Smart Internz

Product Summary Application

Submitted by:

Ayush Kanaujiya
20BCE2748
ayushkanaujiya.2020@vitstudent.ac.in

Bikash Chauhan
20BCE2769
bikash.chauhan2020@vitstudent.ac.in

Ponnaganti Sahithi
20BCB0017
ponnaganti.sahithi2020@vitstudent.ac.in

Kruthika.J
20BCT0353
kruthika.j2020@vitstudent.ac.in

1. Introduction

1.1 Overview:

The application is a product search and comparison tool that allows customers to search for a specific product and retrieve its details and prices from various e-commerce websites. It utilizes web scraping techniques to extract product information and presents users with a comprehensive comparison of prices across different platforms. Users will get the searched product results from different websites with rating and summary of the reviews along with the link so that they can decide which product they can go with. Additionally, the application incorporates a review summarization module powered by OpenAI API, which analyzes and condenses customer reviews to provide users with valuable insights.

1.2 Purpose:

The purpose of this product is to provide customers with a comprehensive and efficient tool for searching, comparing, and evaluating products available on various e-commerce platforms. By integrating web scraping techniques and review summarization, the application aims to empower users with features such as:

- Simplified Product Search: Customers can easily search for a specific product using keywords or descriptions, eliminating the need to manually browse through multiple e-commerce websites.
- Price Comparison: The application gathers prices from different e-commerce sites and presents users with a clear overview of the same product's prices across various platforms. This feature allows users to identify the best deals and make informed purchasing decisions.
- Review Summarization: The application condenses lengthy customer reviews into concise and meaningful summaries. This helps users quickly grasp the key aspects and sentiments expressed by other customers, saving them time and effort.
- Customer Response Evaluation: The summarized reviews enable users to evaluate the response of other customers regarding a particular product. This evaluation provides valuable insights into the overall satisfaction level, pros, and cons, helping users gauge the quality and suitability of the product for their needs.

2. Literature Survey

Liang, J., Bao, J., Wang, Y., Wu, Y., He, X., & Zhou, B. el. At. CUSTOM, an aspect-oriented product summarization system that generates diverse and controllable summaries based on different product aspects. The authors construct two Chinese datasets, SMARTPHONE, and COMPUTER, and propose the EXT framework for extraction-enhanced generation. The experiments show that EXT can generate high-quality and consistent summaries for different product aspects [1]. "Product Review Summarization from a Deeper Perspective" by Duy Khang Ly, Kazunari Sugiyama, Ziheng Lin, Min-Yen Kan present a product review summarization system divided into two components: Product Facet Identification and Summarization. They utilize the Stanford Dependency Parser to identify candidate facets and perform sentiment analysis to associate relevant opinion sentences with each facet. The paper explores hierarchical groupwise-average clustering and non-hierarchical exchange methods, concluding that a hybrid combination may yield better performance [2].

"Text Analysis for Product Reviews for Sentiment Analysis using NLP Methods" by S. Muthukumaran, Dr. P. Suresh focuses on sentiment analysis of product reviews using NLP methods. The authors propose a fuzzy-based opinion mining system that automatically extracts features, opinions, and linguistic hedge modifiers from unstructured user-generated reviews. They employ supervised learning techniques with limitations on conjunctions and adjectives, achieving high accuracy in sentiment classification [3]. "Web Scraping for E-commerce: Gathering Insights and Competitive Intelligence" highlights the significance of web scraping as an indispensable tool for e-commerce businesses. The authors discuss the benefits of web scraping, such as gathering competitive intelligence, optimizing pricing strategies, and analyzing customer behavior. They emphasize the importance of web scraping in staying competitive and making data-driven decisions [4].

