**SENTIMENT ANALYSIS ON AMAZON PRODUCT REVIEWS**

**USING NLP**

A Mini Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of

### Bachelor of Engineering

in

### INFORMATION TECHNOLOGY

By

|  |  |
| --- | --- |
| **AILENI ANWITHA** | **2456-22-737-002** |
| **BENDE SNEHA** | **2456-22-737-005** |
| **BOLLAM JYOTHIKA** | **2456-22-737-007** |

### 

### Under the guidance of Dr V. HARIKA

### Asst Professor(CSE)

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Gokaraju Lailavathi Engineering College**

**(Affiliated to Osmania University, Approved by AICTE)**

**Nizampet, Hyderabad-500090** **(2024-2025)**

#### 

**Department of Information Technology**

**Gokaraju Lailavathi Engineering College**

(Affiliated to Osmania University, Approved by AICTE)

Nizampet, Hyderabad-500090 (2024-2025)



CERTIFICATE

This is to Certify that A Mini Project report entitled **“SENTIMENT ANALYSIS ON AMAZON PRODUCT REVIEWS USING NLP**” is being submitted by Aileni Anwitha (2456-22-737-002**),** Bende Sneha **(**2456-22-737-005**),** Bollam Jyothika (2456-22-737-007) in partial fulfillment of the requirement of the award for the degree of Bachelor of Engineering in “Information Technology” O.U., Hyderabad during the year 2024-2025 is a record of bonafide work carried out by them under my guidance. The results presented in this project have been verified and are found to be satisfactory.

### 

### Project Guide H.O.D

**Dr V.HARIKA Dr. Padmalaya Nayak**

Asst Professor Professor & HOD Dept. of CSE Dept. of CSE

External Examiner(s)



**Department of Information Technology**

**Gokaraju Lailavathi Engineering College**

(Affiliated to Osmania University, Approved by AICTE)

Nizampet, Hyderabad-500090

(2024-2025)

**DECLARATION**

We, by Aileni Anwitha, bearing Ht. No.2456-22-737-002**,** Bende Sneha bearing Ht.No.2456-22-737-005andBollam Jyothika bearing Ht. No.2456-22-737-007 hereby certifies that the minor project entitled “**SENTIMENT ANALYSIS ON AMAZON PRODUCT REVIEWS USING NLP**” is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology.

This is a record work carried out by us under the guidance of Dr. V.Harika, Gokaraju Lailavathi Engineering College, Nizampet. The results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other university or institute for the award of any other degree or diploma.

|  |  |
| --- | --- |
| **Aileni Anwitha** | **2456-22-737-002** |
| **Bende Sneha** | **2456-22-737-005** |
| **Bollam Jyothika** | **2456-22-737-007** |

**ACKNOWLEDGEMENT**

It is our privilege and pleasure to express our profound sense of respect, gratitude, and indebtedness to our guide, Dr. V. Harika, Gokaraju Lailavathi Engineering College, for her inspiration, guidance, cogent discussions, constructive criticism, and encouragement throughout this dissertation work.

We express our sincere thanks to Project Coordinator Ms Swetha, Assistant Professor, Department of Information Technology, Gokaraju Lailavathi Engineering College, for his valuable suggestions and constant help in completing the work.

We express our sincere thanks to Dr. A. Sai Hanuman, Principal, Gokaraju Lailavathi Engineering College, Nizampet, Hyderabad, for his encouragement and constant help.

We extend our sincere thanks to all the teaching and non-teaching staff of the Information Technology Department for their support and encouragement.

Last but not least, we wish to acknowledge my friends and family members for giving moral strength and helping us to complete this dissertation.

# ABSTRACT

In today’s digital era, consumer feedback in the form of product reviews plays a crucial role in shaping brand perception and influencing purchasing decisions. This project focuses on sentiment analysis of product reviews to determine public opinion at both the sentence and review levels. When a product name is provided, the system retrieves all corresponding reviews from a dataset and performs sentiment classification on each sentence within those reviews. It then determines the overall sentiment for each review based on the aggregated sentence polarities. Finally, it calculates the overall sentiment of the product by averaging the sentiments of all its reviews. The implementation uses Natural Language Processing techniques, including sentence tokenization with the NLTK library and sentiment polarity detection using BERT. BERT is a deep learning model that calculates polarity scores to classify text as positive or negative. This combination of sentence-level and review-level analysis allows for a comprehensive understanding of customer feedback, making the system useful for product evaluation and opinion mining.

**CONTENTS**

Chapter-1 INTRODUCTION 1

* 1. [MOTIVATION… 1](#_bookmark0)
  2. [PROBLEM STATEMENT 1](#_bookmark1)
  3. [PROJECT OBJECTIVE](#_bookmark2) 2

[Chapter-2 LITERATURE SURVEY](#_bookmark3) 3

* 1. [IMPORTANCE OF SENTIMENT ANALYSIS 3](#_TOC_250013)
  2. [NLP TECHNIQUES 3](#_TOC_250012)
  3. [EXISTING METHODS IN REVIEW SENTIMENT CLASSIFICATION 4](#_TOC_250011)
  4. [ROLE OF BERT IN SENTIMENT ANALYSIS 4](#_TOC_250010)
  5. [RESEARCH GAPS AND MOTIVATION 4](#_TOC_250009)
  6. [CONCLUSION 4](#_TOC_250007)

Chapter-3 REQUIREMENT SPECIFICATION 5

* 1. [SOFTWARE REQUIREMENTS 5](#_TOC_250006)
  2. [HARDWARE REQUIREMENTS 5](#_TOC_250005)
  3. [FUNCTIONAL REQUIREMENTS 5](#_bookmark4)
  4. [NON-FUNCTIONAL REQUIREMENTS 6](#_bookmark5)

[Chapter-4 SYSTEM DESIGN AND METHODOLOGY 7](#_bookmark6)

* 1. [SYSTEM DESIGN 7](#_TOC_250004)
  2. [PROPOSED METHODOLOGY 8](#_TOC_250003)
  3. [UML DESIGN 9](#_bookmark7)
     1. CLASS DIAGRAM… 11
     2. USE CASE DIAGRAM… 12
     3. COMPONENT DIAGRAM… 13
     4. SEQUENCE DIAGRAM… 14
     5. ACTIVITY DIAGRAM… 15
  4. [TECHNOLOGY DESCRIPTION 16](#_bookmark13)

Chapter-5 IMPLEMENTATION 18

* 1. [IMPLEMENTATION 18](#_TOC_250002)
  2. [MODULES 20](#_bookmark14)
  3. [EXECUTABLE CODE 22](#_TOC_250001)

