Thành viên nhóm: Dương Thanh Phong, Mssv: 22280063

#Task 1

```
import google.generativeai as genai


def translate_texts(texts, api_key):
    """Dịch một danh sách các văn bản từ tiếng Anh sang tiếng Việt sử dụng Gemini API."""
    genai.configure(api_key=api_key)
    model = genai.GenerativeModel('gemini-pro')
    translated_texts = []

    for text in texts:
        try:
            response = model.generate_content(
                f"Translate the following English text to Vietnamese: {text}"
            )
            translated_texts.append(response.text)
        except Exception as e:
            translated_texts.append(f"Lỗi dịch thuật: {str(e)}")

    return translated_texts



if __name__ == "__main__":
    api_key = "AIzaSyChjqhafES9_hR197_QCaH9SZaOuo6QUXo"  # Thay thế bằng API key của bạn
```

```
texts_to_translate2 = ["Hello",
                       "I am John",
                       "Tôi là sinh viên"]
translated_texts2 = translate_texts(texts_to_translate2, api_key)

if translated_texts2:
    print("Văn bản gốc:")
    for text in texts_to_translate2:
        print(text, end=" \t")
    print("\nVăn bản đã dịch:")
    for text in translated_texts2:
        print(text, end=" \t")
```

```
#Task 2
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
import json
import spacy
from googletrans import Translator
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.metrics.pairwise import cosine_similarity


# Cấu hình Selenium
options = Options()
options.add_argument("--headless")  # Chạy trình duyệt ở chế độ ẩn
options.add_argument("--disable-gpu")
service = Service('C:\\Users\\ASUS\\Documents\\chromedriver-win64\\chromedriver-win64.exe')  # Đường dẫn đến chromedriver


# Mở trình duyệt và tải trang web
driver = webdriver.Chrome()
url = "https://www.presight.io/privacy-policy.html"
driver.get(url)


# Lấy nội dung HTML từ trang
html_content = driver.page_source
driver.quit()  # Đóng trình duyệt


# Phân tích HTML với BeautifulSoup
soup = BeautifulSoup(html_content, "html.parser")


# Hàm trích xuất nội dung tiêu đề, đoạn văn và danh sách từ thẻ div cụ thể
def extract_content_from_div(soup, div_class):
    sections = {}
    current_title = None
```

```
    # Tìm thẻ div cụ thể theo class
    target_div = soup.find("div", class_=div_class)
    if not target_div:
        print(f"Không tìm thấy thẻ div với class '{div_class}'")
        return sections

    # Tìm tất cả các thẻ trong div
    for tag in target_div.find_all(['h1', 'h2', 'h3', 'p', 'ul']):  # Chỉ tìm các thẻ trong danh sách này
        if tag.name in ['h1', 'h2', 'h3']:  # Nếu là thẻ tiêu đề
            current_title = tag.get_text(strip=True).replace(" ", "_").replace(":", "").replace(".", "")
            sections[current_title] = {}  # Tạo dictionary mới cho mỗi tiêu đề
        elif tag.name == 'p' and current_title:  # Nếu là đoạn văn thuộc tiêu đề trước đó
            paragraph_text = tag.get_text(strip=True)
            key = f"{current_title}_Content_{len(sections[current_title]) + 1}"
            sections[current_title][key] = paragraph_text
        elif tag.name == 'ul' and current_title:  # Nếu là danh sách không thứ tự
            list_items = [li.get_text(strip=True) for li in tag.find_all('li')]
            key = f"{current_title}_List_{len(sections[current_title]) + 1}"
            sections[current_title][key] = list_items

    return sections

# Gọi hàm trích xuất với class cụ thể
div_class = "chakra-stack css-1uji4px"
```

```
structured_content = extract_content_from_div(soup, div_class)

# Lưu dữ liệu vào file JSON
with open("privacy_policy_filtered.json", "w", encoding="utf-8") as f:
    json.dump(structured_content, f, ensure_ascii=False, indent=4)

print("Dữ liệu đã được lưu vào file privacy_policy_filtered.json")

#Tải mô hình spaCy NLP (tiếng Anh là ngôn ngữ mặc định)
nlp = spacy.load("en_core_web_sm")

# Khởi tạo Google Translator
translator = Translator()

# Tải dữ liệu từ file JSON
with open("C:\\Users\\ASUS\\privacy_policy_filtered.json", "r",encoding='utf-8') as
file:
    data = json.load(file)

# Làm phẳng dữ liệu JSON để tìm kiếm dễ dàng hơn
def flatten_json(json_obj, parent_key='', sep='_'):
    items = []
    for k, v in json_obj.items():
        new_key = f"{parent_key}{sep}{k}" if parent_key else k
        if isinstance(v, dict):
            items.extend(flatten_json(v, new_key, sep=sep).items())
```

```
        elif isinstance(v, list):
            items.append((new_key, ', '.join(v)))
        else:
            items.append((new_key, v))
    return dict(items)

flattened_data = flatten_json(data)

# Xử lý trước các giá trị JSON
documents = list(flattened_data.values())
keys = list(flattened_data.keys())

# Vecto hóa TF-IDF
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(documents)

# Chức năng tìm đoạn văn phù hợp nhất
def find_most_relevant_passage(question):
    question_vector = vectorizer.transform([question])
    similarities = cosine_similarity(question_vector, tfidf_matrix).flatten()
    best_idx = similarities.argmax()
    return keys[best_idx], documents[best_idx]

# Chức năng Chatbot hỗ trợ đa ngôn ngữ
def chatbot():
```

```
    print("Chatbot: Hello! I support multiple languages. Ask me anything about the Privacy Policy.")
    while True:
        question = input("You: ")
        if question.lower() in ["exit", "quit"]:
            print("Chatbot: Goodbye!")
            break

        # Phát hiện ngôn ngữ và dịch sang tiếng Anh nếu cần
        detected_lang = translator.detect(question).lang
        if detected_lang != "en":
            translated_question = translator.translate(question, src=detected_lang, dest="en").text
            print(f"Chatbot: (Translated Question: {translated_question})")
        else:
            translated_question = question

        # Câu hỏi được dịch trước khi xử lý
        doc = nlp(translated_question)
        processed_question = " ".join([token.lemma_ for token in doc if not token.is_stop])

        # Nhận đoạn văn có liên quan nhất
        key, passage = find_most_relevant_passage(processed_question)

        # Dịch đoạn văn trở lại ngôn ngữ của người dùng nếu cần
        if detected_lang != "en":
```

```
            passage = translator.translate(passage, src="en", dest=detected_lang).text


        print(f"Chatbot: {passage}")


# Start the chatbot
chatbot()
```