SENTIMENTAL ANALYSIS USING GOOGLE GEMINI LLM

A DESIGN PROJECT REPORT

Submitted by

NANDHINI K

POOJA P

PRIYADHARSHINI A

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)
Samayapuram – 621 112

NOVENBER,2024

SENTIMENTAL ANALYSIS USING GOOGLE GEMINI LLM

A DESIGN PROJECT WORK

Submitted by

NANDHINI K(811722104098)

POOJA P(811722104108)

PRIYADHARSHINI A(811722104114)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai, Approved by AICTE, New Delhi)
Samayapuram – 621 112

C

K. RAMAKRISHNAN COLLEGE OF TECNOLOGY

(AUTONOMOUS)

SAMAYAPURAM-621 112

BONAFIDE CERTIFICATE

Certified that this project report titled "**SENTIMENTAL ANALYSIS USING GOOGLE GEMINI LLM"** is bonafide work of **NANDHINI K** (**811722104098**), **POOJA P (811722104108), PRIYADHARSHINI A** (**811722104114**) who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate**.**

SIGNATURE

Dr.A.Delphin Carolina Rani M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

SIGNATURE

Ms.S.Uma Mageshwari M.E.,

SUPERVISOR

Assistant Professor

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

Submitted for the viva-voce examination held on.......

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the work entitled **"SENTIMENTAL ANALYSIS USING GOOGLE GEMINI LLM"** is submitted in partial fulfilment of the requirement for the reward of the degree in B.E., Anna University, Chennai, is a record of the work carried out by me during the academic year 2024-2025 under the supervision and guidance of **Ms.S.Uma Mageshwari M.E.,** Assistant Professor**, Department of Computer Science And Engineering in K. Ramakrishnan College of Technology (Autonomous).** The extent and source of information are derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not been submitted for the award of any degree or diploma, either in this or any other University.

SIGNATURE

NANDHINI K

POOJA P

PRIYADHARSHINI A

PLACE: Samayapuram

DATE:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtness to our institution **"K RAMAKRISHNAN COLLEGE OF TECHNOLOGY"**, for providing us with the opportunity to do this project .

We are glad to credit honorable chairman **Dr. K RAMAKRISHNAN B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration to complete it.

We would like to thank **Dr. N VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project with full satisfaction.

We whole heartily thank **Dr. A DELPHIN CAROLINA RANI, M.E., Ph.D.,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing her support to pursue this project.

We express our deep and sincere gratitude and thanks to our project guide **Ms.S.UMA MAGESHWARI M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry our this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Sentiment Analysis Using Google Gemini LLM helps users understand and improve their mental well-being. Using Django, Figma, Google Gemini LLM, and Git, users write daily narratives that are analysed to determine their emotional state. If negative, the system offers personalized suggestions; if positive, it provides uplifting messages. Daily reports offer insights into mental health trends and suggest activities like mindfulness exercises. This project provides an easy, private, and supportive way for users to manage their mental health effectively.

TABLE OF CONTENTS

vii

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.No	**Acronym**	**Abbreviations**
1	LLM	Large Language Model
2	HTML	HyperText Markup Language
3	CSS	Cascading Style Sheets
4	RAM	Random Access Memory
5	SSD	Solid State Drive
6	GPU	Graphic Processing Unit
7	ORM	Object Relational Mapping

CHAPTER 1

INTRODUCTION

In today’s fast-paced world, maintaining good mental health is more important than ever. Many people struggle to find someone to share their thoughts and feelings with, which can lead to feelings of loneliness and sadness. To address this issue, we have developed a project called “Sentimental Analysis using Google Gemini LLM” specifically for the healthcare sector. This innovative project aims to help people understand and improve their mental health through the power of artificial intelligence.

Our project uses Google Gemini, a cutting-edge language model, to analyze the daily narratives of users. A narrative is a simple account of a person’s day, detailing their activities, thoughts, and feelings. Users can write about their experiences, and the system will analyze the text to determine their mental state. By examining the words and phrases used, the model can identify whether a person is feeling good, bad, or somewhere in between.

If the analysis shows that the person is feeling bad, the system will provide helpful suggestions to improve their mental state. These suggestions might include practical advice, such as taking a walk, practicing mindfulness, or reaching out to a friend. The goal is to offer supportive and actionable tips that can help lift the person’s spirits and improve their overall well-being.

On the other hand, if the analysis indicates that the person is in a good mental state, the system will provide positive and uplifting messages. These messages aim to reinforce the positive feelings and encourage the person to continue with the behaviors that contribute to their good mental health.

In addition to providing real-time feedback and suggestions, our project also generates daily tasks designed to improve mental well-being. These tasks might include activities like journaling, meditating, exercising, or spending time outdoors. By completing these tasks, users can build healthy habits that contribute to long-term mental health benefits.

The primary purpose of our project is to make it easier for people to share their thoughts and feelings in a safe and supportive environment. By using advanced AI technology, we can offer personalized mental health support to individuals who might not have access to traditional resources. Whether someone is feeling lonely, stressed, or just needs a little extra encouragement, our system is here to help.

1.1 LARGE LANGUAGE MODEL (LLM)

A Large Language Model (LLM) is a type of artificial intelligence designed to understand and generate human language. These models are built using deep learning techniques, particularly neural networks with many layers, allowing them to process and analyze vast amounts of text data. By training on diverse datasets, LLMs learn grammar, facts about the world, and even nuances of language such as idioms and cultural references.

LLMs operate by predicting the next word in a sentence, which enables them to generate coherent and contextually relevant text. This capability allows them to perform a variety of tasks, including writing essays, answering questions, translating languages, summarizing information, and even engaging in conversations.

One of the key features of LLMs is their ability to fine-tune on specific tasks or datasets, enhancing their performance in particular domains. For instance, an LLM can be fine-tuned to excel in medical diagnosis support, legal document analysis, or customer service interactions.

Despite their powerful capabilities, LLMs have limitations. They can sometimes produce incorrect or nonsensical answers and are sensitive to the data they are trained on, which can introduce biases. Continuous research and development are focused on improving their accuracy, reliability, and ethical use.

1.1.1 Working Of LLM

A Large Language Model (LLM) is an advanced type of artificial intelligence designed to understand and generate human language. Here’s a brief overview of how it works:

Training: An LLM is trained on vast amounts of text data from books, articles, websites, and other sources. This training helps the model learn patterns, grammar, facts, and nuances of language.

Architecture: LLMs use a neural network architecture, typically based on transformers. This architecture allows the model to process and generate text efficiently by understanding context and relationships between words over long distances in a text.

Tokenization: Text is broken down into smaller units called tokens, which can be words, subwords, or characters. This tokenization helps the model manage and process the text data effectively.

Inference: Once trained, the LLM can generate text, answer questions, translate languages, and perform other language-related tasks. It predicts the next word in a sentence based on the context it has learned during training.

Fine-Tuning: For specific tasks or industries, LLMs can be fine- tuned on specialized datasets to improve their performance in particular areas.

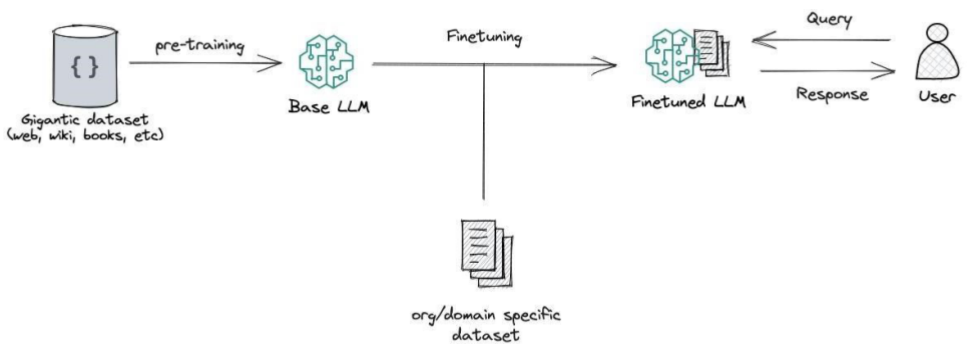

pre-training
Gigantic dataset (web, wiki, books, etc)
Base LLM
Finetuning
Finetuned LLM
Query
Response
User
org/domain specific dataset

FIG 1.1.1 WORKING OF LLM

Overall, LLMs are powerful tools for natural language processing, enabling a wide range of applications from chatbots to automated content creation.