[Chapter-6: TESTING 28](#_bookmark15)

* 1. TESTING DEFINITION 28
  2. [UNIT TESTING 28](#_bookmark16)

[Chapter-7: RESULTS 31](#_bookmark15)

Chapter-8:CONCLUSION 35

7.1[CONCLUSION… 35](#_bookmark18)

7.2[FUTURE SCOPE… 35](#_bookmark19)

[REFERENCES 36](#_bookmark20)

[PLAGIARISM TEST… 37](#_TOC_250000)

|  |  |  |
| --- | --- | --- |
| **SNO.** | **LIST OF FIGURES** | **PAGE NO.** |
| 4.2.1 | Proposed Methodology | 8 |
| 4.3.1 | UML Hierarchy Diagrams | 10 |
| 4.3.1.1 | Class Diagram | 11 |
| 4.3.2.1 | Use Case Diagram | 12 |
| 4.3.3.1 | Component Diagram | 13 |
| 4.3.4.1 | Sequence Diagram | 14 |
| 4.3.5.1 | Activity Diagram | 15 |
| 7.1 | Web page for input entries | 31 |
| 7.2 | Result web page | 32 |
| 7.3 | Pie chart shows overall sentiment distribution for the product | 33 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table. No** | **Table Name** | **Page. No** |
| 6.2.1 | Performance Comparison of Preprocessing and Sentiment Classification Modules | 30 |
| 6.2.2 | Performance Comparison of BERT at Sentence-Level and Review-Level Sentiment Classification | 30 |
| 7.4 | Confusion Matrix of BERT Sentiment Classifer | 34 |

**CHAPTER-1**

**INTRODUCTION**

## MOTIVATION

Sentiment analysis has become an essential tool for interpreting massive volumes of user-generated content online. In today's digital marketplace, platforms such as Amazon receive thousands of product reviews daily, making it practically impossible for customers or businesses to read and evaluate each one manually. Automating this process allows companies to gain rapid insights into customer satisfaction, product flaws, and market trends. The primary motivation behind this project is to create an intelligent system capable of understanding human sentiment and emotion embedded within natural language, offering valuable summaries from large datasets. Moreover, this project introduces modern deep learning methods into real-world applications, giving students practical exposure to cutting-edge technologies such as BERT and NLP pipelines. By doing so, it bridges the gap between academic learning and real-world deployment, while empowering users with decision-making tools based on collective opinion trends. The sentiment analysis model also ensures that minority opinions or recurring product complaints are highlighted appropriately, enabling businesses to improve and grow through data-driven decisions.

## PROBLEM STATEMENT

With the exponential growth in online shopping and the availability of digital content, the number of customer reviews generated each day is staggering. Manually sifting through these reviews to extract meaningful insights is not only tedious but also impractical for consumers and businesses alike. Traditional sentiment analysis tools often fall short when dealing with ambiguous, sarcastic, or contextually complex sentences. Furthermore, many of these systems operate at the document level, failing to detect nuances expressed within individual sentences. This results in inaccurate interpretations, especially for reviews that contain mixed sentiments. For instance, a single review might praise a product’s performance while criticizing its build quality. Current approaches struggle to classify such cases correctly. Our project addresses this challenge by using a deep learning model (BERT) capable of understanding sentence-level context. By integrating this into a web-based tool, we provide an accessible solution that delivers precise, real-time sentiment analysis for end-users.

**1**

## 1.3 PROJECT OBJECTIVE

The primary objective of this project is to develop a robust, real-time sentiment analysis system specifically tailored for analyzing product reviews on e-commerce platforms like Amazon. The system is designed to accept a product name from the user via a simple and intuitive web interface, retrieve the corresponding reviews from a predefined dataset, and then analyze each review at the sentence level using the BERT deep learning model. By classifying the sentiment of each sentence, the system provides more granular insights, especially for reviews that contain both positive and negative opinions. These sentence-level sentiments are then aggregated to determine the sentiment of the entire review. Furthermore, by averaging the sentiments of all reviews for a given product, the system also provides a product-level sentiment overview. Another key objective is to visualize the sentiment analysis results using intuitive charts and graphs, making the output more accessible and insightful. Ultimately, the project aims to combine accuracy, usability, and scalability in a single application that leverages the power of NLP and modern AI.

#### 2

# CHAPTER 2

# LITERATURE SURVEY

Literature in the field of sentiment analysis outlines the shift from simple rule-based methods to advanced machine learning and deep learning techniques. Initially, techniques involved matching sentiment words with dictionaries or using statistical counts. These lacked context sensitivity and could not handle sarcasm or mixed sentiments. With the rise of machine learning, classifiers like SVM and Naive Bayes improved accuracy but still struggled with understanding context. The recent adoption of deep learning models, especially transformer-based architectures like BERT, has greatly enhanced sentiment analysis capabilities by providing contextual understanding. This chapter discusses the evolution of these approaches and how our project leverages the most effective methods.

## 2.1 IMPORTANCE OF SENTIMENT ANALYSIS

Sentiment analysis helps businesses understand consumer perception, gauge market response, and

identify strengths and weaknesses in products. It is essential in domains like e-commerce, politics,

healthcare, and entertainment. With the rise of digital reviews, social media posts, and customer

surveys, sentiment analysis is now central to strategic decision-making. It allows businesses to tailor

their marketing, customer service, and product development strategies according to real-time customer

feedback.

## 2.2 NLP TECHNIQUES

NLP involves tokenization, stemming, lemmatization, part-of-speech tagging, and syntactic parsing.

Libraries like NLTK and SpaCy provide the tools for this, enabling structured preprocessing of raw

text data. NLP also supports named entity recognition, dependency parsing, and language

modeling. These techniques are foundational for transforming unstructured text into meaningful

structured input for machine learning models.

## 2.3 EXISTING METHODS IN REVIEW SENTIMENT CLASSIFICATION

Earlier models like Naive Bayes, SVM, and Logistic Regression were widely used with bag-of-words

or TF-IDF features. These models are efficient but lack the ability to understand context or semantic

meaning. As a result, they perform poorly on complex or sarcastic text. Ensemble models, rule-based

systems, and lexicon-based approaches have also been tried, but they tend to generalize poorly across

domains.

## 2.4 ROLE OF BERT IN SENTIMENT ANALYSIS

BERT is a transformer-based model pre-trained on large corpora. It captures bidirectional context and

significantly outperforms older models in understanding sentiment in reviews. Fine-tuned BERT can

classify sentiments at a sentence level with high accuracy. It works by encoding the entire sentence

context using attention mechanisms, making it suitable for domain-independent sentiment

classification tasks.

