SYNOPSIS

SentiSight is a Python-based sentiment analysis project leveraging Flask Bootstrap for the web interface and Text Blob for natural language processing. It aims to detect positive or negative sentiments in both textual and video content. Through machine learning and NLP techniques, the system provides real-time analysis and multi-language support, catering to various input formats. Users interact with an intuitive interface to input text or upload videos, receiving sentiment analysis results promptly. SentiSight empowers users to understand and monitor sentiment trends, whether analysing customer feedback, social media content, or any other textual or video-based data, making it a versatile tool for sentiment analysis tasks.

WHY IS PARTICULAR TOPIC CHOSEN?

Choosing a topic for my project is a crucial decision that can significantly impact its success and overall learning experience. Here are some compelling reasons to choose the topic of sentiment analysis for your project:

1. Relevance and Timeliness : Sentiment analysis is a highly relevant and timely topic in today's digital age. With the proliferation of social media, online reviews, and customer feedback, understanding sentiment has become increasingly important for businesses, organizations, and individuals to make informed decisions.

2. Interdisciplinary Nature : Sentiment analysis combines concepts from various disciplines such as natural language processing (NLP), machine learning, data analysis, and web development. By choosing this topic I can explore and integrate knowledge from multiple domains, making your project rich and interdisciplinary.

3. Practical Applications : Sentiment analysis has numerous practical applications across industries, including marketing, customer service, brand management, market research, and social media monitoring. By developing a sentiment analysis system, I can create a valuable tool that addresses real-world needs and challenges.

4. Learning Opportunities : Working on a sentiment analysis project provides valuable learning opportunities in areas such as data preprocessing, feature extraction, classification algorithms, model evaluation, and visualization techniques. I can gain hands-on experience with popular libraries and frameworks like NLTK, scikit-learn, TensorFlow, Flask, and more.

5. Personal Interest and Motivation : If I have a personal interest in understanding human emotions, analysing textual data, or building intelligent systems, sentiment analysis can be an engaging and motivating topic for my project. Choosing a topic that aligns with my interests can enhance your enthusiasm and commitment to the project.

6. Flexibility and Customization : Sentiment analysis projects offer flexibility in terms of customization and extension. I can explore different sentiment analysis techniques, experiment with various machine learning models, incorporate domain-specific knowledge or lexicons, and tailor the system to specific use cases or applications.

7. Potential Impact : Developing a sentiment analysis system has the potential to make a meaningful impact by helping businesses improve customer satisfaction, identify emerging trends, mitigate risks, and enhance decision-making processes. My project can contribute to addressing real-world challenges and creating positive outcomes for users and stakeholders.

OBJECTIVE AND SCOPE

OBJECTIVE

The objective of the SentiSight project is to develop a comprehensive sentiment analysis system capable of accurately detecting positive or negative sentiments in both textual and video content. This system aims to fulfil several key goals:

1. Multimedia Sentiment Analysis : Enable sentiment analysis not only for text but also for video content, allowing users to extract sentiment from diverse media formats.
2. Real-time Analysis : Implement real-time sentiment analysis capabilities to provide instant feedback on sentiment trends as new data is inputted.

3. Multi-language Support : Ensure the system can analyse sentiment in multiple languages, catering to a global audience and diverse content sources.

4. User-Friendly Interface : Design an intuitive user interface that simplifies the process of inputting text or uploading videos for sentiment analysis.

5. High Accuracy : Utilize advanced machine learning and NLP techniques to achieve high accuracy in sentiment analysis results, enhancing the reliability of the system.

6. Customization Options : Offer customization features that allow users to fine-tune sentiment analysis models or adjust parameters to suit specific use cases or domains.

7. Scalability : Build a scalable architecture capable of handling large volumes of data, ensuring the system remains responsive and efficient as user demand grows.

8. Educational Purpose : Serve as a learning platform for exploring Python, machine learning, NLP, and web development concepts in a practical project setting, enabling students to gain hands-on experience in building sentiment analysis systems.

SCOPE

1. Sentiment Analysis for Text and Video : This encompasses the core functionality of the project, which is to develop algorithms capable of analysing sentiments in both textual and video content. It involves processing input data, extracting relevant features, and applying machine learning or NLP techniques to classify the sentiment as positive, negative, or neutral.

2. Integration of Flask Bootstrap for Web Interface : Flask Bootstrap will be used to create a visually appealing and responsive web interface for users to interact with the sentiment analysis system. This includes designing input forms for text, video upload functionality, navigation bars, and other UI elements. Flask Bootstrap simplifies the process of creating a polished web interface by providing pre-designed templates and components.

3. Text Blob for Natural Language Processing (NLP) : Text Blob is a Python library that simplifies various NLP tasks, including sentiment analysis. By utilizing Text Blob, the project can leverage pre-trained models and straightforward APIs to analyse the sentiment of textual data. Text Blob handles tokenization, part-of-speech tagging, and other NLP tasks, making it an efficient choice for sentiment analysis.

4. Real-time Analysis Capabilities : Real-time analysis enables the system to process and analyse incoming data as it arrives, providing immediate feedback on sentiment trends. This involves implementing efficient algorithms and processing pipelines to handle data streams in real-time, ensuring prompt analysis and response to user inputs or updates in content.

5. Support for Multiple Languages : To cater to a diverse user base and content sources, the system should support sentiment analysis in multiple languages. This requires incorporating language detection capabilities and adapting the sentiment analysis models to handle different languages effectively. Techniques such as language translation or utilizing multilingual pre-trained models may be employed to achieve this.

6. Customizable Sentiment Analysis Models : Offering customization options allows users to fine-tune sentiment analysis models or adjust parameters based on specific requirements or domains. This could involve providing options to select different sentiment analysis algorithms, adjust threshold values for sentiment classification, or incorporate domain-specific lexicons or features to improve accuracy.

7. Accuracy and Performance Optimization : Achieving high accuracy in sentiment analysis results is crucial for the effectiveness of the system. This involves optimizing machine learning models, feature selection, and parameter tuning to improve the accuracy of sentiment classification. Additionally, optimizing the performance of the system ensures efficient processing of large volumes of data while maintaining responsiveness.

8. Scalability and Efficiency : Designing a scalable architecture ensures that the system can handle increasing loads of data and users without compromising performance. This involves considerations such as distributed computing, load balancing, caching, and resource allocation to ensure efficient utilization of hardware resources and responsiveness under varying loads.

9. User Authentication and Data Security : Implementing user authentication mechanisms ensures that only authorized users can access the system, protecting sensitive content from unauthorized access. Data security measures such as encryption, access control, and secure communication protocols are implemented to safeguard user data and maintain privacy.

10. Visualization of Sentiment Trends : Visualizations or reports help users interpret sentiment analysis results more effectively by presenting trends and insights in a graphical format. This

could include charts, graphs, heatmaps, or other visual representations of sentiment trends over time, across different sources, or in comparison with other metrics.

11. Error Handling and Logging : Robust error handling mechanisms and logging are essential for identifying and diagnosing issues that may arise during system operation. This involves capturing and logging errors, warnings, and other relevant information to facilitate debugging, troubleshooting, and maintenance of the system.

12. Documentation and User Support : Providing comprehensive documentation and user support resources ensures that users can effectively utilize the system and troubleshoot any issues they encounter. This includes user manuals, tutorials, FAQs, and online forums or help desks where users can seek assistance and guidance from the development team or community. Clear and accessible documentation improves user satisfaction and adoption of the system.

METHODOLOGY

1. Requirements Gathering : Defining the project's objectives, scope, and stakeholders is crucial for understanding what the system needs to achieve. Gathering requirements involves identifying functional aspects, such as the ability to analyse text and video, as well as non-functional requirements like performance and security. Prioritizing requirements helps in creating a clear roadmap for development and ensures that the project stays focused on delivering the most important features within the specified timeline and resources.

2. Research and Planning : Researching sentiment analysis algorithms, NLP techniques, and web development frameworks is essential for making informed decisions about the project's architecture and technologies. This phase involves evaluating existing libraries and tools to leverage for text and video analysis, as well as planning the system's architecture to ensure scalability, maintainability, and performance. By conducting thorough research and planning, the project team can lay a strong foundation for successful implementation and deployment.

3. Design : Designing the user interface involves creating wireframes or mockups to visualize the layout and functionality of the web application. Incorporating Flask Bootstrap allows for designing a visually appealing and responsive interface with pre-designed components and

styles. Additionally, designing the sentiment analysis pipeline involves planning the data flow, preprocessing steps, feature extraction techniques, and model selection for both text and video analysis. A well-designed architecture ensures that the system can efficiently process data and provide accurate sentiment analysis results.

4. Implementation : The implementation phase involves actual coding and development of the web application and sentiment analysis algorithms. Using Flask, developers build the web application's backend and frontend components, integrating Flask Bootstrap for UI elements. Text Blob is utilized for text sentiment analysis, while custom or pre-trained models may be employed for video analysis. Implementing real-time analysis capabilities may require asynchronous programming techniques to handle incoming data streams efficiently. Ensuring code quality through best practices and documentation is vital for maintaining and extending the system in the future.

5. Testing : Testing is a critical phase to validate the functionality, correctness, and reliability of the system. Unit tests, integration tests, and end-to-end tests are developed to verify individual components and the system. Functional testing ensures that the system meets the specified requirements, while usability testing gathers feedback on the user interface and experience. Thorough testing helps identify and fix issues early in the development cycle, reducing the risk of bugs and ensuring a high-quality product upon deployment.

6. Deployment : Deploying the web application involves making it accessible to users on a hosting environment such as a cloud platform or dedicated server. Configuration of the deployment environment includes setting up databases, web servers, and security configurations to ensure the application's proper functioning and data integrity. Deployment testing is performed to validate that the application works correctly in the production environment, and any issues are addressed before the system is made available to users.

7. Documentation and Training : Documentation is essential for providing guidance on system architecture, design decisions, and implementation details for future reference and maintenance. User documentation, including manuals, guides, and tutorials, helps users understand how to use the system effectively. Providing training sessions or workshops for users and stakeholders familiarizes them with the system's features and functionalities, promoting adoption and usage.

TESTING TECHNOLOGIES USED

1. Requirements Gathering : Defining the project's objectives, scope, and stakeholders is crucial for understanding what the system needs to achieve. Gathering requirements involves identifying functional aspects, such as the ability to analyse text and video, as well as non-functional requirements like performance and security. Prioritizing requirements helps in creating a clear roadmap for development and ensures that the project stays focused on delivering the most important features within the specified timeline and resources.

2. Research and Planning : Researching sentiment analysis algorithms, NLP techniques, and web development frameworks is essential for making informed decisions about the project's architecture and technologies. This phase involves evaluating existing libraries and tools to leverage for text and video analysis, as well as planning the system's architecture to ensure scalability, maintainability, and performance. By conducting thorough research and planning, the project team can lay a strong foundation for successful implementation and deployment.

3. Design : Designing the user interface involves creating wireframes or mockups to visualize the layout and functionality of the web application. Incorporating Flask Bootstrap allows for designing a visually appealing and responsive interface with pre-designed components and styles. Additionally, designing the sentiment analysis pipeline involves planning the data flow, preprocessing steps, feature extraction techniques, and model selection for both text and video analysis. A well-designed architecture ensures that the system can efficiently process data and provide accurate sentiment analysis results.

4. Implementation : The implementation phase involves actual coding and development of the web application and sentiment analysis algorithms. Using Flask, developers build the web application's backend and frontend components, integrating Flask Bootstrap for UI elements. Text Blob is utilized for text sentiment analysis, while custom or pre-trained models may be employed for video analysis. Implementing real-time analysis capabilities may require asynchronous programming techniques to handle incoming data streams efficiently. Ensuring code quality through best practices and documentation is vital for maintaining and extending the system in the future.

5. Testing : Testing is a critical phase to validate the functionality, correctness, and reliability of the system. Unit tests, integration tests, and end-to-end tests are developed to verify individual components and the system as a whole. Functional testing ensures that the system meets the specified requirements, while usability testing gathers feedback on the user interface and experience. Thorough testing helps identify and fix issues early in the development cycle, reducing the risk of bugs and ensuring a high-quality product upon deployment.

6. Deployment : Deploying the web application involves making it accessible to users on a hosting environment such as a cloud platform or dedicated server. Configuration of the deployment environment includes setting up databases, web servers, and security configurations to ensure the application's proper functioning and data integrity. Deployment testing is performed to validate that the application works correctly in the production environment, and any issues are addressed before the system is made available to users.

7. Documentation and Training : Documentation is essential for providing guidance on system architecture, design decisions, and implementation details for future reference and maintenance. User documentation, including manuals, guides, and tutorials, helps users understand how to use the system effectively. Providing training sessions or workshops for users and stakeholders familiarizes them with the system's features and functionalities, promoting adoption and usage. Clear and comprehensive documentation ensures that users can utilize the system efficiently and effectively.

8. Maintenance and Support : Maintenance involves monitoring the system for performance issues, security vulnerabilities, and user feedback, and addressing them promptly to ensure the system's reliability and usability. Releasing updates and patches as needed improves functionality, fixes bugs, and addresses user feedback, ensuring that the system remains up-to-date and responsive to user needs. Ongoing support, including addressing user queries and troubleshooting issues, fosters user satisfaction and confidence in the system, promoting its continued usage and success.

CONTRIBUTION

1. Advancing Sentiment Analysis Technology : By developing algorithms capable of analysing sentiments in both textual and video content, your project contributes to advancing sentiment analysis technology. This can lead to improvements in understanding human emotions and opinions expressed through various media formats.

2. Enhancing User Understanding and Decision-Making : SentiSight empowers users to gain insights into sentiment trends, enabling them to better understand public opinion, customer feedback, and social media sentiment. This understanding can inform decision-making processes in various domains, including marketing, customer service, and product development.

3. Facilitating Multi-language Sentiment Analysis : Supporting sentiment analysis in multiple languages expands the reach and applicability of the technology, allowing users to analyse sentiments expressed in diverse linguistic contexts. This contribution promotes inclusivity and enables sentiment analysis across global markets and communities.

