Major Project-I Report
on

Product Review Analysis

Submitted in Partial fulfillment for the award of degree of Bachelor of Technology in Artificial Intelligence & Data Science

Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Submitted By:

Jyoti Kalambe (0131AD211028)

Nikita Rajput (0131AD211039)

Abdul Saboor (0131AD223D01)

Under the Guidance of

Professor

Dr. Ayonija Pathre

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Jai Narain College of Technology, Bhopal

Approved by AICTE New Delhi & Govt. of M.P.

Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Session: 2024 – 2025

JAI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

Approved by AICTE New Delhi & Govt. of M.P. & Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

CERTIFICATE

This is to certify that the work embodied in this Project, Dissertation Report entitled as **"Product Review Analysis"** being Submitted by **Jyoti Kalambe (0131AD211028) , Nikita Rajput (0131AD211039) and Abdul Saboor (0131AD223D01)** in partial fulfillment of the requirement for the award of **"Bachelor of Technology" in Artificial Intelligence & Data Science** discipline to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.) during the academic year 2024-25 is a record of bonafide piece of work, carried out under my supervision and guidance in the Department of Artificial Intelligence & Data Science, **Jai Narain College of Technology, Bhopal.**

.

Approved by

Guided by
Dr. Ayonija Pathre

Head of Department
Prof. Ravinder Tanwar

Dean, Academics
Dr. Vivek Dubey

Principal
Dr. Netra Pal Singh

CERTIFICATE OF APPROVAL

This Project **"Product Review Analysis"** being submitted by **Jyoti Kalambe (0131AD211028) , Nikita Rajput (0131AD211039) and Abdul Saboor (0131AD223D01)** has been examined by me & hereby approve for the partial fulfillment of the requirement for the award of **"Bachelor of Technology in Artificial Intelligence & Data Science",** for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but the Project only for the purpose for which it has been submitted.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Date: Date:

CANDIDATE DECLARATION

We hereby declare that the Project dissertation work presented in the report entitled as **"Product Review Analysis System"** submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Artificial Intelligence & Data Science** of Jai Narain College of Technology, Bhopal is an authentic record of our own work.

We have not submitted the part and partial of this report for the award of any other degree or diploma.

Jyoti Kalambe (0131AD211028)
Nikita Rajput (0131AD211039)
Abdul Saboor (0131AD223D01)

Date:

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Guided By:
Dr. Ayonija Pathre

ACKNOWLEDGMENT

We are heartily thankful to the **Jai Narain College of Technology** for providing us all the facilities and infrastructure to take our work to the final stage.

It is the constant supervision, moral support and proper guidance of our respected Principal **Prof. Dr. NetraPal Singh** and Dean, Academics **Prof. Dr. Vivek Dubey,** who motivated throughout the work.

We express a deep sense of gratitude and respect to our learned guide **Dr. Ayonija Pathre** Professor in the Department of Artificial Intelligence & Data Science, during all phases of ourwork. Without his enthusiasm and encouragement this dissertation would not have been completed. His valuable knowledge and innovative ideas helped us to take the work to the final stage. He has timely suggested actions and procedures to follow for which we are really grateful and thankful to him.

We express our gratitude to **Prof. Ravinder Tanwar** Head of Artificial Intelligence & Data Science. Department for providing all the facilities available in the department for his continuous support, advice, and encouragement during this work and also to help to extend our knowledge and proper guidelines.

Constant help, moral and financial support of our loving parents motivated us to complete the work. We express our heartfelt thanks to all our family members for their cooperation.

We really admire the fond support of our class-mates for their cooperation and constant help. It gives immense pleasure to acknowledge the encouragement and support extended by them. Last but not the least we are extremely thankful to all who have directly or indirectly helped us for the completion of the work.

.

Jyoti Kalambe (0131AD211028)
Nikita Rajput (0131AD211039)
Abdul Saboor (0131AD223D01)

Abstract

The project "Product Review Analysis" is a web-based application developed by the students of JNCT College to assist online shoppers in making informed purchasing decisions. In today's digital era, e-commerce platforms like Flipkart and Amazon offer millions of products, making it challenging for customers to choose the right one. Often, customers are overwhelmed by the sheer volume of options and are unable to decide whether a product meets their requirements. This project addresses this problem by providing an automated, data-driven analysis of product reviews to help users evaluate their options effectively.

Our application employs Beautiful Soup, a Python library for web scraping, to extract real-time data such as product reviews, ratings, and other relevant details from e-commerce websites. The collected data undergoes preprocessing and analysis using machine learning techniques, enabling the application to determine the overall sentiment of the reviews. The project also features an API system that allows third-party developers to access the functionality programmatically. Users can generate API keys to integrate this review analysis capability into their own systems, promoting scalability and wider adoption. The project is designed to be user-friendly, ensuring that both end-users and developers can benefit from its functionalities seamlessly.

By providing reliable and unbiased analysis, this project simplifies decision-making for customers and enhances their shopping experience. It demonstrates the practical application of web scraping, sentiment analysis, and API development in solving real-world problems. Future enhancements could include expanding support to additional platforms, refining sentiment analysis algorithms, and adding features like usage analytics for API users.

LIST OF TABLES

S. No.	Table of Contents	Page No.	Table
1	Data Dictionary	16	4.1

LIST OF FIGURES

S. No.	Table of Contents	Page No.	Figure
1	System Architecture Diagram	6	2.1
2	DFD Level 0	10	4.1
3	DFD Level 1	11	4.2
4	DFD Level 2	12	4.3
5	System Flow Diagram	13	4.4
6	ER -Diagram	14	4.5
7	Use Case Diagram	17	4.6

LIST OF GRAPHS

S. No.	Table of Contents	Page No.
1	Review % Distribution	30
2	Review Distribution	30
3	Model Class Probability Distribution	31

ABBREVIATIONS

API - Application Programming Interface
DB - Database
UI - User Interface
HTML - HyperText Markup Language
CSS - Cascading Style Sheets
JSON - JavaScript Object Notation
SQL - Structured Query Language
XML - Extensible Markup Language
NLP - Natural Language Processing
RAM - Random Access Memory
SMTP - Simple Mail Transfer Protocol
URL - Uniform Resource Locator
JWT - JSON Web Token
IDE - Integrated Development Environment
GUI - Graphical User Interface
HTTP - HyperText Transfer Protocol
HTTPS - HyperText Transfer Protocol Secure
WYSIWYG - What You See Is What You Get
CRUD - Create, Read, Update, Delete
CLI - Command Line Interface
XML - Extensible Markup Language
REST - Representational State Transfer
VADER - Valence Aware Dictionary and sEntiment Reasoner
ORM - Object-Relational Mapping
SSL - Secure Sockets Layer
SaaS - Software as a Service
CI/CD - Continuous Integration / Continuous Deployment
IDE - Integrated Development Environment
JSON - JavaScript Object Notation

INDEX

S. No.	Table of Contents	Page No.
1	**Introduction**	1
	Objective	1
	Problem Identification	2
	Proposed Solution	2
2	**Detailed Project Profile**	3
	System overview	3
	Scope	3
	Feasibility Study	4
	System architecture	5
3	**Software Requirement Specification**	6
	Purpose	6
	Scope	6
	Feasibility Study	6
	Hardware Requirement / Software Requirement	7
	Software Process Model Used	9
4	**System Documentation**	10
	DFD	10
	DFD level 0	10
	DFD level 1	11
	DFD level 2	12
	System Flow Chart	13
	ER- Diagram	14
	Data Dictionary	15
	Use Case	17
5	**User Manual**	18
	Introduction and Guidelines	18
	Screen Layouts and Description	20
	Output Reports	27
6	**Limitations**	32
7	**Future Enhancement**	34
8	**Conclusion**	35
9	**Bibliography**	36
11	**References**	37
12	**Appendix – I Source Code**	38

1. Introduction

The rise of online shopping has made it easier for consumers to access a wide variety of products on platforms like Flipkart and Amazon. However, with millions of options available, making informed purchasing decisions can be challenging. Product reviews and ratings play a crucial role, but manually analyzing this data is time-consuming. The "Product Review Analysis" project aims to solve this problem by providing an automated solution to evaluate product reviews.

The project uses Beautiful Soup to scrape real-time data, including product reviews and ratings, from e-commerce websites. This extracted data is processed through sentiment analysis to assess whether a product is viewed positively or negatively by consumers. Additionally, the project features an API that enables third-party developers to incorporate the review analysis functionality into their own applications, enhancing scalability and flexibility. By providing actionable insights from customer feedback, the project simplifies the decision-making process and improves the overall shopping experience for users

Objective

The primary objectives of the "Product Review Analysis" project are as follows:

- **Automate Review Analysis:** To develop a system that automates the process of collecting and analyzing product reviews from e-commerce platforms like Flipkart and Amazon, saving users time and effort in making informed purchasing decisions.

- **Sentiment Analysis:** To implement sentiment analysis techniques that evaluate customer reviews and provide an overall sentiment score, helping users understand the general perception of a product.

- **Web Scraping:** To use Beautiful Soup to extract real-time data from online shopping websites, enabling the system to stay up-to-date with the latest product information and reviews.

- **User-Friendly Interface:** To create a user-friendly web interface that allows consumers to easily input product URLs and receive detailed analysis results in a simple, digestible format.

- **API Development:** To develop an API that allows third-party developers to integrate the review analysis functionality into their own applications, expanding the project's reach and usability.

- **Enhance Consumer Decision-Making:** To provide valuable insights from product reviews that guide consumers in making better purchasing decisions based on real feedback from other users.

Problem identification

- **Overwhelming Choice:** E-commerce platforms like Flipkart and Amazon offer millions of products, making it difficult for consumers to choose the right one.

- **Confusing Reviews:** The vast number of reviews for each product often contains conflicting opinions, leaving users uncertain about the product's quality.

- **Time-Consuming Process:** Manually sifting through numerous reviews to make an informed decision is time-consuming and inefficient for consumers.

- **Lack of Automation:** There is a need for an automated system that can extract, analyze, and summarize product reviews in real-time, providing consumers with clear, unbiased insights.

- **Subjective Judgments:** Consumers often rely on their subjective interpretation of reviews, which may not always provide an accurate assessment of the product's quality.

Proposed Solution

To address the challenges faced by consumers in evaluating products, the "Product Review Analysis" project proposes a comprehensive, automated solution that combines Natural Language Processing (NLP), Machine Learning (ML), web development, and a robust database system. The key components of the solution are:

- **Web Scraping:** The system uses Beautiful Soup to scrape product reviews, ratings, and other relevant data from e-commerce platforms like Flipkart and Amazon.

- **Natural Language Processing (NLP):** NLP techniques process the reviews to extract sentiment, helping the system understand and analyze user feedback effectively.

- **Sentiment Analysis:** Reviews are classified as positive, negative, or neutral, offering a clear and accurate summary of a product's reception by users.

- **Machine Learning (ML):** ML models continuously learn from review data to improve sentiment classification accuracy and adapt to new trends.

- **Web Development with Flask:** The project is built using Flask, a lightweight web framework, to create a user-friendly interface where consumers can input product URLs and receive instant analysis results.

- **SQLite Database:** A local SQLite database is used to store user data, product review data, and analysis results, providing a simple and efficient way to manage and query the data.

- **API Integration:** An API is provided for third-party developers to integrate the review analysis functionality into their own applications, ensuring scalability and versatility.

- **Security:** Security measures, including user authentication and secure data handling, are implemented to protect user data and prevent unauthorized access.

- **User Feedback:** The system includes features for collecting user feedback, allowing continuous improvement based on real-world usage and enhancing the accuracy of the analysis.

This solution integrates web scraping, NLP, ML, web development, database management, and security to offer an automated, scalable, and secure platform for product review analysis, ultimately helping consumers make informed purchasing decisions while ensuring ease of use and protection of user data.

2. Detailed Project Profile

System Overview

The Product Review Analysis System is designed to help online shoppers quickly analyze and assess product reviews from popular e-commerce platforms like Amazon and Flipkart. The system automates the process of scraping product reviews, analyzing sentiment, and summarizing the key points to help consumers make more informed purchasing decisions. The system integrates web scraping, sentiment analysis, and a web interface for a seamless user experience.

Scope

- **Project Goal:**

  The goal of this project is to create a web scraping and sentiment analysis tool for online shopping websites (e.g., Flipkart and Amazon). The system will scrape product reviews, analyze the sentiment of the reviews, and provide insights based on the collected data.

- **Core Functionalities:**

  Web Scraping: Scrape data from online shopping platforms such as Flipkart and Amazon, focusing on product information and reviews.

  Sentiment Analysis: Analyze the sentiment of the reviews to determine if the product has a positive, negative, or neutral reception.

  User Interface: Provide a web-based interface where users can enter a product URL, initiate the scraping process, and view the sentiment analysis results.

  API Generation: Allow users to generate API keys for scraping products programmatically.