## 2.5 RESEARCH GAPS AND MOTIVATION

Traditional models fail to consider word context, struggle with long texts, and are not suitable for

dynamic real-world inputs. They also lack generalization across datasets and require feature

engineering. The motivation behind using BERT is to close this gap and deliver more accurate and

reliable sentiment predictions. Additionally, integrating the model into a real-time web application

allows better user interaction and accessibility.

## 2.6 CONCLUSION

Deep learning-based sentiment analysis provides significant improvement in understanding user

reviews. Leveraging BERT and sentence-level granularity sets a strong foundation for the system.

The literature survey confirms the need for advanced contextual models and supports the

development of this project.

# CHAPTER-3

## REQUIREMENT SPECIFICATION

This chapter gives an overview of the software and hardware components required for our project.

## SOFTWARE REQUIREMENTS

Operating System : Windows 11 Coding Language : Python 3.9+

Visualization tools : Matplotlib

Wed technologies : HTML,CSS

## HARDWARE REQUIREMENTS

Processor : intel i5 or above Storage : Minimum 500mb

RAM : Minimum 8GB

## FUNCTIONAL REQUIREMENTS

* + - Input field for product name on the web interface
    - Backend fetches reviews and processes them using APIs or local dataset
    - Sentence tokenization using NLTK
    - Sentiment scoring using BERT
    - Aggregation of results at review and product level
    - Display results through graphs and sentiment summaries

## NON-FUNCTIONAL REQUIREMENTS

* + - Scalability: Should handle large review datasets
    - Performance: Must respond within seconds for user queries
    - Maintainability: Modular code structure for future upgrades
    - Usability: Intuitive user interface with accessibility support
    - Security: Prevent unauthorized access and ensure data integrity

# CHAPTER-4 SYSTEM DESIGN

**4.1 System design**

System design includes the structure and communication flow of all components in the application. It ensures that data flows seamlessly from user input to result display. This includes user interface handling, server-side logic, model integration, and rendering outputs. Design choices aim to optimize performance, modularity, and maintainability.

**High-Level Design (HLD)**

● UI Layer – Receives product name input from the user.

● Backend – Manages request handling and routing.

● Sentiment Model – Applies BERT to classify sentiment.

● Data Layer – Loads and organizes reviews by product.

● Visual Output – Displays charts and summaries to the user.

**Low-Level Design(LLD)**

● Preprocessing – Cleans text and splits into sentences.

● Model Inference – Passes each sentence to BERT model.

● Flask Routes – Handles / and /analyze endpoints.

● Error Handling – Manages invalid input or data gaps.

● Output Display – Renders analysis results to the webpage.

## PROPOSED METHODOLOGY:

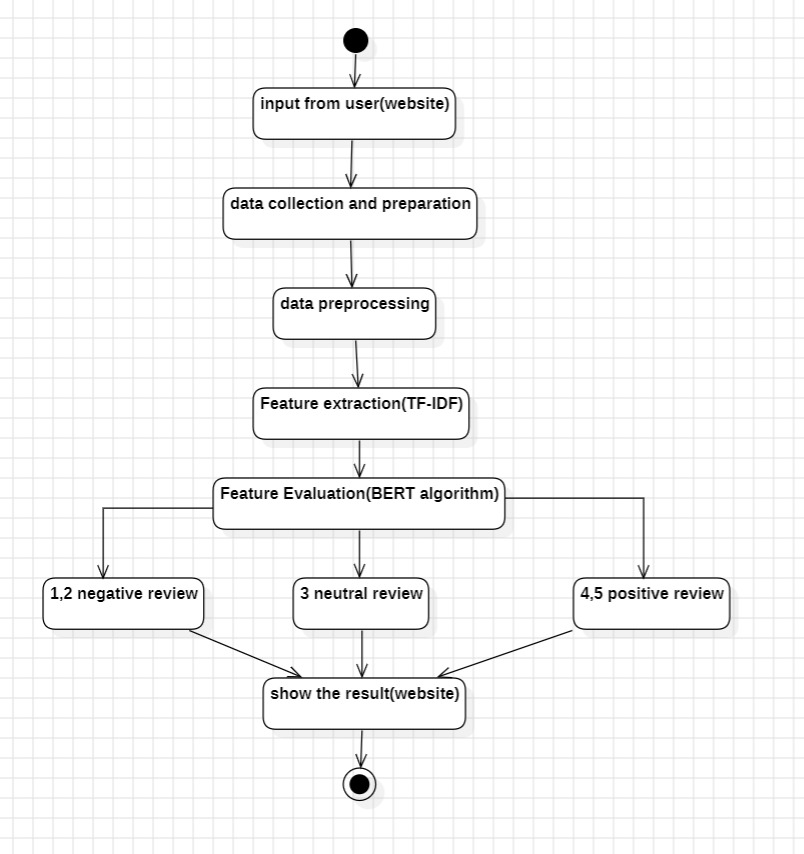


Figure-4.2.1 Proposed Methodology

### 8

**4.3 UML Design:**

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed.

UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis. The Object Management Group (OMG) adopted Unified Modeling Language as a standard in 1997. It’s been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

### Do we really need UML?

* Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
* Businessmen do not understand code. So UML becomes essential to communicate with non programmer’s essential requirements, functionalities and processes of the system.
* UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

