AI-Powered File Organizer

……………………………………………………………………………………………

This project uses the power of Google's Gemini AI to automatically sort a messy folder of files into neatly organized subdirectories. It intelligently analyzes each file's name and content to classify it into categories like Documents, Code, Images, and more.

How It Works

The logic is simple yet powerful:

1. **Input**: The script targets a single folder filled with various unsorted files.
2. **Analyze**: It goes through each file one-by-one. For each file, it sends its name and a preview of its content to the Gemini AI.
3. **Classify & Organize**: The AI returns a category for the file (e.g., "Code"). The script then creates a subfolder with that category name and moves the file into it.

The end result is that your one messy folder becomes neatly organized into multiple, categorized subfolders.

Core Concepts of the Code

This script is built on a few key programming concepts:

- **API (Application Programming Interface)**: An API allows our script to "talk" to Google's AI. We send a request (a "prompt") with the file information, and the Gemini API sends back a response (the file's category). The google-generativeai library handles this communication.
- **File I/O (Input/Output)**: These are the fundamental operations that let Python interact with files on your computer. The script uses:
  - os.path.join(): To correctly construct file paths that work on any operating system.
  - open(): To read the content of each file.

    - os.makedirs(): To create the new category subfolders.
    - shutil.move(): To move the files into their new home.
- **Error Handling & Retries**: The internet isn't always 100% reliable. To prevent the script from crashing due to temporary network issues or API limits, we use a try...except block with a retry loop. If an API call fails, the script waits and tries again automatically. This makes the application robust and resilient.

Code Implementation Breakdown

The Python script is divided into three main parts:

1. Setup and File Creation

- First, it installs the necessary Google library (!pip install).
- It then configures the connection to the Gemini API using a secret API key.
- Finally, it creates a sample my_messy_folder and populates it with various dummy files (.py, .txt, .csv, etc.) to simulate a real-world use case.

2. The get_file_category Function

This is the brain of the operation. It's responsible for classifying a single file. The function reads a preview of the file's content, creates a detailed instruction (a "prompt") for the AI, and sends that prompt to the Gemini model. It then cleans up the AI's text response and returns the category. If an error occurs, it automatically waits and retries the process a few times.

3. The Main Organization Loop

This part ties everything together. It gets a list of all files in the source directory and then loops through each one. Inside the loop, it calls the get_file_category function to get the correct category from the AI. It then creates a target folder (e.g., my_messy_folder/Documents), moves the file into it, and waits for one second before processing the next file to respect API rate limits.

How to Use in Google Colab

1. **Get an API Key**:
   - Go to Google AI Studio.
   - Click **"Create API key"** and copy the key.
2. **Add Key to Colab Secrets**:
   - In your Colab notebook, click the **key icon (🔑)** on the left sidebar.
   - Click **"Add new secret"**.
   - **Name**: GEMINI_API_KEY
   - **Value**: Paste your copied API key here.
   - Ensure the "Notebook access" toggle is on.
3. **Run the Script**:
   - Copy the complete Python script from our previous discussion into a single Colab cell and run it. The script will handle everything else automatically.
4. **Verify the Result**:
   - After the script finishes, run !ls -R my_messy_folder in a new cell to see the newly organized folder structure.

Troubleshooting

- **429 Resource has been exhausted Error**: This means you're making too many requests per minute. The time.sleep(1) in the code is designed to prevent this, but if you still encounter it, you may need to wait a minute before running the script again. Using the gemini-1.5-flash model is also recommended as it's faster and less likely to cause rate-limit issues.