- **Boundaries:**

  In Scope: Scraping product reviews, conducting sentiment analysis, presenting results in a user-friendly web interface, and API key functionality.

  Out of Scope: The project will not cover scraping beyond Flipkart and Amazon, advanced machine learning models, or real-time data scraping on a large scale (e.g., handling millions of requests simultaneously).

Feasibility Study

- **Technical Feasibility:**

  The project is technically feasible because it uses Python for web scraping (via BeautifulSoup and requests) and Flask for the web interface.

  The sentiment analysis component can be implemented using basic NLP techniques (e.g., VADER or TextBlob).

The backend uses SQLite as the database, which is lightweight and suitable for this project's scale.

- **Operational Feasibility:**

  The system is designed for small to medium-scale use, so it will operate efficiently within this scope.The web interface is user-friendly and does not require advanced technical knowledge from users, ensuring smooth operation.

- **Economic Feasibility:**

  The project will be developed with open-source tools and frameworks like Flask, BeautifulSoup, and SQLite, which do not incur additional costs.
  The system does not require any expensive hardware or resources, making it cost-effective and feasible within a small budget.

  The only ongoing costs would be related to hosting the application online if it were to be deployed, but that can be done at low cost using platforms like Heroku or DigitalOcean.The project is tested on ngrok server for temporary testing

System Architecture

- **Frontend (User Interface):** The user interface (UI) built with Flask where users input product URLs and view analysis results.

- **Backend (Processing and Analysis):** The server-side components responsible for scraping product reviews, performing sentiment analysis, and generating summaries. This part may use libraries like BeautifulSoup (for scraping) and TextBlob or VADER (for sentiment analysis).

- **Database:** The system may store user data, review data, and analysis results in a database like SQLite or MySQL. If you're storing API key data, you'd need to describe this as well.

- **API Layer:** If your system exposes an API for external users, describe how the API works and how it integrates with other components.

System Architecture Diagram

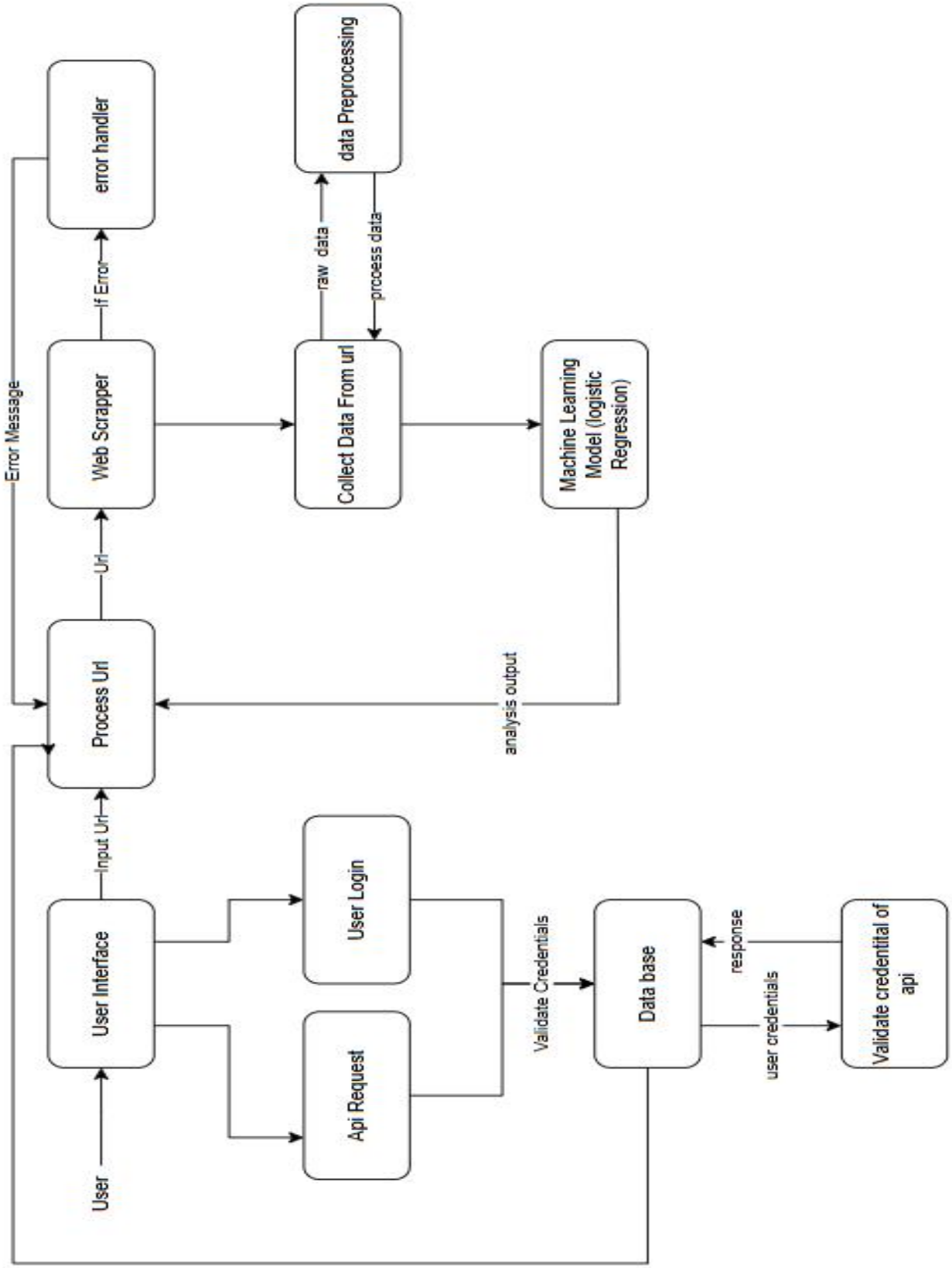


Figure 2.1 System Architecture Diagram

3. Software Requirement Specification

Purpose

The primary purpose of the Product Review Analysis project is to provide consumers with a reliable tool to make informed purchasing decisions by analyzing product reviews from popular e-commerce websites such as Flipkart and Amazon. With the ever-growing number of online products and reviews, it becomes challenging for users to determine whether a product meets their expectations. This project addresses this challenge by automating the process of evaluating product reviews, extracting real-time data, and offering a sentiment-based analysis to help users quickly assess the overall reception of a product.

Scope

- **Web Scraping:** The system extracts real-time product reviews and ratings from popular e-commerce websites, such as Amazon and Flipkart, using BeautifulSoup. It enables the collection of data for a wide range of products to be analyzed.

- **Sentiment Analysis:** By leveraging Natural Language Processing (NLP) techniques, the project performs sentiment analysis on the extracted reviews, categorizing them as positive, negative, or neutral. This helps users understand the overall sentiment toward a product.

- **Machine Learning Integration:** The project uses Machine Learning (ML) models to enhance the accuracy of sentiment analysis, providing more reliable insights into the reviews and ratings.

- **Web Interface:** The project includes a web interface built using Flask, where users can input product URLs from Flipkart or Amazon, analyze reviews, and view results in a user-friendly format.

- **API Access:** A key feature of the project is the ability to integrate the review analysis functionality via an API. This allows external users to access the sentiment analysis of product reviews programmatically.

- **Security and User Management:** The application includes user authentication and API key management to ensure secure access to the review analysis features, allowing users to track their usage and maintain privacy.

- **Deployment:** The project is deployed in a real-time environment using ngrok, allowing users to access the application over the internet.

Feasibility Study

- **Market Feasibility:** Assess the demand for a tool that helps users evaluate product reviews from major e-commerce platforms.

- **Technical Feasibility:** Ensure the project can successfully integrate web scraping, NLP, and machine learning to process data.

- **Economical Feasibility:** Analyze the costs associated with scraping data, machine learning training, and web development, while considering the return on investment.

- **Operational Feasibility:** Ensure smooth integration of the system with real-time review data and API access.

- **Scalability:** Ensure the system can handle multiple users and increasing amounts of product reviews.

- **Risk Analysis:** Identify potential risks such as scraping restrictions, unreliable data sources, and scaling challenges.

- **Legal and Regulatory Compliance:** Ensure compliance with data privacy laws and e-commerce site terms of service.

- **Social Impact:** Assess how the tool can help users make more informed and confident buying decisions, improving their online shopping experience.

Hardware Requirements:

Computer with Internet Access
Minimum 4 GB RAM
Processor: Dual-core or higher

Software Requirements:

Python 3.14
Flask (for web development)
Beautiful Soup (for web scraping)
NLTK or spaCy (for Natural Language Processing)
Scikit-learn (for Machine Learning)
SQLite (for database management)
HTML, CSS, and JavaScript (for front-end development)
Jinja2 (for templating in Flask)

- **External Libraries:**

  requests (for handling HTTP requests)
  pandas (for data handling)
  Matplotlib (for data visualization)

- **API Requirements:**

  Flask-RESTful (for building the API)

- **Security Tools:**

  Flask-Login (for user authentication)

- **Other Tools:**

  Git (for version control)
  Visual Studio Code or any Python IDE

Software Process Model Used

For this project, we adopted the Agile Model, a flexible and iterative approach to software development that allows for continuous improvement and adaptation. This model was well-suited for our team structure and project requirements.

The team consisted of three members:

Abdul Saboor(Leader): Responsible for the integration, backend development, API creation, and overall project coordination.

Nikita Rajput (Analysis) : Focused on data analysis, including dataset selection and building the machine learning model for sentiment analysis.

Jyoti Kalambe (design): Worked on UI/UX design, ensuring an intuitive and user-friendly interface for the web application.

Development Process

Phase 1 – Data Collection and Problem Statement Analysis:

The team first identified the problem and gathered the dataset from sources like Flipkart and Amazon for product reviews and ratings and data set from kaggle for model creation.

Phase 2 – Data Analysis and Model Building:

Nikita worked on cleaning and processing the data. She built the model for sentiment analysis using the dataset.

Phase 3 – UI/UX Design:

While the data model was being developed, Jyoti started designing the UI, ensuring that the web application would be user-friendly and visually appealing.

Phase 4 – Backend Integration and API Development:

Once the UI was ready, I integrated the backend with the web interface and worked on the API functionality to provide product analysis and generate API keys for users.

Phase 5 – Testing and Refinement:

After completing the integration, we tested the entire system and made refinements based on user feedback, ensuring everything worked seamlessly together.By following the Agile Model, we were able to work in parallel and continuously improve each aspect of the project. The iterative nature of Agile allowed us to adapt to changes quickly and incorporate new features as the project progressed.

4. System Documentation

DFD (Data Flow Diagram)

The Data Flow Diagram (DFD) provides a visual representation of the flow of data within the Product Review Analysis System. It depicts how external entities interact with the system, how data flows between the components, and the processes involved in handling the input from the user and generating an analysis report. This DFD illustrates the high-level operations of the system, including the interaction with external e-commerce websites (Amazon and Flipkart) for data scraping.

DFD Level 0 :

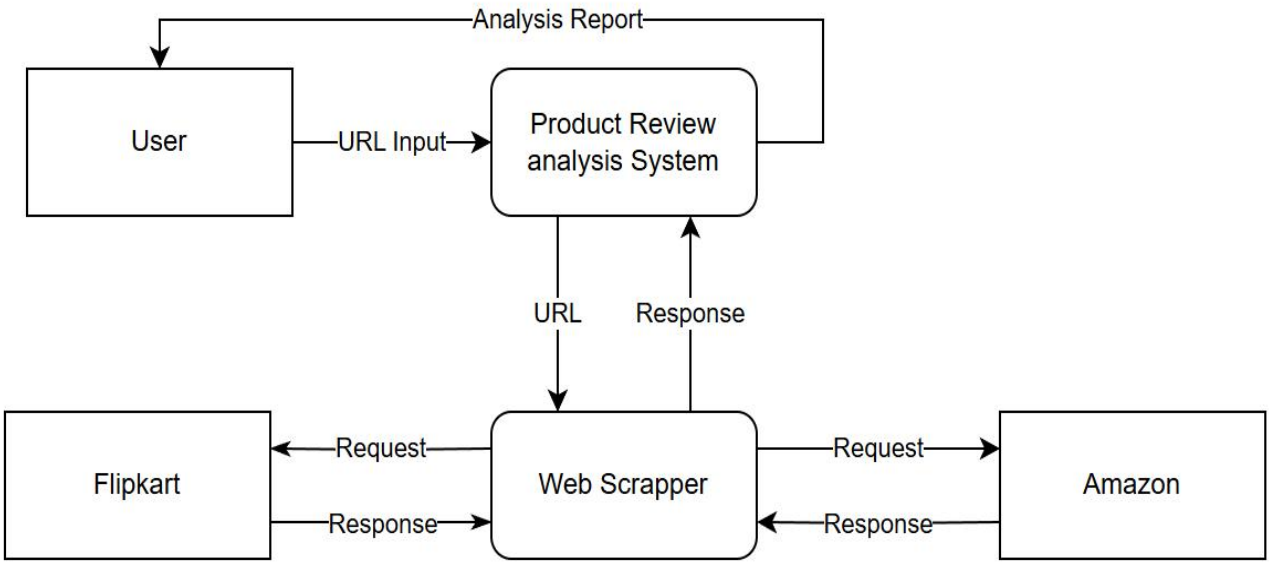


Figure 4.1 Data Flow Diagram level 0

Description of DFD level 0:

User: The external entity who provides the product URL and receives the analysis report.