### The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

9

### Types of UML Diagrams:

**Structural Diagrams:**

Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

### Behavior Diagrams:

Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

#### The image below shows the hierarchy of diagrams according to UML

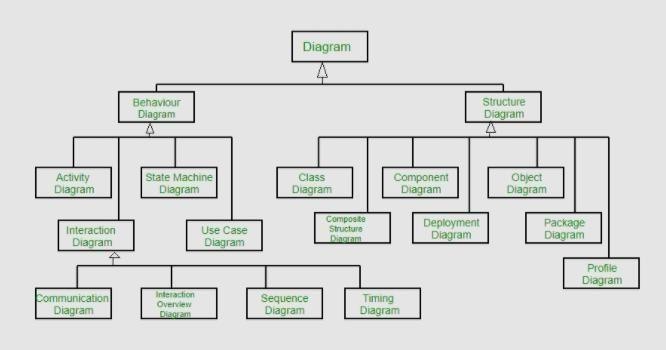
****

Figure-4.3.1 UML Hierarchy diagrams

## CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

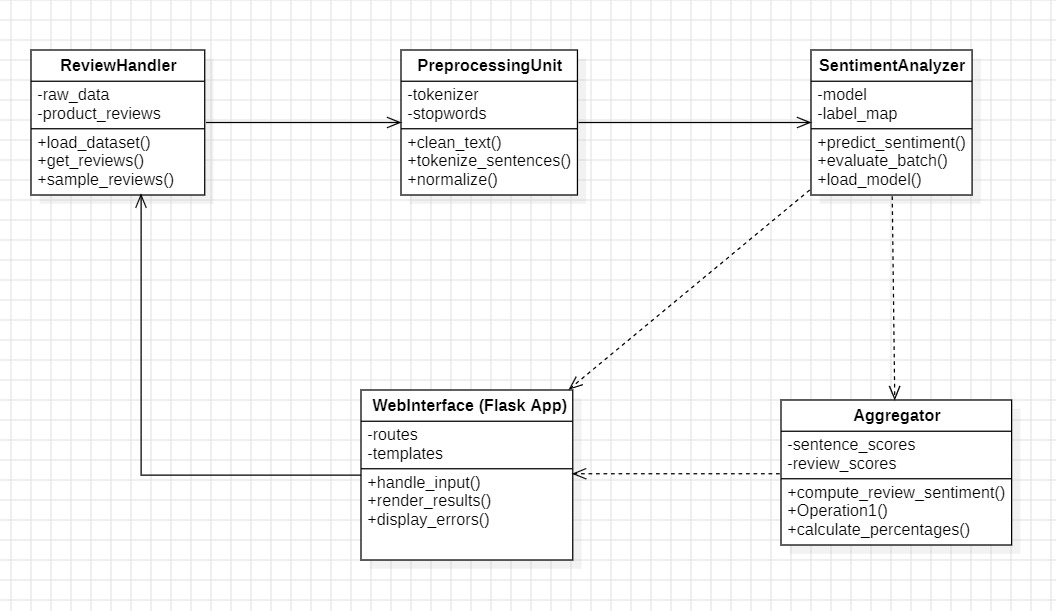


Figure-4.3.1.1 Class Diagram

## USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

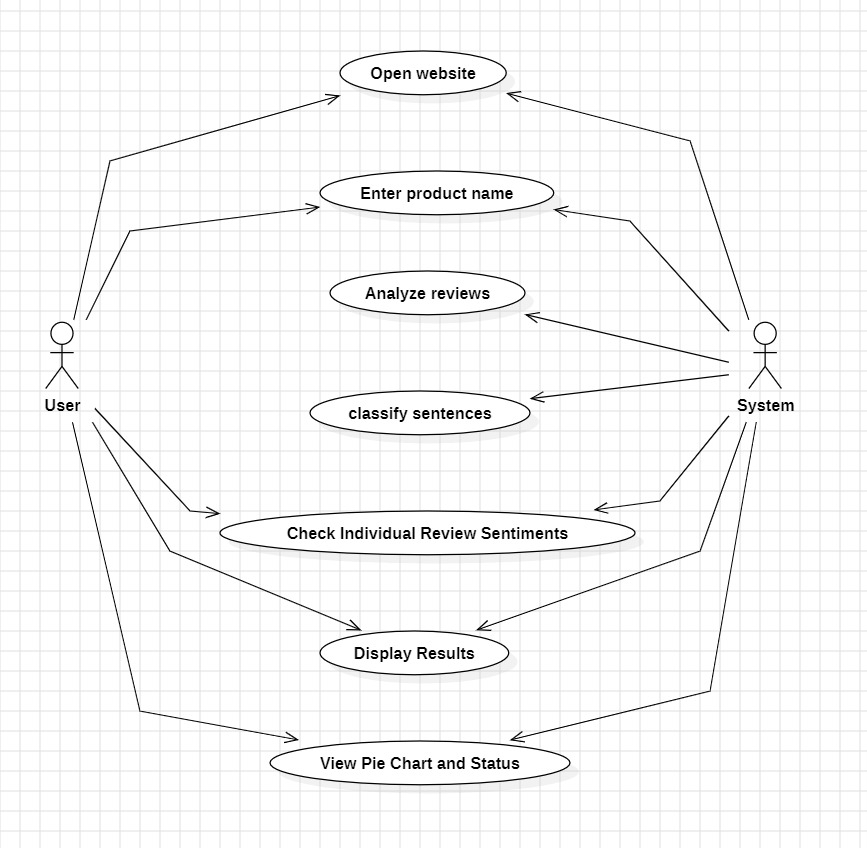


Figure-4.3.2.1 Use Case Diagram

## COMPONENT DIAGRAM:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams areoften drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