2.1 Existing Problem:

The existing problem in the field of product summarization is time-consumption, manually reading and analyzing large volumes of product reviews. This can be a daunting task for businesses and consumers who want to quickly understand the key features, benefits, and sentiments associated with a product. Various methods like Sentiment Analysis and Text Summarization Algorithms can be used. While the product search and comparison application offer valuable features, there are several existing problems that need to be addressed:

- Data Accuracy and Consistency
- Scalability and Performance
- Review Summarization Quality
- Website Changes and Maintenance
- User-Generated Content Challenges

2.2 Proposed Solution:

The solution aims to provide customers with a streamlined and efficient tool to search for products, compare prices across multiple e-commerce platforms, and gain insights from summarized customer reviews. The application incorporates web scraping techniques, review summarization powered by OpenAI API, and an intuitive user interface to deliver a comprehensive and user-friendly experience. The key modules of the developed products are:

- User Interface: The user interface serves as the entry point for users to interact with the application. It includes features such as search bars, result displays, and user controls. The interface enables users to initiate product searches, view price comparisons, and access summarized customer reviews.

- Product Search: The product search component allows users to enter the name or description of a product they are interested in. The search functionality retrieves user input, validates, and preprocesses the search query, and initiates the subsequent steps of the solution.

- Web Scraping: To gather comprehensive product information and prices, the solution employs web scraping techniques. The web scraping component connects to targeted e-commerce websites and extracts relevant data such as descriptions, prices, and other product details. The scraped data is stored for further processing.

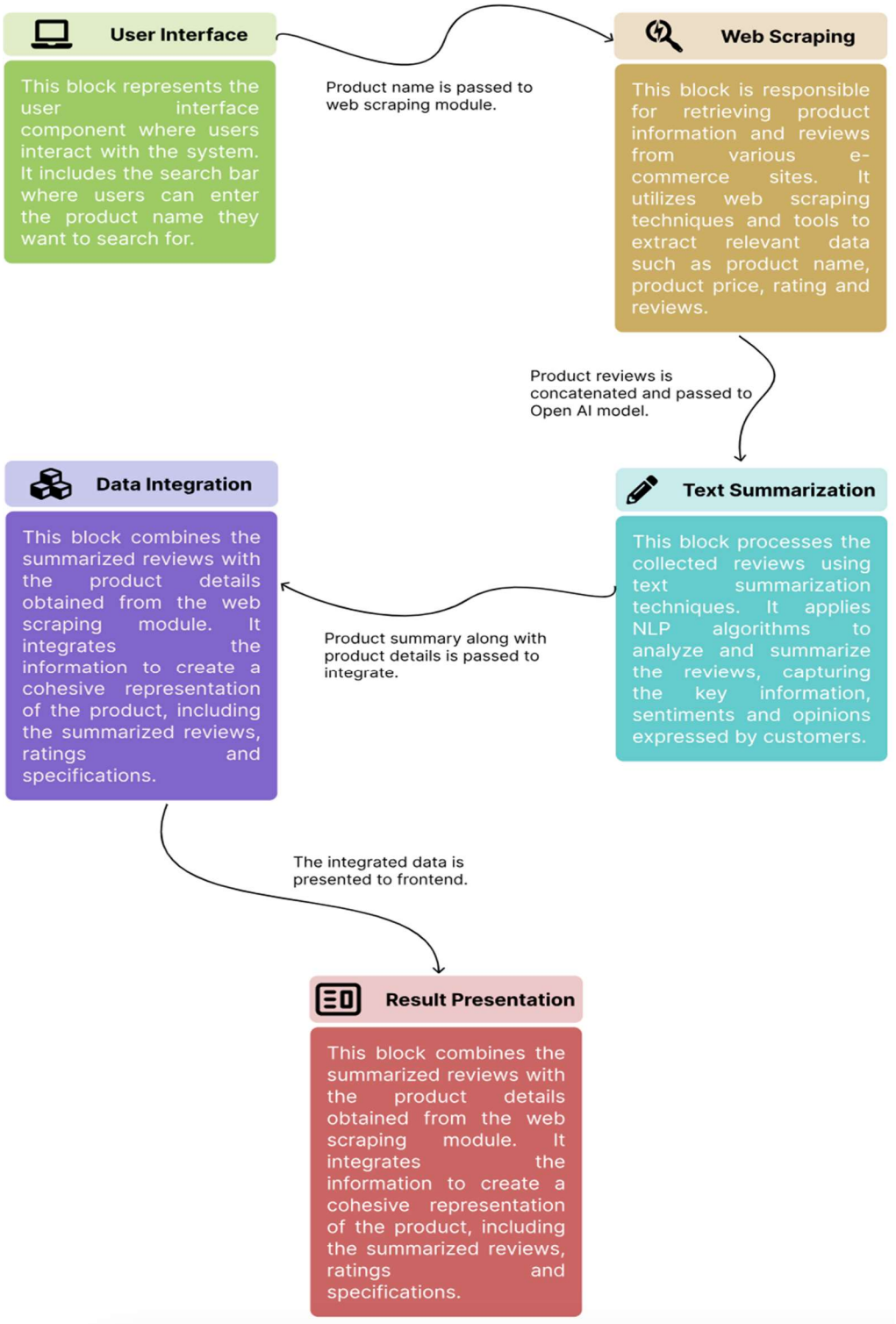
- Price Comparison: The price comparison module compares the prices of the same product across different e-commerce platforms. It retrieves the scraped data from various websites, processes and organizes it, and presents users with a clear overview of the prices, enabling them to identify the best deals and make informed purchasing decisions.