Product Review Analysis System: The main system that processes the input URL, coordinates scraping, and generates the report.

Web Scraper: A sub-component of the system responsible for requesting and fetching data from e-commerce platforms (Amazon, Flipkart).

Amazon/Flipkart: External systems providing the product review data in response to scraping requests.

DFD Level 1:

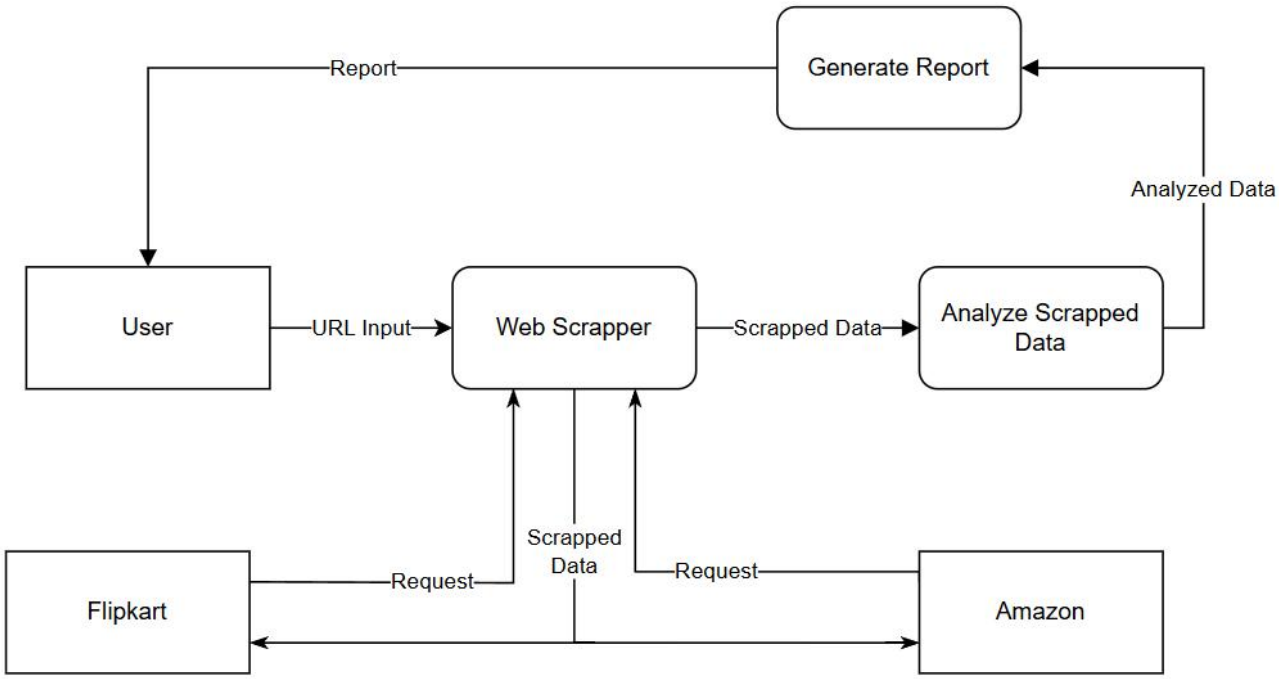


Figure 4.2 Figure 4.1 Data Flow Diagram level 1

Description of DFD level 1:

User : The user provides the URL, which is sent to the Web Scraper. The URL is validated to ensure it is from a supported e-commerce website like Amazon or Flipkart.

Scrape Data: The Web Scraper requests product review data from Amazon and Flipkart. The response contains the scraped product review information.

Analyze Scraped Data: The scraped data is sent to the Analyze Scraped Data process. Here, the reviews are analyzed, including sentiment analysis and other evaluation criteria.

Generate Report: Once the analysis is complete, the results are passed to the Generate Report process, which creates a comprehensive report on the product's review analysis.

Return Report to User: The final report is sent back to the User for review.

DFD Level 2:

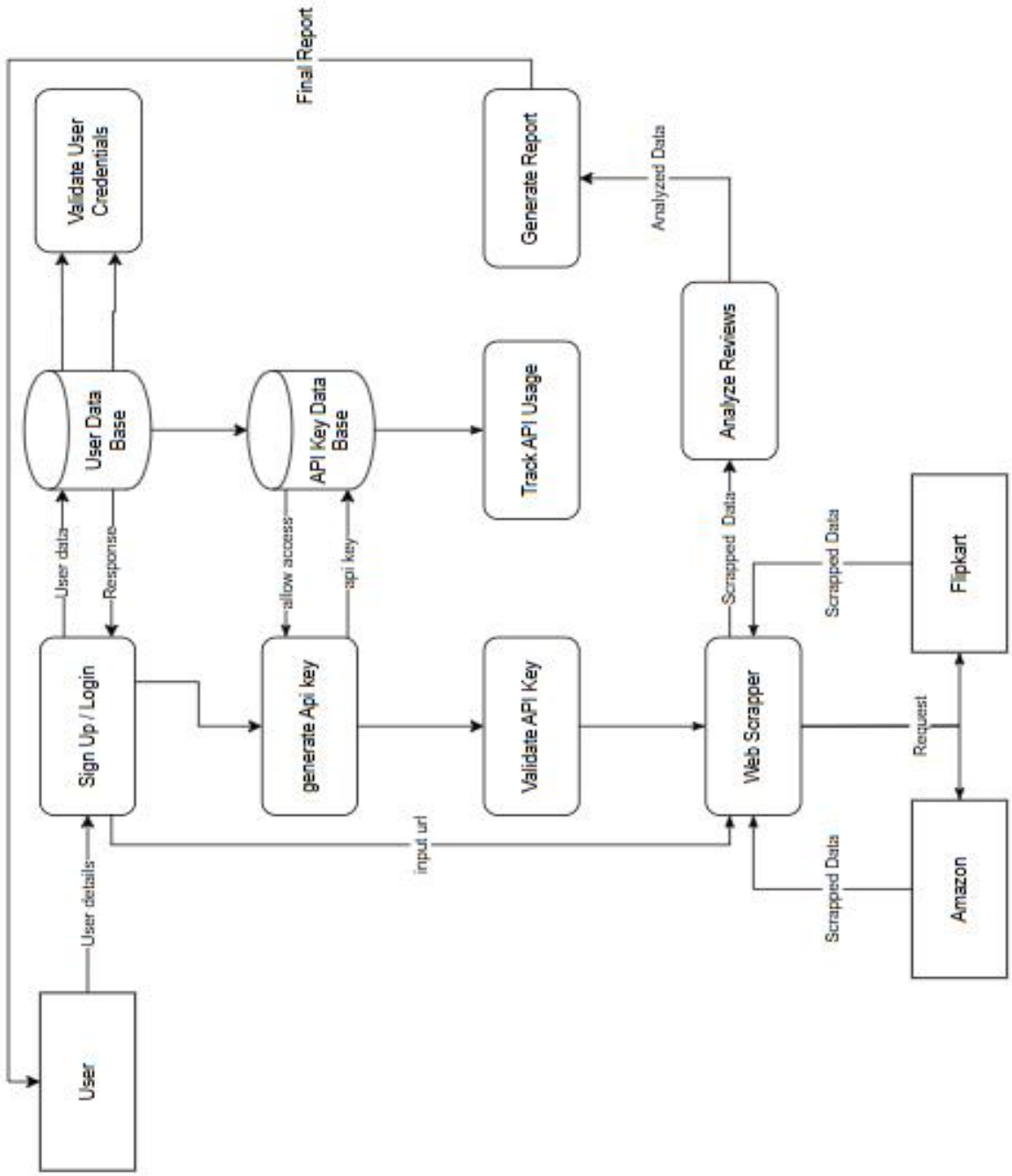


Figure 4.3 DFD Level 2

Description of DFD level 2:

User Account Validation and Authentication: When a User signs up or logs into the system, their credentials are validated by the Sign-Up/Login Process. If valid, the system checks for an existing user account in the User Database. If the account is valid, the system creates a user session or returns an error message if the credentialsare incorrect.

API Key Generation: Once logged in, the Generate API Key process is initiated. This process generates a unique API Key for the user, stores it in the API Key Data Store, and returns the key to the user. API Key Validation:For each API request, the system checks whether the provided API Key is valid. The Validate API Key process verifies the key against the API Key Data Store, checking its validity and ensuring it has not exceeded the usage limits.

Track API Usage: Each time the Scrape Data process is invoked, the system tracks the usage of the API Key. The Track API Usage process logs the usage count in the User Database to monitor the number of requests made by the user.

Scrape Data: The Scrape Data process sends requests to Amazon and Flipkart to fetch product review data. This data is stored temporarily in the Scraped Data Store for further analysis.

Analyze Reviews: The Analyze Reviews process takes the scraped data from the Scraped Data Store, processes it, and generates the analysis results (e.g., sentiment analysis).The analyzed data is stored in the Analysis Results Store before being sent to the Generate Report process.

Generate Report: Finally, the Generate Report process takes the analyzed data from the Analysis Results Store and generates a detailed product review analysis report, which is sent back to the User.

System Flow Chart

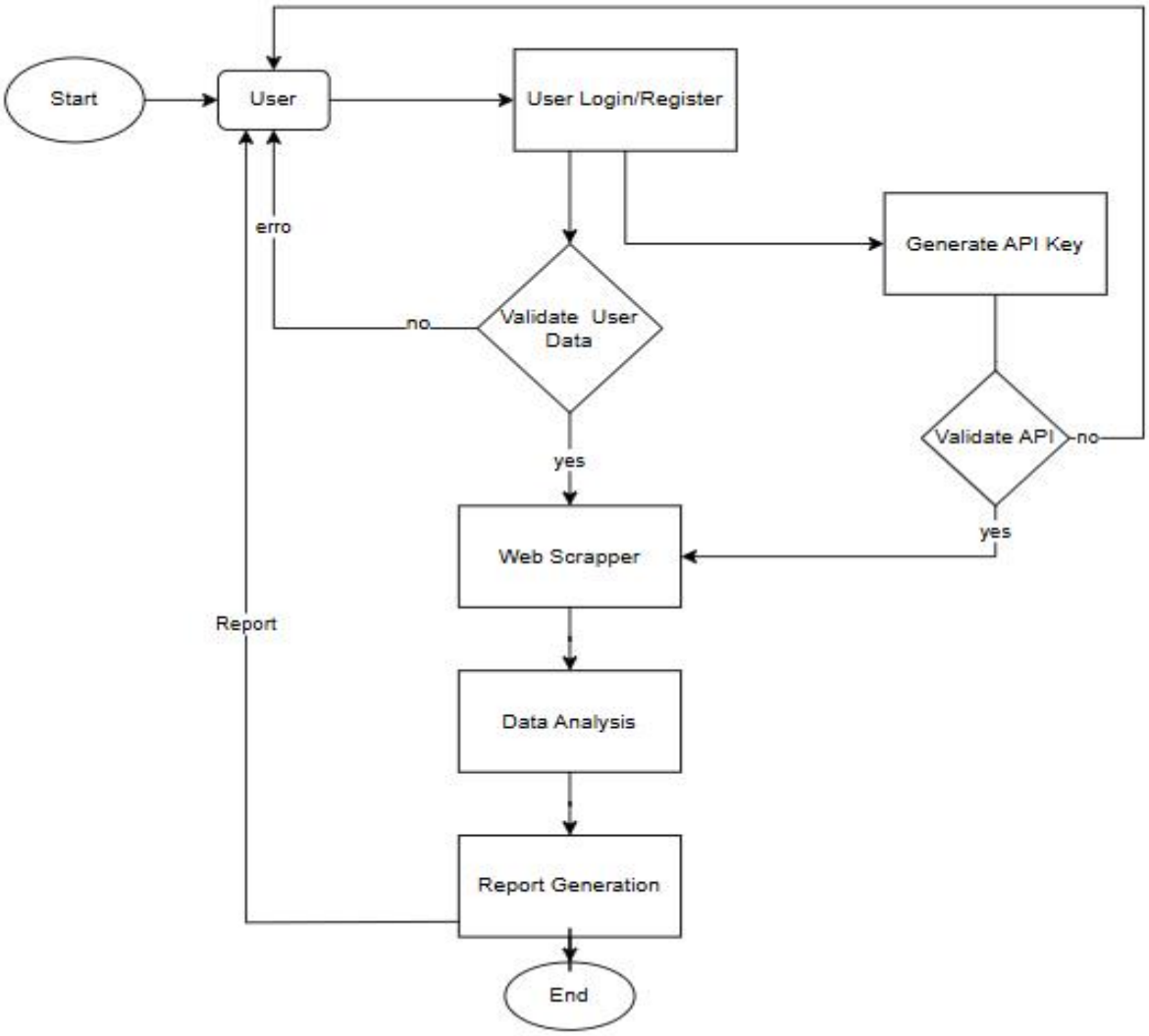


Figure 4.4 System flow chart

Description of System Flow Diagram :

User Login: The user initiates the process by entering their credentials. The system verifies the credentials and, if valid, generates an API key for secure access to system features.