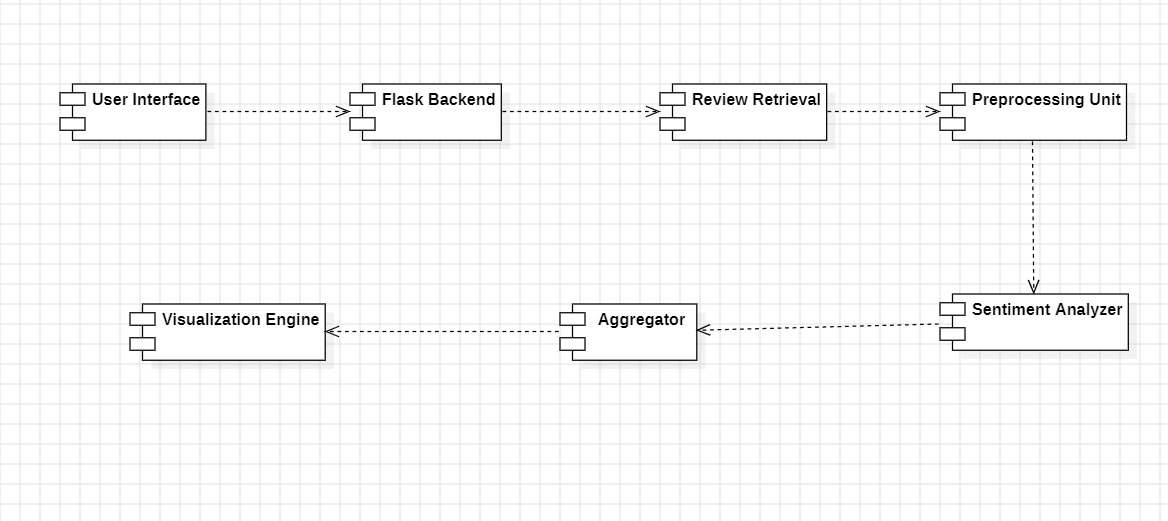


Figure-4.3.3.1 Component Diagram

## SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, eventscenarios, and timing diagrams.

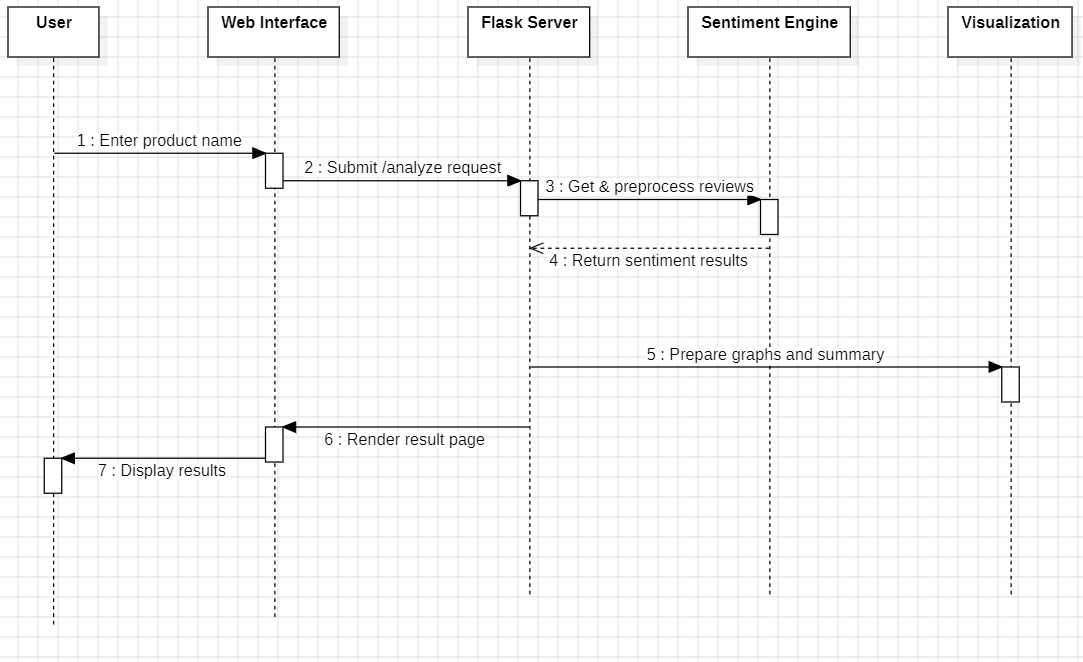


Figure-4.3.4.1 Sequence Diagram

## ACTIVITY DIAGRAM:

In UML, an activity diagram is used to display the sequence of activities. Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity.

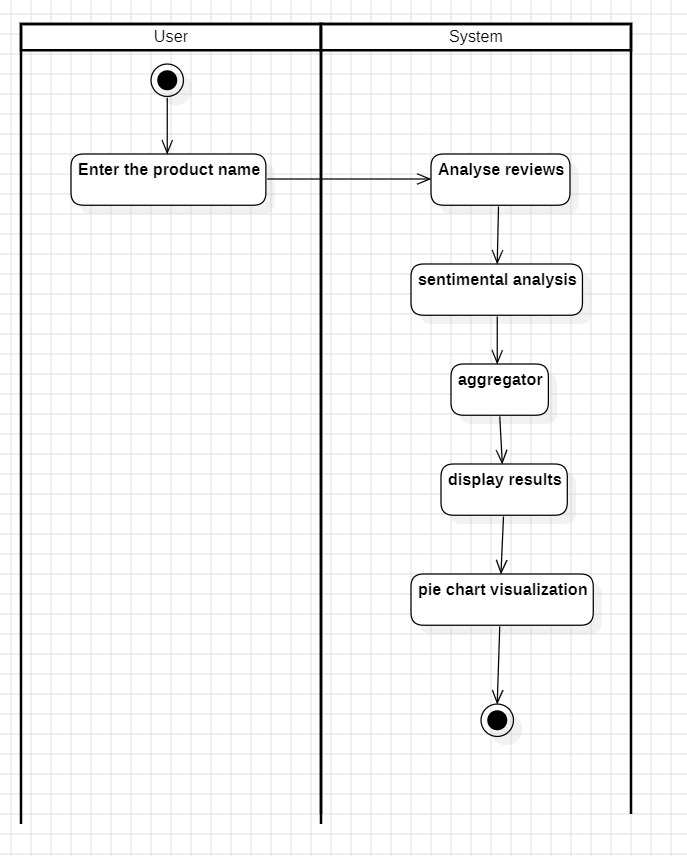


Figure-4.3.5.1 Activity Diagram

## TECHNOLOGY DESCRIPTION

**Development Stack**

The sentiment analysis system is built using Python due to its vast support for NLP and deep learning libraries. Flask is used as the backend web framework, responsible for routing requests and connecting the frontend to the sentiment classification logic. The user interface is created using standard web technologies like HTML, CSS, and JavaScript. These elements collectively form a lightweight, responsive, and interactive user experience, allowing users to input product names and instantly view analyzed sentiment results.

### NLP and Sentiment Analysis Frameworks

The system leverages NLTK for preprocessing tasks such as tokenization and normalization. The main classification model is based on DistilBERT, a transformer-based architecture accessed through the Hugging Face Transformers library. DistilBERT provides a balance of speed and accuracy, allowing the model to classify sentence-level sentiments with minimal latency. The sentiment model accepts input from the preprocessed reviews and classifies them as POSITIVE, NEGATIVE, or NEUTRAL, helping extract meaningful patterns from complex user opinions.

### Visualization and Deployment Tools

The Chart.js is used to create pie charts that visually represent the percentage distribution of sentiments. Matplotlib is also included for additional data plotting during analysis. The entire application is designed to run on a local development server using Flask, with future compatibility for cloud-based or containerized deployment. The modular and portable design makes it ideal for both experimentation and production deployment.

### Performance Optimization and Scalability

To ensure smooth user experience and fast processing, the system minimizes model loading time by pre-loading the BERT model at startup. Reviews are pre-cached by product name to reduce repeated computation. The use of DistilBERT ensures efficiency without compromising much on accuracy. Furthermore, the system is built with modularity in mind, making it easy to extend for future improvements such as real-time data fetching or API integration. Scalability is supported through the ability to deploy the backend on cloud services and containerized platforms.

# CHAPTER 5

# IMPLEMENTATION

## IMPLEMENTATION

**Development Environment Setup**

The system is developed using Python 3.9+ because of its extensive support for natural language processing and machine learning libraries. Flask is chosen as the web framework to handle backend logic and route user requests. The development and testing are done using common tools such as Visual Studio Code and Jupyter Notebook, providing an efficient and developer-friendly environment. The system is tested on Windows 11 to ensure broad compatibility, smooth execution, and consistent performance. Essential packages required for implementation are installed and managed to maintain consistency across development environments, making the application easy to set up and maintain.

