```
!pip install requests beautifulsoup4
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.27.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (4.11.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2022
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4) (2.4
```

```
!pip install requests openai langchain
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.27.1)
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (0.27.8)
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-packages (0.0.202)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2022
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.65.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from openai) (3.8.4)
Requirement already satisfied: PyYAML>=5.4.1 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langch
Requirement already satisfied: dataclasses-json<0.6.0,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from lan
Requirement already satisfied: langchainplus-sdk>=0.0.9 in /usr/local/lib/python3.10/dist-packages (from langchain
Requirement already satisfied: numexpr<3.0.0,>=2.8.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.22.4)
Requirement already satisfied: openapi-schema-pydantic<2.0,>=1.2 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: pydantic<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.10.7)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (23
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->opena
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai)
```

0s completed at 8:28 PM

```
Requirement already satisfied: marshmallow-enum<2.0.0,>=1.5.1 in /usr/local/lib/python3.10/dist-packages (from dat
Requirement already satisfied: typing-inspect>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from dataclasses-
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.
Requirement already satisfied: packaging>=17.0 in /usr/local/lib/python3.10/dist-packages (from marshmallow<4.0.0,
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from typing-insp
```

```
import requests
import openai

# Set up the OpenAI API credentials
openai.api_key = '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'

# Function to get the most technically challenging repository
def get_most_technically_challenging_repo(user_url):
    # Extract the username from the user's GitHub URL
    username = user_url.split('/')[-1]

    # Make a request to the GitHub API to get the user's repositories
    repos_url = f'https://api.github.com/users/{username}/repos'
    response = requests.get(repos_url)
    repositories = response.json()

    # Variables to store the most technically challenging repository information
    max_technical_score = float('-inf')
    most_challenging_repo = None

    # Iterate over the repositories and assess their technical complexity using GPT and LangChain
    for repo in repositories:
        repo_name = repo['name']
        repo_url = repo['html_url']
        readme_url = f'https://raw.githubusercontent.com/{username}/{repo_name}/master/README.md'

        # Fetch the README file content from the repository
        readme_response = requests.get(readme_url)
```

```
            readme_content = readme_response.text

            # Assess the technical complexity of the repository using GPT and LangChain
            technical_score = assess_technical_complexity(readme_content)

            # Update the most technically challenging repository if necessary
            if technical_score > max_technical_score:
                max_technical_score = technical_score
                most_challenging_repo = {
                    'name': repo_name,
                    'url': repo_url,
                    'technical_score': technical_score
                }

    return most_challenging_repo

# Function to assess the technical complexity using GPT and LangChain
def assess_technical_complexity(text):
    # Make a request to the GPT API to assess the technical complexity
    prompt = f'Assess the technical complexity of the following text:\n\n{text}'
    response = openai.Completion.create(
        engine='text-davinci-003',
        prompt=prompt,
        max_tokens=100,
        temperature=0.5,
        n=1,
        stop=None,

    )
    technical_score = response.choices[0].text.strip()

    # Convert the LangChain score to a float
    technical_score = float(technical_score)

    return technical_score

# Example usage
```

```
user_url = 'https://github.com/Jinitha04'
most_challenging_repo = get_most_technically_challenging_repo(user_url)

if most_challenging_repo is not None:
    print(f'The most technically challenging repository is "{most_challenging_repo["name"]}"')
    print(f'URL: {most_challenging_repo["url"]}')
    print(f'Technical Score: {most_challenging_repo["technical_score"]}')
else:
    print('No repositories found for the given user.')
```

```
ERROR: unknown command "install--upgrade" - maybe you meant "install"
```

```
from flask import Flask, render_template, request
import requests
import openai

app = Flask(__name__)

# Set up the OpenAI API credentials
openai.api_key = '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'

# Function to get the most technically challenging repository
def get_most_technically_challenging_repo(user_url):
    # Extract the username from the user's GitHub URL
    username = user_url.split('/')[-1]

    # Make a request to the GitHub API to get the user's repositories
    repos_url = f'https://api.github.com/users/{username}/repos'
    response = requests.get(repos_url)
    repositories = response.json()

    # Variables to store the most technically challenging repository information
    max_technical_score = float('-inf')
    most_challenging_repo = None
```

```
    # Iterate over the repositories and assess their technical complexity using GPT and LangChain
    for repo in repositories:
        repo_name = repo['name']
        repo_url = repo['html_url']
        readme_url = f'https://raw.githubusercontent.com/{username}/{repo_name}/master/README.md'

        # Fetch the README file content from the repository
        readme_response = requests.get(readme_url)
        readme_content = readme_response.text

        # Assess the technical complexity of the repository using GPT and LangChain
        technical_score = assess_technical_complexity(readme_content)

        # Update the most technically challenging repository if necessary
        if technical_score > max_technical_score:
            max_technical_score = technical_score
            most_challenging_repo = {
                'name': repo_name,
                'url': repo_url,
                'technical_score': technical_score
            }

    return most_challenging_repo

# Function to assess the technical complexity using GPT and LangChain
def assess_technical_complexity(text):
    # Make a request to the GPT API to assess the technical complexity
    prompt = f'Assess the technical complexity of the following text:\n\n{text}'
    response = openai.Completion.create(
        engine='text-davinci-003',
        prompt=prompt,
        max_tokens=100,
        temperature=0.5,
        n=1,
        stop=None,

    )
```

```
    technical_score = response.choices[0].text.strip()

    # Convert the LangChain score to a float
    technical_score = float(technical_score)

    return technical_score

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        user_url = request.form['user_url']
        most_challenging_repo = get_most_technically_challenging_repo(user_url)

        if most_challenging_repo is not None:
            return render_template('result.html', repo=most_challenging_repo)
        else:
            return render_template('result.html', repo=None)

    return render_template('index.html')

if __name__ == '_main_':
    app.run(debug=True)
```

Colab paid products - Cancel contracts here