4. Providing Real-time Analysis Capabilities : SentiSight real-time analysis capabilities enable users to monitor sentiment trends as they unfold, providing timely insights that can inform immediate actions or interventions. This contribution enhances the agility and responsiveness of users in addressing sentiment-related issues or opportunities.

5. Supporting Education and Learning : As a college project, SentiSight contributes to the education and learning of students involved in its development. It provides hands-on experience in applying concepts and techniques from Python, machine learning, NLP, and web development to solve real-world problems, preparing students for future careers in AI and software engineering.

6. Fostering Innovation and Collaboration : SentiSight encourages innovation and collaboration by providing a platform for experimentation and exploration in sentiment analysis. It invites contributions from researchers, developers, and practitioners interested in advancing the field of NLP and sentiment analysis through open-source development and collaboration.

7. Promoting Data-driven Decision Making : By providing actionable insights derived from sentiment analysis, SentiSight promotes data-driven decision-making processes across various domains. Users can leverage sentiment analysis results to make informed decisions, tailor their strategies, and address issues proactively based on data-driven insights rather than intuition or assumptions.

Objective and Scope of Project

SentiSight is a sentiment analysis website designed to detect positive or negative sentiment in text data. Leveraging Python for backend development and Flask as the web framework, SentiSight provides users with a seamless experience for inputting text and receiving sentiment analysis results. The frontend interface is crafted using HTML, CSS, and JavaScript, ensuring an intuitive and responsive user experience. Through machine learning techniques and natural language processing algorithms, SentiSight empowers users to gain insights into sentiment trends and make informed decisions based on textual data analysis.

Objective :

The primary objective of the SentiSight project is to develop a robust sentiment analysis website capable of detecting positive or negative sentiment in text data. In today's digital age, the vast amount of textual information generated on social media platforms, review websites, and other online sources presents both opportunities and challenges for individuals and organizations. Understanding the sentiment expressed in this text data is crucial for making informed decisions, managing brand reputation, assessing customer feedback, and monitoring public opinion.

By leveraging the power of Python, Flask, HTML, CSS, and JavaScript, the SentiSight website aims to provide users with a user-friendly platform for conducting sentiment analysis on textual content. Python serves as the primary programming language for implementing the backend logic and sentiment analysis algorithms. Flask, a lightweight web framework, facilitates the development of the web application, enabling seamless integration of backend functionalities with the frontend interface. HTML, CSS, and JavaScript are utilized to design and enhance the user interface, ensuring an intuitive and engaging user experience.

The project's focus on detecting positive or negative sentiment addresses a common need across various domains, including marketing, customer service, market research, and social media analysis. By accurately classifying text data into positive or negative sentiment categories, SentiSight enables users to gain valuable insights into customer sentiment, identify emerging trends, detect sentiment shifts, and evaluate the effectiveness of marketing campaigns or product launches.

One of the key goals of the SentiSight project is to leverage machine learning techniques and natural language processing (NLP) algorithms to achieve high accuracy in sentiment analysis. Through the application of supervised learning algorithms, sentiment analysis models are trained on labeled datasets to recognize patterns and linguistic cues associated with positive or negative sentiment expressions. Additionally, NLP techniques such as tokenization, part-of-speech tagging, and sentiment lexicon-based analysis are employed to extract meaningful features from textual data and enhance the performance of the sentiment analysis system.

Another objective of the SentiSight project is to provide users with real-time feedback on sentiment analysis results. By implementing efficient processing pipelines and asynchronous programming techniques, the website ensures prompt analysis and response to user inputs. Real-time analysis capabilities enable users to monitor sentiment trends as they unfold, facilitating timely interventions and decision-making processes.

Furthermore, the project aims to support customization and scalability to meet the diverse needs of users and accommodate growing volumes of textual data. Customization options allow users to fine-tune sentiment analysis models, adjust parameters, and integrate domain-specific knowledge or lexicons to improve the accuracy and relevance of analysis results. Scalability considerations ensure that the website can handle increasing loads of data and user traffic without compromising performance or responsiveness.

Overall, the objective of the SentiSight project is to develop an advanced sentiment analysis website that empowers users to extract valuable insights from textual data, make informed decisions, and stay ahead in today's fast-paced digital landscape. Through a combination of cutting-edge technologies, robust algorithms, and user-centered design principles, SentiSight aims to become a valuable tool for sentiment analysis across various industries and applications.

Scope :

The scope of the SentiSight project encompasses the development of a comprehensive sentiment analysis website designed to detect positive or negative sentiment in text data. Leveraging a combination of Python, Flask, HTML, CSS, and JavaScript, the project aims to deliver a user-friendly platform for conducting sentiment analysis across various domains and applications.

The primary focus of the project is to provide users with an intuitive and engaging interface for inputting text data and receiving sentiment analysis results in real-time. This involves designing and implementing a responsive web application that integrates backend functionalities with frontend components seamlessly. Flask serves as the web framework for developing the backend logic, while HTML, CSS, and JavaScript are utilized to design and enhance the frontend user interface.

The sentiment analysis functionality of the website is powered by machine learning techniques and natural language processing (NLP) algorithms. The scope includes implementing supervised learning algorithms to train sentiment analysis models on labeled datasets, enabling them to recognize patterns and linguistic cues associated with positive or negative sentiment expressions. Additionally, NLP techniques such as tokenization, part-of-speech tagging, and sentiment lexicon-based analysis are employed to extract meaningful features from textual data and enhance the accuracy of sentiment analysis results.

Customization options are provided to users, allowing them to fine-tune sentiment analysis models, adjust parameters, and integrate domain-specific knowledge or lexicons to improve the relevance and accuracy of analysis results. The scope also includes supporting multi-language sentiment analysis to cater to diverse linguistic contexts and user preferences, further enhancing the versatility and applicability of the website.

Scalability is a key consideration in the scope of the project, ensuring that the website can handle increasing volumes of data and user traffic without compromising performance or responsiveness. This involves designing and implementing a scalable architecture that can accommodate growing demands and user needs while maintaining high levels of reliability and efficiency.

Overall, the scope of the SentiSight project encompasses the development of a robust sentiment analysis website that empowers users to extract valuable insights from textual data, make informed decisions, and stay ahead in today's fast-paced digital landscape. By leveraging advanced technologies, robust algorithms, and user-centered design principles, SentiSight aims to become a valuable tool for sentiment analysis across various industries and applications.

Theoretical Background Definition of Problem

Sentiment analysis, also known as opinion mining, is a field within natural language processing (NLP) that focuses on identifying and extracting subjective information, such as opinions, emotions, and attitudes, from text data. It involves analyzing and categorizing text based on the sentiment expressed, typically into positive, negative, or neutral classes.

Natural language processing (NLP) is a branch of artificial intelligence that deals with the interaction between computers and human language. It encompasses a wide range of tasks, including text processing, speech recognition, machine translation, and sentiment analysis. NLP techniques enable computers to understand, interpret, and generate human language in a meaningful way.

Sentiment analysis relies heavily on NLP methods for tasks such as tokenization, part-of-speech tagging, parsing, and semantic analysis. These techniques help in breaking down text into smaller units, understanding the grammatical structure, and extracting meaningful features that can be used for sentiment classification.

There are various approaches to sentiment analysis, including rule-based methods, machine learning techniques, and hybrid approaches. Rule-based methods involve defining a set of rules or lexicons that associate specific words or phrases with sentiment scores. Machine learning techniques, on the other hand, involve training models on labeled data to learn patterns and make sentiment predictions on new, unseen data. Popular machine learning algorithms used in sentiment analysis include Naive Bayes, Support Vector Machines (SVMs), and deep learning models like Recurrent Neural Networks (RNNs) and Transformers.

Definition of the Problem

SentiSight aims to solve the problem of accurately and efficiently analyzing the sentiment expressed in textual data from various sources, such as social media posts, product reviews, customer feedback, and online discussions. With the massive amount of text-based data generated daily, it becomes increasingly challenging to manually analyze and extract meaningful insights from this data.

SentiSight goal is to provide a robust and scalable sentiment analysis system that can process large volumes of text data in real-time, accurately identifying and quantifying the sentiment expressed. By leveraging machine learning and natural language processing techniques, SentiSight seeks to provide businesses, researchers, and individuals with valuable insights into sentiment trends, enabling them to make informed decisions and take appropriate actions based on the sentiment analysis results.

Challenges and Complexities in Sentiment Analysis

While sentiment analysis has made significant strides, it remains a complex and challenging task due to the intricacies of human language and the subjective nature of sentiment expression. Some of the key challenges and complexities involved in sentiment analysis include:

1. Linguistic Nuances and Context Ambiguity: The same words or phrases can convey different sentiments depending on the context in which they are used. Sentiment analysis models need to account for linguistic nuances, such as sarcasm, irony, and figurative language, which can be difficult to interpret accurately.
2. Domain and Topic Specificity: Sentiment can vary significantly across different domains and topics. For example, the sentiment expressed in a product review may differ from that expressed in a political discussion. Sentiment analysis models often need to be fine-tuned or retrained for specific domains to achieve better accuracy.
3. Multilingual and Cross-Cultural Differences: Sentiment can be expressed differently across languages and cultures. Developing robust sentiment analysis models that can handle multiple languages and cultural nuances is a significant challenge.

4. Evolving Language and Slang: Language is constantly evolving, with new terms, slang, and expressions emerging regularly. Sentiment analysis models need to be continuously updated and adapted to accurately capture these changes.
5. Data Quality and Availability: The performance of sentiment analysis models heavily relies on the quality and quantity of training data available. Obtaining high-quality, labeled data for sentiment analysis can be a time-consuming and expensive process.
6. Computational Complexity and Scalability: As the volume of text data continues to grow, sentiment analysis systems need to be computationally efficient and scalable to handle large datasets in a timely manner.

Despite these challenges, ongoing research and advancements in natural language processing, machine learning, and deep learning techniques are continuously improving the accuracy and robustness of sentiment analysis systems, making them more capable of handling the complexities of human language and sentiment expression.

System Analysis & Design vis-a-vis User Requirements

User Requirements Elicitation

User requirements elicitation is a critical phase in the development of any software system, including SentiSight. It involves gathering, analyzing, and documenting the needs, expectations, and constraints of the end-users and stakeholders. For SentiSight, the user requirements elicitation process was conducted in a systematic and comprehensive manner:

1. Stakeholder Identification and Analysis: The first step was to identify and analyze the key stakeholders who would be using or affected by the sentiment analysis system. This included individuals, businesses, researchers, and organizations from various domains interested in analyzing sentiment trends from textual data. Stakeholder analysis involved understanding their roles, responsibilities, and level of involvement in the project.
2. Requirements Gathering Techniques: Multiple techniques were employed to gather requirements from the stakeholders, ensuring a diverse and comprehensive perspective. These techniques included:
    - **Interviews:** One-on-one interviews were conducted with key stakeholders to understand their specific needs, pain points, and expectations from the sentiment analysis system.
    - **Surveys**: Online and offline surveys were distributed to a broader audience to gather quantitative and qualitative data on user preferences, usage scenarios, and desired features.
    - **Focus Groups:** Interactive focus group sessions were organized, bringing together stakeholders with similar backgrounds or interests to discuss their requirements and provide feedback.
    - **Observation:** Existing sentiment analysis practices and workflows were observed to identify potential areas for improvement and to understand the real-world context in which the system would be used.

3. User Persona Development: Based on the gathered information, user personas were created to represent the different types of users and their specific needs, goals, preferences, and behaviors. These personas helped in designing and tailoring the system to meet the diverse requirements of the target user base.
4. Requirement Analysis and Categorization: The gathered requirements were thoroughly analyzed to identify any ambiguities, conflicts, gaps, or overlaps. This analysis helped in refining and clarifying the requirements. The requirements were then categorized into functional requirements (features and capabilities the system should provide) and non-functional requirements (performance, security, scalability, usability, and other quality constraints).
5. Prioritization and Validation: The categorized requirements were prioritized based on their importance, feasibility, and alignment with the project goals and objectives. This prioritization process involved input from stakeholders and subject matter experts. The prioritized requirements were then validated with the stakeholders to ensure their accuracy, completeness, and relevance.
6. Documentation: Throughout the requirements elicitation process, comprehensive documentation was maintained, including requirement specifications, user stories, and use case diagrams. This documentation served as a reference for the subsequent phases of system analysis and design.

System Analysis

System analysis involves examining the elicited user requirements, identifying the scope and boundaries of the system, and determining the best approach to meet those requirements.

For SentiSight, the system analysis phase was conducted with the following activities:

1. Requirements Review and Refinement: The documented requirements were thoroughly reviewed and refined to ensure their clarity, consistency, and feasibility. Any contradictions or overlaps were addressed, and additional requirements were identified or modified based on stakeholder feedback and technical constraints.

2. Feasibility Study: A comprehensive feasibility study was conducted to assess the technical, economic, operational, and legal viability of the proposed sentiment analysis system. This study involved:

1. Technical Feasibility: Evaluating the available technologies, tools, libraries, and frameworks for sentiment analysis, natural language processing, and web development, and assessing their suitability for the project.
2. Economic Feasibility: Analyzing the project's budget, cost-benefit analysis, and return on investment (ROI) projections to ensure its financial viability.
3. Operational Feasibility: Assessing the available resources, infrastructure, and expertise required for developing, deploying, and maintaining the sentiment analysis system.
4. Legal and Regulatory Feasibility: Evaluating compliance with relevant data privacy and security regulations, as well as any industry-specific guidelines or standards.

3. System Modeling: Various modeling techniques were employed to create a visual representation of the system's components, data flow, and interactions. These included:

1. Data Flow Diagrams (DFDs): DFDs were used to represent the flow of data through the system, highlighting the processes, data stores, and external entities involved.
2. Entity-Relationship Diagrams (ERDs): ERDs were developed to model the database structure, depicting the entities, attributes, and relationships between them.
3. Use Case Diagrams: Use case diagrams were created to illustrate the different actors (users) and their interactions with the system, depicting the system's functionalities from a user's perspective.

4. Architecture Design: Based on the system modeling and analysis, the high-level architecture of the sentiment analysis system was designed. This architecture considered factors such as scalability, performance, maintainability, and modularity. Architectural patterns and principles, such as Model-View-Controller (MVC), microservices, or serverless architectures, were evaluated and selected based on their suitability for the project requirements.