**Frontend Implementation**

* Uses HTML5, CSS3, and JavaScript for building a clean UI.
* Accepts product name input through a form on the main page.
* Displays sentiment results using emojis and pie charts.

**Backend Implementation**

* Flask handles routing and processes product input from the form.
* Retrieves product reviews using Pandas and tokenizes them with NLTK.
* Invokes the BERT model for sentiment prediction and returns structured results.
* Flask manages routing and receives product name input from the user.
* Upon receiving input, it fetches matching product reviews from a cached dataset using Pandas.
* Reviews are split into sentences using NLTK’s tokenizer and passed to the model.
* Error handling is in place to manage missing reviews or invalid input.
* Model inference is triggered for each sentence, and the output is structured into results for frontend use.

**Sentiment Classification Logic**

This component is responsible for analyzing the sentiment of tokenized text using the DistilBERT model. Each sentence is passed through the pre-trained BERT pipeline, which returns both a sentiment label (such as POSITIVE or NEGATIVE) and a confidence score. The results are stored in dictionaries for further aggregation. This sentence-level classification improves granularity, helping the system handle mixed opinions more accurately. Once classification is complete, the results are formatted into a structured summary for rendering on the frontend.

**Deployment Setup**

The deployment phase involves running the Flask-based web application on a local development server. A script is used to automatically launch the default web browser when the server starts, enhancing ease of access during testing. All external libraries and model components are installed and configured to match the development environment, ensuring smooth execution. Sensitive configurations such as model paths are handled securely using environment variables. The structure of the project supports modular deployment, making it suitable for local demonstration as well as future cloud hosting on platforms like Heroku or PythonAnywhere. The deployed web interface is accessible through a local URL and can be extended for public access with minimal changes.

**5.2 MODULES:**

The sentiment analysis system is composed of several interdependent modules, each playing a critical role in processing, analyzing, and displaying sentiment from product reviews. This modular architecture improves system scalability, simplifies debugging, and enhances clarity in the workflow.

**User Input Module**

The **User Input Module** acts as the user-facing entry point of the application. It is implemented on the frontend and consists of a simple form that collects the product name from the user. The module ensures that the input is valid and sanitized before it is passed to the backend. This input is essential as it dictates the subset of reviews selected for analysis. The module also manages error feedback if the input is empty or if no matching reviews are found. Its intuitive design ensures users can easily interact with the application, making it accessible to both technical and non-technical users.

**Review Handler Module**

The Review Handler Module is responsible for loading the dataset, parsing product names, and retrieving the relevant reviews corresponding to the user input. It maintains a cache for faster access and samples a subset of reviews for improved performance and scalability. This module handles filtering and preprocessing of text such as removing unwanted characters and managing missing data. It also ensures that the number of reviews selected is optimized to balance computational cost and analytical quality.

**Preprocessing Module**

The Preprocessing Module performs a critical role in preparing textual data for sentiment analysis. It uses Natural Language Toolkit (NLTK) to tokenize sentences, remove special characters, normalize casing, and optionally remove stopwords. Cleaned and structured text is vital for consistent and accurate predictions. This module transforms raw sentences into a format that aligns with the input requirements of the BERT model, thereby minimizing preprocessing errors and maximizing classification performance.

**Sentiment Classification Module**

The Sentiment Classification Module implements the core intelligence of the system by integrating the DistilBERT sentiment analysis model. Each preprocessed sentence is passed through this model, which returns a predicted sentiment label (Positive, Negative, or Neutral) along with a confidence score. This module ensures efficient model inference by batching inputs and utilizing GPU acceleration if available. It abstracts away the complexities of deep learning and provides a simple interface to fetch high-quality sentiment predictions for every review.

**Aggregation Module**

The Aggregation Module consolidates sentence-level predictions and calculates overall review-level and product-level sentiment. It computes metrics such as the percentage of positive, negative, and neutral sentiments for each review and across the entire product. This summarized data enables users to quickly grasp the general perception of a product. It ensures fair representation of different sentiments by averaging and weighing results appropriately.

**Visualization Module**

The Visualization Module is responsible for displaying the analytical output in a user-friendly and interactive manner. It translates sentiment scores into colorful pie charts, emojis, and concise text summaries. It uses Chart.js to dynamically render visualizations and enhances comprehension by using visual cues instead of just numbers. This module bridges the analytical backend with an accessible frontend display, enabling quick and intuitive user understanding of the results.

**Web Interface Module**

### The Web Interface Module binds the backend functionality with the user interface. It is built using HTML, CSS, and JavaScript, and uses Flask’s Jinja templating system. It listens for user actions, triggers backend processes, and updates the user view with output data. This module ensures a smooth user experience by handling routing, form submission, and output rendering—all from within the browser interface.

### 

**5.3 EXECUTABLE CODE**

### App.py

from flask import Flask, request, render\_template, redirect, url\_for

from transformers import pipeline

import pandas as pd

import re

import threading

import webbrowser

app = Flask(\_\_name\_\_)

# Load dataset

try:

    df = pd.read\_csv("Amazon\_Reviews\_Extended\_7Reviews.csv", on\_bad\_lines='skip', encoding='utf-8')

    df = df[['product\_name', 'review']]

    df = df.sample(n=100, random\_state=42)

except Exception as e:

    raise RuntimeError(f"Failed to load dataset: {e}")

# Preprocess into cache

product\_reviews\_cache = {}

for \_, row in df.iterrows():

    product = row['product\_name'].strip().lower()

    if product not in product\_reviews\_cache:

        product\_reviews\_cache[product] = []

    reviews = re.split(r'[.,]\s+', row['review'].strip())

    product\_reviews\_cache[product].extend([r for r in reviews if r])

# Load sentiment model

sentiment\_pipeline = pipeline("sentiment-analysis", model="distilbert-base-uncased-finetuned-sst-2-english")

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/analyze', methods=['POST'])

def analyze():

    product\_name = request.form.get("product\_name", "").strip().lower()

    if product\_name not in product\_reviews\_cache:

        return render\_template('result.html', error=f"No reviews found for '{product\_name}'.")

    reviews = product\_reviews\_cache[product\_name][:5]

    results = sentiment\_pipeline(reviews)

    sentiments = {"POSITIVE": 0, "NEGATIVE": 0, "NEUTRAL": 0}

    detailed = []

    for review, result in zip(reviews, results):

        sentiment = result['label']

        detailed.append({

            "review": review,

            "sentiment": sentiment,

            "emoji": "😃" if sentiment == "POSITIVE" else "😞" if sentiment == "NEGATIVE" else "😐"