URL Validation: Once the user logs in and provides a valid URL, the system checks if the URL belongs to a supported platform (Amazon or Flipkart). If the URL is invalid, the user is notified with an error message.

Scraping Product Data: If the URL is valid, the system sends the URL to the Web Scraper, which fetches product review data from the specified platform.

Data Analysis: The scraped data is then passed to the Data Analysis module. This module performs sentiment analysis and extracts key insights from the reviews.

Report Generation: After analysis, the system generates a detailed report summarizing the insights derived from the product reviews. The report is then sent to the user.

End: The user receives the final report, and the process concludes.

Entity Relationship Diagram

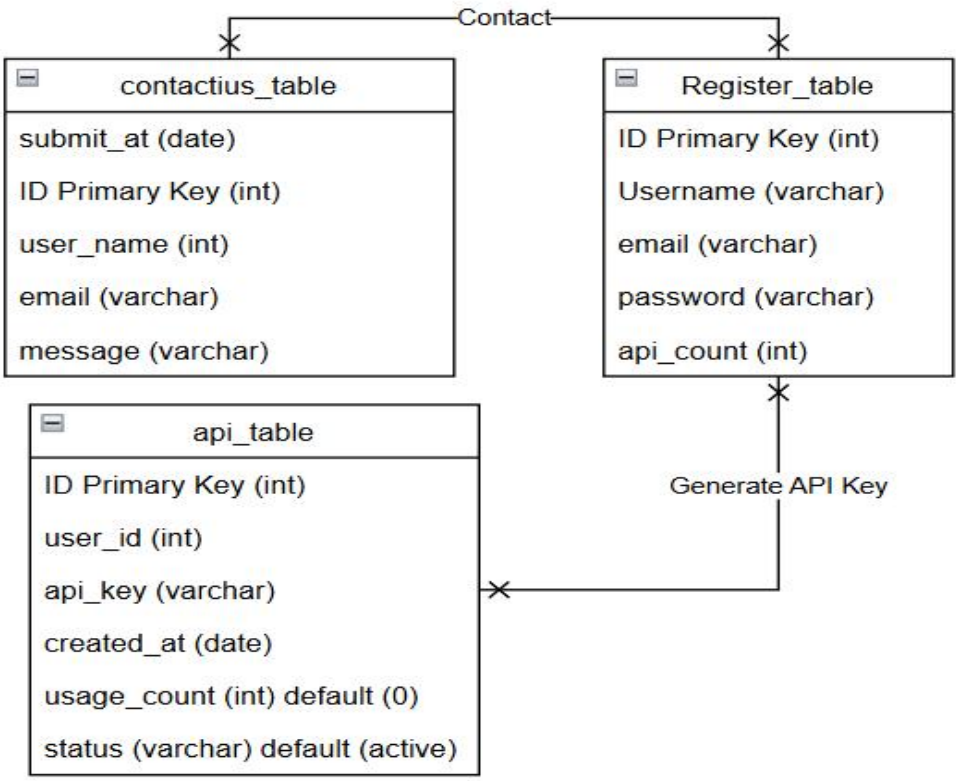


Figure 4.5 Entity Relationship Diagram

- **Entity:**

  Registertable:

  Represents users of the application.

  Attributes include:

  id: Unique identifier for each user (Primary Key).
  username: The user's chosen username.
  email: The user's email address.
  password: The user's hashed password.
  api_count: A count of how many API calls the user has made.

  APItable:

  Represents API keys associated with users.

  Attributes include:

  id: Unique identifier for each API key (Primary Key).
  user_id: Foreign Key referencing the Registertable, indicating which user owns API
  api_key: The actual API key string.
  created_at: Timestamp indicating when the API key was created.
  status: Current status of the API key (e.g., active, inactive).
  usage_count: Number of times the API key has been used.

  ContactUstable:

  Represents messages submitted by users through a contact form.

  Attributes include:

  id: Unique identifier for each contact message (Primary Key).
  username: The name of the user submitting the message.
  email: The email address of the user submitting the message.
  submit_at: Timestamp indicating when the message was submitted.
  message: The content of the user's message.

- **Relationships:**

  Registertable to APItable: A one-to-many relationship, where a single user (from Registertable) can have multiple API keys (in APItable). This is represented by the line connecting the two entities, with Registertable on the "one" side and APItable on the "many" side.

  Registertable to ContactUstable: A one-to-many relationship, where a user can submit multiple contact messages. This relationship indicates that each contact message can be associated with a specific user, although the ContactUstable does not have a direct foreign key reference to Registertable

Data Dictionary

Data Element	Description	Data Type	Format	Constraints
product_name	Name of the product scraped from the e-commerce site.	String	Alphanumeric characters	Non-empty
price	Price of the product.	Float	Decimal (e.g., 199.99)	Positive number
rating	Rating of the product (1-5).	Integer	5-Jan	Integer between 1 and 5
reviews	Number of user reviews.	Integer	Positive integer	Non-negative integer
review_sentiment	Model's sentiment label: -1 for "Negative", 0 for "Neutral", 1 for "Positive".	Integer	-1, 0, 1	Must be one of -1, 0, or 1
review_sentiment_label	Human-readable sentiment: "Negative", "Neutral", "Positive".	String	"Negative" / "Neutral" / "Positive"	Derived from review_sentiment
helpfulness_score	Confidence level of sentiment prediction, expressed as a percentage (0-100%).	Float	0-100	Between 0 and 100
url	URL provided by the user for scraping.	String	Valid URL format	Must be valid (Amazon or Flipkart only)
scraped_data	Raw data scraped from the website, including product details.	String	JSON format	N/A
scraping_status	Status of scraping process: "Success" or "Failure".	String	"Success" / "Failure"	N/A
report	Generated report with analyzed data (JSON or PDF format).	String	JSON or PDF format	N/A
api_key	API key assigned to a user.	String	Alphanumeric characters	Unique and non-empty
user_id	Unique user identifier for accessing the API.	Integer	Positive integer	Unique integer
created_at	Timestamp when the API key was created.	DateTime	ISO 8601 format (e.g., 2024-11-17T00:00:00Z)	N/A
last_used_at	Timestamp of the most recent use of the API key.	DateTime	ISO 8601 format	N/A
usage_count	Number of times the API key has been used.	Integer	Non-negative integer	N/A

Table 4.1 Data Dictionary

Use Case Diagram

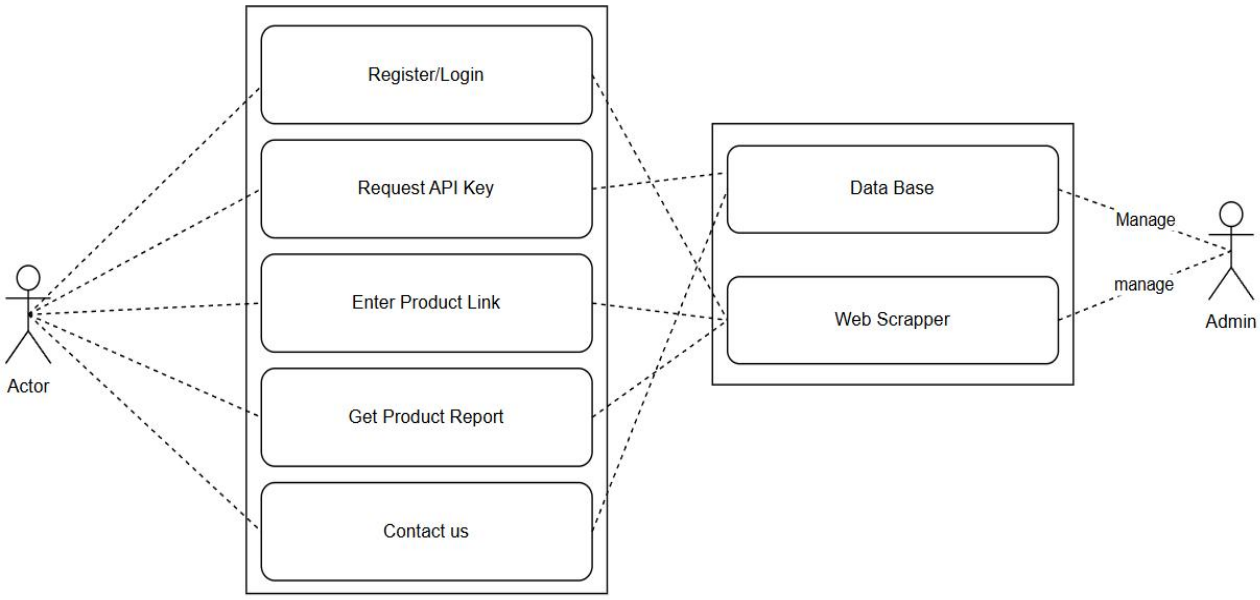


Figure 4.6 Use Case

Use Case Diagram Description

User Interactions:

Register: Users can create an account by providing necessary information, enabling them to access the system's features.

Login: Registered users can log into their accounts using their credentials.

Submit Product URL: Users can input product URLs from platforms like Amazon or Flipkart for the system to scrape.

Request API Key: user can generate and request for API KEY.

Contact Us: User can contact us by filling a form or can submit a feed back

Admin Interactions:

Manage Database: Admins have the ability to oversee and manage the database, ensuring data integrity and facilitating updates as necessary.

Web Scrapper: Admins have to manage web scrapper to see every thing working

5. User Manual

Introduction and Guidelines

- **Introduction**

  Welcome to the Web Scraping System! This platform is designed to help users extract and analyze product reviews from leading e-commerce websites such as Amazon and Flipkart. The system performs sentiment analysis on reviews, categorizing them as Positive, Neutral, or Negative, and provides a helpfulness score in percentage form. Additionally, it suggests alternative products based on the data.

  This manual aims to guide users in effectively interacting with the system, understanding its features, and troubleshooting common issues. Whether you are a regular user or an admin, this manual provides the necessary steps for utilizing the system's capabilities.

- **Guidelines**

  User:

  Registration and Login: Create an account and log in to access features.
  API Key Request: Generate an API key to access the system programmatically.
  Product Data Scraping: Input a product URL and retrieve detailed Analysis.
  Sentiment Analysis: View the breakdown of sentiments in the reviews.

  Admin:

  Database Management: Manage user accounts, product data, and reviews.
  Monitoring the Analysis System: Ensure sentiment analysis accuracy

  System Requirements

  To use this platform effectively, ensure your system meets the following prerequisites:

  Web Browser: Google Chrome (preferred) or any modern browser.
  Internet Connection: Required for accessing the platform and scraping product data.
  Supported Devices: Compatible with desktops, laptops, and tablets.

  Steps to Access the System

  Registration:

  Navigate to the Registration Page.
  Enter your email address, password, and other details.
  Submit the form to create an account.

  Login:

  Use your registered credentials to log in on the Login Page.
  Access the dashboard upon successful login.

Requesting an API Key:

After logging in, go to the API Key Request Page.
Click on Request API Key. The system will generate a unique key for you.

Scraping Product Data:

Input the product URL (Amazon or Flipkart) on the Product Scraping Page.
Click Analyze to scrape the reviews and analyze them.