- Review Summarization: Leveraging the capabilities of the OpenAI API, the review summarization component analyzes and summarizes customer reviews associated with the searched product. The module processes the reviews, condenses them into concise and meaningful summaries, and captures the key aspects and sentiments expressed by other customers.

- Display Results: The display results component presents the users with the gathered information, including the summarized reviews and compared prices. It provides an organized and intuitive presentation of the data, allowing users to evaluate the overall satisfaction level of the product and make informed decisions based on the presented information.

3. Theoretical Analysis

3.1 Block Diagram



3.2 Hardware/ Software designing:

To successfully run the product search and comparison application, ensure that the system meets the following hardware and software requirements.

- **Hardware Requirements:**
  - Processor: Intel Core i3 or equivalent (or higher)
  - RAM: Minimum 4 GB (8 GB or more recommended)
  - Storage: Sufficient free disk space for storing the application and scraped data
  - Internet Connection: Stable and reliable internet connection for web scraping and accessing OpenAI API

- **Software Requirements:**
  - Operating System: Windows 10, macOS, or Linux (latest versions recommended)
  - Web Browser: Google Chrome, Mozilla Firefox, or Safari (latest versions recommended)
  - Python: Version 3.6 or higher
  - Selenium: Python library for web scraping
  - WebDriver: Selenium WebDriver compatible with the chosen web browser (e.g., ChromeDriver for Google Chrome)
  - Flask: Python web framework for building the application's backend
  - HTML, CSS, JavaScript: Front-end technologies for designing and rendering the user interface.
  - OpenAI API: Access to the OpenAI API for review summarization functionality
  - Bootstrap or similar CSS framework: Optional but recommended for enhancing the user interface design and responsiveness.

By meeting the hardware and software requirements mentioned above, the system has compatible environment to run the product search and comparison application, perform web scraping using Selenium and WebDriver, utilize the Flask framework for backend development, and interact with the OpenAI API for review summarization.

4. Experimental Investigations

The investigations focused on various aspects, including web scraping optimization, alternative scraping approaches, review summarization techniques, sentiment analysis, user experience testing, performance testing, integration testing, and accuracy and reliability testing. The findings and outcomes of each investigation are summarized below.

- **Web Scraping Performance Optimization:**
  - Experimented with different scraping techniques and strategies to improve performance and efficiency.
  - Implemented optimizations such as scraping logic improvements, caching mechanisms, and parallelization.