1.1.2 Prompt Engineering

Prompt engineering is the process of crafting inputs, known as prompts, to optimize the performance of large language models (LLMs).
These models, like GPT-4, respond to the prompts they receive. Well- designed prompts can guide the model to produce more accurate, relevant, and high-quality outputs.
To engineer effective prompts, one must:

Be Clear and Specific: Use precise language to direct the model towards the desired response.

Provide Context: Include relevant background information to help the model understand the task.

Iterate and Refine: Experiment with different phrasing and structures to find the most effective prompt.

Test and Evaluate: Continuously test the outputs and adjust prompts to improve the results.

Prompt engineering is crucial for applications in customer service, content creation, and data analysis, ensuring LLMs perform efficiently and effectively.

1.2 AVAILABLE MODELS

OpenAI's Language Model (LLM) and Google's Gemini LLM are two powerful tools for natural language processing. OpenAI's LLM offers a diverse range of models, from small to extra-large, catering to various needs.

It's renowned for its versatility and effectiveness in generating human-like text. Google's Gemini LLM, on the other hand, boasts advanced capabilities in understanding context and generating responses, especially in conversational settings. Both models leverage vast amounts of data and cutting-edge algorithms to comprehend and produce language with remarkable fluency and accuracy.

1.2.1 OpenAI's LLM

OpenAI's Language Model (LLM) is an advanced artificial intelligence designed to understand and generate human-like text. It's like having a smart conversation partner who can help you write, learn, and create. LLM uses a technique called deep learning, where it learns patterns from vast amounts of text data. This allows it to understand language in a way that’s surprisingly similar to how humans do.

Imagine LLM as a super-smart librarian in a giant library filled with books, articles, and websites. When you ask it a question or give it a prompt, it quickly searches through its library to find the most relevant information and then generates a response for you. Whether you need help with writing an essay, coming up with ideas for a story, or just want to learn something new, LLM is there to assist.

It’s important to note that while LLM is incredibly helpful, it’s not perfect. Sometimes it might misunderstand your question or provide an answer that isn’t quite what you were looking for. However, it’s constantly learning and improving, so the more you interact with it, the better it gets at understanding and generating text.

1.2.2 Google Gemini LLM

Google Gemini is an advanced language model developed by Google, designed to understand and analyze text data with remarkable accuracy.
Gemini stands for “Generating Multimodal Inferences,” indicating its capability to comprehend various types of information, including text, images, and more.

In the realm of healthcare, Google Gemini LLM (Large Language Model) plays a pivotal role in sentiment analysis. Sentiment analysis involves evaluating the emotions, opinions, and attitudes expressed in textual data. By leveraging Gemini’s natural language processing abilities, healthcare professionals can gain valuable insights from patient reviews, social media posts and other sources of healthcare-related text.

Using Gemini LLM for sentiment analysis in healthcare allows practitioners to monitor patient satisfaction, identify areas for improvement in healthcare services, and detect potential issues or concerns early on. This technology enables healthcare organizations to better understand patient experiences, tailor services to meet their needs, and ultimately enhance overall patient care.

Google Gemini LLM’s robust capabilities in natural language understanding and sentiment analysis make it a powerful tool for improving healthcare outcomes, fostering patient-centric care, and driving continuous improvement in the healthcare industry

1.3 WORKING OF GOOGLE GEMINI LLM

Google Gemini LLM (Language Model) is an advanced tool designed for understanding and analyzing text data. Using cutting-edge machine learning algorithms, Gemini LLM can comprehend the nuances of language, including context, tone, and sentiment. The process begins with feeding the model a large amount of text data to train it to recognize patterns and relationships within language.

Once trained, Gemini LLM can perform various tasks, including sentiment analysis. Sentiment analysis involves examining text to determine the emotions or attitudes expressed within it. Gemini LLM can accurately identify whether a piece of text conveys positive, negative, or neutral sentiment, enabling businesses and researchers to gain valuable insights into customer opinions, market trends, and public perception.

By leveraging the power of Google's infrastructure and machine learning expertise, Gemini LLM offers a robust and efficient solution for sentiment analysis and other natural language processing tasks. Its ability to understand language at a deep level makes it a valuable tool for a wide range of applications, from social media monitoring to customer feedback analysis.

1.4 OBJECT OF INVENTION

The purpose of this project is to create a system that helps people understand and improve their mental health using advanced technology. Many people find it hard to share their feelings with others, which can affect their mental well-being. This system uses Google Gemini LLM, an advanced language model, to analyze a person's daily thoughts and feelings.

When a user provides a narrative of their day, the system analyzes it to determine their mental state. If the analysis shows that the user is feeling good, the system sends a positive and uplifting message to encourage them. If the user is feeling bad, the system suggests ways to improve their mental state.

Additionally, the system generates a report based on the user's mental state over time and creates daily tasks to help improve their mental well-being. These tasks are simple and aim to make the user feel better gradually.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: Sentimental Analysis with Amazon Review Data

AUTHORS: Ming XiangChen and Stanford.

In their study, Chen and Mal analyzed Amazon review data to predict product ratings using sentiment analysis. Their methodology involved traditional machine learning algorithms, focusing on text classification to determine sentiment polarity. However, the results indicated a modest accuracy rate of 46.66%, revealing significant room for improvement. The study underscored the challenges of accurately capturing consumer sentiment using basic machine learning techniques, which often fail to understand the contextual nuances of natural language. This limitation suggests a need for more sophisticated models that can enhance predictive accuracy.

2.2TITLE: Sentiment Analysis of Amazon Products Using Ensemble Machine Learning Algorithms

AUTHORS: Jayakumar Sadhasivam and R.Kalivaradhan.

Sadhasivam and Kalivaradhan expanded on traditional approaches by employing ensemble machine learning algorithms for sentiment analysis of Amazon product reviews. Their approach aimed to combine the strengths of multiple algorithms to improve sentiment classification. Despite these efforts, the study reported unsatisfactory accuracy levels. The ensemble methods did not significantly outperform individual algorithms, indicating that the inherent complexities and subtleties of human language posed substantial challenges.

2.3TITLE: Sentimental Analysis of Twitter Data using Classifier Algorithms

AUTHORS: Sharvil Shah, K. Kumar, and Ra. K.Sarvananguru.

Shah, Kumar, and Sarvananguru focused on sentiment analysis of Twitter data, which presents unique challenges due to the platform's brevity, informal language, and extensive use of hashtags and sarcasm. Their study utilized various classifier algorithms to analyze tweets, but encountered difficulties in accurately interpreting sentiment. The algorithms struggled particularly with hashtag classification and sarcasm detection, leading to suboptimal performance. This research highlighted the limitations of traditional classifier algorithms in handling the dynamic and nuanced nature of social media language, emphasizing the need for more sophisticated tools to improve sentiment analysis accuracy.

TITLE: A Sentimental Analysis System Using Zero-Shot Machine Learning Technique

AUTHORS: Shreya Ganga and A. Solanki

Ganga and Solanki introduced a zero-shot machine learning technique for sentiment analysis, aiming to address the shortcomings of previous methods. Zero-shot learning enables the model to predict sentiments without requiring extensive labeled training data for every new task. Their study showed promising results in terms of accuracy and the ability to handle diverse datasets. However, the authors pointed out that the system still struggled with understanding nuanced sentiments and context- specific expressions. This research suggested that while zero-shot learning offers a step forward, there remains a need for models that can better comprehend the intricacies of human language.

TITLE: Google Gemini LLM :Revolutionizing multimodel sentiment analysis

AUTHORS: DeepMind Team

YEAR:2023

This work introduces Gemini, a multimodal large language model by Google, which demonstrates advanced capabilities in sentiment analysis. It emphasizes Gemini's strengths in handling multilingual text, understanding sarcasm, and providing fine-grained sentiment classifications. The study benchmarks Gemini against GPT-4 and finds superior performance in certain complex sentiment tasks.

TITLE: Sentiment analysis with large language models(LLM)

AUTHORS: Why Labs Team

YEAR:2023

This article explores how modern LLMs like Google Gemini are transforming sentiment analysis by leveraging contextual understanding and cross-domain adaptability. It outlines practical implementations, such as real-time monitoring and customer feedback analysis, and provides insights into fine-tuning LLMs for domain-specific sentiment tasks.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System

The current system for sentiment analysis in healthcare predominantly relies on traditional methods and manual analysis, which are often limited in scope and accuracy.