Viewing Results:

Access a detailed report showing:
Sentiments (Positive, Neutral, Negative).
And view the charts of reviews and model classification probility

Important Notes

Valid URLs: Ensure you enter valid product URLs from Amazon or Flipkart.
Account Security: Do not share your login credentials or API key with others.
Troubleshooting: Refer to the FAQ section in this manual for common issues

Screen Layouts and Descriptions

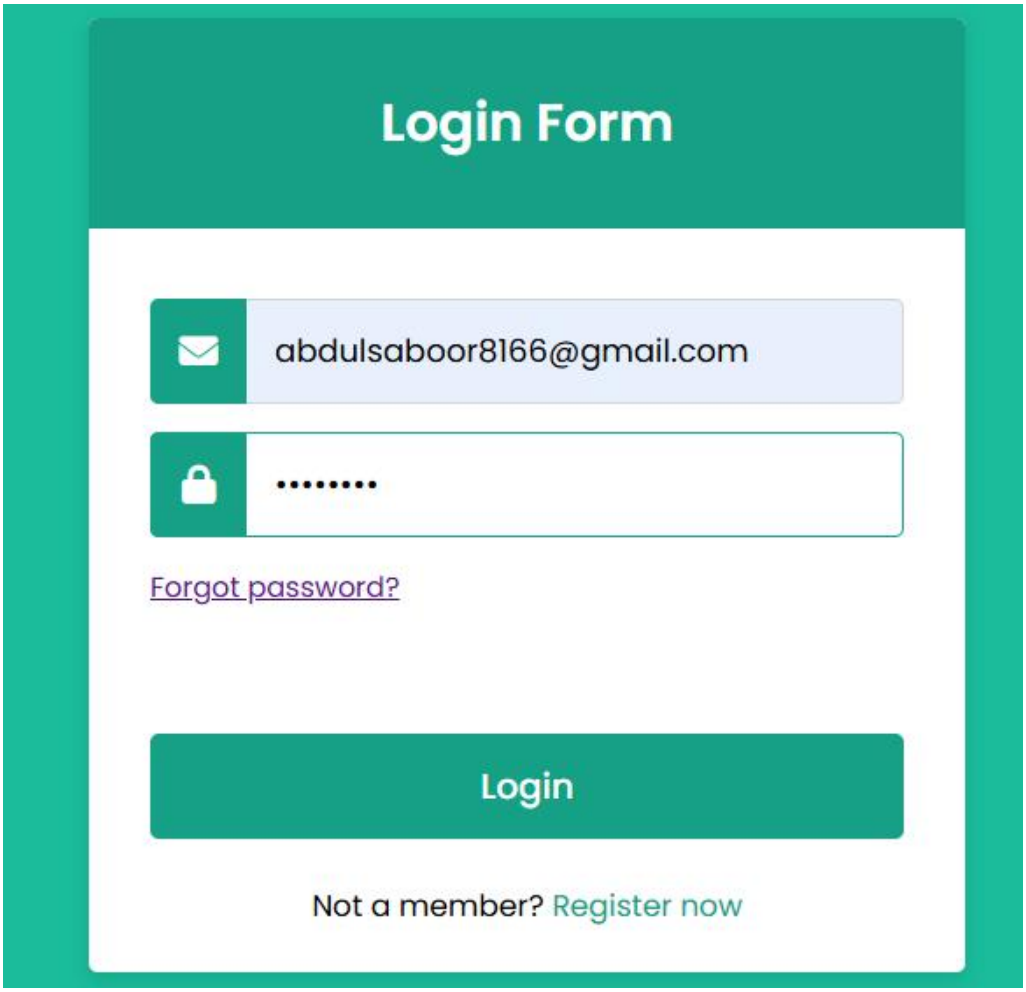
This section provides an overview of the key web pages in the system, their purpose, and how userscan interact with them. Screenshots are included to help users familiarize themselves with the interface.

- **Login Page**

  Purpose: To allow registered users to authenticate and access the system.

  Description: Users enter their email and password in the provided fields and click the Login button. An error message is displayed for invalid credentials.

  Screenshot:

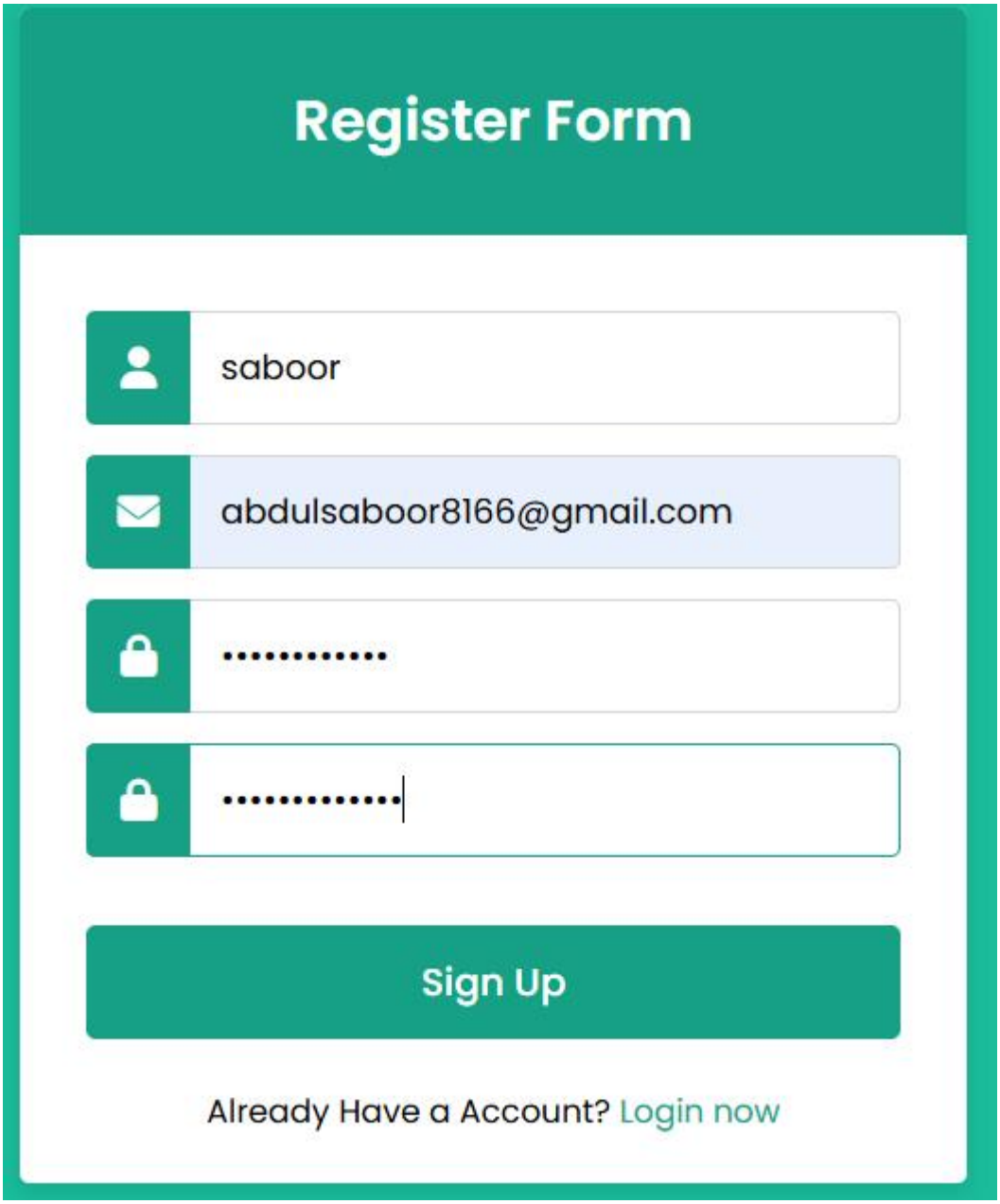


- **Registration Page**

  Purpose: To enable new users to create an account.

  Description: Users provide their name, email, password, and confirm the password. Upon successful registration, they can log in.

  Screenshot:



- **Home Page**

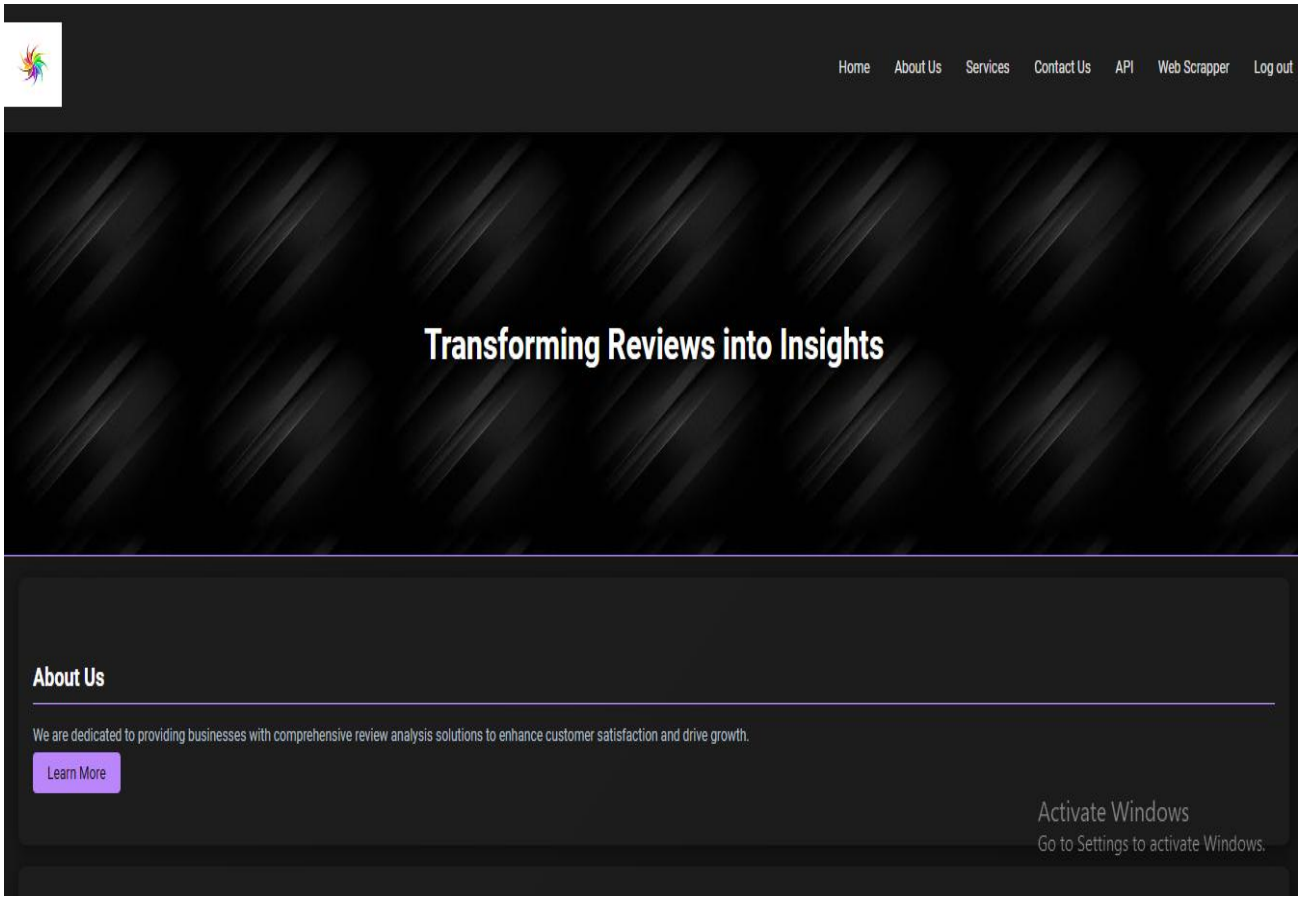
Purpose: Serves as the landing page for the system, providing a welcoming interface and navigation to key features.

Description:

The home page typically includes:

A navigation bar with links to Login, Registration, and API Key Request pages.A brief description of the system's features, such as product scraping, sentiment analysisAny call-to-action buttons like "Get Started" or "Explore Features"This page is the starting point for new users and provides an overview of the system.