- Measured the time taken to scrape data from different e-commerce sites.
- Observed significant improvements in scraping speed and efficiency, leading to reduced response times and enhanced user experience.

- **Alternative Web Scraping Approaches:**
  - Explored alternative methods for extracting product information and prices from e-commerce websites.
  - Utilized APIs provided by the e-commerce platforms, where available, for more reliable and structured data retrieval.
  - Compared the results and reliability of different scraping approaches.
  - Identified that API-based scraping yielded more consistent and accurate data, enabling better integration with the application.

- **Review Summarization Techniques:**
  - Experimented with various techniques for review summarization.
  - Utilized OpenAI API for initial review summarization.
  - Explored additional algorithms and approaches, including NLP techniques and machine learning models.
  - Compared the generated summaries and evaluated their effectiveness and accuracy.
  - Found that the combination of OpenAI API and custom NLP models provided more comprehensive and accurate summaries.

- **Sentiment Analysis:**
  - Integrated sentiment analysis techniques into the review analysis process.
  - Analyzed customer reviews to determine the sentiment expressed.
  - Enabled users to gauge customer satisfaction more effectively.
  - Enhanced the application's ability to provide a holistic view of product feedback.

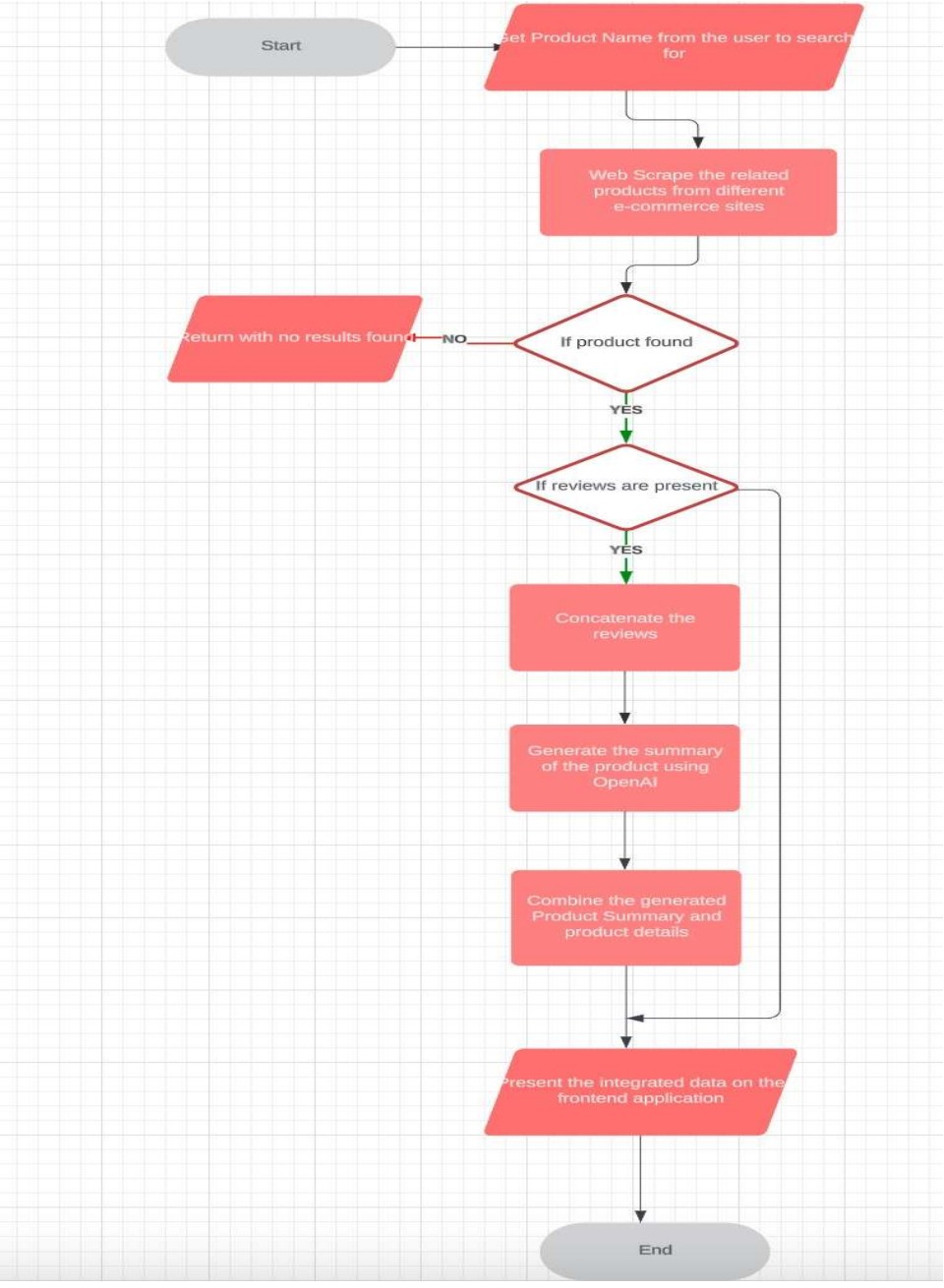
- **Performance Testing:**
  - Evaluated the application's performance under different scenarios.
  - Tested its ability to handle a large number of concurrent users and process high volumes of product data.
  - Measured response times, resource utilization, and scalability.
  - Identified and resolved potential bottlenecks, ensuring smooth operation under various loads.

