Project Documentation: Gift Idea Generation

Overview This project is an automated tool designed to generate personalized gift ideas. It uses personal interest data extracted from a CSV file and OpenAI's API to suggest gifts aligned with individual preferences. Additionally, it generates Amazon links for these gifts and provides explanations as to why each gift is suitable.

Key Features

- Data Extraction: Reads a CSV file to obtain specific interests.
- Gift Generation: Uses OpenAI's API to suggest gifts based on the extracted data.
- Amazon Links: Creates direct links to Amazon for easy purchase of suggested gifts.
- Personalized Explanations: Provides AI-generated reasons why a gift is suitable.
- Creative Visualization: Generates a Christmas-themed image using DALL·E 3, showcasing the suggested gifts.

Requirements

- Python: The tool is written in Python, requiring a Python installation.
- Python Libraries: csv and re for data processing, openai and os for OpenAI API integration.
- OpenAI API Key: A valid OpenAI API key is required for AI services access.
- CSV File: A CSV file containing personal interest data, generated by the Chrome extension described in the README.

Usage

1. Dependency Installation: Install necessary libraries via pip.
2. API Key Setup: Provide the OpenAI API key to the program.
3. Data Loading: Enter the location and details of the CSV file generated by the Chrome extension.
4. Gift Idea Generation: The program will process the data and use OpenAI's API for gift suggestions.
5. Purchase Links and Explanations: The program will provide Amazon links for suggested gifts and explain why they are suitable.
6. Gift Visualization: A Christmas image with the gifts will be generated using DALL·E 3.
7. Cost Calculation: A cost calculation will be generated based on the tokens used in execution.

Security and Privacy

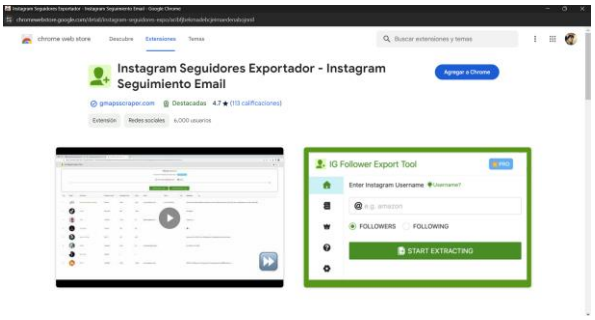
- The OpenAI API key should be handled securely and not exposed in the source code. The code requests it directly from the user.

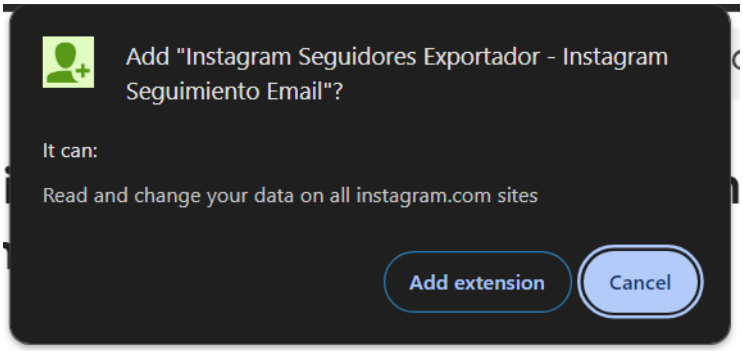
Contributions and Support

- Contributions are welcome through GitHub.
- Currently, there is no long-term support.