Traditional approaches struggle with understanding context, detecting sarcasm, and processing nuanced language, leading to lower accuracy in sentiment classification.

Existing systems often require extensive preprocessing of data, including manual tagging and categorization, which is time-consuming and prone to errors.

These systems typically utilize rule-based algorithms or basic machine learning models that do not effectively capture the complexities of human language.

There is a significant challenge in scaling these traditional systems to handle large datasets, limiting their applicability in real-time scenarios.

Overall, the existing systems are less efficient, require substantial manual effort, and fail to provide the depth of analysis needed for accurate sentiment detection in healthcare.

3.2 Proposed System

Google Gemini LLM (Large Language Model): This cutting-edge technology acts as the core engine of our sentiment analysis system. It processes the user’s narrative using advanced natural language processing algorithms to gauge their emotional state accurately.

Data Processing Modules: Within the system, there are specialized modules dedicated to processing the incoming data. These modules clean and structure the narrative data, making it ready for analysis by Google Gemini LLM.

Machine Learning Algorithms: Supporting Google Gemini LLM are machine learning algorithms trained on extensive datasets of emotional expressions and mental health indicators. These algorithms enhance the accuracy and depth of the sentiment analysis results.

Recommendation Engine: For instances where the analysis indicates a negative or suboptimal mental state, our system incorporates a recommendation engine. This engine suggests personalized strategies and activities aimed at improving the user's mental well.

Positive Messaging System: Conversely, if the analysis indicates a positive mental state, the system responds with uplifting and encouraging messages. These positive reinforcements contribute to maintaining and enhancing the user's positive mental state.

Reporting and Monitoring Tools: Our system generates comprehensive reports based on the sentiment analysis results. These reports offer insights into the user's emotional trends over time, helping both users and healthcare providers track progress and identify areas for improvement.

Daily Task Generation: To actively promote mental wellness, our system generates daily tasks aligned with the user's emotional needs. These tasks are designed to be manageable yet impactful, contributing to a holistic approach to mental health care.

Chapter 4

SYSTEM DESIGN

4.1 Architecture Diagram

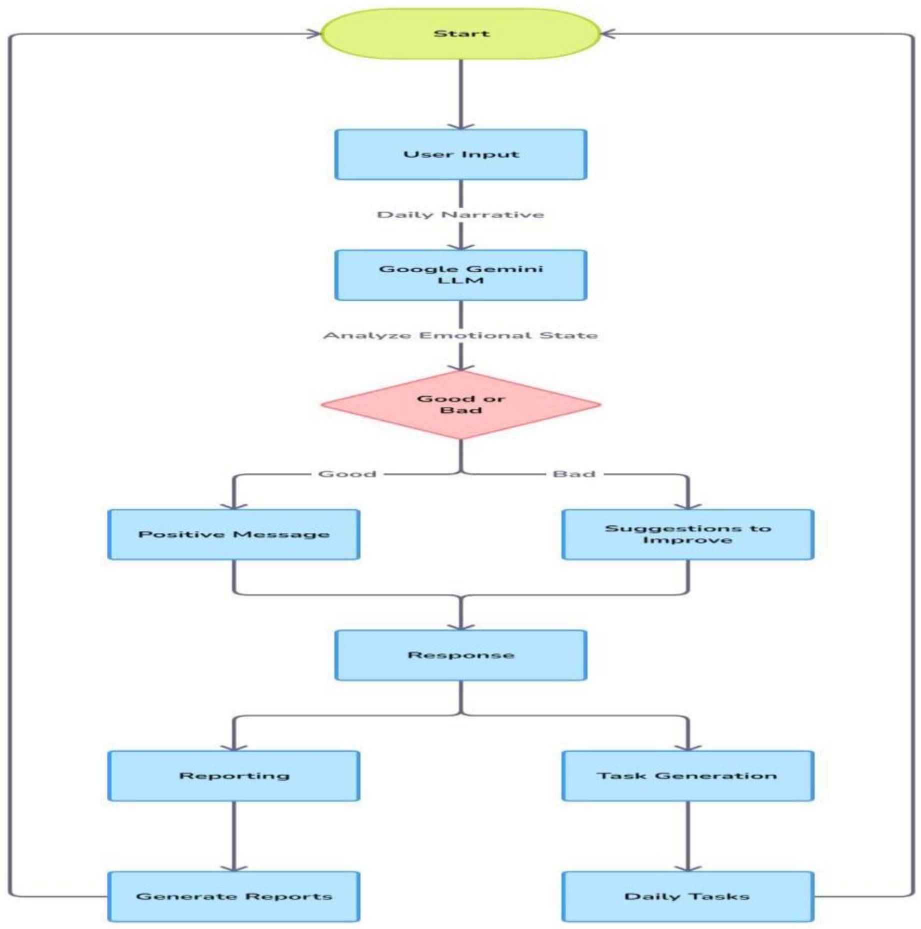


Fig 4.1 Architecture Diagram

CHAPTER 5

MODULE DESCRIPTION

5.1 Mental State Analysis

The Mental State Analysis module page of our web application is a pivotal component designed to provide advanced sentiment analysis tailored specifically for the healthcare sector. Leveraging the robust capabilities of Google's Gemini Large Language Model (LLM), this module is developed using Django for backend functionality and Figma for frontend design, ensuring a seamless and user-friendly experience.

At its core, this module excels in analyzing textual statements to gauge the user's mental condition accurately. Through a user-friendly interface, individuals can input their statements into the designated text box. Upon clicking the "Analyze mental state" button, the system meticulously evaluates the sentiment, distinguishing between positive and negative sentiments with remarkable precision.

What sets this module apart is its proactive approach towards mental well-being. In addition to identifying negative sentiments, it goes a step further by offering actionable insights to improve the user's mental state. This proactive feedback mechanism plays a vital role in promoting mental wellness by empowering users with personalized recommendations based on their sentiment analysis results.

The module also encompasses comprehensive reporting functionalities. Users can access detailed sentiment reports based on past analysis providing invaluable insights into their mental health trends over time. This feature is invaluable for both individuals and healthcare providers, facilitating informed decision-making and timely interventions when necessary.

Furthermore, to encourage proactive mental health management, the module generates daily tasks customized to address specific mental health needs. These tasks are intelligently designed based on sentiment analysis results, empowering users to take proactive steps towards enhancing their mental well-being.

The integration of Google Gemini LLM significantly enhances the module's capabilities. It enables the system to accurately interpret complex emotions, understand nuances such as sarcasm, and process diverse datasets with exceptional accuracy. This technological advancement translates into more reliable sentiment analysis compared to conventional methodologies, making it a game-changer in mental health management.

5.2 Get Report

The “Get Report” module page within our innovative sentiment analysis project is designed to provide users with a detailed and insightful report on their past sentiments. Leveraging the advanced capabilities of Google Gemini LLM, this module aims to revolutionize mental wellness in healthcare by accurately assessing mental states and offering personalized recommendations for improvement.

Upon landing on the “Get Report” page, users are greeted with a visually engaging interface. The page’s header prominently displays “Overall Report on Your Past Sentiments” in vibrant, multi-colored text, setting the tone for a comprehensive analysis. Below this header, a meticulously crafted black text box presents a thorough examination of the user’s predominant emotional trends.

The analysis delves into the user’s historical emotional expressions, highlighting patterns and tendencies. For instance, if the user consistently exhibits positive emotions such as happiness and encouragement, the report reflects a strong sense of self-confidence and motivation. It suggests that the user’s positive outlook is accompanied by a belief in their capabilities and a proactive approach to overcoming life’s challenges.

The core functionality of this module lies in its ability to perform real- time analysis on textual inputs, accurately determining the user’s emotional tone and overall mental state. For users identified as experiencing a negative mental state, the system offers tailored suggestions and strategies to facilitate positive emotional shifts. These recommendations are rooted in evidence-based psychological techniques and are customized to suit the user’s unique profile.

Furthermore, the “Get Report” module generates detailed reports, providing valuable insights into mental well-being trends, identifying patterns, and pinpointing triggers. This empowers users to track their emotional journey over time and make informed decisions for self-improvement. As part of its holistic approach to mental wellness, the platform also generates daily tasks aimed at enhancing emotional resilience, promoting positive habits, and creating a supportive environment for maintaining optimal mental health.

