**API Documentation**

**I. Google Gemini API**

* **Overview:**
  + The Google Gemini API is used for generating realistic text-based scenarios and (potentially later) for handling conversational interactions. The model we will be using during the development phase is gemini-1.5-pro-latest, other models might be considered in future iterations.
* **Base URL:**
  + The main URL to interact with the Gemini API depends on the specific library you are using, for google-generativeai we will be using the model.generate\_content(prompt) method.
* **Authentication:**
  + **Method:** Workload Identity Federation is recommended for production environments, or OS environment variables when developing.
  + **Credentials:** Obtain a Google Cloud API key from your project's credentials page. You may want to create a service account for proper management, and give it the minimum necessary permissions for your project.
  + **Environment Variables:** Your API key must be stored as GOOGLE\_API\_KEY.
* **Text Generation Endpoint (model.generate\_content(prompt)):**
  + **Description:** This endpoint is used to generate text responses based on provided prompts. You will use this endpoint to call the Gemini API with different instructions and prompt templates for scenario generation.
  + **Method:** model.generate\_content(prompt) uses the gemini-1.5-pro-latest model for generating text.
  + **Request Parameters (through the model.generate\_content method):**
    - prompt (required): A string containing the text prompt (using one of the templates described in "Prompt Templates" ClickUp doc) for the model to generate.
    - Example: "Generate a realistic scenario for a data analyst in a e-commerce company"
    - model (required): A string containing the name of the model. You will use gemini-1.5-pro-latest.
    - generation\_config (optional): A dictionary with additional parameters, such as:
      * temperature (float): Controls the randomness of the output. Higher values make the text more diverse.
      * top\_p (float): Controls how random the response may be.  
        \* top\_k (integer): Controls how random the response may be.  
        \* max\_output\_tokens (integer): Sets the maximum tokens for the response.
    - Example:
  + generation\_config = {
  + "temperature": 0.9, # Increased temperature
  + "top\_p": 0.9, # Adjusted top-p
  + "top\_k": 10, #Adjusted top-k
  + "max\_output\_tokens": 2048,
  + }

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487).

* + - safety\_settings (optional): You should include safety filters to avoid inappropriate responses, for example:
    - safety\_settings = [
    - {"category": "HARM\_CATEGORY\_HARASSMENT", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},
    - {"category": "HARM\_CATEGORY\_HATE\_SPEECH", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},
    - {"category": "HARM\_CATEGORY\_SEXUALLY\_EXPLICIT", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},
    - {"category": "HARM\_CATEGORY\_DANGEROUS\_CONTENT", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},
    - ]

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487).

* + **Response Format:** The response from this endpoint contains the text generated from the model, this data will be returned in a response.text format.
* **Error Handling:**
  + The Gemini API may return errors related to authorization, rate limits, or invalid requests. You need to handle these errors correctly and provide feedback to the user. This can be done by catching specific HTTP exceptions or using try-catch blocks.
  + You should log every API error, and provide feedback to the user if the call fails.

**II. ClickUp API**

* **Overview:**
  + The ClickUp API will allow us to manage our development process programmatically.
* **Base URL:**
  + https://api.clickup.com/api/v2/
* **Authentication:**
  + You will require an API Token.
  + To create a new ClickUp API token, follow these instructions:
    1. In your ClickUp workspace, go to your profile settings.
    2. Go to the “Apps” section, and click on the button called "API Token".
    3. Create a new token. This will generate a unique string of characters that will be used as an authorization token.
  + Store the token using Google Cloud Secret Manager (recommended) or Environment Variables (less secure), and manage them carefully.
  + The token should be passed in the HTTP header as Authorization: <Your API Key>.
* **Relevant API Endpoints**
  + **Get Task:** GET https://api.clickup.com/api/v2/task/<task\_id> retrieves the information from a single task.
  + **Update Task:** POST https://api.clickup.com/api/v2/task/<task\_id> changes a specific information from a task, use this to update their state, assignees, or change the time tracked.
  + **Add Comment:** POST https://api.clickup.com/api/v2/task/<task\_id>/comment adds a comment to the task.
  + **Create Task:** POST https://api.clickup.com/api/v2/list/<list\_id>/task creates a task inside a specific list using it's list\_id.
  + **Get list:** GET https://api.clickup.com/api/v2/list/<list\_id> gets all information from the specified list.
  + **Get Folder:** GET https://api.clickup.com/api/v2/folder/<folder\_id> gets all information from the specified folder.
* **Key Concepts:**
  + task\_id: Unique identifier for a specific task.
  + list\_id: Unique identifier for a specific list.
  + team\_id: Unique identifier for a workspace.
  + user\_id: Unique identifier for a user within the workspace.

**III. Code Example**

Here is an example of how to call the Gemini API in your python code, using the google-generativeai library.

import google.generativeai as genai

import os

import logging

GOOGLE\_API\_KEY = os.getenv('GOOGLE\_API\_KEY')

genai.configure(api\_key=GOOGLE\_API\_KEY)

generation\_config = {

"temperature": 0.9,

"top\_p": 0.9,

"top\_k": 10,

"max\_output\_tokens": 2048,

}

safety\_settings = [

{"category": "HARM\_CATEGORY\_HARASSMENT", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},

{"category": "HARM\_CATEGORY\_HATE\_SPEECH", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},

{"category": "HARM\_CATEGORY\_SEXUALLY\_EXPLICIT", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},

{"category": "HARM\_CATEGORY\_DANGEROUS\_CONTENT", "threshold": "BLOCK\_MEDIUM\_AND\_ABOVE"},

]

def generate\_scenario(domain):

model = genai.GenerativeModel(

model\_name="gemini-1.5-pro-latest",

generation\_config=generation\_config,

safety\_settings=safety\_settings,

)

prompt\_template = select\_prompt(domain)

try:

response = model.generate\_content(prompt\_template)

logging.info(f"Successfully generated a scenario for domain:{domain}")

return response.text

except Exception as e:

logging.error(f"Error with Gemini API: {e}", exc\_info=True)

return "API Error, could not retrieve scenario"

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487).Python

**IV. Using this Documentation:**

* **Clarity:** Use this document as a central resource for all your API related information and instructions.
* **Implementation:** This information should be used when implementing the code for scenario generation and API integration.
* **Future Reference:** This should also serve as a reference point when working on future features that may require working directly with the API endpoints.