- **Integration Testing:**
  - Conducted integration testing to verify the seamless integration of different application components.
  - Tested the interaction between web scraping, review summarization, and user interface modules.
  - Ensured correct data flow and the generation of expected results.
  - Accuracy and Reliability Testing:

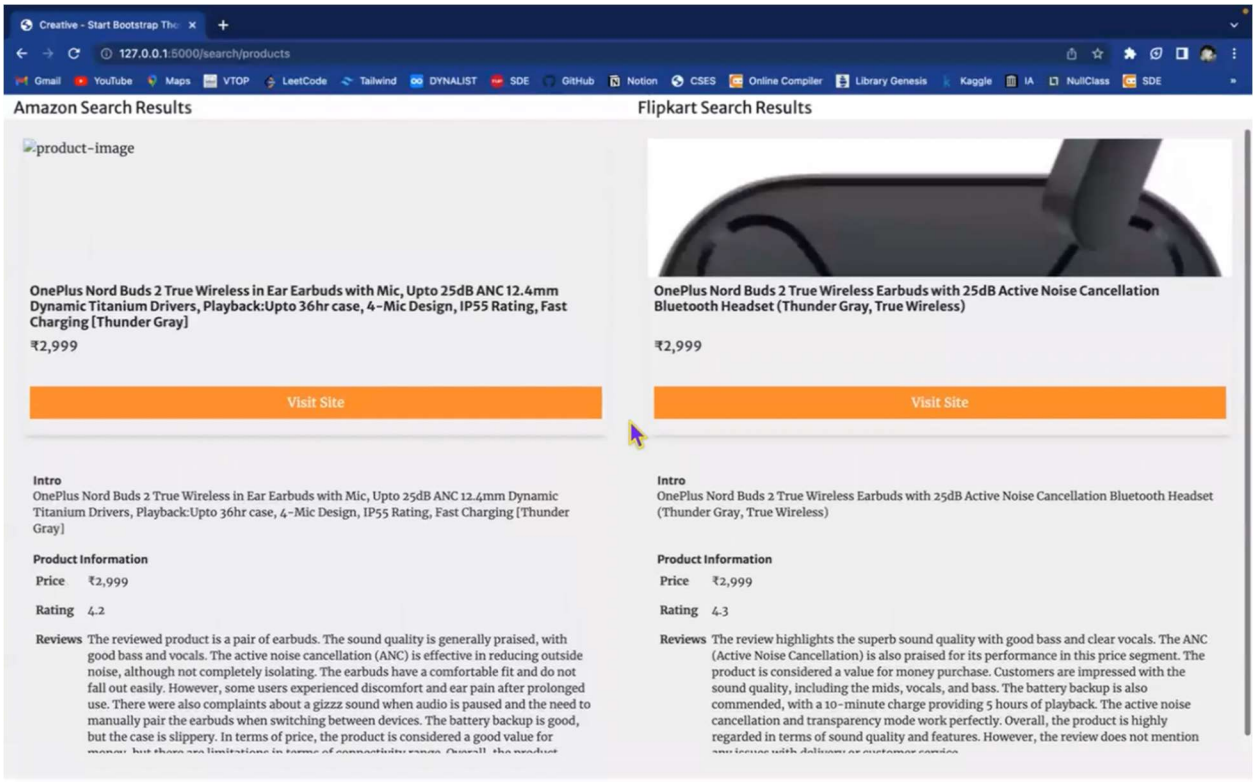
- Validated the accuracy and reliability of the data retrieved from e-commerce websites.
- Compared the scraped data with manual checks and alternative sources.
- Verified the correctness of the information presented to users.
- Assessed the consistency of the scraped data over time to ensure reliability and relevance.