5.3 Get Today’s Task

The “Get Today’s Task” module is an integral part of our mental wellness platform, designed to leverage the power of Google Gemini LLM for sentiment analysis and personalized mental health recommendations. This module is accessible through the homepage, where users can engage with daily tasks aimed at enhancing emotional resilience and well-being.

The primary function of the “Get Today’s Task” module is to provide users with a daily task specifically crafted to promote mindfulness and emotional stability. For example, the current task may revolve around mindfulness meditation, encouraging users to find a comfortable position, close their eyes, and focus on their breath. This practice fosters a sense of calmness and aids in stress reduction, contributing to overall mental wellness.

Beyond offering daily tasks, this module conducts real-time analysis of user-provided text to assess emotional tone and the user’s overall mental state. If a negative mental state is detected, the platform responds with tailored strategies designed to promote a positive shift in emotions. These recommendations are rooted in evidence-based psychological techniques and are customized to each user’s unique profile, ensuring relevance and effectiveness in improving mental well-being.

Moreover, the “Get Today’s Task” module generates detailed reports on users’ mental well-being trends, available on a weekly or monthly basis.

These reports provide insights into patterns, triggers, and progress over time, empowering users to make informed decisions for self-improvement and track their emotional journey effectively.

Technologically, this module utilizes Django for robust backend development, ensuring scalability and reliability. Figma is employed for intuitive and user-friendly interface design, enhancing the overall user experience. Version control is managed through Git, facilitating efficient project management and collaboration among team members.

At its core, the “Get Today’s Task” module serves as a comprehensive tool for ongoing mental health assessment and improvement, harnessing advanced technology to deliver personalized and impactful mental wellness support to users.

CHAPTER 6
SYSTEM REQUIREMENTS

6.1 Hardware Specifications

For optimal performance, the system requires a robust hardware setup. The server hosting the Django backend should be equipped with at least 16GB of RAM and a multi-core processor, preferably quad-core or higher. This configuration ensures the server can efficiently handle concurrent user requests. Additionally, incorporating a solid-state drive (SSD) with a minimum of 500GB storage capacity is crucial for fast data retrieval and storage operations.

To effectively manage real-time sentiment analysis through the Google Gemini LLM integration, it is recommended to use a dedicated GPU, such as an NVIDIA Tesla or an equivalent model, with at least 12GB of VRAM. This specification is essential to handle the intensive computational requirements of real-time sentiment analysis.

Furthermore, a reliable internet connection with high bandwidth is necessary to ensure seamless data processing and efficient communication between the system components. This comprehensive hardware setup is pivotal in maintaining the performance and reliability of the sentiment analysis system.

6.2 Software Specification

6.2.1 Software Tools Requirements

6.2.1.1 HTML

HTML, or Hyper Text Markup Language, is the standard language used for creating and designing webpages. It structures content on the web, allowing the incorporation of text, images, links, and other elements. HTML is fundamental to web development and works alongside CSS (Cascading Style Sheets) and JavaScript to create visually appealing and interactive websites.

One of the primary advantages of HTML is its simplicity and ease of use. It is relatively straightforward to learn, making it accessible for beginners. HTML is also highly compatible with all web browsers, ensuring that webpages are displayed consistently across different platforms. Additionally, HTML5, the latest version, introduces several powerful features, such as improved multimedia support (audio and video), enhanced graphic capabilities with the canvas element, and better semantic elements that provide more meaning to the content (e.g., <header>, <footer>, <article>, and <section>).

HTML is used in a wide range of applications, from creating static webpages to developing complex web applications. It serves as the backbone for all websites, enabling the organization and presentation of content. HTML is also crucial in the development of web forms, which facilitate user interaction and data collection on website.

Key features of HTML include its ability to embed multimedia elements,
its support for hyperlinks that connect different webpages, and its use of tags to structure and format content. HTML documents are plain text files with a .html or .html extension, and they are interpreted by web browsers to render the visual presentation of the content.

Overall, HTML is an essential technology for web development, offering a robust framework for creating and managing online content. Its ease of use, wide compatibility, and powerful features make it an indispensable tool for web developers and designers.

6.2.1.2 CSS

CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation of a document written in HTML or XML. CSS controls the layout of multiple web pages simultaneously, allowing for consistent and attractive design. One of its primary advantages is the separation of content from design, enabling developers to make style changes across an entire website by modifying just one file. This separation also enhances maintainability and reduces complexity in HTML files.

CSS offers numerous advantages. It improves page load times by reducing HTML file size and streamlining the style application process. CSS provides enhanced control over layout, allowing for precise positioning of elements, custom fonts, colors, and responsive designs that adapt to various screen sizes and devices. This flexibility is crucial for creating user-friendly and accessible websites.

In terms of applications, CSS is fundamental in web development, used extensively to style websites and web applications. It is also employed in mobile app development, email template design, and any other context where HTML is used for content structuring. Advanced CSS features, such as animations and transitions, enable the creation of dynamic and interactive user interfaces.

CSS includes various features that enhance web development. Selectors allow developers to target HTML elements efficiently, applying styles based on their attributes, classes, or IDs. Box model properties, including margins, padding, and borders, provide control over spacing and element sizing. Flexbox and Grid layouts offer sophisticated methods for creating complex, responsive designs.

Media queries enable the implementation of responsive web design, ensuring that web pages look good on all devices. Furthermore, CSS preprocessors like Sass and LESS extend CSS capabilities with variables, nesting, and functions, making it easier to write and manage styles.

Overall, CSS is a powerful tool that plays a crucial role in modern web development, offering numerous advantages and features that enhance both the aesthetics and functionality of web pages.

6.2.1.3 JAVASCRIPT

JavaScript is a versatile and widely-used programming language primarily known for its role in web development. It enables interactive and dynamic content on websites, enhancing user experience. As a high-level, interpreted language, JavaScript is easy to learn and implement, making it accessible for both beginners and experienced developers.

One of JavaScript’s significant advantages is its ability to run in any web browser without requiring additional plugins, ensuring broad compatibility and ease of deployment. Its asynchronous nature, enabled through features like callbacks, promises, and async/await, allows for efficient handling of operations, such as server requests, without blocking the main thread. This contributes to smoother performance and a better user experience.

JavaScript’s applications extend beyond web development. It is used in server-side programming through environments like Node.js, which allows developers to use JavaScript for backend development, creating scalable and high-performance applications. Additionally, JavaScript is pivotal in developing mobile applications, desktop applications, and even game development, thanks to frameworks like React Native, Electron, and Babylon.js, respectively.

Key features of JavaScript include its event-driven programming capabilities, which enable responsive and interactive web pages. The language supports object-oriented, imperative, and functional programming paradigms, providing flexibility in coding styles. The extensive ecosystem of libraries and frameworks, such as React, Angular, and Vue.js, significantly accelerates development processes and enhances functionality.

JavaScript also benefits from a large and active community, which contributes to a wealth of resources, tutorials, and open-source projects. This community support makes it easier for developers to find solutions, share knowledge, and collaborate on projects. Overall, JavaScript's versatility, ease of use, and extensive ecosystem make it an indispensable tool in modern software development.

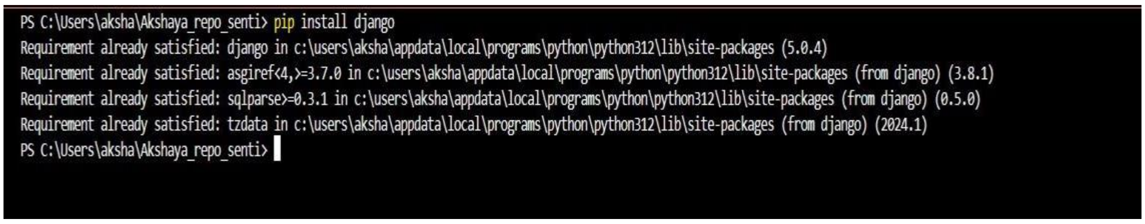
6.2.1.4 DJANGO

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is known for its "batteries- included" philosophy, which means it comes with many built-in features and utilities that simplify web development. Django's advantages include its robust security features, scalability, and versatility. It provides protection against common security threats such as SQL injection, cross-site scripting, and cross-site request forgery, making it a secure choice for web applications.

Django is widely used in various applications, ranging from simple websites to complex web applications. It is particularly well-suited for developing content management systems, social networks, scientific computing platforms, and e-commerce sites. Its scalability allows it to handle high traffic and large amounts of data, making it a preferred choice for many large-scale projects.