5. Risk Analysis and Mitigation: Potential risks associated with the development, implementation, and operation of the sentiment analysis system were identified and analyzed. These risks included technical risks (e.g., integration challenges, performance bottlenecks), operational risks (e.g., resource constraints, data quality issues), and external risks (e.g., regulatory changes, market shifts). Mitigation strategies and contingency plans were developed to address these risks proactively.

6. Prototyping and Proof of Concept: Where necessary, prototypes or proof-of-concept implementations were developed to validate the feasibility of specific components or technologies, such as sentiment analysis algorithms or natural language processing pipelines. These prototypes helped in identifying potential issues and refining the system design before proceeding with full-scale development.

System Design

The system design phase translates the user requirements and system analysis into a detailed blueprint for the sentiment analysis system.

This phase involved the following activities:

1. **Component Design and Decomposition:** The system was decomposed into various components or modules, each responsible for specific functionalities. This included components such as:
    - Frontend User Interface: Responsible for displaying the user interface, accepting user inputs, and presenting sentiment analysis results.
    - Backend Sentiment Analysis Engine: Responsible for processing textual data, applying sentiment analysis algorithms, and generating sentiment scores or classifications.
    - Data Preprocessing Module: Responsible for cleaning, tokenizing, and normalizing textual data before sentiment analysis.
    - Database Module: Responsible for storing and retrieving textual data, sentiment analysis results, and user-related information.
    - Integration and Communication Modules: Responsible for facilitating communication and data exchange between different components and external systems.
2. **Interface Design:** The user interfaces for interacting with the sentiment analysis system were designed, focusing on usability, accessibility, and intuitive user experience. This involved creating wireframes, mockups, and prototypes of the web interfaces, ensuring consistency with industry standards and best practices.
3. **Database Design**: The database schema and data structures were designed to store and manage the textual data, sentiment analysis results, user preferences, and any other relevant information. This involved identifying the entities, attributes, and relationships, as well as defining appropriate data types, constraints, and indexing strategies for optimal performance.

4. **Algorithm Design and Selection:** Appropriate algorithms and techniques for sentiment analysis, such as machine learning models (e.g., Naive Bayes, Support Vector Machines, Deep Learning), natural language processing pipelines (e.g., tokenization, stemming, part-of-speech tagging), and data preprocessing methods (e.g., text normalization, stop word removal), were selected and designed based on the requirements and constraints.
5. **Integration and Deployment Planning:** Strategies for integrating the various components of the sentiment analysis system were developed, considering factors such as inter-component communication, data exchange formats, and synchronization mechanisms. Additionally, plans for deployment, testing, monitoring, and maintenance were established, including infrastructure provisioning, containerization, and continuous integration/continuous deployment (CI/CD) pipelines.
6. **Security and Data Privacy Considerations:** Appropriate security measures and data privacy protocols were incorporated into the system design to ensure the confidentiality, integrity, and availability of user data and sentiment analysis results. This included implementing authentication and authorization mechanisms, data encryption, secure communication channels, and compliance with relevant data privacy regulations (e.g., GDPR, CCPA).
7. **Performance and Scalability Considerations:** The system design took into account performance and scalability requirements, addressing aspects such as load balancing, caching mechanisms, horizontal and vertical scaling strategies, and the use of cloud computing resources or serverless architectures to accommodate growing volumes of data and user traffic.
8. **Documentation:** Comprehensive documentation was created throughout the system design phase, including technical specifications, component diagrams, data models, algorithm descriptions, and integration guides. This documentation served as a reference for the development team, facilitated knowledge transfer, and supported future maintenance and enhancement efforts.
9. **Testing and Quality Assurance Planning:** A comprehensive testing strategy was developed, encompassing unit testing, integration testing, system testing, and user acceptance testing. This strategy aimed to ensure the correctness, reliability, and quality of the sentiment analysis system. Appropriate testing frameworks, tools, and environments were selected, and test cases were designed to validate functional and non-functional requirements, edge cases, and error handling scenarios.

10. **DevOps and Continuous Integration/Continuous Deployment (CI/CD):** Practices and tools related to DevOps and CI/CD were incorporated into the system design to streamline the development, testing, and deployment processes. This included version control systems (e.g., Git), issue tracking tools, automated build and deployment pipelines, and monitoring and logging frameworks.
11. **System Evolution and Maintenance Strategy**: A strategy for system evolution and maintenance was established to ensure the sentiment analysis system remains up-to-date and adaptable to changing requirements, technological advancements, and emerging trends in sentiment analysis and natural language processing. This strategy involved planning for regular software updates, model retraining, and incorporating mechanisms for user feedback and feature requests.
12. **Documentation and Knowledge Transfer:** Throughout the system design phase, comprehensive documentation was maintained, including technical specifications, architectural diagrams, data models, component descriptions, integration guides, and user manuals. This documentation served as a central knowledge base for the development team, stakeholders, and future maintainers, facilitating effective knowledge transfer and ensuring a smooth transition between project phases.

The system design phase involved close collaboration between various stakeholders, including domain experts, software architects, developers, and end-users. Iterative and incremental approaches were followed, allowing for continuous refinement and improvement based on feedback and changing requirements. Regular design reviews and walkthroughs were conducted to ensure the system's alignment with user needs, technical feasibility, and industry best practices.

By following a structured and methodical approach to system design and analysis, SentiSight aimed to deliver a robust, scalable, and user-friendly sentiment analysis system that effectively addressed the challenges and complexities involved in sentiment analysis while meeting the diverse needs of its users. The comprehensive design documentation and well-defined system architecture provided a solid foundation for the subsequent development, testing, and deployment phases of the project.

System Planning (PERT Chart)

To outline a PERT (Program Evaluation and Review Technique) chart for the SentiSight project, we need to identify the major tasks or activities involved, their dependencies, and the estimated time required to complete each task. Here's a step-by-step approach to outlining the PERT chart:

1. Identify the major tasks or activities: List out all the major tasks or activities required for the SentiSight project.

2. Determine the task dependencies: Analyze the dependencies between the tasks. Some tasks may need to be completed before others can start. For example, Requirements Gathering must be completed before System Analysis and Design, and System Design must be completed before Development can begin.

3. Estimate task durations: For each task, estimate the optimistic time (the shortest time it could take), the most likely time, and the pessimistic time (the longest time it could take). These estimates can be based on historical data, expert opinions, or educated guesses.

4. Calculate the expected time for each task: Using the formula: Expected Time = (Optimistic Time + 4 × Most Likely Time + Pessimistic Time) / 6, calculate the expected time for each task.

5. Determine the critical path: The critical path is the longest sequence of dependent tasks that determines the minimum time required to complete the project. Identify the tasks that lie on the critical path, as any delay in these tasks will directly impact the project's completion time.

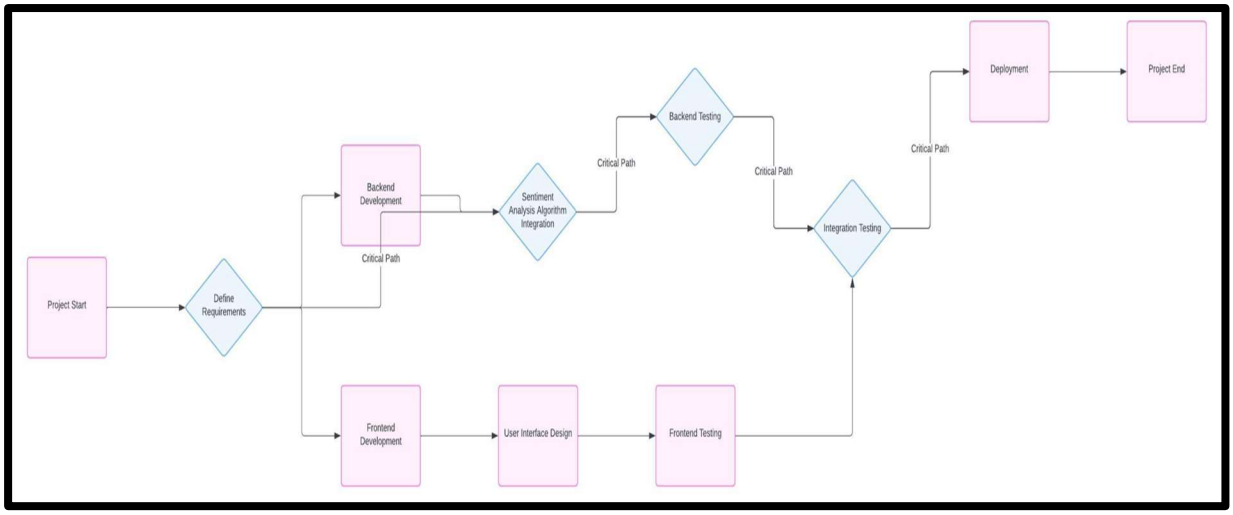
6. Create the PERT chart: With the identified tasks, dependencies, and expected times, create a visual representation of the PERT chart. This can be done using project management software or drawing tools.

- Represent each task as a node or box in the chart.
- Connect the dependent tasks with arrows, indicating the direction of the dependency.
- Label each task with its expected time or duration.
- Highlight the critical path tasks or activities.

7. Add milestones and deadlines: Identify important milestones or deadlines in the project, such as project kickoff, requirements sign-off, design review, alpha/beta testing, and final deployment. Add these milestones to the PERT chart to track progress and ensure timely completion of critical tasks.

8. Review and refine: Review the PERT chart with the project team and stakeholders. Refine the task dependencies, durations, and critical path as needed based on feedback and changing project requirements.

By outlining the PERT chart, you can visualize the project timeline, identify potential bottlenecks or areas of risk, and plan for resource allocation and project monitoring effectively. The PERT chart serves as a valuable tool for project management, facilitating communication among team members, and ensuring the timely delivery of the SentiSight project.



The PERT chart starts with the "Project Start" milestone, followed by the "Define Requirements" task, which is a crucial initial step to gather and document the user requirements for the sentiment analysis system.

From the "Define Requirements" task, the chart branches into two parallel paths:

1. Frontend Development Path:

1. Frontend Development: This task involves the development of the user interface and frontend components of the sentiment analysis application.
2. User Interface Design: This task focuses on designing the user interface, including wireframes, mockups, and ensuring a user-friendly experience.
3. Frontend Testing: This task involves testing the frontend components, user interface, and functionality.

2. Backend Development Path:

1. Backend Development: This task covers the development of the backend components, including the sentiment analysis engine, data processing, and server-side logic.
2. Sentiment Analysis Algorithm Integration: This task involves integrating and implementing the sentiment analysis algorithms and natural language processing techniques into the backend.
3. Backend Testing: This task focuses on testing the backend components, sentiment analysis functionality, and integration with other modules.

Both the Frontend Testing and Backend Testing tasks converge into the "Integration Testing" task, which is marked as a critical path. The critical path represents the longest sequence of dependent tasks that determine the minimum time required to complete the project.

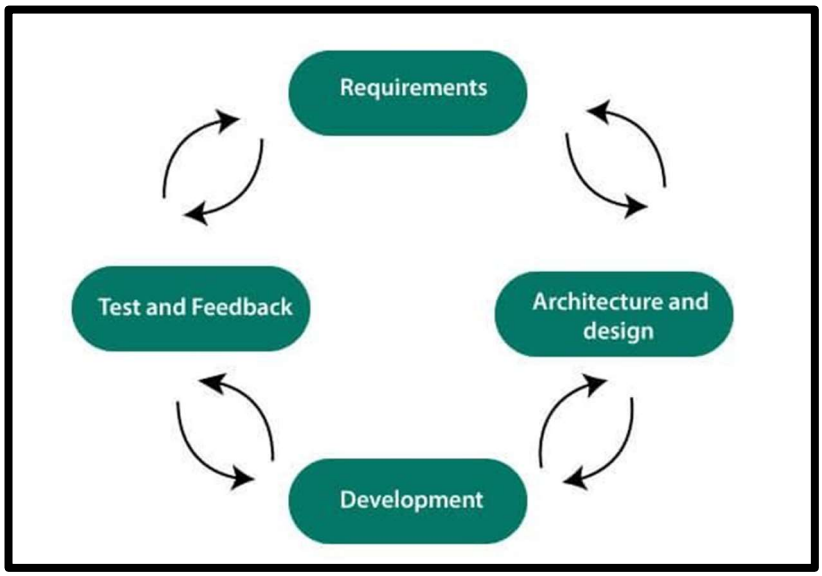
After successful Integration Testing, the project moves to the "Deployment" task, where the sentiment analysis system is deployed to the production environment.

Finally, the PERT chart concludes with the "Project End" milestone, signifying the completion of the SentiSight project. It's important to note that the PERT chart provides a visual representation of the project timeline, task dependencies, and critical path. It helps in identifying potential bottlenecks, facilitating resource allocation, and ensuring timely delivery of the project by focusing on the critical path tasks.

METHODOLOGY

AGILE SDLC MODEL

Agile Software Development Life Cycle (SDLC) is the combination of both iterative and incremental process models. It focuses on process adaptability and customer satisfaction by rapiddelivery of working software product. Agile SDLC breaks down the product into small incremental builds. These builds are provided into iterations, as shown.



In the agile SDLC development process(Figure 2), the customer is able to see the result and understand whether he/she is satisfied with it or not. This is one of the advantages of the agile SDLC model. One of its disadvantages is the absence of defined requirements so, it is difficult toestimate the resources and development cost.

Each iteration of agile SDLC consists of cross-functional teams working on various phases:

- Requirement gathering and analysis
- Design the requirements
- Construction/ iteration
- Deployment
- Testing
- Feedback

Requirements gathering and analysis:

In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

Design the requirements:

When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features andshow how it will apply to your existing system.

Construction/ Iteration:

When the team defines the requirements, the work begins. The designers and developers start working on their project. The aims of designers and developers deploy the working product within the estimated time. The product will go into various stages of improvement, so it includessimple, minimal functionality.