Screenshot:

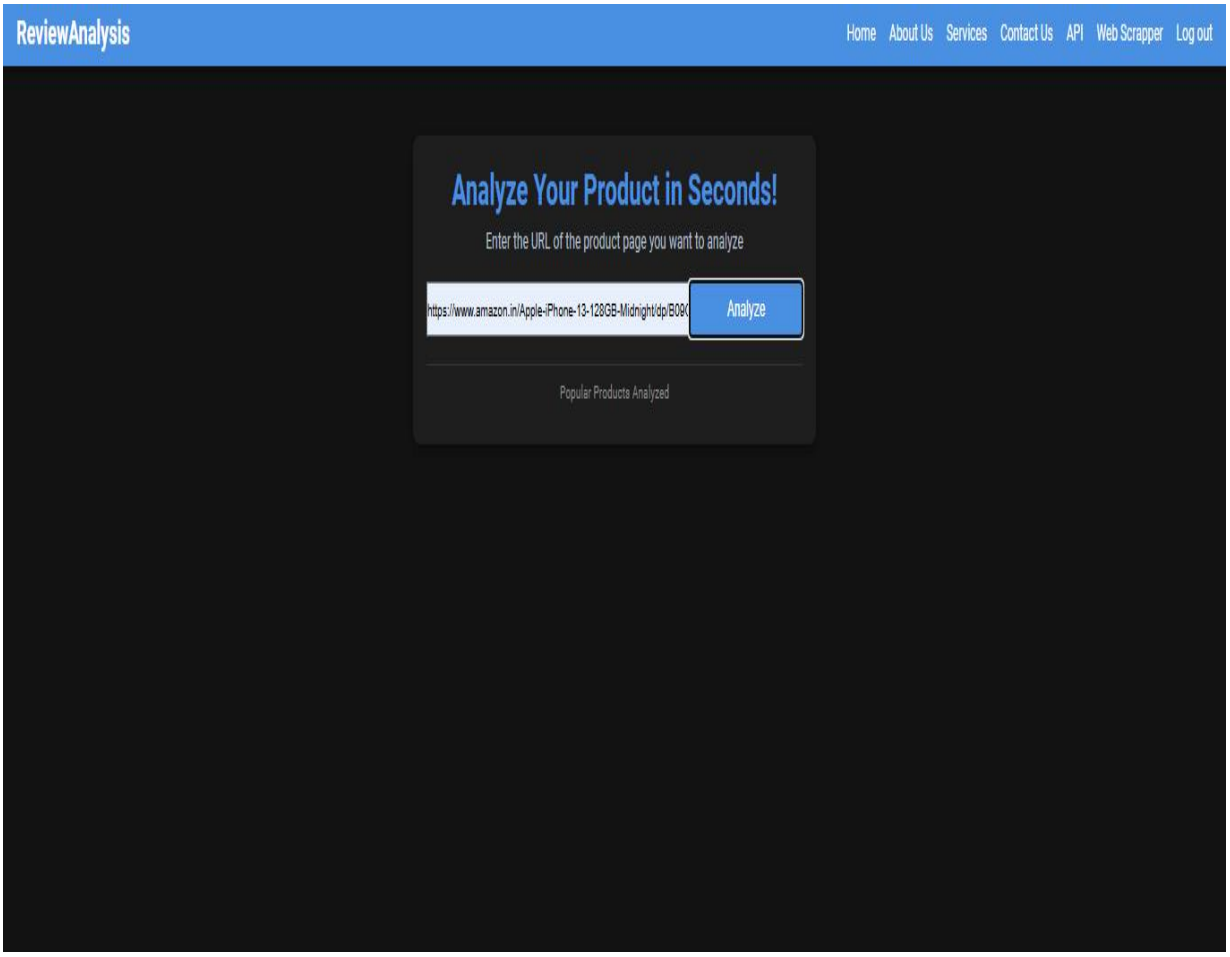


- **Product Scraping Page**

  Purpose: To enable users to input a product URL and retrieve data.

  Description: Users paste a valid Amazon or Flipkart product URL into the input field and click Submit. The system processes the URL, scrapes reviews, and performs sentiment analysis.

  Screenshot:

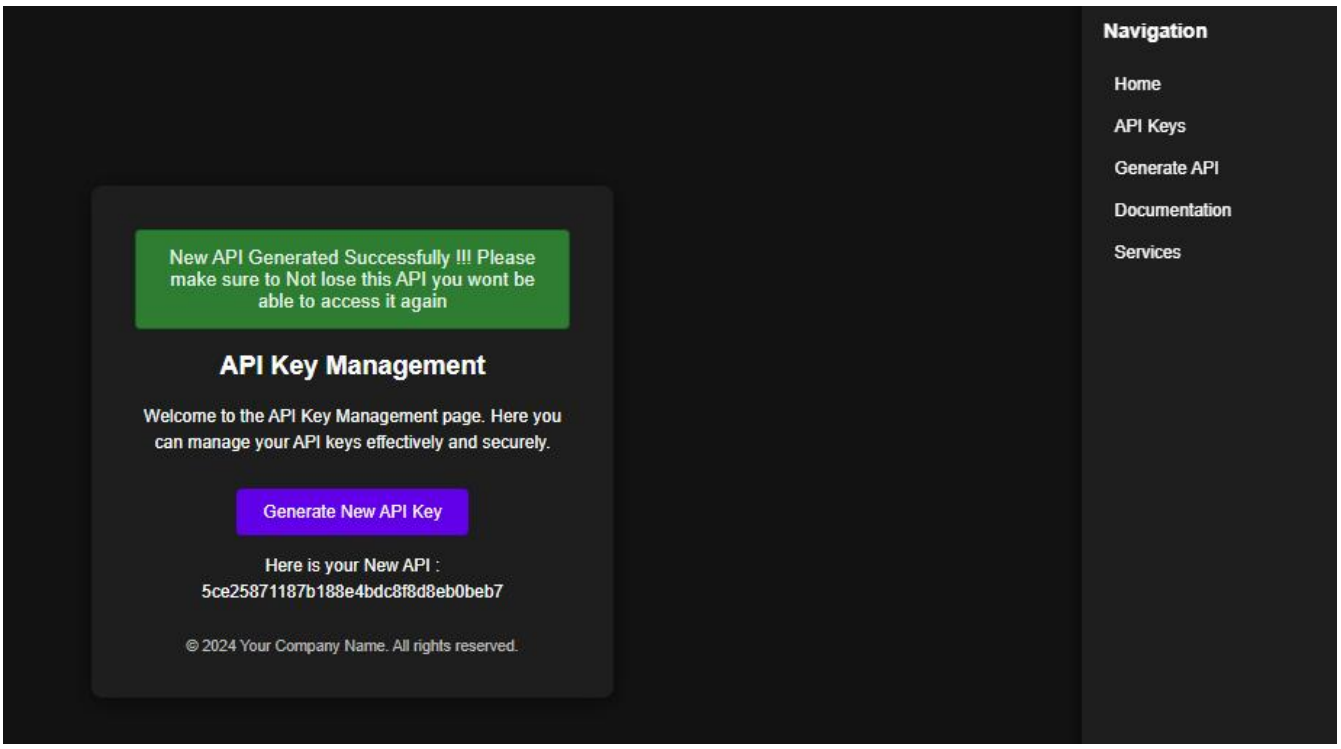


- **API Key Request Page**

  Purpose: To allow users to generate an API key for programmatic access to the system.

  Description: A simple interface where users click the Request API Key button to generate a unique key. The API key is displayed on the same page for the user to copy and save.

  Screenshot:

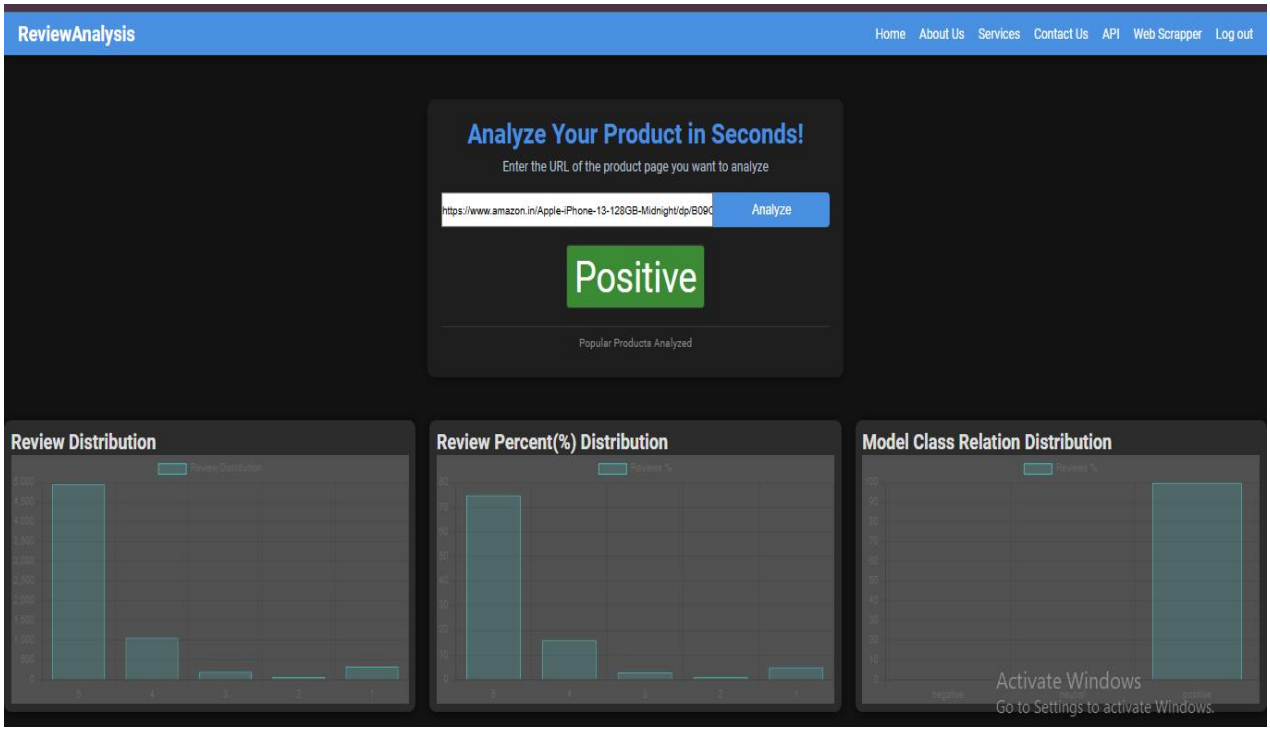


- **Product Review Results Page**

  Purpose: To display the analysis of scraped reviews.
  Description:

  Sentiments: The sentiments have 3 classes positive , negative and neutral

  Positive : means product have a good rating and good reviews user can buy it
  Negative: means product have less rating and bad reviewsmean user should not buy it
  Neutral: means product have a average rating and user can buy it

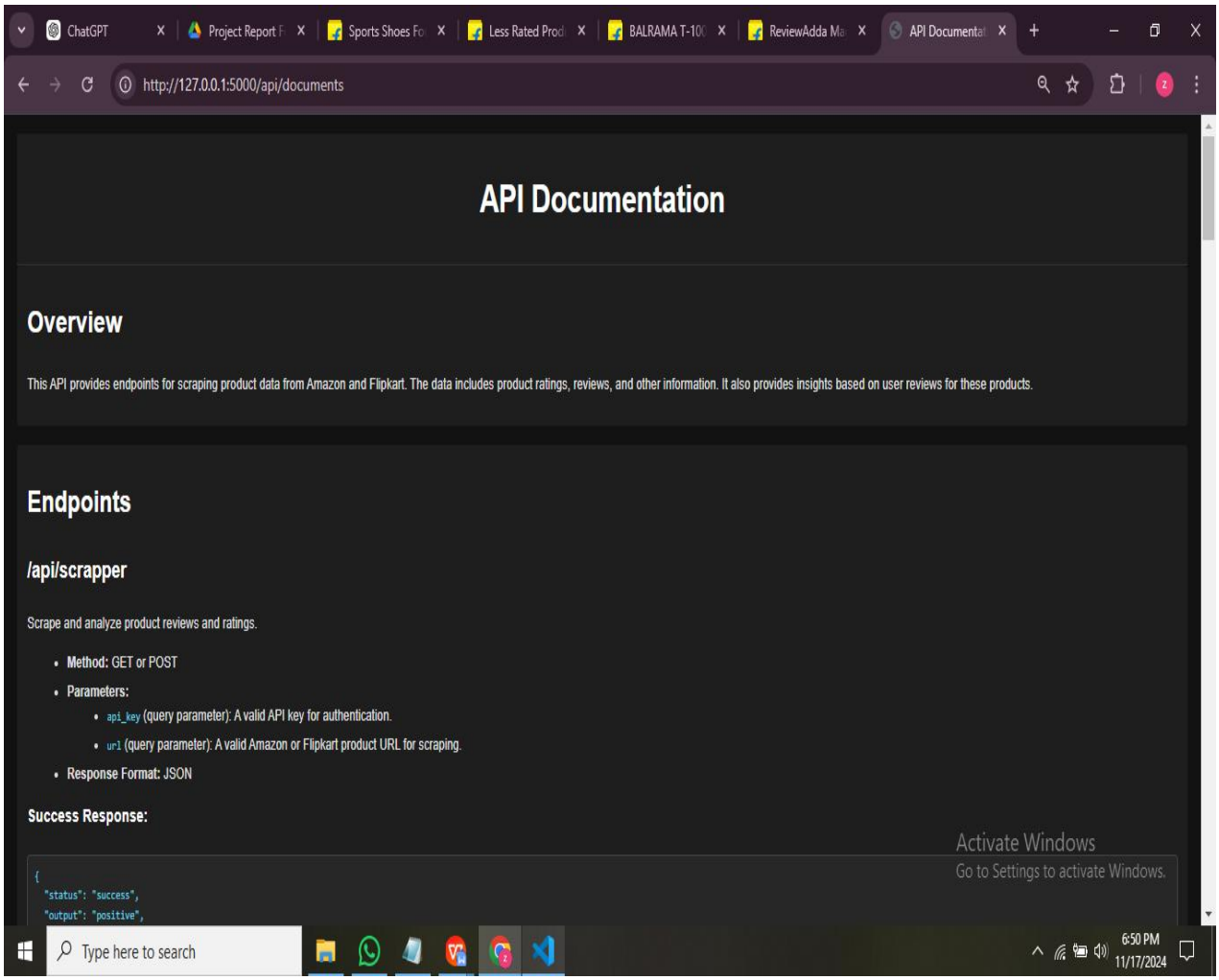


- **API Documentation Page**

  Purpose: To guide developers on how to integrate the API into their applications.