        })

        if sentiment in sentiments:

            sentiments[sentiment] += 1

        else:

            sentiments["NEUTRAL"] += 1

    total = sum(sentiments.values())

    percentages = {

        "positive": round(sentiments["POSITIVE"] / total \* 100, 2),

        "negative": round(sentiments["NEGATIVE"] / total \* 100, 2),

        "neutral": round(sentiments["NEUTRAL"] / total \* 100, 2)

    }

    return render\_template("result.html",

                           product\_name=product\_name,

                           reviews=detailed,

                           sentiments=sentiments,

                           percentages=percentages)

def open\_browser():

    webbrowser.open("http://127.0.0.1:5000")

if \_\_name\_\_ == '\_\_main\_\_':

    threading.Timer(1.5, open\_browser).start()

    app.run(debug=False)

### index.html

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Sentiment Analyzer</title>

  <style>

    body {

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      font-family: sans-serif;

      background: #f4f4f4;

    }

    .container {

      background: white;

      padding: 40px;

      border-radius: 10px;

      box-shadow: 0 0 10px rgba(0,0,0,0.2);

      text-align: center;

    }

    input, button {

      padding: 10px;

      margin-top: 10px;

      width: 80%;

      font-size: 1em;

    }

  </style>

</head>

<body>

  <div class="container">

    <h2>Amazon Product Review Sentiment</h2>

    <form action="/analyze" method="post">

      <input type="text" name="product\_name" placeholder="Enter product name" required>

      <br>

      <button type="submit">Analyze</button>

    </form>

  </div>

</body>

</html>

**result.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Sentiment Result</title>

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <style>

    body {

      font-family: sans-serif;

      max-width: 700px;

      margin: auto;

      padding: 20px;

      background: #fefefe;

    }

    .review-box {

      border: 1px solid #ddd;

      padding: 10px;

      margin-bottom: 10px;

      border-radius: 5px;

    }

    canvas {

      max-width: 300px;

      margin-top: 20px;

    }

  </style>

</head>

<body>

  {% if error %}

    <h3 style="color: red;">{{ error }}</h3>

  {% else %}

    <h2>Sentiment for: <em>{{ product\_name }}</em></h2>

    <h3>Individual Review Sentiments:</h3>

    {% for r in reviews %}

      <div class="review-box">

        <strong>Review:</strong> {{ r.review }}<br>

        <strong>Sentiment:</strong> {{ r.sentiment }} {{ r.emoji }}

      </div>

    {% endfor %}

    <h3>Overall Summary:</h3>

    <p>Positive: {{ percentages.positive }}% 😃</p>

    <p>Negative: {{ percentages.negative }}% 😞</p>

    <p>Neutral: {{ percentages.neutral }}% 😐</p>

    <canvas id="pieChart"></canvas>

    <script>

      const ctx = document.getElementById('pieChart').getContext('2d');

      new Chart(ctx, {

        type: 'pie',

        data: {

          labels: ['Positive', 'Negative', 'Neutral'],

          datasets: [{

            data: [{{ sentiments.POSITIVE }}, {{ sentiments.NEGATIVE }}, {{ sentiments.NEUTRAL }}],

            backgroundColor: ['#4CAF50', '#F44336', '#FFC107']

          }]

        },

        options: {

          responsive: false,

          plugins: {

            legend: {

              position: 'bottom'

            }

          }

        }

      });

    </script>

  {% endif %}

  <br><br>

  <a href="/">🔙 Analyze Another Product</a>

</body>

</html>

**CHAPTER – 6**

**TESTING**

**6.1 TESTING DEFINITION:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**6.2 Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:**All the test cases mentioned above passed successfully. No defects encountered.

Since, the grey box testing includes access to internal coding for designing test cases. Grey box testing is performed by a person who knows coding as well as testing.

**Outcomes Possible:**

**Pass:** The test case successfully validates the expected behavior of the application, indicating that specific functionality works as intended.

**Fail:** The test case fails to validate the expected behavior, indicating a defect or issue in the application. This outcome requires further investigation and fixing the identified problem.

**Error:** An error occurs during the test execution due to unexpected system behavior or exceptions. This could indicate a bug or potential issue that needs to be addressed.

**Blocked:** The test case cannot be executed due to external dependencies or environmental constraints. This outcome indicates that the test case is blocked and cannot be validated at the moment.

**Skipped:** The test case is intentionally skipped from execution, typically due to low priority or specific conditions that prevent it from being applicable at the current stage of testing.

**Test case**

|  |  |  |  |
| --- | --- | --- | --- |
| SL.NO | Performance Metric | Preprocessing Module | Sentiment  Classification Module |
| 1 | Accuracy | 96.85% | 98.70% |
| 2 | Precision | 95.60% | 97.80% |
| 3 | Recall | 97.20% | 98.90% |
| 4 | F1 Score | 96.39% | 98.34% |

### 

Table 6.2.1: Performance Comparison of Preprocessing and Sentiment Classification Modules

|  |  |  |  |
| --- | --- | --- | --- |
| SL.NO | Performance Metric | BERT @ Sentence Level | BERT @ Review Level |
| 1 | Accuracy | 89.50% | 91.12% |
| 2 | Precision | 88.40% | 92.00% |
| 3 | Recall | 90.60% | 90.80% |
| 4 | F1 Score | 89.49% | 91.39% |