The framework includes several powerful features that streamline development. Its ORM (Object-Relational Mapping) system simplifies database interactions by allowing developers to use Python code instead of SQL. The admin interface, which is automatically generated, provides an easy way to manage application data. Django's templating engine allows for the creation of dynamic web pages with ease. Additionally, the framework supports URL routing, form handling, and internationalization, making it a comprehensive tool for web development.

Overall, Django stands out due to its emphasis on reusability and "don't repeat yourself" (DRY) principles. This approach not only speeds up the development process but also ensures that the codebase remains clean and maintainable. Django's extensive documentation and active community further contribute to its appeal, offering robust support and resources for developers.



```
PS C:\Users\aksha\Akshaya_repo_senti> pip install django
Requirement already satisfied: django in c:\users\aksha\appdata\local\programs\python\python312\lib\site-packages (5.0.4)
Requirement already satisfied: asgiref<4,>=3.7.0 in c:\users\aksha\appdata\local\programs\python\python312\lib\site-packages (from django) (3.8.1)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\aksha\appdata\local\programs\python\python312\lib\site-packages (from django) (0.5.0)
Requirement already satisfied: tzdata in c:\users\aksha\appdata\local\programs\python\python312\lib\site-packages (from django) (2024.1)
PS C:\Users\aksha\Akshaya_repo_senti>
```

Figure 6.2.1.4: Pip install Django

6.2.1.5 GOOGLE GEMINI LLM

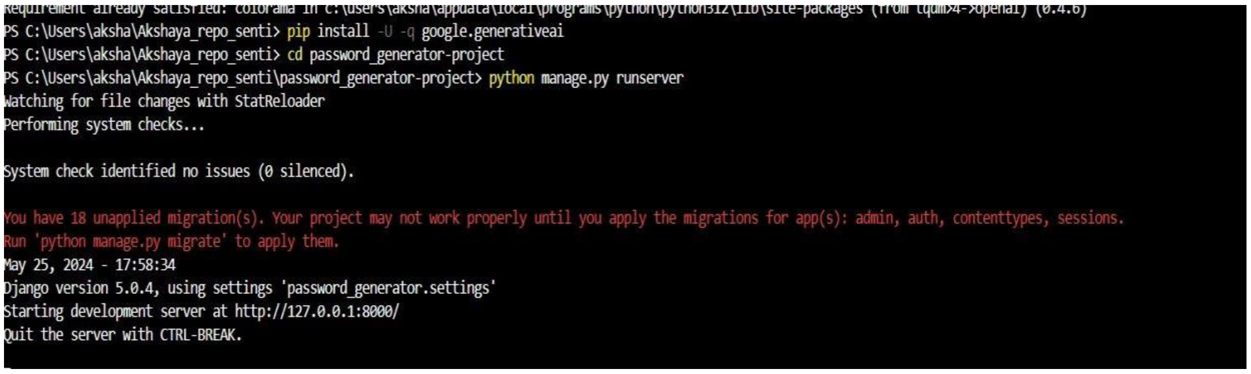
Google Gemini LLM is a state-of-the-art language model designed to perform advanced natural language processing tasks. It leverages deep learning algorithms to understand and generate human-like text, making it highly effective for applications that require language comprehension and generation. One of the key advantages of Google Gemini LLM is its ability to process and analyze large volumes of text data with remarkable accuracy and speed. This makes it particularly useful in fields like healthcare, where understanding nuanced language and emotional contexts is crucial.

The applications of Google Gemini LLM are diverse. In the healthcare sector, it can be used for sentiment analysis, where it interprets patients' narratives to gauge their mental state. This capability is invaluable for providing timely interventions and personalized recommendations for mental well-being. Beyond healthcare, Google Gemini LLM is employed in customer service to handle queries and provide automated support, in content creation for generating high-quality articles and reports, and in educational tools for tutoring and providing personalized learning experiences.

Google Gemini LLM boasts several advanced features. Its deep learning architecture allows it to understand context, detect sarcasm, and process complex linguistic structures, which traditional models struggle with.

The model is also capable of continuous learning, meaning it can improve over time as it is exposed to more data. Additionally, it supports multilingual processing, making it versatile for global applications. The integration of machine learning algorithms further enhances its ability to provide accurate and detailed analysis, contributing to a more refined and user-centric experience.

In summary, Google Gemini LLM stands out for its advanced language processing capabilities, diverse applications, and innovative features, making it a powerful tool in modern AI-driven solutions.



```
Requirement already satisfied: colorama in c:\users\aksha\appdata\local\programs\python\python312\lib\site-packages (from tqdm>4->openai) (0.4.6)
PS C:\Users\aksha\Akshaya_repo_senti> pip install -U -q google.generativeai
PS C:\Users\aksha\Akshaya_repo_senti> cd password_generator-project
PS C:\Users\aksha\Akshaya_repo_senti\password_generator-project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 25, 2024 - 17:58:34
Django version 5.0.4, using settings 'password_generator.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 6.2.15 : Pip install -U -q google.generativeai

CHAPTER 7

SYSTEM TESTING

7.1 Integration Testing

Integration testing involves validating the seamless interaction and functionality of various system components. This testing ensures that the Google Gemini LLM engine integrates effectively with other modules such as data processing, response generation, reporting, and task generation. By simulating real-world scenarios and testing the interactions between these components, integration testing verifies that the system operates as intended, delivering accurate sentiment analysis, appropriate responses, insightful reports, and relevant daily tasks for mental well-being improvement.

CHAPTER 8

8.1 PERFORMANCE ANALYSIS

The performance analysis involves evaluating the system’s efficiency, accuracy, and responsiveness. This includes assessing the time taken for sentiment analysis, from receiving the user’s input to generating a response, to ensure real-time processing. The system’s ability to handle large volumes of data and maintain consistent performance under varying workloads is also a key aspect of performance analysis. Furthermore, analyzing the accuracy of sentiment classification, comparing the system’s predictions with actual user emotions, helps validate its effectiveness. Performance metrics such as throughput, latency, and resource utilization are monitored to identify bottlenecks and optimize system performance. Regular performance testing and monitoring ensure that the sentiment analysis system delivers reliable and timely results, meeting the requirements of healthcare applications for prompt and accurate mental state assessment.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 Conclusion

Advanced AI Technology: Utilizing Google Gemini LLM's advanced natural language processing capabilities ensures accurate emotional state analysis.

Personalized Recommendations: Providing tailored suggestions and positive messages based on the user's emotional state enhances user engagement and mental well-being.

Efficiency and Scalability: The system's efficiency in processing large volumes of data and scalability for real-time analysis make it suitable for widespread adoption in healthcare settings.

Comprehensive Reporting: Generating detailed reports on emotional trends allows for informed decision-making and continuous improvement of mental health interventions.

Active Promotion of Mental Wellness: The integration of daily task generation promotes a holistic approach to mental wellness, empowering users to take proactive steps towards improving their mental health.

9.2 Future Enhancement

In the future, the "Sentimental Analysis Using Google Gemini LLM" project can be enhanced in several ways to further improve its effectiveness and user experience. One area of enhancement could be the integration of real-time data streams, allowing the system to analyze and respond to users' emotions instantaneously. This would enable timely interventions and more proactive support for mental well-being.

Another potential enhancement is the incorporation of personalized machine learning models within Google Gemini LLM. These models can be trained on individual user data to better understand their unique emotional patterns and preferences. This personalized approach can lead to more accurate analysis and tailored recommendations for each user, enhancing the overall impact of the system.

Furthermore, enhancing the reporting and analytics capabilities of the system can provide deeper insights into emotional trends and patterns over time.

Additionally, expanding the language support and cultural sensitivity of Google Gemini LLM can make the system more inclusive and effective in diverse settings. This includes improving its ability to understand regional dialects, idiomatic expressions, and cultural nuances, ensuring accurate and culturally sensitive analysis and response generation.

CHAPTER 10

APPENDIX

10.1 Sample Coding

home.html

```
<head>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.google apis.com/css2?family=Pacifico&display=swap" rel="stylesheet">

<style> body{
        display: grid; place-content:center; background-
        image:
 url('https://www.parents.com/thmb/FmihFthdF62Ou0EoUlRI7Ikeb4k=/1500x0/filters:n
 o_upscale():max_bytes(150000):strip_icc()/GettyImages-884378360-2000-
 6d8b0cd7ce0c4daf96ed15c80e3bfc16.jpg');
   background-size:100%;
   background-position: center;
   font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida
   Sans Unicode', Geneva, Verdana, sans-serif; }
           .container{
            display:grid;
     place-content:                    center;
     padding:100px; background: linear-
     gradient(to    right,    rgb(228,    228,
     101),rgb(97,   200,   216));   border-
     radius:30px;
            }