Deployment:

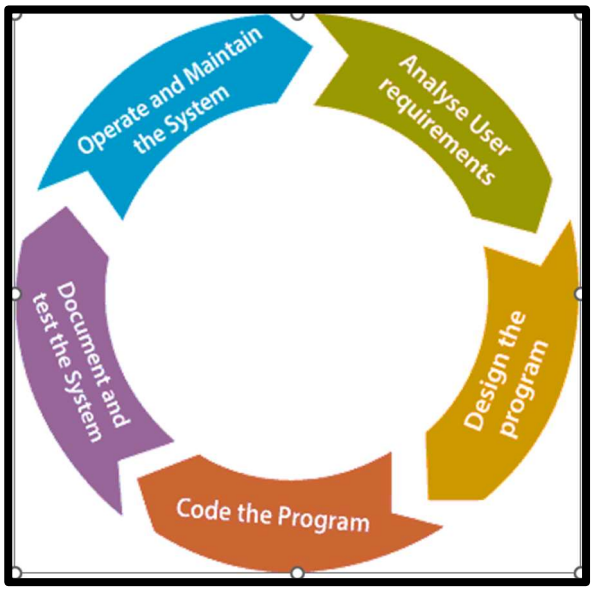
In this phase, the team issues a product for the user's work environment.

Testing:

In this phase, the Quality Assurance team examine the product's performance and look for thebug.

Feedback

After releasing of the product, the last step is to feedback it. In this step, the team receives feedbackabout the product and works through the feedback.



Agile SDLC Process Flow

1. **Concept:** Project are imagined and prioritized.
2. **Inception:** Team members are created, funding is put in place, and basic environments andrequirements are discussed.
3. **Iteration/Constriction:** The software development team works to deliver workingsoftware. It is based on requirement and feedback.
4. **Release:** Perform quality assurance (QA) testing, provides internal and external training,documentation development, and final version of iteration into the product.
5. **Production:** It is ongoing support of the software.

Advantages of Agile SDLC

1. Project is divided into short and transparent iterations.
2. It has a flexible change process.
3. It minimizes the risk of software development.
4. Quick release of the first product version.
5. The correctness of functional requirement is implemented into the development process.
6. Customer can see the result and understand whether he/she is satisfied with it or not.

Disadvantages of Agile SDLC

1. The development team should be highly professional and client-oriented.
2. New requirement may be a conflict with the existing architecture.
3. With further correction and change, there may be chances that the project will cross theexpected time.
4. There may be difficult to estimate the final coast of the project due to constant iteration.
5. A defined requirement is absent.

System Implementation

HTML (User Interface)

The hypertext markup language (HTML) is a simple markup language. Used to create a hypertext documents that are portable from one platform to another HTML documents are SGML (Standard generalized mark up language) documents with generic semantics that are appropriate for representing information from a wide range of applications. This specification defines HTML version 3.2. HTML 3.2 aims to capture recommended practice as of early ’96 and as such a replacement for HTML2.0 (RFC 1866). A set of instructions embedded in a document is called mark up language. These instructions describe what the document text means and how it should look like in a display. Hyper Text Mark Up language (HTML) is the language used to encode World Wide Web documents.

CSS (Cascading Style Sheets)

CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation of a document written in markup language like HTML. It enables web developers to control the layout, appearance, and behavior of HTML elements on a webpage.

Syntax: CSS consists of a set of rules, each composed of a selector and one or more declarations. The selector specifies which HTML elements the rule applies to, and the declarations define the style properties of those elements.

```
h1
{
   Color: blue;
   Font-size 24px;
}
```

JavaScript

Javascript is a high-level, interpreted programming language primarily used for creating dynamic and interactive web pages. It is one of the core technologies of the World Wide Web, along with HTML and CSS.

In my SentiSight project, JavaScript is used for several purposes, including:

1. **User Interface Interactions**: JavaScript is used to enhance the user experience by adding interactivity and responsiveness to the web application. In the `analyzed.html` file, JavaScript is used to handle the sentiment analysis results returned from the server and update the DOM (Document Object Model) with the appropriate values and styles. This includes updating the sentiment type, polarity, positive/neutral/negative percentages, sentiment score progress bar, and overall display of the results.
2. **Asynchronous Data Fetching:** JavaScript's ability to perform asynchronous operations is leveraged to fetch the sentiment analysis results from the server without blocking the main thread. The `fetch` API is used in the `analyzed.html` file to send an HTTP GET request to the `/process` route and retrieve the sentiment analysis data in JSON format. This allows the application to remain responsive while waiting for the server's response.
3. **DOM Manipulation:** JavaScript is used to manipulate the DOM, which represents the structure of the web page. In the `analyzed.html` file, JavaScript selects various HTML elements using `document.querySelector` and updates their content or styles based on the sentiment analysis results. This includes showing or hiding elements, changing text content, and adjusting the width and color of the sentiment score progress bar.
4. **Event Handling:** While not prominently used in the provided code, JavaScript can also handle user events, such as button clicks, form submissions, or keyboard input. This allows for creating interactive and responsive user interfaces.

In the SentiSight project, JavaScript plays a crucial role in enhancing the user experience by enabling dynamic updates of the sentiment analysis results, fetching data from the server asynchronously, and manipulating the DOM to reflect the analysis outcome. While the provided code focuses on the frontend functionality, JavaScript can also be used on the server-side with technologies like Node.js, enabling the development of full-stack web applications.

Python

Python is a high-level, interpreted, and dynamically typed programming language. It was created by Guido van Rossum in the late 1980s and first released in 1991. Python is designed to be readable and easy to learn, with a clean and concise syntax that emphasizes code readability.

How Python is used in your project:

In my project, Python is used as the primary programming language for building the sentiment analysis web application. The `app.py` file contains the Flask application, which is a lightweight Python web framework.

1. **Flask:** Flask is a Python web framework that is used to create the web application and handle the routes and HTTP requests.
2. **TextBlob:** TextBlob is a Python library for processing textual data. It is used in your project to perform sentiment analysis on the input text and calculate the polarity score.
3. **NLTK (Natural Language Toolkit):** NLTK is a suite of libraries and programs for working with human language data in Python. In your project, NLTK's `SentimentIntensityAnalyzer` is used to analyze the sentiment of the input text and provide detailed scores for positive, negative, and neutral sentiments.
4. **Template Rendering:** Flask's template rendering functionality is used to generate dynamic HTML pages (`index.html`, `form.html`, `analyzed.html`) based on the data and sentiment analysis results.
5. **URL Routing:** Flask's URL routing mechanism is used to define different routes (`/`, `/result`, `/process`) and handle the corresponding HTTP requests and responses.
6. **Request Handling:** Flask's `request` object is used to retrieve the input text submitted through the form.

In summary, Python is the backbone of your project, providing the necessary tools and libraries to build a web application for sentiment analysis. Flask serves as the web framework, while TextBlob and NLTK handle the core text processing and sentiment analysis functionality.

Flask

Flask is a lightweight and flexible Python web framework for building web applications. It's designed to be simple, minimalistic, and easy to get started with, while still providing powerful features and tools for building robust applications.

How Flask is used in your project:

In your project, Flask is used as the web framework to build the sentiment analysis web application. Here's how Flask is utilized:

1. **Application Setup:** The `app.py` file creates a Flask application instance, which serves as the entry point for your web application.
2. **Routing and URL Handling:** Flask's routing mechanism is used to define different routes (`/`, `/result`, `/process`) and associate them with corresponding Python functions (`index()`, `analyse()`, `processText()`). These functions handle the different HTTP requests and generate the appropriate responses.
3. **Template Rendering:** Flask's template rendering functionality is used to render HTML templates (`index.html`, `form.html`, `analyzed.html`) with dynamic data. The rendered templates are returned as responses to the client.
4. **Request Handling:** Flask's `request` object is used to retrieve the input text submitted through the form in the `analyse()` function.
5. **Development Server:** Flask's built-in development server is used to run and test the application locally during development.

In summary, Flask is the backbone of your web application, providing the necessary tools and features for handling HTTP requests, defining routes, rendering templates, and managing the overall application flow. Its simplicity and lightweight nature make it an excellent choice for building small to medium-sized web applications like your sentiment analysis project.

Hardware and Software used.

Development Machine:

Processor: Intel® Core™ i3 CPU @ 2.30GHz 2.30GHz

Installed Memory (RAM): 8.00GB

System Type: 64-bit Operating System, x64-based processor

The development machine is equipped with an Intel Core i3 processor, providing the computational power needed for coding, testing, and debugging the EasyRation application. With8GB of RAM, the system can efficiently handle the development environment and various software tools.

Software Tools:

Visual Studio Code

Visual Studio Code is the integrated development environment (IDE) used for coding and debugging during the development of EasyRation. Its lightweight nature and extensive pluginsupport contribute to an efficient development process.

Web Browsers (Google Chrome, Windows Explorer)

Web browsers, including Google Chrome and Windows Explorer, are essential for testing andcompatibility checks. Ensuring that EasyRation functions seamlessly across different browserscontributes to a positive user experience.

Detailed Life Cycle of SentiSight

Here's a detailed life cycle of your sentiment analysis web application project:

1. Planning and Requirements Gathering

- Identify the project goals and objectives (building a web application for sentiment analysis)
- Determine the target audience and their needs
- Define the project scope and features (input text, sentiment analysis, displaying results)
- Research and select the appropriate technologies and tools (Python, Flask, TextBlob, NLTK, Tailwind CSS)

2. Design

- Plan the overall application architecture and structure.
- Design the user interface (UI) and user experience (UX) for the web application.
- Create wireframes or mockups to visualize the UI design.
- Decide on the color scheme, typography, and branding elements.

3. Development

- Set up the development environment (install Python, Flask, and other dependencies).
- Implement the backend logic using Python and Flask.
- Create the Flask application (`app.py`).
- Define routes for handling different URLs (`/`, `/result`, `/process`).
- Implement sentiment analysis functionality using TextBlob and NLTK.
- Develop the frontend using HTML, CSS (Tailwind CSS), and JavaScript.
- Create the HTML templates (`index.html`, `form.html`, `analyzed.html`).
- Style the pages using Tailwind CSS utility classes.
- Add interactivity and handle asynchronous requests using JavaScript.

- Integrate the backend and frontend components.

4. Testing

- Unit testing: Test individual components and functions (e.g., sentiment analysis logic)
- Integration testing: Test the integration between different components (e.g., Flask routes, template rendering).
- End-to-end testing: Test the entire application flow from the user's perspective.
- Cross-browser and cross-device testing: Ensure the application works correctly on different browsers and devices.

5. Deployment

- Choose a hosting platform or server for deploying the web application.
- Configure the production environment (e.g., web server, database if needed).
- Build and package the application for deployment.
- Deploy the application to the hosting platform or server.

6. Maintenance and Updates

- Monitor the application's performance and user feedback.
- Address any bugs or issues that arise.
- Implement new features or enhancements based on user feedback or changing requirements.
- Keep the application and its dependencies up-to-date with the latest security patches and updates.

7. Documentation

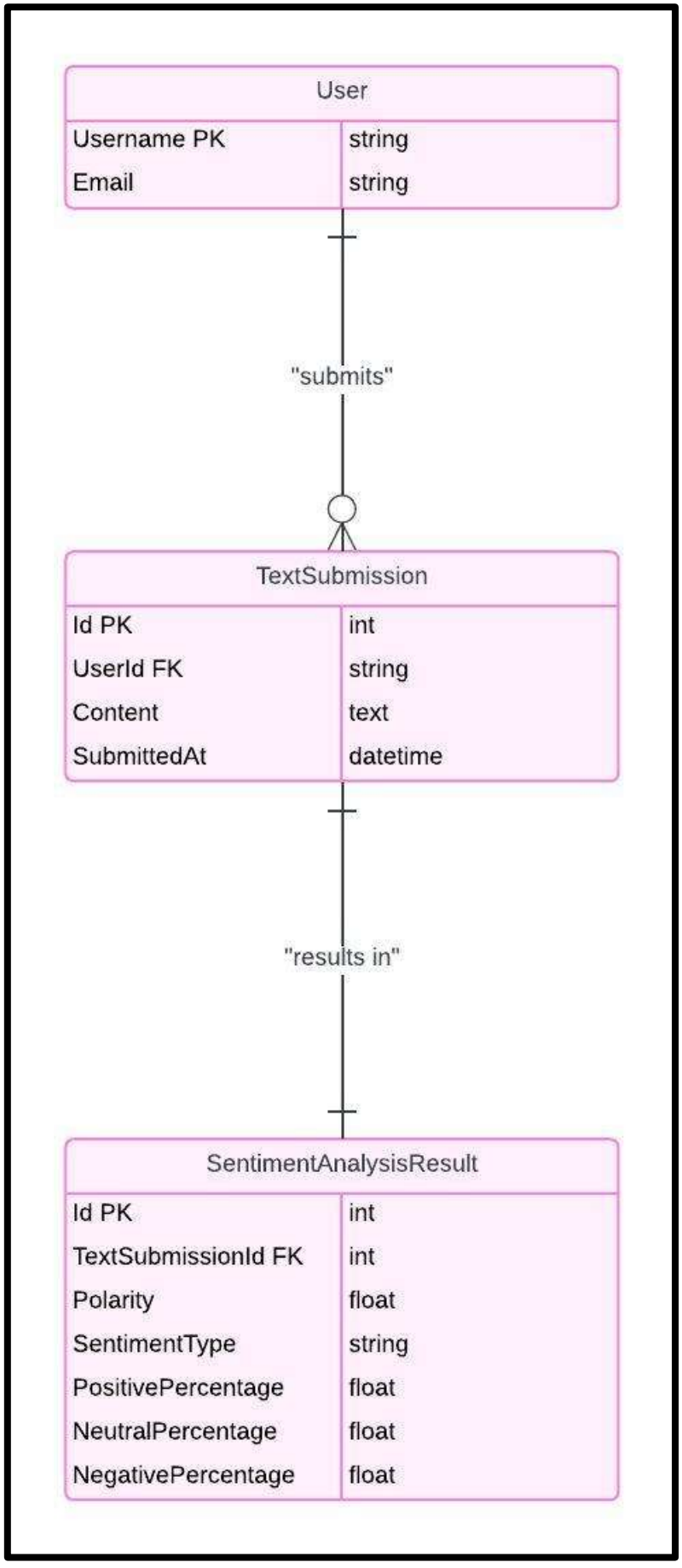
- Document the project throughout the life cycle.
- Create user documentation or help guides for end-users.
- Maintain technical documentation for developers (e.g., code comments, architecture diagrams).

