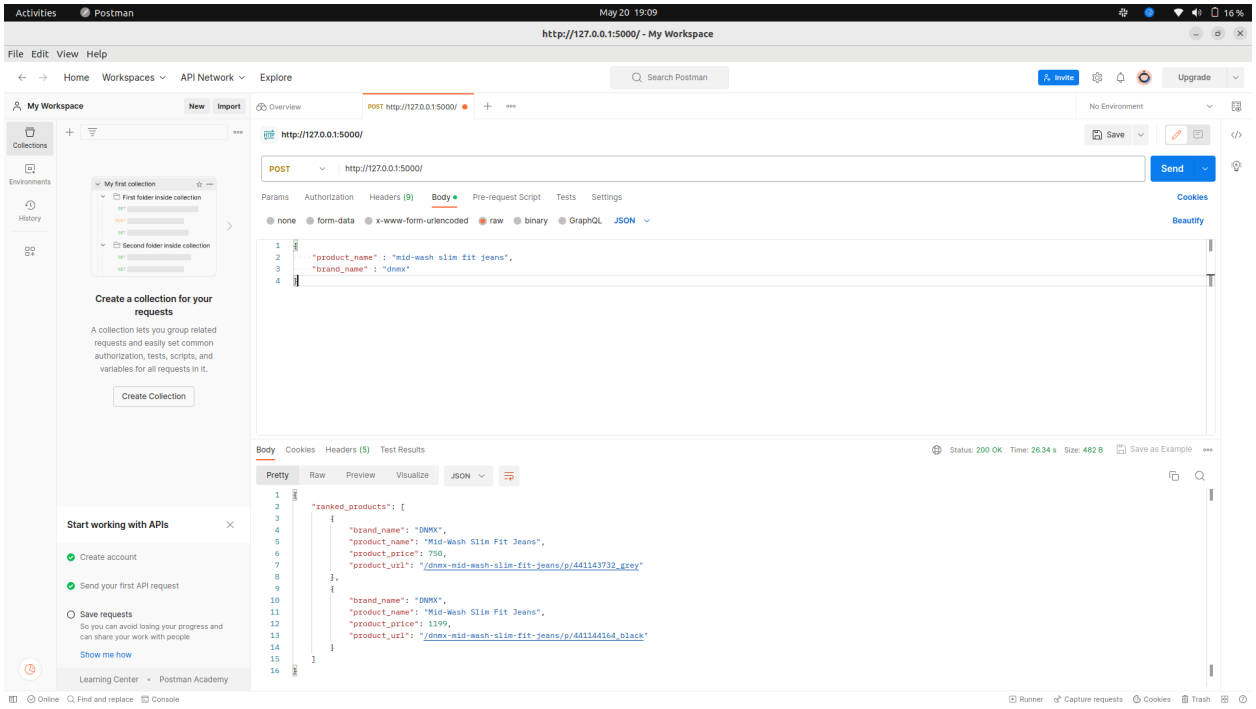
```python
import requests
from bs4 import BeautifulSoup
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from fastapi import FastAPI
import pandas as pd

url = "https://www.myntra.com/women-clothing"
response = requests.get(url)
print("RESPONSE",response)
soup = BeautifulSoup(response.content, "html.parser")

product_cards = soup.find_all("div", class_="product-item")
print("$$$$$", product_cards)
for product_card in product_cards:
    product_title = product_card.find("h2", class_="product-name").text
    product_description = product_card.find("div", class_="product-description").text

    product_features = {
        "product_title": product_title,
        "product_description": product_description    }

    with open("clothing_similarity_data.csv", "a") as f:
        f.write(",".join([str(k) for k in product_features.keys()]) + "\n")
        f.write(",".join([str(v) for v in product_features.values()]) + "\n")

data = pd.read_csv("clothing_similarity_data.csv")

vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(data["product_description"])

cosine_similarities = cosine_similarity(X, X)

def get_similar_products(product_title, n=5):
    index = data[data["product_title"] == product_title].index[0]

    similarities = cosine_similarities[index, :]

    similarities.sort(reverse=True)

    top_n_indices = similarities.argsort()[:n]

    top_n_products = data.iloc[top_n_indices]["product_title"].tolist()

    return top_n_products
```

I have tried web scraping using NLP and TF-IDF, Word2Vec, GloVe for **Clothing Similarity** And I didn't get solution. Let me check above SS. and I got solution another option used **selenium** library with NLP and APIs. let me check below SS.

```python
from bs4 import BeautifulSoup
from selenium import webdriver

from flask import Flask,request


app = Flask(__name__)

@app.route("/",methods=['POST'])
def get_data():

    data = request.get_json()

    input_product = data['product_name']

    input_brand = data['brand_name']

    url = "https://www.ajio.com/men-jeans/c/8302160017query=%3Arelevance&gridColumns=5"
    path = '/media/gsa/Data/Work_Space/interview_task/chromedriver_linux64/chromedriver'
    driver = webdriver.Chrome(path)

    driver.get(url)

    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

    soup = BeautifulSoup(driver.page_source, "html.parser")

    product_cards = soup.find_all("div", class_="item rilrtl-products-list__item item")

    product_name_list = []

    brand_name_list = []

    product_price_list = []

    product_url_list = []

    for product_card in product_cards:

        brand_name = product_card.find("div", class_="brand").text

        product_name = product_card.find("div", class_="nameCls").text

        product_price = product_card.find("span", class_="price").text

        product_url = product_card.find("a", class_="rilrtl-products-list__link").get("href")

        split_value = product_price.split("₹")
```

I create rest api using flask with product name and brand name. And I tested this api on postman. I gave input product name and brand name and get output top-N most similar items.



And last deploy with docker file. I have issue for billing with credit card. So please i request you if you have already billed on GCP. you can share credentials detail . i can work this.

No worry, I have worked with deploy with docker image. It is worked for cloud run with docker container image.

run.sh

services > stochastic > run.sh

```
V=29
SERVICE_NAME=stochastic-user-service
docker buildx build --platform linux/amd64 -t us-central1-docker.pkg.dev/randomly-dev/stochastic-images/${SERVICE_NAME}:$V .
docker push us-central1-docker.pkg.dev/randomly-dev/stochastic-images/${SERVICE_NAME}:$V
gcloud run deploy $SERVICE_NAME --image us-central1-docker.pkg.dev/randomly-dev/stochastic-images/${SERVICE_NAME}:$V
```

Dockerfile

services > stochastic > Dockerfile > ...

```
FROM us-central1-docker.pkg.dev/randomly-dev/stochastic-images/common-image:29
ENV PYTHONUNBUFFERED True
ENV APP_HOME /app
WORKDIR $APP_HOME
COPY requirements.txt .
RUN /usr/local/bin/python -m pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD exec gunicorn --bind :$PORT --workers 1 --threads 4 -k uvicorn.workers.UvicornWorker --timeout 0 main:app
```