    #enter{     width:500px;     height:
        40px;         border-radius:20px;
        padding-left:   20px;   padding-
    right:20px; border:none;
        }

  #btn{   color:black;   border:   2px   solid   red;
    background-color: transparent; height: 40px;
    border-radius:10px; transition:0.25s ease-in;
         }

  #btn:hover{    background-color:    red;
    transition:0.25s                    ease-in;
    color:white; cursor:pointer;
            }
  #result{     max-width:     400px;     height:     auto;
    padding:20px;   background-color:   rgb(48,   48,
    48); color:white; border-radius: 20px;
             }

  h2{   text-align:   center;   font-size:   40px;   font-family:   “Pacifico”,   cursive;
    background: linear-gradient(to right, red, rgb(99, 69, 232));
                                                   -
    webkit-background-clip:  text;  background-
    clip: text; color:   transparent;
             }


     </style>
             </head>

             <body>

                    <div class=”container”>
                     <h2>SentiVerse</h2>
                     <div id=”response”>
              <form>


     <center><input      id=”enter”      placeholder=”Enter      your      statement”      name=”length”
type=”text”></center>
```

```
            <br>
            <br>

        <center><inputtype=”submit”id=”btn”onclick=”document.getElementById(‘result’).style.di
splay = block” value=”Analyse mental state”></center>
        </form>
            </div>
            <br>

          <center>
           <div id=”result”> {{password}} </div>
          </center>

          <br>

          <center><button id=”btn” class=”speak”>Speak</button></center>

            <br>
          <form action=”{% url ‘password’ %}”>
<center><button id=”btn” class=”speak”>Get today’s task</button></center>
            </form>
            <br>

         <form action=”{% url ‘about’ %}”>
         <center><button id=”btn” class=”speak”>Get report</button></center>
         </form>
          </div>
        <scriptsrc=”https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js”
        integrity=”sha384-
        geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Siy3Te4Bkz”
        Cross origin=”anonymous”></script>

         <script>
              // Check if browser supports speech synthesis
        if(‘speech Synthesis’ in window) {
        const speak Button = document .query Selector(‘.speak’); const
        content = document .get Element By Id(‘result’);
// Event listener for button click
 Speak Button .add Event Listener('click', () => {

 // Create a new speech synthesis object const speech Synth =
 window. Speech Synthesis;

  // Get the text content of the div const text To Read = content
 .text Content;

// Create a new Speech Synthesis Utterance object const utterance = new Speech
Synthesis Utterance (text To Read);

const voices = speech Synthesis. Get Voices();
 const zira Voice = voices. find(voice => voice.name === 'Microsoft David Desktop –
 English    (United States)');

// Set the Zira voice utterance.voice=
ziraVoice;

//   Function   to   speak   after   a   delay   function
speakAfterDelay() {
 // Speak the text speechSynth.speak(utterance);
  }

  // Speak the text speechSynth.speak(utterance);
  });
       } else { alert('Your browser does not support speech    synthesis.');
            }
 </script>
 </body>
```

about.html

```
<head>
<link rel=”preconnect” href=”https://fonts.googleapis.com”>
<link rel=”preconnect” href=”https://fonts.gstatic.com” crossorigin>
<linkhref=”https://fonts.googleapis.com/css2?family=Pacifico&display=swap” rel=”stylesheet”>
<style> body{
      display:         grid;         place-content:center;
      background-image:
url(‘https://www.parents.com/thmb/FmihFthdF62Ou0EoUlRI7Ikeb4k=/1500x0/filters:n
o_upscale():max_bytes(150000):strip_icc()/GettyImages-884378360-2000-
6d8b0cd7ce0c4daf96ed15c80e3bfc16.jpg’);
      background-size: 100%;
      background-position: center;
      font-family: ‘Lucida Sans’, ‘Lucida Sans Regular’, ‘Lucida Grande’, ‘Lucida
  Sans Unicode’, Geneva, Verdana, sans-serif; }
   .container{ display:grid; place-content: center; padding:100px; background: linear-gradient(to
      right, rgb(228, 228, 101),rgb(97, 200, 216)); border-radius:30px;
    }


        #enter{
      width:500px;   height:   40px;
      border-radius:20px; padding-
      left:        20px;        padding-
      right:20px; border:none;
         }


   #btn{
      color:black;
      border: 2px solid red;
      background-color:transparent;      height:
      40px;
      border-radius:10px;         transition:0.25s
      ease-in;
   }

#btn:hover{
     background-color:red;
     transition:0.25s
     ease-in;
     color:white;
     cursor:pointer;

}

#result{
    max-width:   400px;   text-
     wrap:  wrap;  height:  auto;
     padding:20px;
     background-color:   rgb(48,
     48, 48); color:white;
    border-radius: 20px;
                                              }


     h2{  text-align:  center;  font-size:  40px;  font-family:  "Pacifico",  cursive;
     background: linear-gradient(to right, red, rgb(99, 69, 232));
                                                  -
     webkit-background-clip:                     text;
     background-clip: text; color: transparent;
            }

         </style>

          </head>
           <body>
                      <div class="container">
                               <h2>SentiVerse</h2>

                                             <div id="response">
           <form>

               <center><h2>Overall report on your past sentiments</h2></center>
               <center> <div id="result"> {{password}} </div> </center>
```

```
                    <br>

               <center><button id=”btn” class=”speak”>Speak</button></center>
                </div>
    <script                    src=”https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js”
integrity=”sha384-
      geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Siy3Te4Bkz”
      crossorigin=”anonymous”></script>

          <script>
          // Check if browser supports speech synthesis if
  (‘speechSynthesis’ in window) {
  const speakButton = document.querySelector(‘.speak’); const content =
  document.getElementById(‘result’);

  // Event listener for button click
  speakButton.addEventListener(‘click’, () => {
   // Create a new speech synthesis object const speechSynth =
    window.speechSynthesis;
   // Get the text content of the div const textToRead =
    content.textContent;

    // Create a new SpeechSynthesisUtterance object const utterance = new
    SpeechSynthesisUtterance(textToRead);

          const voices = speechSynthesis.getVoices();
    const ziraVoice = voices.find(voice => voice.name === ‘Microsoft David Desktop –
    English (United States)’);

    // Set the Zira voice utterance.voice =
    ziraVoice;
// Function to speak after a delay function
speakAfterDelay() {
  // Speak the text speechSynth.speak(utterance);
                                              }

// Speak the text speechSynth.speak(utterance);
      });
                } else { alert('Your browser does not support speech
            synthesis.');
         }
      </script>
      </body>
```

feedback.html

```
<head>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
<style> body{
            display: grid; place-content:center;
            background-image:
url('https://www.parents.com/thmb/FmihFthdF62Ou0EoUlRI7Ikeb4k=/1500x0/filters:n
o_upscale():max_bytes(150000):strip_icc()/GettyImages-884378360-2000-
6d8b0cd7ce0c4daf96ed15c80e3bfc16.jpg');
            background-size: 100%; background-
            position: center;
            font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida
  Sans Unicode', Geneva, Verdana, sans-serif; }
      .container{ display:grid; place-content: center; padding:100px; background: linear-
            gradient(to right, rgb(228, 228, 101),rgb(97, 200, 216)); border-radius:30px;
  }
      #enter{      width:500px;
            height: 40px; border-
            radius:20px; padding-
```

```
    left:   20px;   padding-
    right:20px; border:none;
}

  #btn{ color:black; border: 2px solid red;
    background-color:        transparent;
    height:   40px;   border-radius:10px;
    transition:0.25s ease-in;
  }

  #btn:hover{   background-color:
    red;  transition:0.25s  ease-in;
    color:white; cursor:pointer;
  }

  #result{          max-width:
    400px;          text-wrap:
    wrap;    height:    auto;
    padding:20px;
    background-color:
    rgb(48,      48,      48);
    color:white;       border-
    radius: 20px;
   }

  h2{ text-align: center; font-size: 40px; font-family: “Pacifico”, cursive;
    background: linear-gradient(to right, red, rgb(99, 69, 232));
                                                     -
    webkit-background-clip:           text;
    background-clip:      text;       color:
    transparent;
   }