This life cycle represents a general workflow for developing and deploying a web application, but it may vary depending on the project's specific requirements, team structure, and development methodologies (e.g., agile, waterfall).

Entity Relationship diagram

An Entity-Relationship Diagram (ERD) is a visual representation of the data model for a system or application. It illustrates the logical structure of the database, including the entities (tables), attributes (columns), and relationships between entities. ERDs are commonly used in database design and help developers and stakeholders understand the structure and relationships within the data.

Even though your current sentiment analysis web application does not have a persistent data storage mechanism like a database, we can still create a basic ERD to represent a potential data model for storing user inputs and their corresponding sentiment analysis results.


User
Username PK
string
Email
string
"submits"
TextSubmission
Id PK
int
UserId FK
string
Content
text
SubmittedAt
datetime
"results in"
SentimentAnalysisResult
Id PK
int
TextSubmissionId FK
int
Polarity
float
SentimentType
string
PositivePercentage
float
NeutralPercentage
float
NegativePercentage
float

In this ERD, we have two entities:

1. **User**:
    - id (Primary Key): A unique identifier for each user.
    - username: The username of the user.
    - email: The email address of the user.
2. **AnalysisResult**:
    - id (Primary Key): A unique identifier for each analysis result.
    - text: The input text submitted by the user for sentiment analysis.
    - polarity: The polarity score of the input text.
    - sentiment_type: The sentiment type (positive, negative, or neutral) of the input text.
    - positive_pcnt: The percentage of positive sentiment in the input text.
    - neutral_pcnt: The percentage of neutral sentiment in the input text.
    - negative_pcnt: The percentage of negative sentiment in the input text.
    - user_id (Foreign Key): A reference to the id of the User entity, establishing a one-to-many relationship between User and AnalysisResult. This means that a user can have multiple analysis results, but each analysis result belongs to only one user.

By implementing this data model, you could store user information and their sentiment analysis results in a database. This would allow you to persist the data, retrieve historical analysis results, and potentially add more features, such as user authentication, result tracking, or data analysis.

DATA FLOW DIAGRAM

Data Flow Diagram (DFD): A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system or process. It is a widely used technique in structured analysis and design methodologies, providing a visual representation of how data moves within a system, and how it is processed, stored, and transformed.

DFDs are typically organized into different levels, with each level providing more detailed information about the processes and data flows.

DFD (Level 0) or Context-Level DFD: The Level 0 DFD, also known as the Context-Level DFD, provides an overview of the entire system and its interactions with external entities. It represents the system as a single process and shows the main data flows between the system and external entities (e.g., users, other systems, or external data sources). The Context-Level DFD gives a high-level understanding of the system's boundary and its interactions with the external environment.

DFD (Level 1): The Level 1 DFD provides a more detailed representation of the internal processes and data flows within the system. It breaks down the main process from the Context-Level DFD into sub-processes, showing the data flows between them and any data stores involved. The Level 1 DFD offers a more granular view of how the system works internally, revealing the various processes and the flow of data between them.

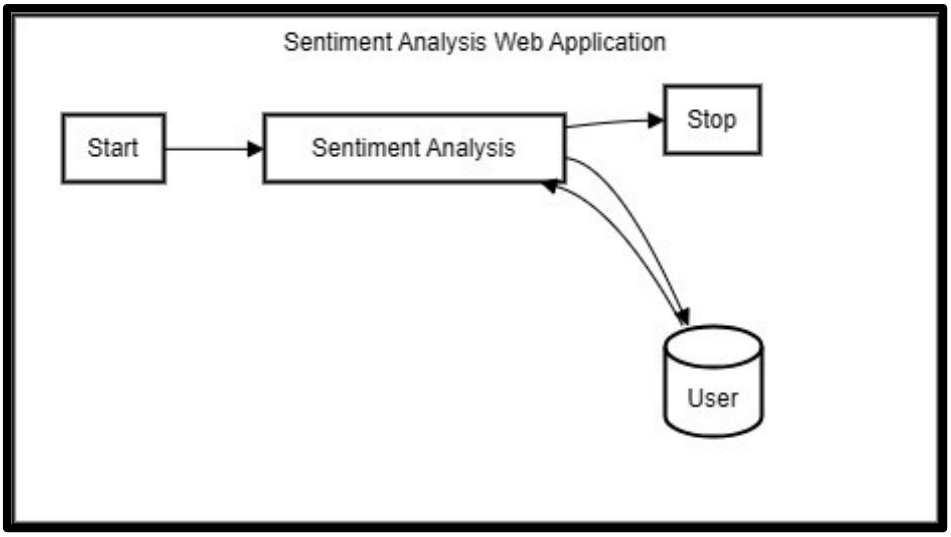
In the context of your sentiment analysis web application project, DFDs can help you visualize and understand the flow of data and the processes involved in the application.

Level 0 (Context-Level) DFD: The Level 0 DFD for your project would represent the overall sentiment analysis web application as a single process and show its interaction with the external entity, which is the user. It would include:

- External Entity: User
- Main Process: Sentiment Analysis Web Application
- Data Flows:

- User Input Text (from User to Sentiment Analysis Web Application)
- Sentiment Analysis Results (from Sentiment Analysis Web Application to User)

This Context-Level DFD provides a high-level overview of the system, showing that the user interacts with the sentiment analysis web application by providing input text and receiving sentiment analysis results.



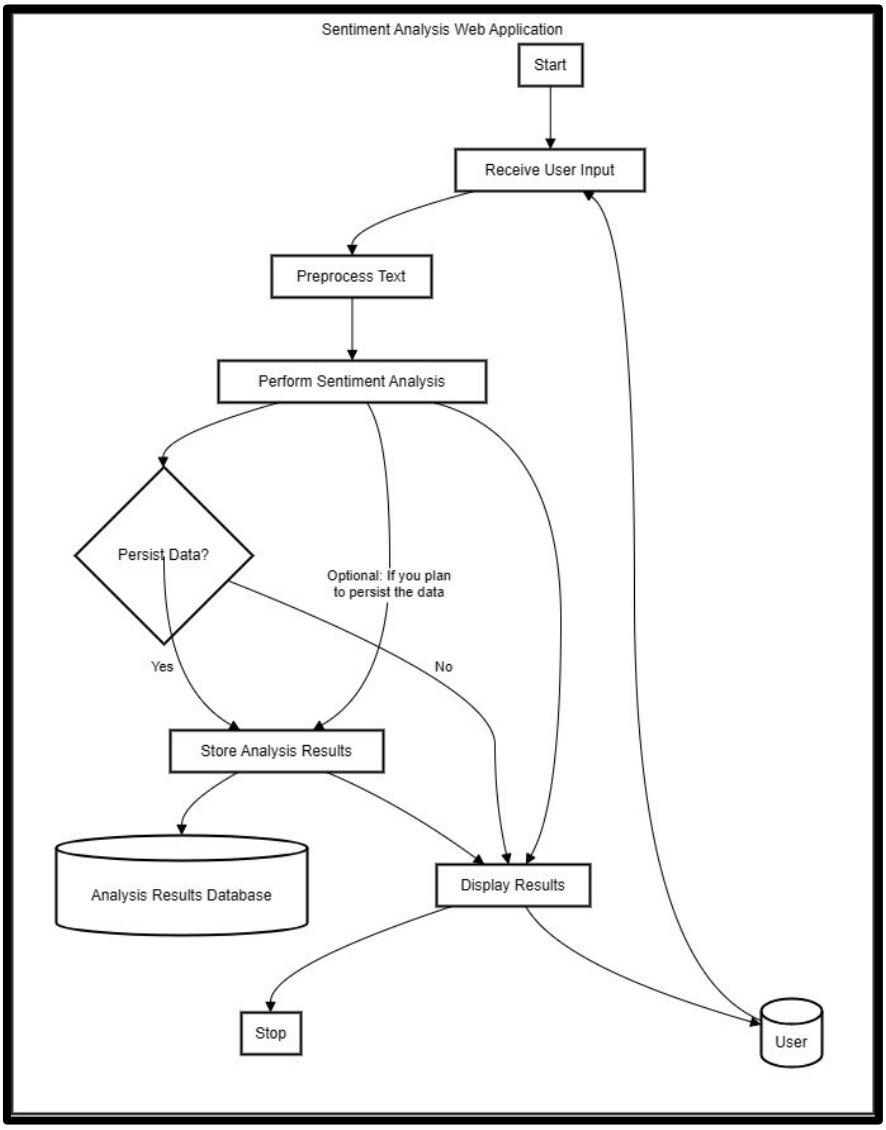
DFD level – 0

Level 1 DFD: The Level 1 DFD for your project would break down the main process (Sentiment Analysis Web Application) into sub-processes and show the data flows between them. It might include processes such as:

- Receive User Input
- Preprocess Text
- Perform Sentiment Analysis
- Store Analysis Results (if you plan to persist the data)
- Display Results

The Level 1 DFD would also show the data flows between these processes, such as:

- User Input Text (from external entity User to Receive User Input process)
- Preprocessed Text (from Preprocess Text process to Perform Sentiment Analysis process)
- Analysis Results (from Perform Sentiment Analysis process to Store Analysis Results process and Display Results process)
- Stored Analysis Results (from Store Analysis Results process to Display Results process, if applicable)
- Displayed Results (from Display Results process to external entity User)



DFD Level 1

FLOWCHART

A flowchart is a graphical representation of a process or a sequence of steps, using standardized symbols and diagrams. It is a visual tool that illustrates the flow of control, decisions, and actions within a process or algorithm. Flowcharts are commonly used in various fields, including computer programming, project management, process optimization, and business process modeling.

Flowcharts typically consist of different shapes, such as rectangles, diamonds, and arrows, each representing a specific type of step or decision in the process. These shapes are connected by arrows that show the direction of the flow and the sequence of operations.

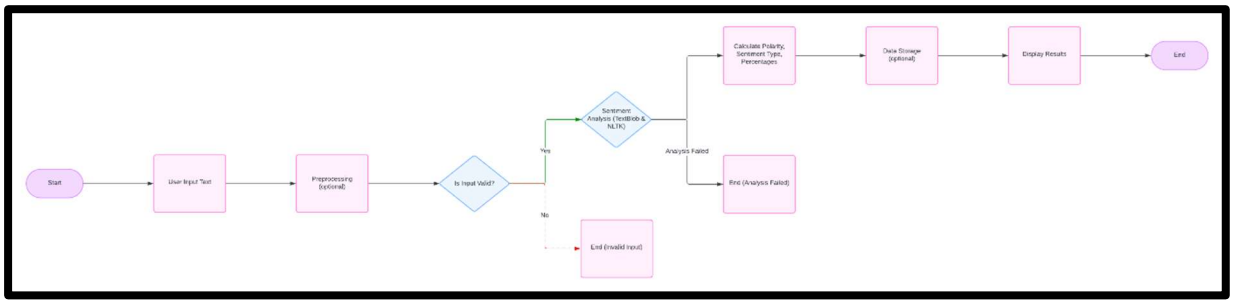
In the context of your sentiment analysis web application project, a flowchart can be used to visually represent the logical flow and sequence of steps involved in processing user input, performing sentiment analysis, and displaying the results.

A flowchart for your project might include the following steps:

- Start
- Receive user input text (from the web form)
- Preprocess the text (if necessary, e.g., remove punctuation, convert to lowercase)
- Perform sentiment analysis using TextBlob and NLTK libraries
- Calculate polarity score, sentiment type, and percentages for positive, neutral, and negative sentiments
- Store the analysis results (if you plan to persist the data)
- Render the HTML template with the analysis results
- Display the sentiment analysis results to the user
- End

The flowchart would also include decision points, such as checking if the user input is valid, or if the sentiment analysis process was successful or encountered any errors.

By creating a flowchart for your project, you can visualize the logical flow of the application, identify potential bottlenecks or areas for optimization, and communicate the process more effectively with your team members or stakeholders.



Flow chart.

TESTING METHODOLOGY

1. Unit Testing : Unit testing is a software testing technique where individual components or units of a software application are tested in isolation. The primary purpose of unit testing is to validate that each unit of the software performs as designed. A "unit" in this context refers to the smallest testable part of a program, often a function, method, or procedure.