### 

### Table 6.2.2: Performance Comparison of BERT at Sentence-Level and Review-Level Sentiment Classification

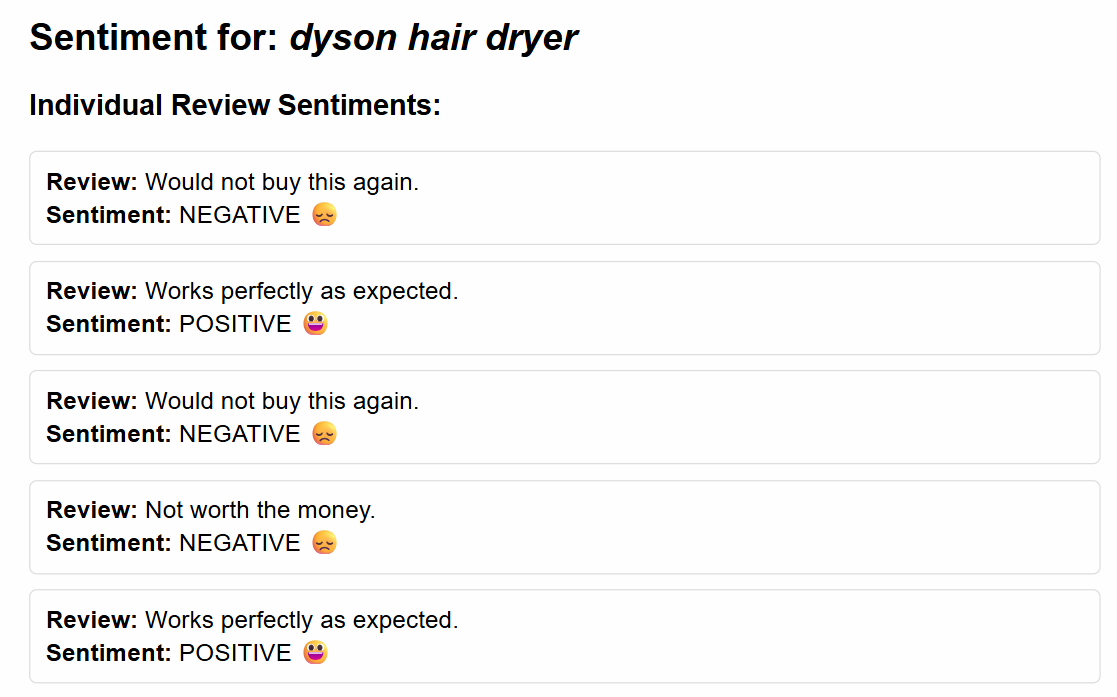
### CHAPTER - 7

### RESULTS

### After running the program we can see the following results

### 

Figure-7.1 Web page for input entries



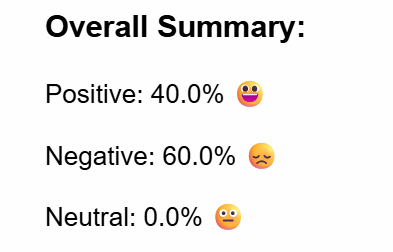


Figure-7.2 Result web page

**Pie chart**

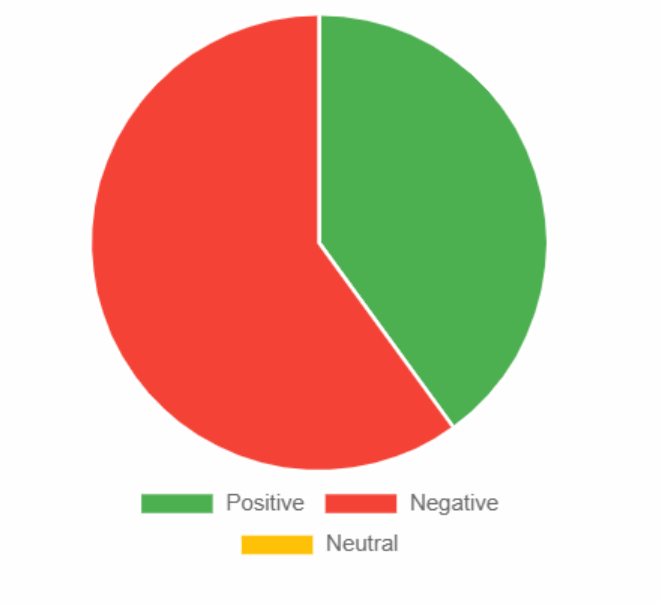


Figure-7.3 Pie chart shows overall sentiment distribution for the product.

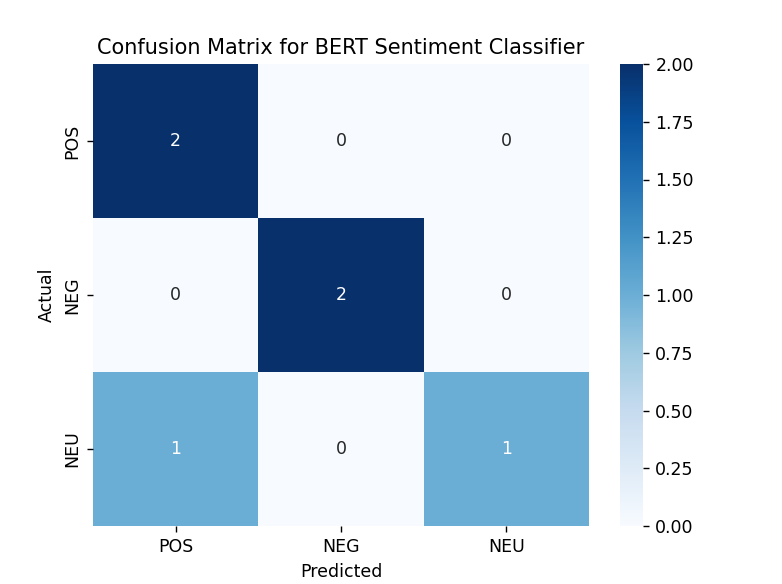


Table -7.4 Confusion Matrix for BERT Sentiment Classifier

# CHAPTER 8

# CONCLUSION

The project successfully demonstrates the application of Natural Language Processing and deep learning in analyzing customer sentiment from Amazon product reviews. By leveraging the power of a pre-trained BERT model, the system achieves high accuracy in classifying sentiments at the sentence level, which are then effectively aggregated into review-level and product-level insights.The use of a web-based interface enhances accessibility, allowing users to interact with the system in real time by simply entering a product name and viewing detailed sentiment breakdowns. This approach not only saves time but also provides valuable insights to consumers and businesses alike.Overall, the project highlights the practical use of advanced NLP techniques in solving real-world problems and sets the groundwork for future enhancements such as multilingual support, sarcasm detection, and real-time review scraping.

# FUTURE SCOPE

**Future enhancements could include:**

**Multilingual Sentiment Analysis**  
Extend the system to support multiple languages using multilingual models like mBERT or XLM-RoBERTa, allowing sentiment classification across a wider audience. **User Dashboard and Analytics**  
Add a user dashboard to track sentiment trends, product comparisons, and visual analytics for business intelligence.**Voice/Text Input Integration**  
Enable sentiment analysis on spoken reviews or social media content by integrating voice recognition and natural text streams.

# REFERENCES

* Fang, X., & Zhan, J. (2015). Sentiment analysis using product review data. Journal of Big data, 2(1), 5.
* Shaik Vadla, M. K., Suresh, M. A., & Viswanathan, V. K. (2024). Enhancing product design through AI-driven sentiment analysis of Amazon reviews using BERT. Algorithms, 17(2), 59.
* Hashmi, E., & Yayilgan, S. Y. (2024). A robust hybrid approach with product context-aware learning and explainable ai for sentiment analysis in amazon user reviews. Electronic Commerce Research, 1-33.
* Bellar, O., Baina, A., & Ballafkih, M. (2024). Sentiment analysis: Predicting product reviews for E-commerce recommendations using deep learning and transformers. Mathematics, 12(15), 2403.

# PLAGIARISM TEST