  </style>
```

```
     </head>

          <body>

                    <div class="container">

                     <h2>SentiVerse</h2>

                       <div id="response">
          <form>

                    <center><h2>TODAY'S TASK</h2></center>
                     <center><div id="result"> {{password}} </div> </center>

             <br>


          <center><button id="btn" class="speak">Speak</button></center>

             </div>
                <script>
   src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
   integrity="sha384-
   geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Siy3Te4Bkz"
    crossorigin="anonymous"></script>

  <script>
          // Check if browser supports speech synthesis if
   ('speechSynthesis' in window) {
 const speakButton = document.querySelector('.speak'); const content =
document.getElementById('result');


// Event listener for button click
speakButton.addEventListener('click', () => {

// Create a new speech synthesis object const speechSynth
  = window.speechSynthesis;

 // Get the text content of the div const textToRead =
  content.textContent;


  // Create a new SpeechSynthesisUtterance object const utterance = new
  SpeechSynthesisUtterance(textToRead);
const voices = speechSynthesis.getVoices();
const ziraVoice = voices.find(voice => voice.name === ‘Microsoft David Desktop –
English (United States)’);

// Set the Zira voice utterance.voice =
ziraVoice;

// Function to speak after a delay function
speakAfterDelay() {
  // Speak the text speechSynth.speak(utterance);
  }
 // Speak the text speechSynth.speak(utterance);
 });
     } else { alert(‘Your browser does not support speech
   synthesis.’);
    }
</script>
</body>
```

views.py

```
from 48jango.shortcuts import render from
48jango.http import HttpResponse import
random; import openai import os import
textwrap import google.generativeai as genai

GOOGLE_API_KEY=’’
genai.configure(api_key=GOOGLE_API_KEY) model =
genai.GenerativeModel(‘gemini-pro’)
                                    x = []
                                    sentiment = “”
```

```
def to_markdown(text):
def to_markdown(text):
  text   =   text.replace('•',   '   *')
return textwrap.indent(text, '', predicate=lambda _: True)

def home(request):

  global x
  length=request.GET.get('length') instructions = f""" You are
given with a role of a psychiatrist
You will be provided with a sentence delimited within triple backticks or an paragraph and you should analyse the sentiment of the paragraph and tell the emotions of the writer of the paragraph.you should only specify the exact emotion of the person whether they are happy,sad or having mixed emotions like you seem to sad. If it is sad or mixed recommend some suggestions which are tasks to improve their mental state or if it is happy show some uplifting message like "You are stronger than you think, and you have the power to overcome any challenge that comes your way." "Believe in yourself, trust your journey, and keep moving forward one step at a time." "You are capable of achieving great things. Keep working hard and stay focused on your goals." Like that but not these alone along with it. -the result should be within 50 words and the suggestion should be a task like Mindfulness meditation ,Physical activity,Social support, Self-care, Gratitude practice, Limiting negative influences, Setting realistic goals, Deep breathing exercises but not these things alone. If the sentence is about any context other than sentiment analysis, then you should return Iam designed for sentiment analysis only. This is very important. For example if the sentence is about coding a software, then tell Iam designed for sentiment analysis only.
-sentence = ```{length}```

 -You should not assist them in any other context other than sentiment analysis. You should say   Iam
 designed for sentiment analysis only if the context is not about sentiment analysis.
 -you should not use the word writer instead you may address them as you """

  # thepassword = get_completion(instructions)

 response   =   model.generate_content(instructions)   result   =
  to_markdown(response.text)
```

```
    chatbot_response = result sentiment =result
    x.append(chatbot_response)    return    render(request,    ‘generator/home.html’,    {‘password’:
    chatbot_response })

    def password(request):
    global       x       global
    sentiment
    length=request.GET.get(‘length’) instructions = f”’” You are given with a role of a psychiatrist. You
    will be provided
  with a sentence delimited within triple backticks or an paragraph and you should analyse the sentiment of
  the paragraph and tell the emotions of the person who said the paragraph.

    - If the sentence is about any context other than sentiment analysis, then you should return I dont know.
  This is very important. For example if the sentence is about coding a software, then tell I dont know.

 -sentence = ```{length}```

    -You should not assist them in any other context other than sentiment analysis. You should say I dont
  know if the context is not about sentiment analysis.

                                        “’”’

   # thepassword = get_completion(instructions)

 response = model.generate_content(instructions) result =
to_markdown(response.text)
    chatbot_response = result sentiment = result
    x.append(chatbot_response)

    return render(request, ‘generator/home.html’, {‘password’: chatbot_response })

       def give_feedback(request):
    global       x       global
    sentiment
```

instructions = f””” You are given with a role of a psychiatrist.

- You have to generate an task and should give that as today’s task to the user to improve their mental state by doing that task for that day alone and should not not repeat the task.

-paragraph = ```{sentiment}``` -please dont give the array as the response. -do not use bold texts.

-Use only 51jango51. 51jango51e lines with line breaks. This is important.

Give a single paragraph answer within 100 words. This is very very important.

“”” response = model.generate_content(instructions) result =

to_markdown(response.text) chatbot_response = result

return render(request, ‘generator/feedback.html’, {‘password’: chatbot_response })

def generate_report(request):
global x global
sentiment

instructions = f””” You are given with a role of a psychiatrist. You will be provided with a array delimited within triple backticks. The array consists of the past sentiment analysis of the user.

- You have to give the overall report to the user based on the past sentiments in the array. Consider the array is sorted that is first element in the array is the first sentiment, and the last is the latest sentiment.

-paragraph = ```{x}```

-please dont give the array as the response.

-do not use bold texts.

-Use only 52jango52. 52jango52e lines with line breaks. This is important.

Give a single paragraph answer within 250 words. This is very very important.

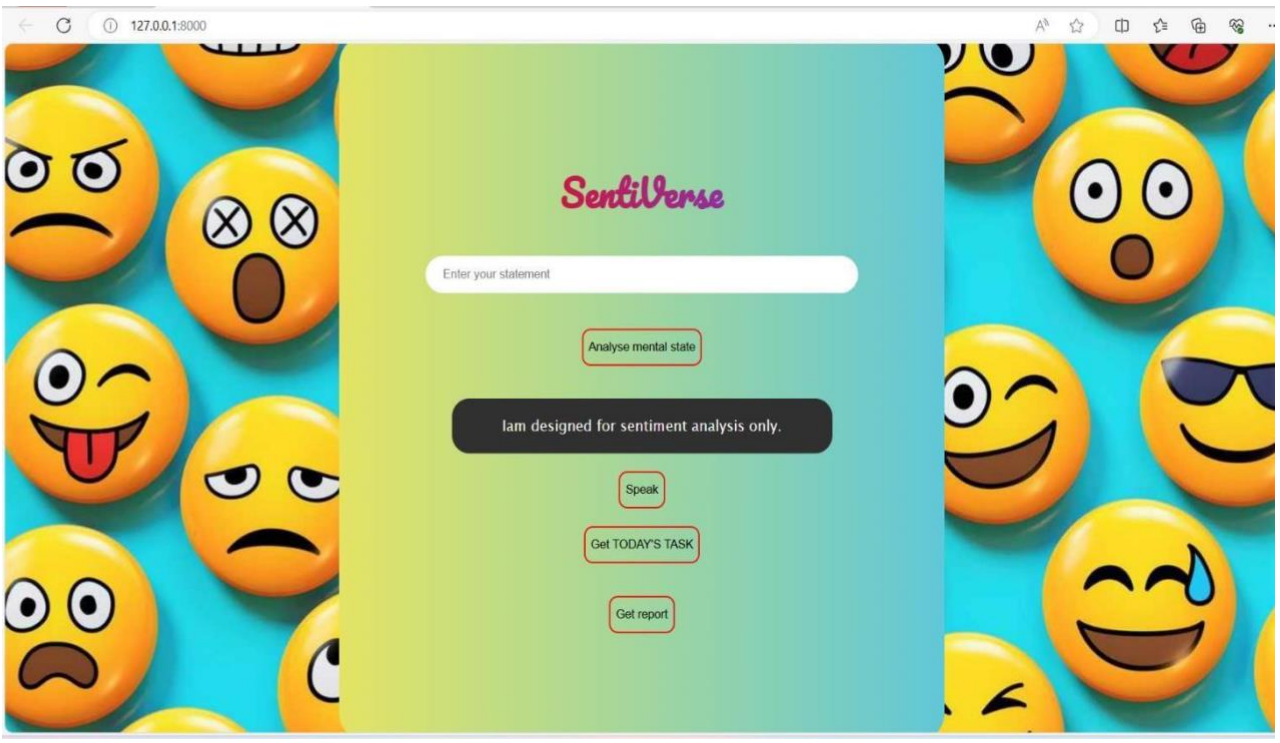
""""

response = model.generate_content(instructions) result = to_markdown(response.text) chatbot_response = result return render(request, 'generator/about.html', {'password': chatbot_response })

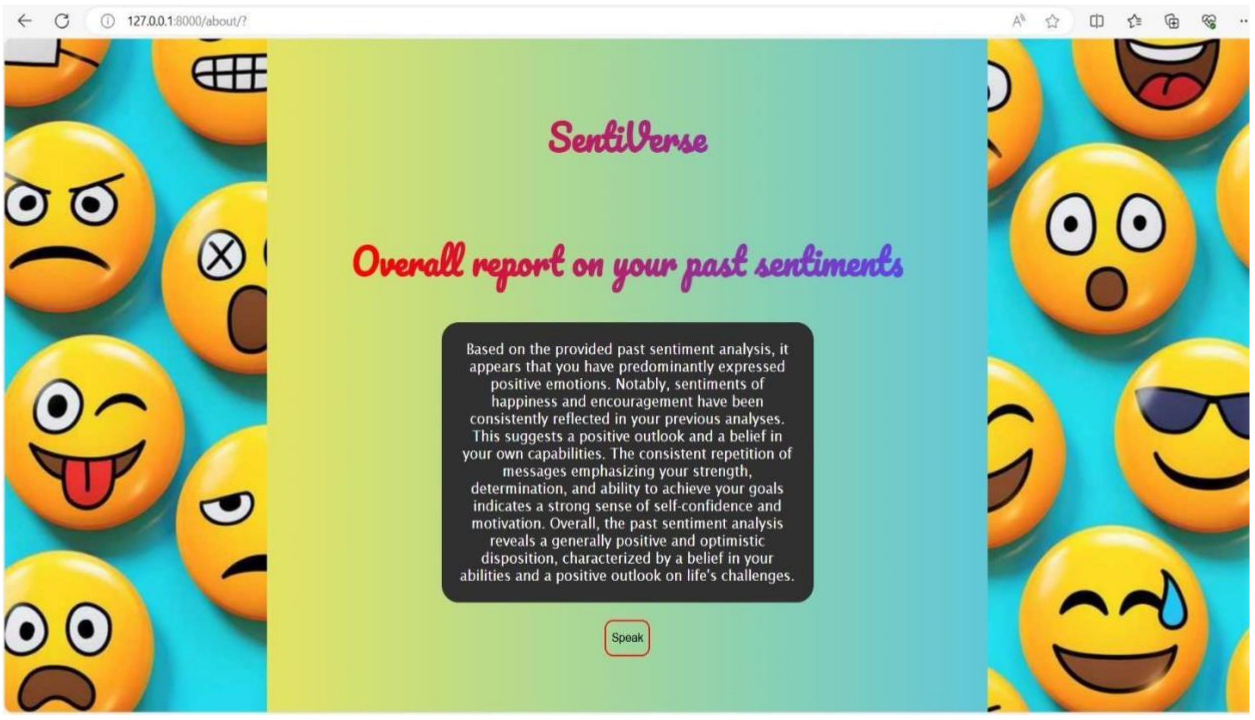
```
  def about(request):
    return render(request,'generator/about.html')