Process of Unit Testing

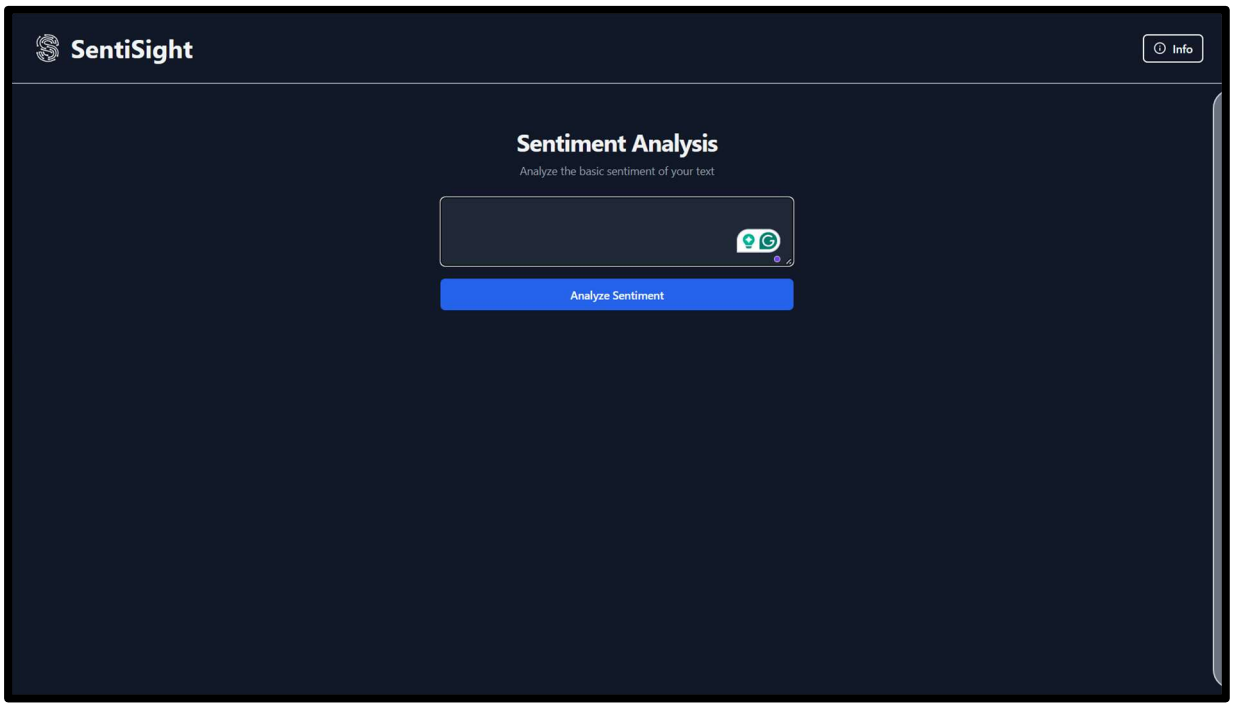
- Identification of Units: Developers identify individual units of code, such as functions, methods, or classes, that can be tested independently.
- Creation of Test Cases: Test cases are designed to assess the functionality of each unit. Test cases include inputs, expected outputs, and conditions to be verified.
- Execution of Tests: The automated testing framework is used to execute the unit tests. Each unit is tested in isolation, and the results are recorded.
- Analysis of Results: Developers analyze the test results to identify any failures or deviations from expected behavior. If issues are found, developers make necessary adjustments to the code.
- Regression Testing: After making changes to address issues, developers rerun unit tests to ensure that existing functionality remains intact. This is known as regression testing.

2. Integration Testing : Integration testing is a level of software testing where individual units or components of a software application are combined and tested as a group. The primary goal of integration testing is to ensure that the integrated components work together as expected, uncovering any issues that may arise due to the interaction between these components. It validates the correct flow of data and functionality between the interconnected units, verifying that they collaborate seamlessly to achieve the intended behavior of the system.

Process of Integration Testing

- Identification of Integration Points: Identify the points where individual units or components interface with each other. These interfaces are crucial for data exchange and function calls.
- Selection of Integration Approach: Decide whether to follow a top-down or bottom-up integration approach. In top-down integration, higher-level modules are tested first, while in bottom-up integration, lower-level modules are tested first.
- Creation of Test Cases: Develop test cases that exercise the interactions between integrated components. Test cases should cover different scenarios, including normal operation, error conditions, and boundary cases.
- Execution of Tests: Execute the integration tests, combining the selected units or components to assess their collective behavior. Monitor the outputs and interactions to identify any discrepancies.
- Analysis of Results: Analyze the test results to detect and diagnose integration issues. If problems are identified, the development team addresses them by refining the code or modifying the interfaces.
- Regression Testing: After resolving issues, perform regression testing to ensure that changes do not introduce new problems or impact existing functionality.

Screenshots of project

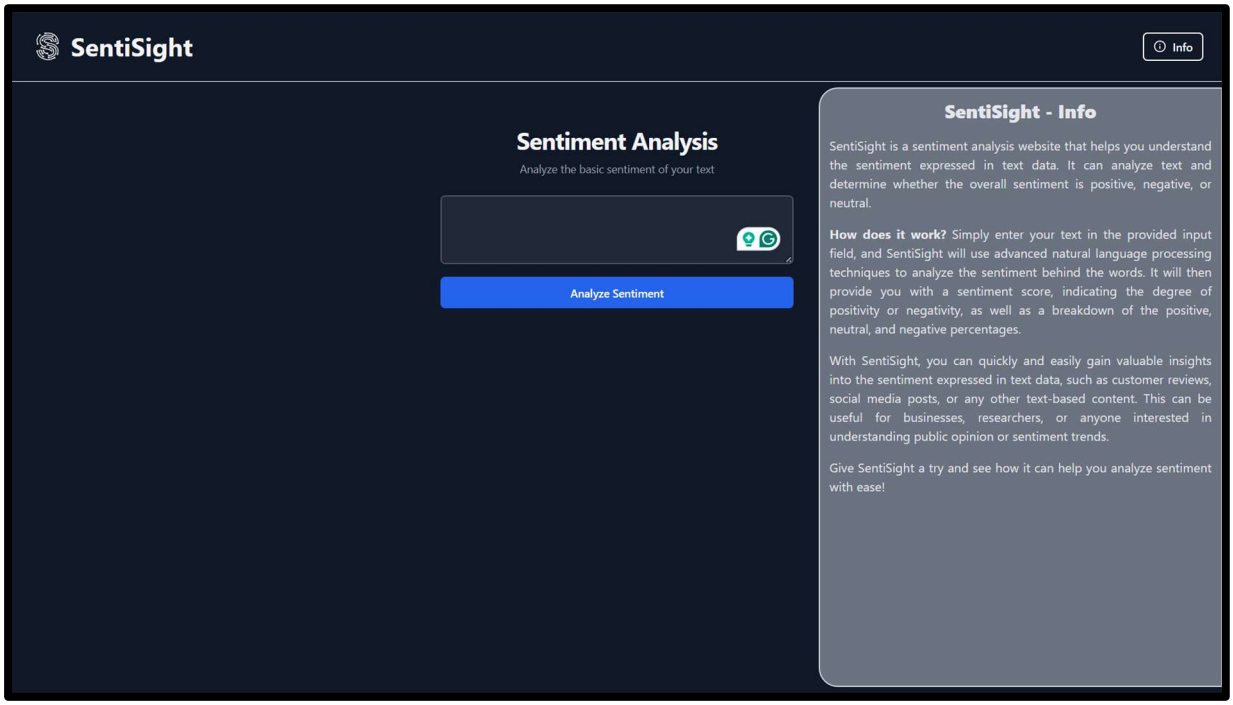


The provided image shows the front webpage of the SentiSight website, which is a sentiment analysis tool. Here's an explanation of the different elements and components present on this page:

1. Header: The header section displays the name "SentiSight" along with a small "Info" button. This button likely provides additional information or documentation about the website and its functionality.
2. Main Content Area: The main content area of the page is focused on the sentiment analysis feature. It includes the following elements:
    a. Heading: The heading "Sentiment Analysis" clearly indicates the primary purpose of the website.
    b. Subheading: The subheading "Analyze the basic sentiment of your text" provides a brief description of the website's functionality, which is to analyze the sentiment (positive, negative, or neutral) of text input.
    c. Text Input Field: There is a text input field where users can enter the text they want to analyze for sentiment. The input field has a placeholder text that says "Enter text to analyze..." to guide the user on what needs to be done.
    d. Analyze Sentiment Button: Below the text input field, there is a blue button labeled "Analyze Sentiment". This button likely triggers the sentiment analysis process when clicked, submitting the entered text for analysis.

2. Color Scheme: The website follows a dark color scheme, with a black background and white/light text and elements. This color scheme creates a sleek and modern look, while also providing good contrast for readability.
3. Minimalistic Design: The overall design of the front webpage is minimalistic, focusing solely on the core functionality of sentiment analysis. This approach helps to minimize distractions and provides a clear and straightforward user experience.

The front webpage of the SentiSight website effectively communicates its purpose and invites users to input text for sentiment analysis. The clean and minimalistic design, along with clear instructions and a prominent call-to-action button, makes it easy for users to understand and engage with the website's primary functionality.

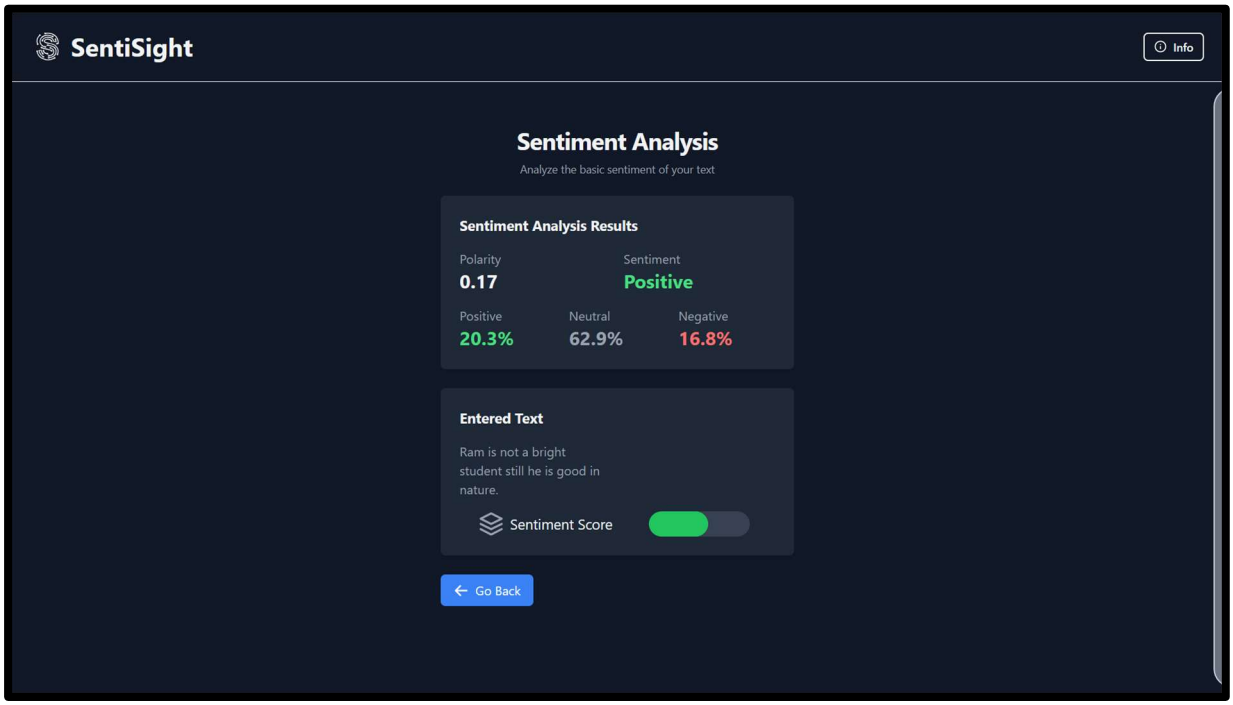


This webpage is an updated version of the SentiSight website, which includes an "Info" section in addition to the main sentiment analysis feature.

1. Info Section: A new section titled "SentiSight - Info" has been added to the right side of the main content area. This section provides a detailed explanation of the SentiSight website and its functionality. It includes the following information:
    a. Introduction: The first paragraph introduces SentiSight as a sentiment analysis website that helps users understand the sentiment expressed in text data.
    b. How it Works: The second paragraph explains how SentiSight works, mentioning that users need to enter their text in the provided input field, and the website will use advanced natural language processing techniques to analyze the sentiment behind the words.

c. Result Details: The third paragraph elaborates on the results provided by SentiSight, including a sentiment score indicating the degree of positivity or negativity, as well as a breakdown of positive, neutral, and negative percentages.
d. Use Cases: The fourth paragraph highlights potential use cases for SentiSight, such as analyzing customer reviews, social media posts, or any other text-based content, making it useful for businesses, researchers, or anyone interested in understanding public opinion or sentiment trends.
e. Call to Action: The final paragraph encourages users to try SentiSight and see how it can help them analyze sentiment with ease.

The addition of the "SentiSight - Info" section provides users with a comprehensive understanding of the website's purpose and functionality. It explains the sentiment analysis process, the types of results users can expect, and the potential applications of the tool. This informative section enhances the overall user experience by providing context and clarity to users, especially those who may be unfamiliar with sentiment analysis or natural language processing.



This webpage appears to be the main user interface for the sentiment analysis functionality of your SentiSight web application. Let me break down the different components and elements present on this page:

1. Header: The header section displays the "SentiSight" logo, indicating the name of your application.

2. Main Content Area:
    i. Heading: The heading "Sentiment Analysis" clearly indicates the purpose of this page, which is to perform sentiment analysis on text.
    ii. Subheading: The subheading "Analyze the basic sentiment of your text" provides a brief description of the functionality offered on this page.

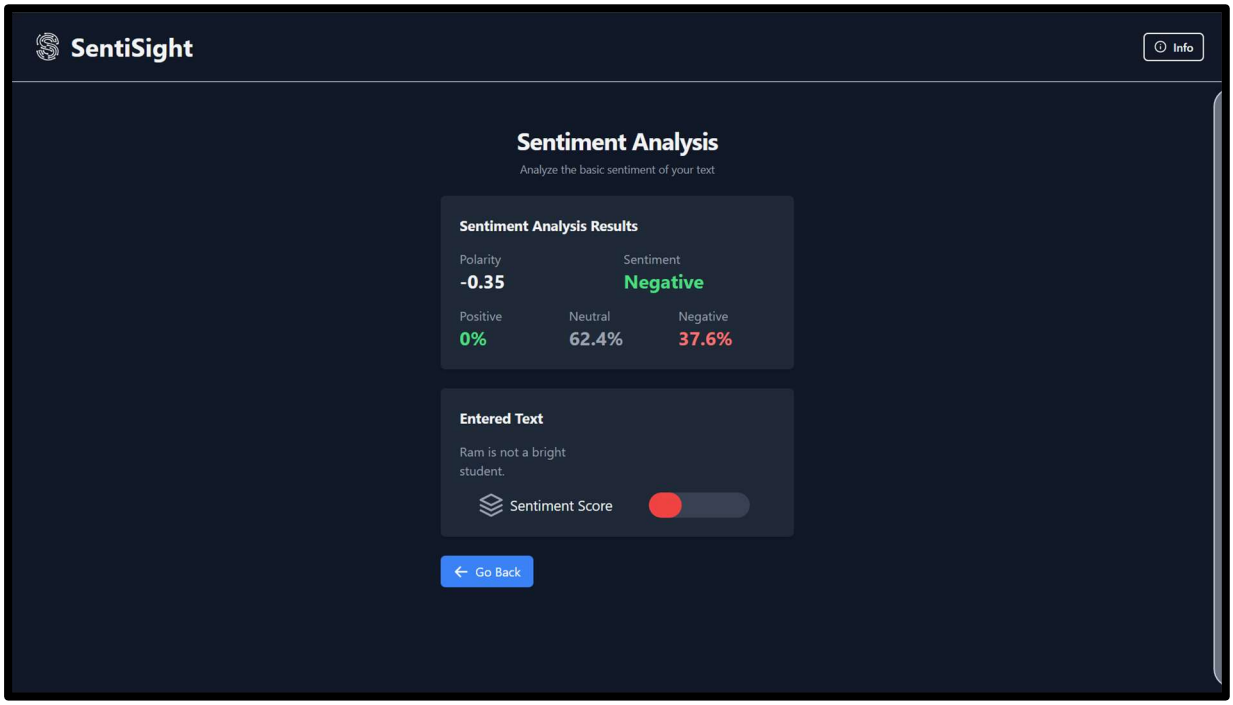
2. Sentiment Analysis Results Section:
    i. Heading: "Sentiment Analysis Results" signifies that this section displays the results of the sentiment analysis performed on the input text.
    ii. Polarity Score: A numerical value (in this case, 0.17) represents the polarity score of the input text, which indicates the overall sentiment (positive or negative).
    iii. Sentiment Type: The sentiment type "Positive" is displayed, indicating that the overall sentiment of the input text is considered positive based on the polarity score.
    iv. Sentiment Percentages: Three values are displayed, representing the percentages of positive (62.9%), neutral (20.3%), and negative (16.8%) sentiments present in the input text.