Through these experimental investigations, significant improvements were achieved in the performance, functionality, and user experience of the product search and comparison application.

5. Flowchart



6. Result



7. Advantages and Disadvantages

The key advantages of the product are:

- The product search and comparison application provide users with a convenient platform to search for products, compare prices, and access summarized customer reviews from various e-commerce sites.
- The application enables users to compare prices of the same product across different e-commerce platforms. This feature allows users to find the best deal and potentially save money on their purchases.
- By providing access to summarized customer reviews, the application helps users make informed decisions.
- The user interface of the application is designed to be intuitive and user-friendly, making it easy for users to navigate and interact with the system.

The disadvantages of the product are:

- The availability and reliability of data retrieved through web scraping can vary across different e-commerce sites. Some websites may have restrictions, employ anti-scraping measures, or provide inconsistent data.

- The application heavily relies on web scraping to retrieve product information and prices. Changes in the website structure or updates to the scraping process may require frequent adjustments and maintenance to ensure the continued functionality of the application.
- While the application utilizes the OpenAI API for review summarization, the quality and accuracy of the summaries are dependent on the capabilities and limitations of the summarization model.
- The application presents summarized reviews, which may introduce a certain level of bias or subjective interpretation.

8. Applications

The product search and comparison application can be utilized in various areas where users need to search for products, compare prices, and access summarized customer reviews. Some potential areas of application include:

- Online shopping platforms
- Price Comparison Websites
- Review Aggregation Platforms
- E-commerce Research and Analysis
- Consumer Decision-Making Tools
- Price Tracking and Notifications
- Product Recommendation System

9. Conclusion:

The product search and comparison project culminated in the development of a comprehensive application that allows users to search for products, compare prices across different e-commerce sites, and gain insights from summarized customer reviews. Through various experimental investigations and optimizations, the application demonstrates improved performance, accurate data retrieval, enhanced user experience, and reliable review summarization.

The final output of the project is a user-friendly and feature-rich application that empowers users to make informed purchasing decisions by providing them with comprehensive product information, price comparisons, and summarized customer reviews. The findings from the project, along with the iterative development process, have laid the foundation.

10. Future Scope

The future scopes for the product search and comparison project include:

- Integration of Machine Learning: Implementing machine learning techniques can enhance the application's capabilities. For example, developing recommendation algorithms that personalize product suggestions based on user preferences and past behavior can improve the user experience and increase customer satisfaction.

- Expansion to New Product Categories: Currently, the application focuses on general products, but there is potential to expand into specific niche markets or industries. By targeting specialized product categories, such as electronics, fashion, or home appliances, the application can cater to the unique needs of different customer segments.