# def get_completion(prompt, model="gpt-3.5-turbo"):
# Andrew mentioned that the prompt/ completion paradigm is preferable for this class
   #          messages = [{"role": "user", "content": prompt}]
   #          response = openai.ChatCompletion.create(
   #          model=model,
   #           messages=messages,
   #               temperature=0, # this is the degree of randomness of the model's output
   #     )
   #          return response.choices[0].message["content"]
```

10.2 Screenshots

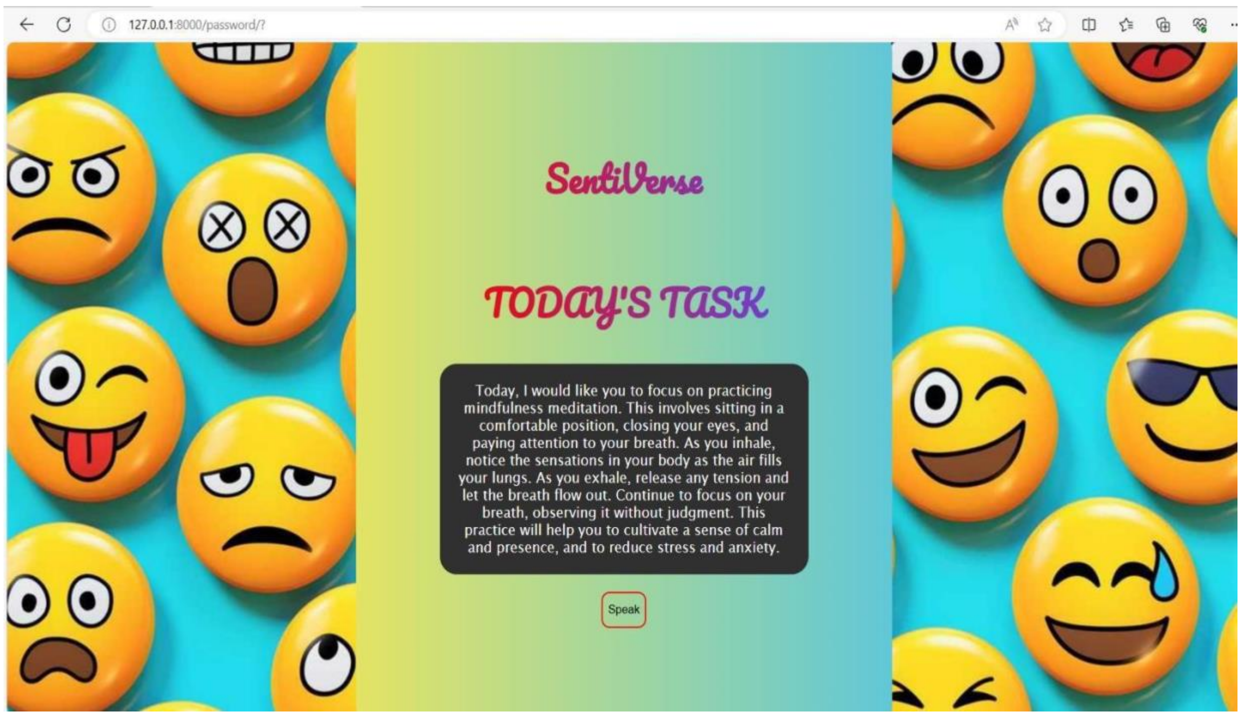


Iam designed for sentiment analysis only.



Overall report on your past sentiments

Based on the provided past sentiment analysis, it appears that you have predominantly expressed positive emotions. Notably, sentiments of happiness and encouragement have been consistently reflected in your previous analyses. This suggests a positive outlook and a belief in your own capabilities. The consistent repetition of messages emphasizing your strength, determination, and ability to achieve your goals indicates a strong sense of self-confidence and motivation. Overall, the past sentiment analysis reveals a generally positive and optimistic disposition, characterized by a belief in your abilities and a positive outlook on life's challenges.



Today, I would like you to focus on practicing mindfulness meditation. This involves sitting in a comfortable position, closing your eyes, and paying attention to your breath. As you inhale, notice the sensations in your body as the air fills your lungs. As you exhale, release any tension and let the breath flow out. Continue to focus on your breath, observing it without judgment. This practice will help you to cultivate a sense of calm and presence, and to reduce stress and anxiety.

10.3 References

[1] Chen, M., Mall, S. "Sentimental Analysis with Amazon Review Data."

[2] Sadhasivam, J., Kalivaradhan, R. "Sentiment Analysis of Amazon Products Using Ensemble Machine Learning Algorithm."

[3] Shah, S., Kumar, K., Sarvananguru, R.K. "Sentimental Analysis of Twitter Data using Classifier Algorithms."

[4] Ganga, S., Solanki, A. "A Sentimental Analysis System Using Zero- Shot Machine Learning Technique."