  Description: This page lists available endpoints, request formats, and response examples.

  Screenshot:



Output Reports

- **Overview:**

  The output reports generated by the web scraping and review analysis system provide valuable insights into the product reviews scraped from online shopping websites such as Amazon and Flipkart. The reports contain information on sentiment classification, review statistics, and product details, allowing users to assess the product's performance based on customer feedback.

  Sentiment Analysis Report

  Description: The sentiment analysis report provides an overall prediction of the sentiment of the product reviews, categorized into three labels: Positive, Neutral, and Negative. This analysis helps users understand how customers feel about the product based on the reviews.

  Outputs:

  output: The sentiment label predicted by the model:

  "Positive": Majority of the reviews are favorable.
  "Neutral": Reviews are balanced with no strong positive or negative bias.
  "Negative": Majority of the reviews are unfavorable.

  rating_dict: The count of reviews per star rating (1 to 5) for the product.

  Example: { "5": 50, "4": 30, "3": 10, "2": 5, "1": 5 }

  percent_dict: The percentage distribution of reviews across each star rating.

  Example: { "5": 50.0, "4": 30.0, "3": 10.0, "2": 5.0, "1": 5.0 }.

  probability_dict: The model's prediction probabilities for each sentiment class.

  Example: { "Positive": 0.85, "Neutral": 0.10, "Negative": 0.05 }.

  Product Data Report

  Description: The product data report includes key product details such as the product title, price,
  average rating, and total review count. This data helps users assess the product's overall popularity
  and quality based on customer reviews.

  Outputs:

  title: The product's name.

  Example: "Sample Product Title".

  price: The product's price.

  Example: "99.99".

rating: The average customer rating of the product.

Example: 4.5.

reviews_count: The total number of reviews for the product.

Example: 1200.

API Responses

Description: For API calls made to /api/scrapper and /api/product_data, the system returns structured JSON responses containing the product details, sentiment analysis results, and review
statistics. These responses enable the user to integrate or further analyze the data in their own
systems.

Sample API Response for /api/scrapper:
Format json:


```
{
    "status": "success",
    "output": "Positive",
    "rating_dict": {
        "5": 50,
        "4": 30,
        "3": 10,
        "2": 5,
        "1": 5
    },
    "percent_dict": {
        "5": 50.0,
        "4": 30.0,
        "3": 10.0,
        "2": 5.0,
        "1": 5.0
    },
    "probability_dict": {
        "Positive": 0.85,
        "Neutral": 0.10,
        "Negative": 0.05
    }
}
```

Sample API Response for /api/product_data:

Format : json


```
{
    "status": "success",
    "product_data": {
        "title": "Sample Product Title",
        "price": "$99.99",
        "rating": 4.5,
        "reviews_count": 1200
    }
}
```


Error Handling in Outputs

Description: If there is an error in the API request or processing, the system will provide an error
message in the output. The error message will include details about the cause of the error to assist
the user in resolving the issue.

Common Error Responses:

Invalid API Key:

Format :json


```
{
    "status": "error",
    "message": "Invalid API"
}
```


Invalid URL: When the URL provided is not from Amazon or Flipkart.

Format :json


```
{
    "status": "error",
    "message": "Invalid URL",
    "reason": "the scrapper works only on flipkart and amazon"
}
```


General Error: If an exception occurs during data processing.
Format : json


```
{
    "status": "error",
    "message": "An error occurred: <error details>"
}
```

- **Output Screen short :**

Review (%) Distribution

Graph 5.1 Review (%) Distribution

This is the graph of distribution of rating of product in % here each bar represent rating from 1 to 5

Review Distribution

Graph 5.2 Review Distribution

This is the graph of distribution of rating of product in each bar represent rating from 1 to 5

Model Class Probability Distribution

Graph 5.3 model class relation Distribution

This is the graph of distribution of rating classes the model thing the product belong this show the distribution of how much probability product belong to specific class

6. Limitations

While the web scraping and review analysis system offers valuable insights into product reviews, there are certain limitations to consider regarding its functionality and accuracy. Below are the key limitations of the system:

- **Limited to Amazon and Flipkart**

  Currently, the system only supports scraping and analysis of product reviews from Amazon and Flipkart. The scraper does not work with other e-commerce platforms or websites. This restriction means that users who wish to analyze products from other sources will not be able to use the system.

- **Dependence on Website Structure**

  The accuracy and effectiveness of the scraper depend on the website's structure. If Amazon or Flipkart changes the layout or HTML structure of their pages, the scraper may fail to extract relevant data, leading to errors or incomplete information in the reports. This dependency means that the system may require maintenance or updates whenever the structure of these websites changes.

- **Review Data Quality**

  The quality of sentiment analysis is based on the data retrieved from user reviews. If reviews are sparse, inconsistent, or contain irrelevant content, the system may not provide accurate sentiment predictions. Additionally, product reviews may not always be a true reflection of the product's performance, as users may leave reviews that are not fully representative of the product’s quality or features.

- **Model Accuracy**

  While the sentiment classification model works well in general, it is not flawless. The accuracy of the model may vary based on the complexity of the reviews. In some cases, reviews with ambiguous sentiment or complex language may result in incorrect sentiment labels (positive, negative, or neutral). The model's effectiveness is limited to the training data it was built on, and further improvements are necessary to handle more diverse review content.

- **Lack of Product Suggestions**

  Due to time constraints, the feature for providing product suggestions based on user preferences and review analysis has not been implemented. As a result, users are unable to receive recommendations for similar or alternative products from the system.

- **API Limitations**

  - **API Rate Limiting:** To avoid misuse or overuse, the API may have a limited number of calls that can be made within a specific time frame. Excessive use of the API might lead to throttling or temporary suspension of access.

  - **API Key Validation:** Users must obtain a valid API key to use the API endpoints. If an invalid API key is provided, the request will be denied, which could impact users who fail to manage their keys correctly.

- **Performance Constraints**

  The system's performance might degrade with large-scale data scraping or analysis due to server or resource limitations. If users attempt to scrape or analyze an extensive number of products simultaneously, the system may experience delays or slow responses. This can impact the overall user experience, particularly for high-traffic use cases.

- **No Real-Time Data Fetching**

  The system does not fetch real-time data continuously. Instead, it scrapes data at the time of request, and the product reviews or information retrieved may not reflect the most recent changes or trends on the website. Users should be aware that the product data may be slightly outdated by the time they receive the report.

- **Legal and Ethical Considerations**

  Web scraping, particularly for large-scale data extraction, can raise legal and ethical concerns. Some websites, including Amazon and Flipkart, may have terms of service that prohibit scraping or automated data extraction. Users must ensure that they comply with relevant terms and conditions before using the system for scraping data.

7. Future Enhancements

- **Support for Additional E-commerce Platforms**

  Expanding support for other platforms (e.g., eBay, Walmart) will broaden the system's reach and allow users to analyze more products.

- **Real-Time Data Fetching**

  Introducing real-time data fetching and periodic scraping could ensure the most up-to-date product details and reviews.

- **Product Suggestions Feature**

  A recommendation engine could be implemented to suggest products based on sentiment, ratings, and reviews, improving user experience.

- **Enhanced Sentiment Analysis Model**

  Refining the model with advanced NLP techniques (e.g., BERT) would improve sentiment classification accuracy, especially for complex or ambiguous reviews.

- **Multi-Language Support**

  Adding support for multiple languages would allow the system to analyze reviews in various languages, making it more inclusive.

- **User Dashboard for Review Analytics**

  A dashboard could be developed to visualize review trends, sentiment analysis, and API usage statistics, enhancing user engagement.

- **Improved API Management**

  Features like usage quotas, API analytics, and access control (IP whitelisting, expiration) would enhance API security and usability.

- **Integration with Data Analytics Tools**

  Allowing data exports to platforms like Excel or Tableau would enable users to perform more advanced analysis on scraped data.

- **Integration with Chatbots**

  Integrating the system with virtual assistants like Alexa or Google Assistant would enable voice-controlled product reviews and analysis.

- **Mobile Application**

  A mobile app for iOS and Android would provide users with on-the-go access to scraping andanalysis features.

8. Conclusion

The web scraping and review analysis system provides valuable insights into product reviews, helping users make informed purchasing decisions based on sentiment analysis and review data. Despite its current success, there are several opportunities to enhance the system's functionality and improve user experience.

By expanding the platform's capabilities to support additional e-commerce websites, implementing real-time data fetching, and introducing features like product suggestions and multi-language support, the system can become even more versatile and user-friendly. Additionally, security improvements, enhanced reporting features, and a mobile application could further increase accessibility and user engagement.

Overall, with these planned future enhancements, the system will not only meet the evolving needs of its users but also keep pace with emerging trends in data analysis, e-commerce, and machine learning, making it a powerful tool for both consumers and businesses alike.

9. Bibliography

- **Books**

Lang, M., & Yang, H. (2019). Practical Web Scraping for Data Science: Best Practices and Techniques for Web Scraping. O'Reilly Media, 2019.

- **Research Papers/Articles**

Kumar, S., & Mehta, P. (2020). "Sentiment Analysis of Product Reviews Using Machine Learning." International Journal of Computer Applications, 174(1), 29-34.
Patel, R., & Shah, M. (2021). "Web Scraping Techniques and Tools for Data Extraction." International Journal of Computer Science and Technology, 9(3), 156-162.

- **Websites**

Amazon, "Product Reviews API Documentation," Amazon Web Services, https://aws.amazon.com/reviews, Accessed: November 2024.
Flask, "Flask Web Framework," Flask Documentation, https://flask.palletsprojects.com, Accessed: November 2024.

- **Software/Tools**

Beautiful Soup, Version 4.11.1, https://www.crummy.com/software/BeautifulSoup/, 2024.
Scikit-learn, Version 0.24, https://scikit-learn.org, 2024.
TensorFlow, Version 2.6.0, https://www.tensorflow.org, 2024.

- **Datasets**

Amazon Review Dataset, "amazone Product Reviews," Kaggle, 2024, “https://drive.usercontent.google.com/download?id=1Z_y9nw9nQCmuQUJyFUF2DkVw_7KX346b&export=download&authuser=0&confirm=t&uuid=545a5976-e0e2-4b11-a9a5-41083376e5e5&at=AO7h07dxkHIDUH8lVPSIREQnlDzB%3A1727204151871”

Python Software Foundation, Python Documentation, 2024, https://docs.python.org.
Jupyter Project, Jupyter Notebook Documentation, 2024, https://jupyter.org.

10. References

- Lang, M., & Yang, H. (2019). Practical Web Scraping for Data Science: Best Practices and Techniques for Web Scraping. O'Reilly Media.

- Kumar, S., & Mehta, P. (2020). "Sentiment Analysis of Product Reviews Using Machine Learning." International Journal of Computer Applications, 174(1), 29-34.

- Patel, R., & Shah, M. (2021). "Web Scraping Techniques and Tools for Data Extraction." International Journal of Computer Science and Technology, 9(3), 156-162.