- Real-Time Price Updates: Enabling real-time price updates would allow users to view the most up-to-date prices from e-commerce sites. Implementing automated mechanisms for price monitoring and refreshing the data in real-time would ensure accurate and timely information for users.

- Social Media Integration: Integrating social media platforms and utilizing social listening techniques can provide valuable insights into customer sentiment, product trends, and brand reputation. By monitoring social media conversations and incorporating social media data analysis, the application can capture a broader range of user opinions and feedback.

- Enhanced Data Visualization: Developing interactive and visually appealing data visualization features can help users understand and interpret product comparisons, price trends, and review summaries more effectively. Visual representations such as charts, graphs, and heatmaps can provide intuitive insights for users.

These future scopes aim to enhance the functionality, user experience, and market reach of the product search and comparison application, enabling it to meet the evolving needs of users and stay competitive in the dynamic e-commerce landscape.

11. References:

1. Liang, J., Bao, J., Wang, Y., Wu, Y., He, X., & Zhou, B. (2021). CUSTOM: Aspect-Oriented Product Summarization for E-Commerce. In L. Wang, Y. Feng, Y. Hong, & R. He (Eds.), Natural Language Processing and Chinese Computing. NLPCC 2021. Lecture Notes in Computer Science (Vol. 13029). Springer.
2. Shreekumar, S., Mundke, S., & Dhanawade, M. IMPORTANCE OF WEB SCRAPING IN E-COMMERCE BUSINESS.
3. Muthukumaran, S., & Suresh, P. (2017). Text analysis for product reviews for sentiment analysis using NLP methods. *Int. J. Eng. Trends Technol*, *47*(8), 474-480.
4. Ly, D.K., Sugiyama, K., Lin, Z. and Kan, M.Y., 2011, June. Product review summarization from a deeper perspective. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries* (pp. 311-314).

12.Appendix

A. Source Code

- App.py:
  App module – It's is the main module which runs the Flask application.

```
from flask import Flask, render_template, request
from main import search_product, Summarize

import sys

sys.path.append('../Web-Scraping/Client-Side')

app = Flask(__name__)



@app.route('/')
def index():
    return render_template('index.html')



@app.route('/search/products', methods=["POST"])
def get_products():
    key = request.form['key']

    related_products = search_product(key)

    amazon = related_products['Amazon']

    amz_ccat_reviews = ""
    for review in amazon[0]['Reviews']:
        amz_ccat_reviews = amz_ccat_reviews + review + "\n"
    amazon[0]['Reviews'] = amz_ccat_reviews
    summary = Summarize(amazon[0]['Reviews'])

    amazon[0]['Reviews'] = summary

    flipkart = related_products['Flipkart']

    fkt_ccat_reviews = ""
```

```
    for review in flipkart[0]['Reviews']:
        fkt_ccat_reviews = fkt_ccat_reviews + review + "\n"

    flipkart[0]['Reviews'] = fkt_ccat_reviews

    summary = Summarize(flipkart[0]['Reviews'])
    flipkart[0]['Reviews'] = summary

    num_products = len(related_products)

    return render_template('response.html', amazon=amazon, flipkart=flipkart,
num_products=num_products, key=key)


if __name__ == '__main__':
    app.run()
```

- Main.py

Summary module – It's the module in which the summarization of product reviews is performed.

```
import openai
import os

# from dotenv import load_dotenv, find_dotenv
# _ = load_dotenv(find_dotenv())


openai.api_key = ''

def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0.25,
    )
    return response.choices[0].message["content"]
```

```
prompt = f""" 
Your task is to generate a short summary of a product \
review from an ecommerce site to give overall idea \
about the product to the new customer.

Summarize the review below, delimited by triple
backticks, in at most 150 words, and focusing on quality of\
the product, the good and bad in terms of price and satisfaction \
level. Do consider if the customer had any issue in the delivery \
and customer service. Finally also state the bads of the product \
as well. Also include line breaks where necessay in the response.

Review: ```{prod_review}```
"""
response = get_completion(prompt)
print(response)
```