Installing the Chrome extension

https://ig-follower-extractor.gmapsscraper.com/





Program Execution Images

```
[3]  import csv
     import re
     import openai
     import os
     import unicodedata

     # Requesting relevant user information
     csv_filename = input("Enter the CSV file name (including the .csv extension): ")
     gift_recipient_name = input("Enter the name of the person you are giving the gift to: ")
     recipient_gender = input("Enter the gender of the person: ")
     recipient_age = input("Enter the approximate age of the person: ")
     person_data = "The person is " + recipient_age + " years old and their gender is " + recipient_gender
```

```
Enter the CSV file name (including the .csv extension): IG-Follower-Email-Scraper_1702104246421.csv
Enter the name of the person you are giving the gift to: Paola
Enter the gender of the person: Female
Enter the approximate age of the person: 22
```

```
[4]  # Requesting the user to enter the API key
     openai_api_key = input("Enter your OpenAI API key: ")
     os.environ["OPENAI_API_KEY"] = openai_api_key
```

```
Enter your OpenAI API key: [redacted]
```

```
[5]  # Read the CSV file, limit to a maximum of 1000 users, and clean data for AI readability
     def read_and_clean_csv(file_name, limit=1000):
         followed_users = []
         with open(file_name, mode='r', encoding='utf-8') as csvfile:
             reader = csv.reader(csvfile)
             for row in reader:
                 if row and len(followed_users) < limit:  # Check if the row is not empty and the limit is not reached
                     username = row[4]  # Assuming usernames are in the fifth column (index 4)
                     clean_username = re.sub(r'[0-9]', '', username)  # Remove numbers
                     clean_username = clean_username.replace('_', ' ').replace('.', ' ')  # Replace '_' and '.' with spaces
                     followed_users.append(clean_username)
                 else:
                     break  # Break the loop if the limit is reached
         return followed_users
```

```
[6]  followed_users = read_and_clean_csv(csv_filename)
     def generate_gift_recommendations(followed_users, openai_api_key):
         openai.api_key = openai_api_key

         # Create a prompt including followed interests and request gift recommendations
         prompt = "Based on the following interests and followed accounts: {}\n".format(", ".join(followed_users))
         prompt += "Please suggest 3 affordable and easy-to-find Christmas gifts. I only require short titles, as I will search for the products on Amazon."
         prompt += "Also, please consider the following factors of gender and approximate age: {}\n".format(", ".join(person_data))

         try:
             response = openai.ChatCompletion.create(
                 model="gpt-4",  # Make sure this is the correct model
                 messages=[
                     {"role": "system", "content": "Your system prompt here"},
                     {"role": "user", "content": prompt}
                 ]
             )
             return response.choices[0].message['content'].strip().split('\n')
         except Exception as e:
             print(f"Error in generating recommendations: {e}")
             return []

     # Store the gift recommendations
     gifts = generate_gift_recommendations(followed_users, openai_api_key)
```

```
[7]  # Remove accents for correct Amazon link generation
     def remove_accents(text):
         return unicodedata.normalize('NFKD', text).encode('ASCII', 'ignore').decode('ASCII')

     # Generate Amazon Mexico search links
     def generate_amazon_link(search_query):
         # Remove quotes and other unwanted characters
         clean_search = remove_accents(search_query.replace('"', '').replace('.', '').replace(',', ''))
         # Split words and join with '+'
         words = clean_search.split()
         amazon_url = "https://www.amazon.com.mx/s?k=" + '+'.join(words)
         return amazon_url

     # Store the links
     amazon_links = [generate_amazon_link(gift) for gift in gifts]

     # Introductory text for the gift list
     print(f"Given the list of followed accounts you provided for {gift_recipient_name}, I believe these gifts would be liked: ")

     for gift, link in zip(gifts, amazon_links):
         print(f"Gift: {gift}, Amazon Link: {link}")
```

```
Given the list of followed accounts you provided for Paola, I believe these gifts would be liked:
Gift: 1. A Set of Aromatherapy Candles, Amazon Link: https://www.amazon.com.mx/s?k=1+A+Set+of+Aromatherapy+Candles
Gift: 2. Minimalist Jewelry Pieces, Amazon Link: https://www.amazon.com.mx/s?k=2+Minimalist+Jewelry+Pieces
Gift: 3. Skincare Gift Box Set, Amazon Link: https://www.amazon.com.mx/s?k=3+Skincare+Gift+Box+Set
```

```
[9]  def generate_explanations(gifts, gift_recipient_name):
         # Simplify the prompt to avoid exceeding character limit
         prompt = f"Explain why you think the following gifts are suitable for {gift_recipient_name} based on the data analyzed from their Instagram followings, who has varied interests:\n"
         for gift in gifts:
             prompt += f"- {gift}\n"
         prompt += "Explanation:"

         try:
             response = openai.ChatCompletion.create(
                 model="gpt-4",
                 messages=[
                     {"role": "system", "content": "You are a helpful assistant."},
                     {"role": "user", "content": prompt}
                 ]
             )
             return response.choices[0].message["content"]
         except Exception as e:
             print(f"Error generating explanations: {e}")
             return ""

     # Example of using the function
     explanations = generate_explanations(gifts, gift_recipient_name)
     print("The explanation for these gifts is:\n", explanations)
```

```
The explanation for these gifts is:
 Based on analysis of Paola's Instagram followings:

1. A Set of Aromatherapy Candles: Paola follows several wellness and home decor pages. These candles not only promote relaxation but also enhance the atmosphere of the room. Thus, a se

2. Minimalist Jewelry Pieces: The Instagram pages followed by Paola suggest a keen interest in fashion, specifically with a minimalist style. Therefore, giving her minimalist jewelry p

3. Skincare Gift Box Set: Paola follows multiple skincare influencers and beauty brands on Instagram. This indicates an evident interest in skincare and beauty routines. Therefore, a s
```

```
[10] # Generate the image with Dall-E 3
     def generate_dalle_image(description, openai_api_key):
         openai.api_key = openai_api_key

         try:
             response = openai.Image.create(
                 model="dall-e-3",  # DALL·E 3 model
                 prompt=description,
                 n=1,  # Number of images to generate
                 size="1024x1024"  # Image size
             )
             return response.data[0]['url']  # URL of the generated image
         except Exception as e:
             print(f"Error generating the image: {e}")
             return ""

     # Print the image
     description = f"A Christmas scene with a decorated tree and the three suggested gifts, all visible. Which are: {gifts}"
     image_url = generate_dalle_image(description, openai_api_key)
     print("Generated image URL:", image_url)

     Generated image URL: https://oaidalleapiprodscus.blob.core.windows.net/private/org-xiWjKf2e54NggRLrsfih5JFQ/user-cfoGGhomQXBmh3hOlZhgp0dr/img-u0ETBU3VCFgAwLVtwIzJBHIl.png?st=2023-12-
```

```
[11] # Example of cost calculation considering the length of ChatGPT responses and a DALL-E image

     # Estimated rates
     PRICE_PER_CHATGPT_TOKEN = 0.00002  # Estimated price per token for ChatGPT
     PRICE_PER_DALLE_IMAGE = 0.02       # Price per image generated by DALL·E

     # Token estimation: Assuming each token is approximately 4 characters
     num_tokens_gifts = sum(len(gift) for gift in gifts) / 4
     num_tokens_amazon = sum(len(link) for link in amazon_links) / 4
     num_tokens_explanation = len("The explanation for these gifts is: " + explanations) / 4

     # Calculate the total cost of tokens
     token_cost = (num_tokens_gifts + num_tokens_amazon + num_tokens_explanation) * PRICE_PER_CHATGPT_TOKEN

     # Calculate the total cost including a DALL·E image
     total_cost = token_cost + PRICE_PER_DALLE_IMAGE

     print(f"The total cost in US dollars for this execution was: ${total_cost:.5f}")


     The total cost in US dollars for this execution was: $0.02657
```