3. Entered Text Section:
    i. Heading: "Entered Text" indicates that this section displays the original text entered by the user for sentiment analysis.
    ii. Text Area: A multi-line text area displays the input text provided by the user, which in this case says "Thank you for your wonderful support on this great project. Weather today is good."

4. Sentiment Score Slider:
    i. Icon and Label: An icon and the label "Sentiment Score" represent a visual indicator or slider for the sentiment score.
    ii. Slider: A green slider or progress bar visually represents the sentiment score, which appears to be positive in this case.

5. Button:
    i. Button Label: The button label "Go Back" suggests that clicking this button will navigate the user back to the previous page or step in the application flow.

Overall, this webpage provides a clean and visually appealing interface for users to input text, perform sentiment analysis, and view the results, including the polarity score, sentiment type, and percentages of positive, neutral, and negative sentiments. The design appears to be well-structured and user-friendly, making it easy for users to understand and interact with the sentiment analysis functionality of your SentiSight web application.



This webpage appears to be the result page of your SentiSight sentiment analysis web application, displaying the analysis results for a specific input text. Here's a breakdown of the different components and elements present on this page:

1. Header The header section displays the "SentiSight" logo, indicating the name of your application. It also includes an "Info" button, likely providing additional information about the application.
2. Main Content Area
    i. Heading The heading "Sentiment Analysis" clearly indicates the purpose of this page, which is to perform sentiment analysis on text.
    ii. Subheading The subheading "Analyze the basic sentiment of your text" provides a brief description of the functionality offered on this page.

2. Sentiment Analysis Results Section
    i. Heading "Sentiment Analysis Results" signifies that this section displays the results of the sentiment analysis performed on the input text.
    ii. Polarity Score A numerical value (-0.35) represents the polarity score of the input text, which indicates an overall negative sentiment.

iii. Sentiment Type The sentiment type "Negative" is displayed, indicating that the overall sentiment of the input text is considered negative based on the polarity score.
iv. Sentiment Percentages Three values are displayed, representing the percentages of positive (0%), neutral (62.4%), and negative (37.6%) sentiments present in the input text.

3. Entered Text Section
    i. Heading "Entered Text" indicates that this section displays the original text entered by the user for sentiment analysis.
    ii. Text Area A single-line text area displays the input text provided by the user, which in this case says "Ram is not a bright student."

4. Sentiment Score Indicator
    i. Icon and Label An icon and the label "Sentiment Score" represent a visual indicator for the sentiment score.
    ii. Indicator A red circle or dot visually represents the negative sentiment score for the input text.

5. Button
    i. Button Label The button label "Go Back" suggests that clicking this button will navigate the user back to the previous page or step in the application flow.

Overall, this webpage displays the sentiment analysis results for a specific input text, including the polarity score, sentiment type (negative in this case), and percentages of positive, neutral, and negative sentiments. The design is consistent with the previous example, providing a clear and user-friendly interface for understanding the sentiment analysis results. The red indicator for the sentiment score effectively communicates the negative sentiment associated with the input text.

Coding of the project

App.py

```
from flask import Flask, render_template, request, jsonify

from flask import url_for

from textblob import TextBlob

import nltk



nltk.download(info_or_id="vader_lexicon")

from nltk.sentiment.vader import SentimentIntensityAnalyzer



app = Flask(__name__)



FORM_TEMPLATE: str = "form.html"

ANALYSE_TEMPLATE: str = "analyzed.html"



@app.route("/", methods=["GET"])

def index():

    print(url_for("static", filename="logo.svg"))

    return render_template(FORM_TEMPLATE, blob_sentiment=0)
```

```
@app.route("/result", methods=["POST"])
def analyse():
    text = request.form.get("rawtext")
    return render_template(ANALYSE_TEMPLATE, text=text)


@app.route("/process", methods=["GET"])
def processText():
    # print("Processing...")
    text = request.args.get("text")


    sid = SentimentIntensityAnalyzer()
    process_dict: dict[str, float] = sid.polarity_scores(text)


    compound: float = process_dict["compound"]
    if compound == 0:
        sentiment_type = "Neutral"
    elif compound > 0:
        sentiment_type = "Positive"
    elif compound < 0:
        sentiment_type = "Negative"
```

```
    data = dict()
    data["sentiment_type"] = sentiment_type
    data["polarity"] = round(TextBlob(text=text).sentiment.polarity, 2)
    data["positive_pcnt"] = round(process_dict["pos"] * 100, 2)
    data["neutral_pcnt"] = round(process_dict["neu"] * 100, 2)
    data["negative_pcnt"] = round(process_dict["neg"] * 100, 2)
    data["text"] = text
    return jsonify(data)


if __name__ == "__main__":
    app.run(debug=True)
```

Header.html

```
<header

    class="sticky bg-gray-100 dark:bg-gray-900 px-4 md:px-6 py-4 flex
items-center justify-between border-b border-gray-800 dark:border-white ">

    <div class="flex items-center p-1">

        <svg width="35" height="35" viewBox="0 0 564 697" fill="none"
xmlns="http://www.w3.org/2000/svg">

            <path class="fill-black dark:fill-white"

                d="M0.297302 453.777L35.8632 451.913L37.7272
487.479L2.16123 489.342L0.297302 453.777Z" />

            <path class="fill-black dark:fill-white"

                d="M435 632.613L464.526 612.697L484.441 642.223L454.915
662.139L435 632.613Z" />

            <path class="fill-black dark:fill-white"

                d="M103.019 378.406L122.935 348.88L152.461 368.796L132.545
398.322L103.019 378.406Z" />

            <path class="fill-black dark:fill-white"

                d="M372.197 481.267L362.979 446.866L397.38 437.648L406.598
472.049L372.197 481.267Z" />

            <path class="fill-black dark:fill-white"

                d="M90.1046 38.6889L112.031 66.7537L83.9664 88.6804L62.0398
60.6156L90.1046 38.6889Z" />

            <path class="fill-black dark:fill-white"
```

```
                d="M38.2234 267.039L56.0308 297.883L25.1875 315.69L7.38011
284.847L38.2234 267.039Z" />
            <path class="fill-black dark:fill-white"
                d="M261.934 547.701L263.745 512.133L299.314 513.944L297.502
549.513L261.934 547.701Z" />
            <path class="fill-black dark:fill-white"
                d="M155.186 442.419L189.941 434.64L197.72 469.395L162.965
477.174L155.186 442.419Z" />
            <path class="fill-black dark:fill-white"
                d="M150.002 617.374L165.422 585.27L197.525 600.69L182.105
632.794L150.002 617.374Z" />
            <path class="fill-black dark:fill-white"
                d="M137.918 533.68L163.55 508.953L188.277 534.584L162.645
559.312L137.918 533.68Z" />
            <path class="fill-black dark:fill-white"
                d="M294.369 599.893L294.969 564.283L330.578 564.883L329.978
600.493L294.369 599.893Z" />
            <path class="fill-black dark:fill-white"
                d="M372.458 370.213L365.559 405.153L330.619 398.254L337.518
363.314L372.458 370.213Z" />
            <path class="fill-black dark:fill-white"
                d="M292.229 652.422L290.773 616.837L326.358 615.382L327.814
650.967L292.229 652.422Z" />
```

```
            <path class="fill-black dark:fill-white"
                d="M266.405 292.806L301.207 300.371L293.641 335.173L258.839
327.607L266.405 292.806Z" />
            <path class="fill-black dark:fill-white"
                d="M212.262 338.511L247.063 346.077L239.498 380.878L204.696
373.313L212.262 338.511Z" />
            <path class="fill-black dark:fill-white"
                d="M406.695 71.4L436.514 90.8737L417.041 120.693L387.221
101.219L406.695 71.4Z" />
            <path class="fill-black dark:fill-white"
                d="M398.165 196.688L407.103 231.163L372.628 240.1L363.69
205.625L398.165 196.688Z" />
            <path class="fill-black dark:fill-white"
                d="M555 539.743L539.473 571.795L507.421 556.268L522.948
524.216L555 539.743Z" />
            <path class="fill-black dark:fill-white"
                d="M135.988 182.696L171.553 184.582L169.667 220.147L134.102
218.261L135.988 182.696Z" />
            <path class="fill-black dark:fill-white"
                d="M488.288 142.974L503.847 175.01L471.811 190.57L456.251
158.533L488.288 142.974Z" />
            <path class="fill-black dark:fill-white"
```

```
                d="M536.072 257.789L537.129 293.388L501.53 294.445L500.473
258.846L536.072 257.789Z" />
            <path class="fill-black dark:fill-white"
                d="M232.429 0L239.532 34.8993L204.632 42.0022L197.529
7.10289L232.429 0Z" />
            <path class="stroke-black dark:stroke-white"
                d="M163.675 604.667C66.6885 540.115 68.5964 448.535 68.5964
448.535H124.878C124.878 483.832 154.771 526.442 154.771 526.442M22.6356
487.28C40.1326 621.155 230.054 753.486 446.367 645.688M137.282
382.393C259.126 430.662 296.718 408.627 375.138 451.715M74.1233
73.6677C13.1156 144.376 4.48378 191.324 25.4711 279.089M179.575
465.706C194.52 519.764 268.293 529.622 268.293 529.622M363.753
392.569C438.735 395.749 502.333 541.07 327.121 583.362M289.917
314.662C402.485 343.281 486.116 345.824 497.564 466.978C497.564 556.969
437.782 617.705 320.444 633.286M212.963 357.272C88.6298 327.063 65.5497
209.829 118.203 136.27C161.449 75.8519 272.109 38.0111 407.255
92.0692M535.404 536.936C549.174 499.448 550.995 481.924 546.543
439.631C542.091 397.339 516.794 364.06 479.438 329.925C431.351 285.984
332.298 273.706 320.434 271.415C302.318 267.917 222.821 251.7 211.055
239.298C199.29 226.897 196.11 202.553 218.051 183.968C239.992 165.383
326.168 139.45 385.631 206.863M154.453 193.541C160.335 147.369 228.842
120.913 259.39 117.224C317.632 110.191 373.028 117.224 406.937
156.939C422.2 174.816 441.597 223.081 441.597 223.081L496.292
224.671L485.162 177.926M229.817 19.2498C393.581 -4.59933 552.576 69.8101
552.576 273.323L530.952 275.867"
                stroke="" stroke-width="22" />
        </svg>
```

```
        <h1 class="text-[2em] font-bold text-gray-900 dark:text-gray-100
ms-3">SentiSight</h1>
    </div>
    <button id="info-button"
        class="inline-flex items-center justify-center whitespace-nowrap
dark:text-white text-sm font-medium ring-offset-background transition-
colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-
offset-2 disabled:pointer-events-none disabled:opacity-50 border-2 border-
blue-500 dark:border-white h-9 rounded-md px-3"
        onclick="showPane()">
        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none"
            stroke="currentColor" stroke-width="2" stroke-linecap="round"
stroke-linejoin="round" class="h-4 w-4 mr-2">
            <circle cx="12" cy="12" r="10"></circle>
            <path d="M12 16v-4"></path>
            <path d="M12 8h.01"></path>
        </svg>
        Info
    </button>
</header>
```

