create automated, LLM-driven VR scenarios using structured JSON input.

The trainer prompts in the provided LLM, the desired scenario (fire, malfunctioning incident or even both) and the LLM return a selection of the selected robotic arm which will virtually be set on fire. When the trainee starts the VR application, the library retrieves the structured output of the LLM, which indicated the robotic devices to be set on fire, malfunction or in case of a combination (a selection of max. 2 robotic devices) of the incidents which corresponding robotic device suffers which disaster.

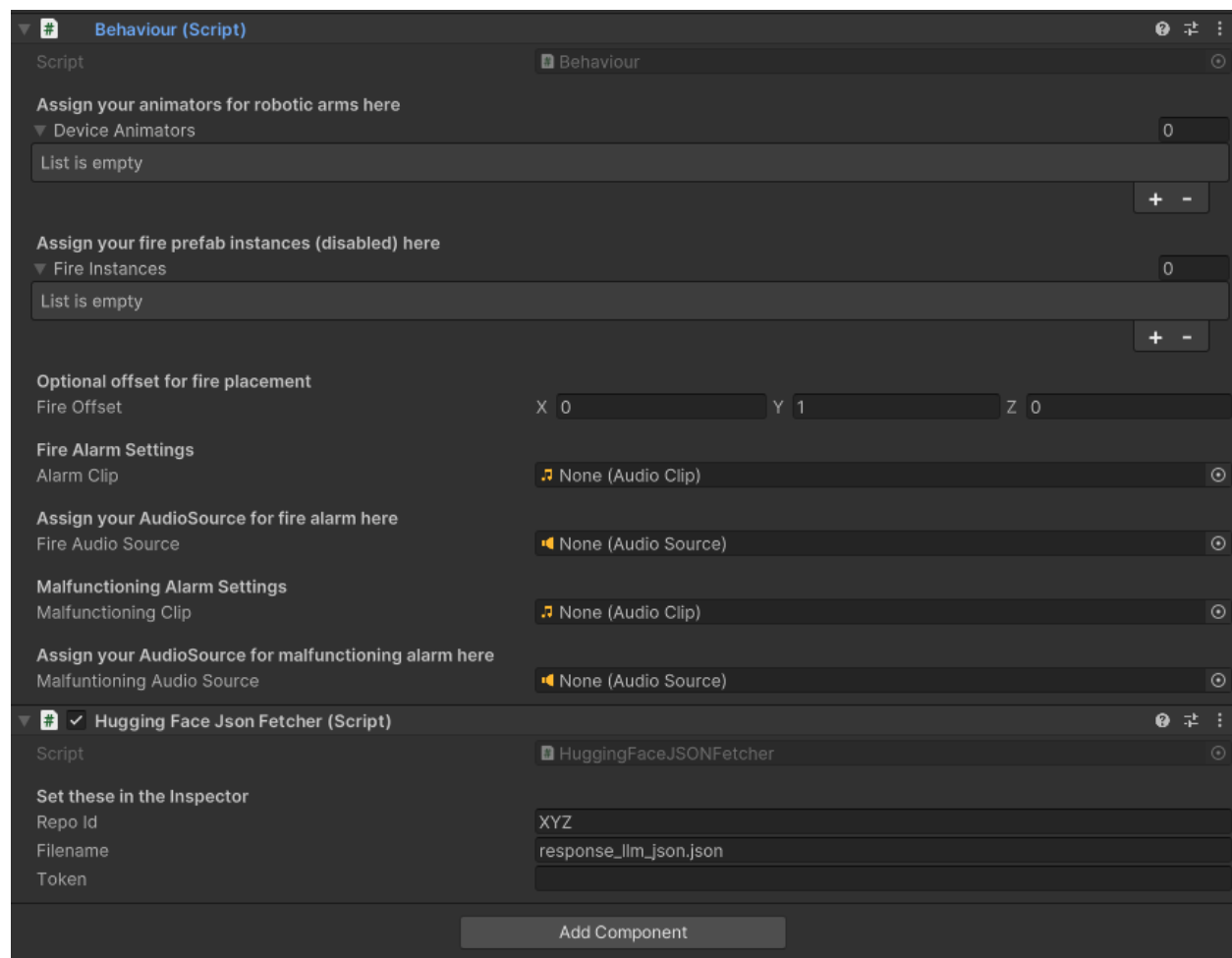
An example structured output is shown here, presenting also the JSON structure of the output:

```
{
  "Robotic_Arm_Virtual_Fire": 1,
  "Robotic_Arm_Malfunctioning": 2,
  "Reason_of_selection": "Robotic Arm Number 1 is the closest to the base with a distance of 1.43, which"
}
```

This component, retrieves the LLM structured output from a dataset repository, which must be created in HuggingFace (see the corresponding documentation). In order to successfully retrieve this information, we have to correctly set the following fields, shown in the table below:

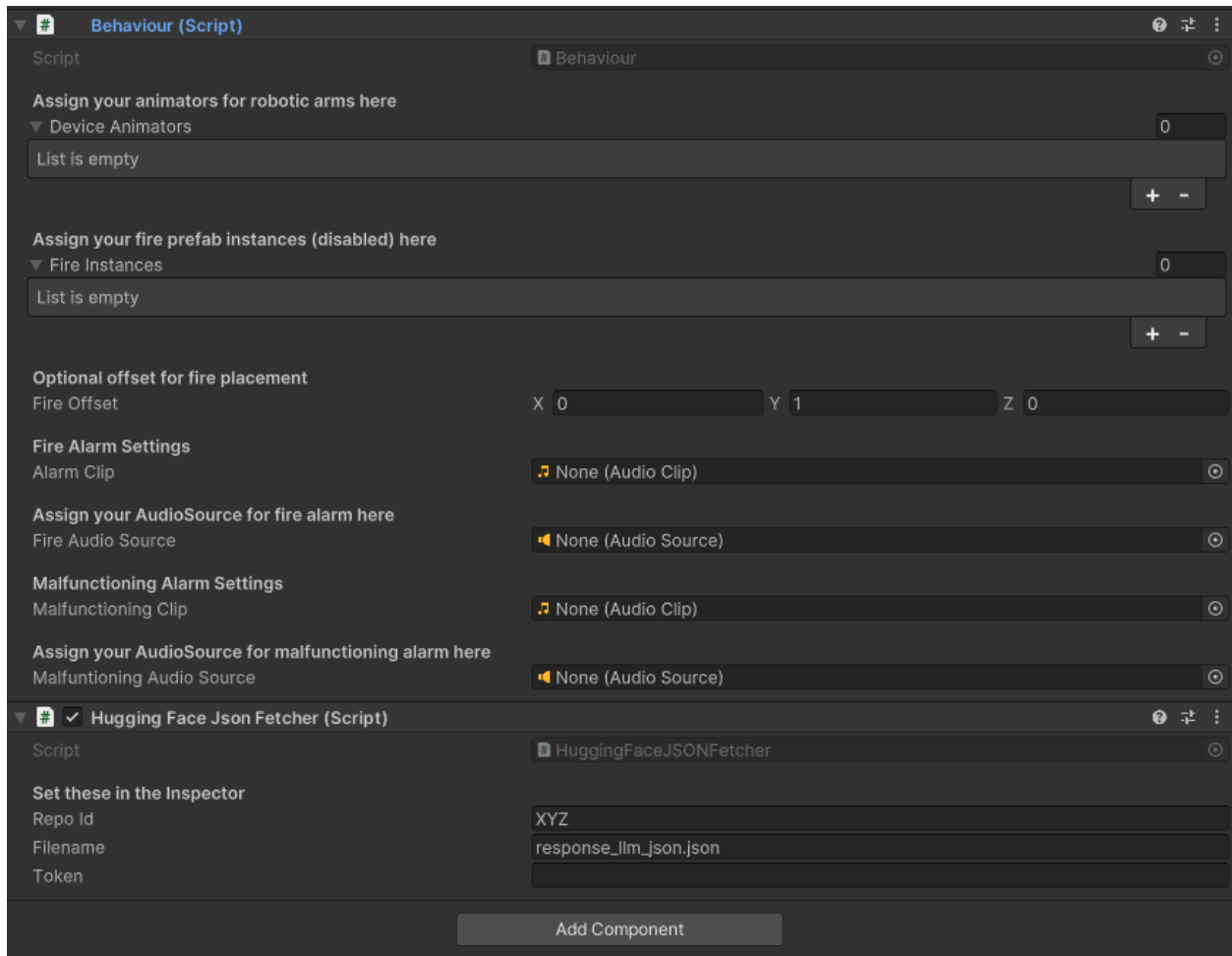


| Parameter | Type   | Description  |
|-----------|--------|--|
| repoId    | string | Hugging Face dataset repository ID (e.g., username/dataset_name).    |
| filename  | string | JSON file name inside the dataset.                                   |
| token     | string | (Optional) Hugging Face access token. Required for private datasets. |



*Example showing the HuggingFaceJsonFetcher component fields in the Unity Inspector.*

Below we show how a fully configured Behaviour and HuggingFaceJsonFetcher component looks like.



Example showing how both components can be configured in the Unity Inspector. This is a fully functional configuration.

### 3.1 HuggingFaceJsonFetcher Field Descriptions

**3.1.1 repoId (string)** The repoId is the name of the created dataset in Hugging Face (this is also the name that must be used in the LLM module in HuggingFace). No links or any url of any kind is needed, only the profile name followed with the dataset name e.g. my-user-name/dataset-name. Inside this repository there must be at least one file (default can be the file with name response\_llm\_json.json ) in which the LLM will write its structured output.

**3.1.2 filename (string)** The name of the file the LLM writes the structured output in the designated dataset (see above field).

**3.1.3 token (string)** In order to access the Hugging Face dataset, we need to create a token from the Hugging Face user account in order for the library to be authenticated and authorized to access the file mentioned above (same logic as in github, gitlab, etc). Refer to this link for additional information on this topic. Be aware to set the correct privileges, otherwise the library will now be able to access the file, resulting in a 401 error response.

**BE CAREFULL not to publish or accidentally share the generated token. Line other tokens, is must not be publically available!**