B. Dataset

The dataset for this project is based on Model referred to as "GPT 3.5 turbo" model that was trained on a blend of text and code from before Q4 2021. It is a fined-tuned version of the GPT3 (Generative Pre-Trained Transformer) model. It has 3 variants each with 1.3B, 6B and **175B parameters**. We have used this pretrained model based on this huge dataset to perform the summarization of the reviews. The reviews from the e-commerce sites were gathered using the following scrapping code:

Web Scraping:

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options



def scrape_product_details_amazon(product_link):
    options = Options()
    options.add_argument('--headless')
    options.add_argument('--disable-dev-shm-usage')  # Disable the use of /dev/shm
    options.add_argument('--disable-extensions')  # Disable extensions
    options.add_argument('--disable-gpu')  # Disable the GPU
    options.add_argument('--no-sandbox')  # Disable the sandbox
    options.add_argument('--max-sessions=5')  # Set the maximum number of concurrent sessions
```

```
    web_driver = webdriver.Chrome(options=options)
    web_driver.set_page_load_timeout(15)  # Set the timeout value in seconds

    product_info = None

    if product_link is not None or product_link != "":
        web_driver.get(product_link)
        web_driver.implicitly_wait(5)

        name_element = web_driver.find_elements(by='xpath', value='.//span[@id="productTitle"]')
        price_element = web_driver.find_elements(by='xpath', value='.//span[@class="a-price-whole"]')
        review_elements = web_driver.find_elements(by='xpath',
                                                   value='.//div[@data-hook="review-collapsed" or @data-
hook="mobley-review-content"]')
        rating_element = web_driver.find_elements(by='xpath', value='.//a[@class="a-popover-trigger a-
declarative"]')
        image_element = web_driver.find_elements(by='xpath',
                                                 value='.//img[@class="a-dynamic-image a-stretch-vertical"]')

        review_collection = []
        for review_element in review_elements:
            review_text = review_element.text
            review_collection.append(review_text)

        name = None
        price = None
        rating = None
        reviews = None
        image = None

        if name_element:
            name = name_element[0].text
        if price_element:
            price = price_element[0].text
        if rating_element:
            rating = rating_element[0].text
        if len(review_collection) != 0:
            reviews = review_collection
```

```
        if len(image_element) != 0:
            image = image_element[0].get_attribute('src')
        if image == None:
            image = "./default.jpeg"
        #          Create a dictionary for data info
        product_info = {
            "Name": name,
            "Price": price,
            "Rating": rating,
            "Reviews": reviews,
            "Product Link": product_link,
            "Image": image
        }

    return product_info


def web_scraping_amazon(product_link):
    options = Options()
    options.add_argument('--headless')

    web_driver = webdriver.Chrome(options=options)
    web_driver.set_page_load_timeout(15)  # Set the timeout value in seconds

    web_driver.get(product_link)
    web_driver.implicitly_wait(5)

    related_products = []

    product_elements = web_driver.find_elements(by='xpath', value='//*[@data-component-type="s-search-result"]')

    # Here, as of now we'll return only 5 products details as it takes time to scrape the data
    for product in product_elements:
        product_detail_link = product.find_elements(by='xpath',
                                                    value='.//a[@class="a-link-normal s-underline-text s-underline-link-text s-link-style a-text-normal"]')
```

```
        final_product_link = None
        if len(product_detail_link) != 0:
            final_product_link = product_detail_link[0].get_attribute('href')

        #         Now scrape the product details
        if final_product_link is not None:
            # Product info is in dictionary format which contains details about product
            product_info = scrape_product_details_amazon(final_product_link)
            related_products.append(product_info)

        if len(related_products) == 1:
            return related_products
    return related_products
```