analyzed.html

```
{% extends "index.html" %}

{% block content %}


<div id="sentiment_result" class="bg-white dark:bg-gray-800 rounded-md h-0
hidden">
    <h2 class="text-lg font-bold text-gray-900 dark:text-gray-
100">Sentiment Analysis Results</h2>
    <div class="grid grid-cols-2 gap-4">
        <div>
            <p class="text-gray-600 dark:text-gray-400">Polarity</p>
            <p id="polarity" class="text-2xl font-bold text-gray-900
dark:text-gray-100"></p>
        </div>
        <div>
            <p class="text-gray-600 dark:text-gray-400">Sentiment</p>
            <p id="sentiment_type" class="text-2xl font-bold text-green-500
dark:text-green-400">Positive</p>
        </div>
    </div>
    <div class="grid grid-cols-3 gap-4">
        <div>
```

```
            <p class="text-gray-600 dark:text-gray-400">Positive</p>
            <p id="positive_pcnt" class="text-2xl font-bold text-green-500
dark:text-green-400">75%</p>
        </div>
        <div>
            <p class="text-gray-600 dark:text-gray-400">Neutral</p>
            <p id="neutral_pcnt" class="text-2xl font-bold text-gray-500
dark:text-gray-400">20%</p>
        </div>
        <div>
            <p class="text-gray-600 dark:text-gray-400">Negative</p>
            <p id="negative_pcnt" class="text-2xl font-bold text-red-500
dark:text-red-400">5%</p>
        </div>
    </div>
</div>

<div id="result">
    <div class="relative bg-white dark:bg-gray-800 rounded-md shadow-md p-6
space-y-4">
        <h2 id="result_heading" class="text-lg font-bold text-gray-900
dark:text-gray-100">Getting Sentiment Results
        </h2>
```

```
        <div class="loader"></div>
        <div id="text_div" class="grid grid-cols-2 gap-4 hidden">
            <div>
                <p id="text" class="text-gray-600 text-md dark:text-gray-
400"></p>
            </div>
        </div>
        <div class="mx-auto flex px-6 my-6 items-center justify-between">
            <div class="flex items-center">
                <svg class="h-8 w-8 text-gray-500 dark:text-gray-400"
viewBox="0 0 24 24" fill="none"
                    stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round">
                    <path d="M12 2L2 7l10 5 10-5-10-5z"></path>
                    <path d="M2 17l10 5 10-5"></path>
                    <path d="M2 12l10 5 10-5"></path>
                </svg>
                <span class="ml-2 text-lg text-gray-900 dark:text-gray-
100">Sentiment Score</span>
            </div>
            <div class="flex items-center">
                <div class="relative h-8 w-32">
```

```
                    <div class="absolute inset-0 overflow-hidden rounded-
full bg-gray-200 dark:bg-gray-700">
                        <div class="absolute inset-0 rounded-full bg-green-
500" id="sentiment_score"></div>
                    </div>
                </div>
                <span id="blob_sentiment" class="ml-2 text-gray-900
dark:text-gray-100"></span>
            </div>
        </div>
    </div>
</div>
<button onclick="history.back();"
    class="flex items-center gap-2 px-4 py-2 bg-blue-500 text-white
rounded-md hover:bg-blue-600 transition-colors duration-200">
    <svg class="w-5 h-5" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg">
        <path fill-rule="evenodd"
            d="M9.707 16.707a1 1 0 01-1.414 0l-6-6a1 1 0 010-1.414l6-6a1 1
0 011.414 1.414L5.414 9H17a1 1 0 110 2H5.414l4.293 4.293a1 1 0 010 1.414z"
            clip-rule="evenodd"></path>
    </svg>
    Go Back
```

```
</button>

{% endblock %}

{% block script %}

<script>

    function goBack() {

        window.history.back();

    }



    const url = "{{ url_for('processText', text=text) }}";



    fetch(url)

        .then(response => response.json())

        .then(data => {

            // console.log(data)

            setTimeout(() => {

                document.querySelector(".loader").classList.add("hidden");

                document.querySelector("#text_div").classList.remove("hidde
n");

                document.querySelector("#sentiment_result").classList.remov
e("hidden")

                document.querySelector("#sentiment_result").classList.remov
e("h-0");
```

```
                document.querySelector("#sentiment_result").classList.add("
shadow-md", "p-6", "space-y-4", "mt-4");

                document.querySelector("#text").textContent = data.text;

                document.querySelector("#polarity").textContent =
data.polarity;

                document.querySelector("#sentiment_type").textContent =
data.sentiment_type;

                document.querySelector("#positive_pcnt").textContent =
data.positive_pcnt + "%";

                document.querySelector("#neutral_pcnt").textContent =
data.neutral_pcnt + "%";

                document.querySelector("#negative_pcnt").textContent =
data.negative_pcnt + "%";

                document.querySelector("#result_heading").textContent =
"Entered Text";

                document.querySelector("#sentiment_score").style.width =
`${(50 * (data.polarity + 1))}%`;

                var bgcolor = "rgb(107 114 128)"

                if (data.sentiment_type == "Positive") {

                    bgcolor = "rgb(34 197 94)"

                }

                if (data.sentiment_type == "Negative") {

                    bgcolor = "rgb(239 68 68)";
```

```
                }

                document.querySelector("#sentiment_score").style.background
Color = bgcolor

            }, 500)



        })

        .catch(error => {

            console.error("Error:", error);

        });

</script>

{% endblock %}
```

Form.html

```
{% extends 'index.html' %} {% block content %}

<form class="space-y-4" method="post" action="{{ url_for('analyse') }}">

  <div>

    <label for="rawtext" class="sr-only"> Enter text </label>

    <textarea id="rawtext" name="rawtext" rows="3"

      class="block w-full rounded-md border-2 border-gray-400 shadow-sm
focus:border-indigo-500 focus:ring-indigo-500 dark:border-gray-600 dark:bg-
gray-800 dark:text-gray-100 dark:focus:border-indigo-500 sm:text-sm p-3"

      placeholder="Enter text to analyze..."></textarea>

  </div>

  <button

    class="inline-flex items-center justify-center whitespace-nowrap
rounded-md text-sm font-medium ring-offset-background transition-colors
focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring
focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-
50 bg-blue-600 text-white hover:bg-primary/90 h-10 px-4 py-2 w-full"

    type="submit">

    Analyze Sentiment

  </button>

</form>

{% endblock %}
```

```
{% block script %}
<script>
  document.addEventListener("load", () => {
    rawText = document.querySelector('#rawtext');
    rawText.value = "";
  }
  )
</script>
{% endblock %}
```

Index.html

```
<!DOCTYPE html>

<html lang="en">



<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <title>SENTISIGHT Sentiment Analysis website</title>

  <script src="https://cdn.tailwindcss.com"></script>

  <style>

    #sentiment_score {

      --blob_sentiment: 0;

      width: var(--blob_sentiment);

      transition: width 500ms ease-in-out;

      container-type: inline-size;

      container-name: sentimen-score;

    }



    .loader {

      width: 50px;

      aspect-ratio: 1;
```

```
  margin: auto;
  display: grid;
  border: 4px solid #0000;
  border-radius: 50%;
  border-right-color: #25b09b;
  animation: l15 1s infinite linear;
}

.loader::before,
.loader::after {
  content: "";
  grid-area: 1/1;
  margin: 2px;
  border: inherit;
  border-radius: 50%;
  animation: l15 2s infinite;
}

.loader::after {
  margin: 8px;
  animation-duration: 3s;
```

```
    }

    @keyframes l15 {
      to {
        transform: rotate(1turn)
      }
    }

    #info-pane {
      transition: right 300ms linear;
    }
  </style>
</head>

<body class="relative bg-gray-100 dark:bg-gray-900">
  <div class="relative flex flex-col">
    {% include 'header.html' %}
    <div class="flex flex-col items-center justify-center px-4 py-8">
      <div class="max-w-md w-full space-y-6">
        <div>
          <h1 class="mt-6 text-center text-3xl font-bold tracking-tight
text-gray-900 dark:text-gray-100">
```

```
            Sentiment Analysis
          </h1>
          <p class="mt-2 text-center text-sm text-gray-600 dark:text-gray-
400">
            Analyze the basic sentiment of your text
          </p>
        </div>
        {% block content %}{% endblock %}
      </div>
    </div>
    <div id="info-pane"
      class="bg-gray lg:w-1/3 sm:w-3/4 dark:bg-gray-500 bg-opacity-20
border-2 rounded-l-3xl border-gray-300 fixed top-24 right-[-500px] bottom-
2">
      {% include "info.html" %}
    </div>
  </div>
  {% block script %}
  {% endblock %}
  <script>
    infoPane = document.getElementById("info-pane")
    displaying = false;
```

```
    function showPane() {
      if (displaying) {
        infoPane.classList.remove("right-0");
        infoPane.classList.add("right[-500px]");
        displaying = false;
      } else {
        infoPane.classList.add("right-0");
        infoPane.classList.remove("right[-500px]");
        displaying = true;
      }
    }
  </script>
</body>
</html>
```

Info.html

```
<div class="p-3 text-justify flex flex-col gap-4 text-black dark:text-gray-200">
    <h1 class="font-extrabold text-center text-2xl">SentiSight - Info</h1>
    <p>SentiSight is a sentiment analysis website that helps you understand the sentiment expressed in text data. It can
        analyze text and determine whether the overall sentiment is positive, negative, or neutral.</p>
    <p><strong>How does it work?</strong> Simply enter your text in the provided input field, and SentiSight will use advanced natural
        language processing techniques to analyze the sentiment behind the words. It will then provide you with a
        sentiment score, indicating the degree of positivity or negativity, as well as a breakdown of the positive,
        neutral, and negative percentages.</p>
    <p>With SentiSight, you can quickly and easily gain valuable insights into the sentiment expressed in text data,
        such as customer reviews, social media posts, or any other text-based content. This can be useful for
        businesses, researchers, or anyone interested in understanding public opinion or sentiment trends.</p>
    <p>Give SentiSight a try and see how it can help you analyze sentiment with ease!</p>
</div>
```

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
    content: ["./**/templates/*.{html,js}"],
    theme: {
        extend: {},
    },
    plugins: [],
};
```

Logo.svg

```
<svg width="564" height="697" viewBox="0 0 564 697" fill="none"
xmlns="http://www.w3.org/2000/svg">
            <path class="fill-black dark:fill-white"
                d="M0.297302 453.777L35.8632 451.913L37.7272
487.479L2.16123 489.342L0.297302 453.777Z" />
            <path class="fill-black dark:fill-white"
                d="M435 632.613L464.526 612.697L484.441 642.223L454.915
662.139L435 632.613Z" />
            <path class="fill-black dark:fill-white"
                d="M103.019 378.406L122.935 348.88L152.461 368.796L132.545
398.322L103.019 378.406Z" />
```

```
            <path class="fill-black dark:fill-white"
                d="M372.197 481.267L362.979 446.866L397.38 437.648L406.598
472.049L372.197 481.267Z" />
            <path class="fill-black dark:fill-white"
                d="M90.1046 38.6889L112.031 66.7537L83.9664 88.6804L62.0398
60.6156L90.1046 38.6889Z" />
            <path class="fill-black dark:fill-white"
                d="M38.2234 267.039L56.0308 297.883L25.1875 315.69L7.38011
284.847L38.2234 267.039Z" />
            <path class="fill-black dark:fill-white"
                d="M261.934 547.701L263.745 512.133L299.314 513.944L297.502
549.513L261.934 547.701Z" />
            <path class="fill-black dark:fill-white"
                d="M155.186 442.419L189.941 434.64L197.72 469.395L162.965
477.174L155.186 442.419Z" />
            <path class="fill-black dark:fill-white"
                d="M150.002 617.374L165.422 585.27L197.525 600.69L182.105
632.794L150.002 617.374Z" />
            <path class="fill-black dark:fill-white"
                d="M137.918 533.68L163.55 508.953L188.277 534.584L162.645
559.312L137.918 533.68Z" />
            <path class="fill-black dark:fill-white"
```

```
                d="M294.369 599.893L294.969 564.283L330.578 564.883L329.978
600.493L294.369 599.893Z" />
            <path class="fill-black dark:fill-white"
                d="M372.458 370.213L365.559 405.153L330.619 398.254L337.518
363.314L372.458 370.213Z" />
            <path class="fill-black dark:fill-white"
                d="M292.229 652.422L290.773 616.837L326.358 615.382L327.814
650.967L292.229 652.422Z" />
            <path class="fill-black dark:fill-white"
                d="M266.405 292.806L301.207 300.371L293.641 335.173L258.839
327.607L266.405 292.806Z" />
            <path class="fill-black dark:fill-white"
                d="M212.262 338.511L247.063 346.077L239.498 380.878L204.696
373.313L212.262 338.511Z" />
            <path class="fill-black dark:fill-white"
                d="M406.695 71.4L436.514 90.8737L417.041 120.693L387.221
101.219L406.695 71.4Z" />
            <path class="fill-black dark:fill-white"
                d="M398.165 196.688L407.103 231.163L372.628 240.1L363.69
205.625L398.165 196.688Z" />
            <path class="fill-black dark:fill-white"
                d="M555 539.743L539.473 571.795L507.421 556.268L522.948
524.216L555 539.743Z" />
```

```
            <path class="fill-black dark:fill-white"
                d="M135.988 182.696L171.553 184.582L169.667 220.147L134.102
218.261L135.988 182.696Z" />
            <path class="fill-black dark:fill-white"
                d="M488.288 142.974L503.847 175.01L471.811 190.57L456.251
158.533L488.288 142.974Z" />
            <path class="fill-black dark:fill-white"
                d="M536.072 257.789L537.129 293.388L501.53 294.445L500.473
258.846L536.072 257.789Z" />
            <path class="fill-black dark:fill-white"
                d="M232.429 0L239.532 34.8993L204.632 42.0022L197.529
7.10289L232.429 0Z" />
            <path class="stroke-black dark:stroke-white"
                d="M163.675 604.667C66.6885 540.115 68.5964 448.535 68.5964
448.535H124.878C124.878 483.832 154.771 526.442 154.771 526.442M22.6356
487.28C40.1326 621.155 230.054 753.486 446.367 645.688M137.282
382.393C259.126 430.662 296.718 408.627 375.138 451.715M74.1233
73.6677C13.1156 144.376 4.48378 191.324 25.4711 279.089M179.575
465.706C194.52 519.764 268.293 529.622 268.293 529.622M363.753
392.569C438.735 395.749 502.333 541.07 327.121 583.362M289.917
314.662C402.485 343.281 486.116 345.824 497.564 466.978C497.564 556.969
437.782 617.705 320.444 633.286M212.963 357.272C88.6298 327.063 65.5497
209.829 118.203 136.27C161.449 75.8519 272.109 38.0111 407.255
92.0692M535.404 536.936C549.174 499.448 550.995 481.924 546.543
439.631C542.091 397.339 516.794 364.06 479.438 329.925C431.351 285.984
```

```
332.298 273.706 320.434 271.415C302.318 267.917 222.821 251.7 211.055
239.298C199.29 226.897 196.11 202.553 218.051 183.968C239.992 165.383
326.168 139.45 385.631 206.863M154.453 193.541C160.335 147.369 228.842
120.913 259.39 117.224C317.632 110.191 373.028 117.224 406.937
156.939C422.2 174.816 441.597 223.081 441.597 223.081L496.292
224.671L485.162 177.926M229.817 19.2498C393.581 -4.59933 552.576 69.8101
552.576 273.323L530.952 275.867"
                stroke="" stroke-width="22" />
        </svg>
```