- Amazon Web Services. (2024). "Product Reviews API Documentation." Amazon Web Services. Retrieved November 2024 from https://aws.amazon.com/reviews

- Flask Documentation. (2024). Flask Web Framework. Retrieved November 2024 from https://flask.palletsprojects.com

- Beautiful Soup. (2024). "Beautiful Soup Documentation." Retrieved from https://www.crummy.com/software/BeautifulSoup/

- Scikit-learn. (2024). Scikit-learn Documentation. Retrieved from https://scikit-learn.org

- TensorFlow. (2024). TensorFlow Documentation. Retrieved from https://www.tensorflow.org

- Amazon Review Dataset. (2024). "Flipkart Product Reviews." Kaggle. Retrieved from https://drive.usercontent.google.com/download?id=1Z_y9nw9nQCmuQUJyFUF2DkVw_7KX346b&export=download&authuser=0&confirm=t&uuid=545a5976-e0e2-4b11-a9a5-41083376e5e5&at=AO7h07dxkHIDUH8lVPSIREQnlDzB%3A1727204151871

- Python Software Foundation. (2024). Python Documentation. Retrieved from https://docs.python.org

- Jupyter Project. (2024). Jupyter Notebook Documentation. Retrieved from https://jupyter.org

11. Appendix – I Source Code

```
import secrets
import pickle
import traceback
from datetime import datetime
import os
import json
from PIL import Image
from reviewanalysis import app,db,bcrypt,mail,amazon,flipkart
from flask import render_template,flash,redirect,url_for,request,abort,send_from_directory
from reviewanalysis.forms import
RegisterForm,LoginForm,UpdateAccountForm,RequestResetForm,ResetPasswordForm,ProductLin
k,APIGenerator,DeleteAPI,ContactUsForm
from reviewanalysis.models import Registertable,APItable,ContactUstable
from flask_login import login_user,current_user,logout_user,login_required
from flask_mail import Message
from flask import jsonify,session

@app.route("/register", methods=['POST', 'GET'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegisterForm()
    if form.validate_on_submit():
        print("Form validation successful!")   # Add a print statement to check if form validation is
successful
        hashed = bcrypt.generate_password_hash(form.password.data).decode("utf-8")
        userdata = Registertable(username=form.username.data.lower(),
email=form.email.data.lower(), password=hashed,api_count=0)
        db.session.add(userdata)
        db.session.commit()
        flash("Your account is created and you can login now!", 'success')
        return redirect(url_for('login'))
    else:
        print("Form validation failed!")
    return render_template("register.html", title="Register", form=form)
@app.route("/")
def home():
    return render_template("home.html")
@app.route("/login",methods=['POST','GET'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form=LoginForm()
    if form.validate_on_submit():
        userdata=Registertable.query.filter_by(email=form.email.data.lower()).first()
        if(userdata and bcrypt.check_password_hash(userdata.password,form.password.data)):
            login_user(userdata,remember=form.remember.data)
            next_page=request.args.get('next')
            return redirect(next_page) if next_page else   redirect(url_for("home"))
        elif(userdata):
            flash("check your password","danger")
```

```
            else:
                flash('Wrong Email Entered',"danger")
    return render_template("login.html",title="Login",form=form)
def save_picture(form_picture):
    random_hex=secrets.token_hex(8)
    _,f_ext=os.path.splitext(form_picture.filename)
    picture_fn=random_hex+f_ext
    picture_path=os.path.join(app.root_path,'static/pictures',picture_fn)
    output_size=(125,125)
    i=Image.open(form_picture)
    i.thumbnail(output_size)
    i.save(picture_path)
    return picture_fn
@app.route("/aboutus")
def aboutus():
     return render_template("aboutus.html")
@app.route("/contactus",methods=["POST","GET"])
def contactus():
    form=ContactUsForm()
    if form.validate_on_submit():
                                                                                                    userdata
 =ContactUstable(username=form.name.data,email=form.email.data,message=form.message.data)
db.session.add(userdata)
        db.session.commit()
        flash("Your Message Recieved Successfully","success")
        return render_template("contactus.html",form=form)
    return render_template("contactus.html",form=form)
@app.route("/services")
def service():
    return render_template("services.html")
@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for("home"))
def send_reset_email(user):
    token=user.get_reset_token()
     msg=Message('Password
Reset Request',sender="saboorabdul627@gmail.com",recipients=[user.email])
    msg.body=f''' to Rese your password visit the following link :
http://127.0.0.1:5000/{url_for('reset_token',token=token,external=True)}
if you did not request reset then ignore this email'''
    mail.send(msg)
@app.route("/reset_password",methods=['POST','GET'])
def reset_request():
    if current_user.is_authenticated:
         return redirect(url_for('home'))
    form=RequestResetForm()
    if form.validate_on_submit():
        user_data=Registertable.query.filter_by(email=form.email.data).first()
        send_reset_email(user_data)
        flash("check your email to reset password !","info")
        return redirect(url_for('login'))
    return render_template('resetrequest.html',form=form)
```

```
@app.route("/reset_password/<token>",methods=['POST','GET'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user_data=Registertable.verify_reset_token(token)
    if user_data is None:
        flash("That is an Inavalid token the link is Expired ",'warning')
        return redirect(url_for('reset_request'))
    form=ResetPasswordForm()
    if form.validate_on_submit():
      print("Form validation successful!")  # Add a print statement to check if form validation is successful
        hashed = bcrypt.generate_password_hash(form.password.data).decode("utf-8")
        user_data.password=hashed
        db.session.commit()
        flash("Your Password is reset !", 'success')
        return redirect(url_for('login'))
    return render_template('resetpassword.html',form=form)
@app.route("/api/documents")
def document():
    return render_template("documentation.html")
@app.route("/scrapper", methods=['POST', 'GET'])
@login_required
def scrapper():
    form = ProductLink()
    if form.validate_on_submit():
        url = form.product_link._value()
        domain = amazon.get_domain(url) if "amazon" in url else flipkart.get_domain(url)

        # Validate URL
        valid = amazon.validate_url(url) if "amazon" in url else flipkart.validate_url(url)
        if not valid:
            flash("Sorry, the website only supports Flipkart or Amazon links", "danger")
            return render_template("webscrapper.html", form=form, valid=valid)

        # Fetch source code and parse
        sourcecode = amazon.get_source_code(url)
        soup = amazon.get_soup_code(sourcecode)
        error=""

        try:
            # Process based on domain
            if "amazon" in domain:
                # Amazon-specific functions
                total = amazon.findtotalreviewsNumber(soup)
                text_percentages = amazon.findReviewsPercentages(soup)
                error="text_percetage"
                int_percentages = amazon.convertPercentageToInt(text_percentages)
                avg_rating = amazon.findavgrating(total, int_percentages)
                rating_number = amazon.getAllRatingNumber(total=total, percentages=int_percentages)
                reviews = amazon.getReviews(soup)
                helpfullness = amazon.getHelpFullness(soup, total)
```

```
            elif "flipkart" in domain:
                # Flipkart-specific functions
                total = flipkart.findtotalreviews(soup)
                review_int = flipkart.findnumberofreviews(soup)
                percent_int = flipkart.percentageconvertion(total, review_int)
                avg_rating = flipkart.findavgrating(total, percent_int)
                reviews = flipkart.getReviews(soup)
                helpfullness = flipkart.getHelpFullness(soup)
                rating_number = review_int
                int_percentages = percent_int
            # Load model and transform features
            model = amazon.load_model()
            cvec = amazon.load_vec()
            tfidf = amazon.load_tfidf()
            rating_dict = amazon.converttodict([5, 4, 3, 2, 1], rating_number)
            per_dict = amazon.converttodict([5, 4, 3, 2, 1], int_percentages)
                        finalinputs = amazon.combineparameter(helpful=helpfullness, model=model,
overall=avg_rating,
                                                                reviewstxt=reviews, tfidf_review=tfidf,
vectorizer_review=cvec)
            output = amazon.model_predict(model=model, inputs=finalinputs)
            prob_dict = amazon.find_probability(model, finalinputs)
            return render_template("webscrapper.html", form=form, valid=valid, output=output,
                                    rating_dict=rating_dict, per_dict=per_dict, prob_dict=prob_dict)

        except Exception as e:
            print(f"Exception: {e}")
            print("Traceback:")
            traceback.print_exc()
            flash(f"The web page does not contain sufficient data for analysis {e}", "warning")
            return render_template("webscrapper.html", form=form,
                                    error=f"Sorry for the inconvenience, but the provided link does not
have the required data for analysis {e}")

    # If form is not validated or no link is submitted
    return render_template("webscrapper.html", form=form)
@app.route("/api", methods=['POST', 'GET'])
@login_required
def api():
    form = APIGenerator()
    user=current_user
    if form.validate_on_submit():
        if user.api_count >= 5:
            flash("You have reached the limit of 5 APIs.", "warning")
            return render_template("api.html",form=form)
        api_key = secrets.token_hex(16)
        new_api = APItable(
            user_id=current_user.id,   # Link the API key to the logged-in user
            api_key=api_key,
            created_at=datetime.utcnow(),   # Set the creation time
            status='active',   # Set the default status to 'active'
            usage_count=0   # Initialize usage count to 0
        )
```

```
        # created a api
        db.session.add(new_api)
        db.session.commit()
        # update the api count
        user.api_count += 1
        db.session.commit()
        flash("New API Generated Successfully   !!! Please make sure to Not lose this API you wont
 be able to access it again","success")
        return render_template("api.html",form=form,api_key=api_key)
    return render_template("api.html",form=form)
@app.route("/api/manageapi", methods=['POST', 'GET'])
@login_required
def manageapi():
    user_api=APItable.query.all()
    form= DeleteAPI()

    return render_template("manageapi.html",user_api=user_api,form=form)
@app.route("/api/manageapi/delete<int:id>", methods=["POST", "GET"])
@login_required
def deleteapi(id):
    try:
        api = APItable.query.get(id)
        if not api:
            flash("API key not found.", "error")
            return redirect(url_for('manageapi'))

        # Delete the API entry
        db.session.delete(api)

        # Update the usage count for the user who owns the API
        current_user.api_count -= 1   # Adjust as needed, e.g., increment or decrement
        db.session.commit()

        flash(f"Successfully deleted the API {api.api_key}", "success")

    except Exception as e:
        db.session.rollback()
        flash("An error occurred while deleting the API key.", "error")
        print(e)   # or log the error as needed

    return redirect(url_for('manageapi'))
#    api routes :
@app.route("/api/scrapper", methods=["POST", "GET"])
def scrapperapi():
    api_key = request.args.get("api_key")
    key_record = APItable.query.filter_by(api_key=api_key).first()
    if key_record is None:
        return jsonify({"status": "error", "message": "Invalid API"}), 400
    key_record.usage_count += 1
    db.session.commit()
    url = request.args.get("url")
    domain = amazon.get_domain(url) if "amazon" in url else flipkart.get_domain(url)
    # Validate URL forma
```

```
    valid = amazon.validate_url(url) if "amazon" in url else flipkart.validate_url(url)
    if not valid:
        return jsonify({"status": "error", "message": "Invalid URL", "reason": "the scrapper works only on flipkart and amazon"}), 400
    sourcecode = amazon.get_source_code(url)
    soup = amazon.get_soup_code(sourcecode)
    try:
        if "amazon" in domain:
            total = amazon.findtotalreviewsNumber(soup)
            text_percentages = amazon.findReviewsPercentages(soup)
            int_percentages = amazon.convertPercentageToInt(text_percentages)
            avg_rating = amazon.findavgrating(total, int_percentages)
            rating_number = amazon.getAllRatingNumber(total=total, percentages=int_percentages)
            reviews = amazon.getReviews(soup)
            helpfullness = amazon.getHelpFullness(soup, total)
        elif "flipkart" in domain:
            total = flipkart.findtotalreviews(soup)
            review_int = flipkart.findnumberofreviews(soup)
            percent_int = flipkart.percentageconvertion(total, review_int)
            avg_rating = flipkart.findavgrating(total, percent_int)
            reviews = flipkart.getReviews(soup)
            helpfullness = flipkart.getHelpFullness(soup)
            rating_number = review_int
            int_percentages = percent_int
        # Pre-load the model and vectorizer only once
        model = amazon.load_model()
        cvec = amazon.load_vec()
        tfidf = amazon.load_tfidf()
        rating_dict = amazon.converttodict([5, 4, 3, 2, 1], rating_number)
        per_dict = amazon.converttodict([5, 4, 3, 2, 1], int_percentages)
        finalinputs = amazon.combineparameter(helpful=helpfullness, model=model, overall=avg_rating,
                                              reviewstxt=reviews, tfidf_review=tfidf, vectorizer_review=cvec)
        output = amazon.model_predict(model=model, inputs=finalinputs)
        prob_dict = amazon.find_probability(model, finalinputs)
        output_dict = {
            "status": "success",
            "output": output,
            "rating_dict": rating_dict,
            "percent_dict": per_dict,
            "probability_dict": prob_dict
        }
        return jsonify(output_dict)
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500
@app.route("/api/product_data", methods=['POST', "GET"])
def product_data_api():
    api_key = request.args.get("api_key")
    key_record = APItable.query.filter_by(api_key=api_key).first()
    if key_record is None:
        return jsonify({"status": "error", "message": "Invalid API"}), 400
    key_record.usage_count += 1
```

```
db.session.commit()
try:
    url = request.args.get("url")
    domain = amazon.get_domain(url) if "amazon" in url else flipkart.get_domain(url)
    if domain == "amazon":
        source_code = amazon.get_source_code(url)
        soup = amazon.get_soup_code(source_code)
        title = amazon.get_product_title(soup)
        price = amazon.get_product_price(soup)
        total = amazon.findtotalreviewsNumber(soup)
        text_percentages = amazon.findReviewsPercentages(soup)
        int_percentages = amazon.convertPercentageToInt(text_percentages)
        rating = amazon.findavgrating(total, int_percentages)
    elif domain == "flipkart":
        source_code = flipkart.get_source_code(url)
        soup = flipkart.get_soup_code(source_code)
        title = flipkart.get_product_title(soup)
        price = flipkart.get_product_price(soup)
        total = flipkart.findtotalreviews(soup)
        review_int = flipkart.findnumberofreviews(soup)
        percent_int = flipkart.percentageconvertion(total, review_int)
        rating = flipkart.findavgrating(total, percent_int)
    product_data = {
        "title": title,
        "price": price,
        "rating": rating,
        "reviews_count": total,
    }
    return jsonify({
        "status": "success",
        "product_data": product_data
    })
except Exception as e:
    return jsonify({
        "status": "error",
        "message": str(e)
    })
```