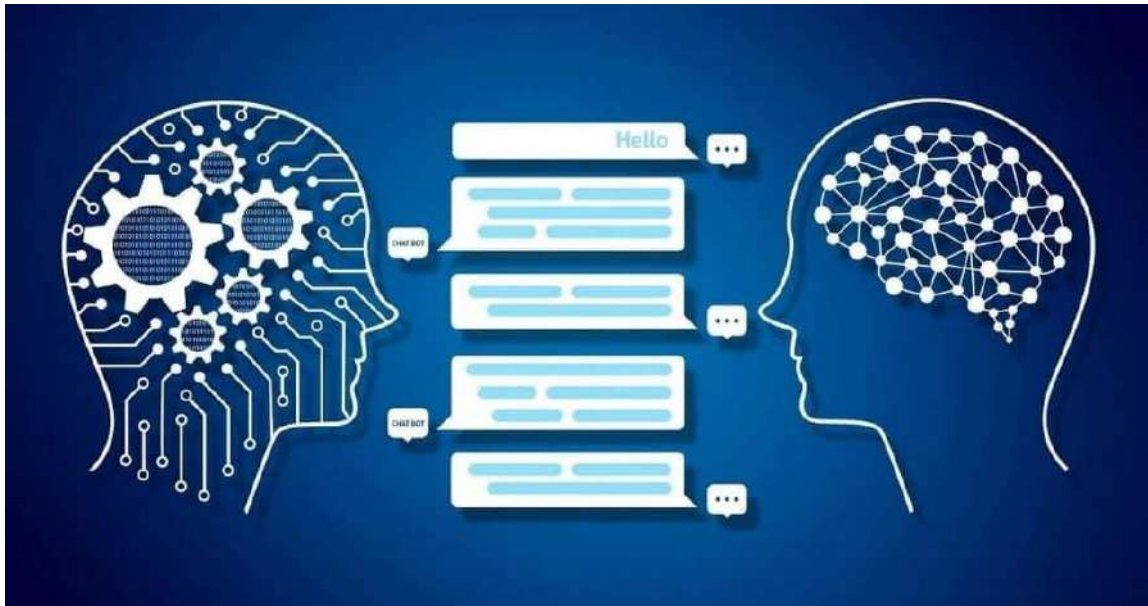


Department of CSE AIML

Beyond Words: Student Innovations in NLP

**(A technical magazine comprising collection of Student Mini
projects)**



Prepared By

BE CSE AIML Students

(AY 2025-26)

Under Guidance of

Prof. Amey V. Phanse

In the rapidly evolving era of Artificial Intelligence, the ability of machines to understand, interpret, and generate human language has become one of the most transformative technological advancements. Natural Language Processing (NLP), a core domain within AI and Machine Learning, plays a pivotal role in bridging the gap between human communication and computational intelligence. With its widespread applications in healthcare, education, business analytics, social media, governance, and automation, NLP has become an essential competency for future AI engineers.

This magazine is a compilation of mini-projects designed and developed by the BE CSE (AIML) students, as a part of their curriculum-based experiential learning component. The objective of this initiative was not only to enable students to apply the theoretical foundations of NLP but also to encourage them to design, develop, and demonstrate working models that solve meaningful real-world problems. Through these projects, students learned how language data can be pre-processed, modeled, and intelligently interpreted using modern NLP techniques, tools, and libraries such as NLTK, spaCy, scikit-learn, and neural language models.

Each mini-project showcased in this magazine reflects a unique idea, practical implementation, and analytical insight into NLP applications. The projects range across diverse areas such as sentiment analysis, text summarization, fake news detection, chatbots, emotion recognition, topic modeling, machine translation, speech-to-text systems, and more—highlighting the creativity, curiosity, and technical capabilities of our students. The magazine also documents the methodology, implementation steps, challenges faced, and outcomes achieved, giving readers a holistic view of the project development journey.

This student-driven publication stands as a testament to collaborative learning, innovation, and problem-solving skills nurtured throughout the course. It emphasizes the importance of applying domain-specific knowledge, using modern AI tools, and adopting ethical considerations while building intelligent language systems.

We hope this magazine inspires fellow learners, educators, and AI enthusiasts to explore NLP beyond the classroom and encourages them to contribute to the future of human-centric AI solutions. The work presented herein reflects the dedication and passion of our budding AI engineers, who are ready to shape the next generation of intelligent technologies

- Team BE CSE AIML

Publication Date – 08/11/2025

BE MINI PROJECT REPORT

On

AI Assistant for Event Management System

By

Atharv Vidhyadhar Goythale(08)

Siddhi Santosh Palshetkar(28)

Rutuja Ganesh Salunkhe(42)

1. Introduction

Organizing events involves multiple tasks such as sending invitations, tracking registrations, providing real-time updates, answering attendee queries, and ensuring smooth communication between organizers and participants. Traditionally, these tasks are handled manually or through separate systems, which can lead to delays, miscommunication, and poor user experience.

With the growing adoption of conversational interfaces, AI-powered chatbots are increasingly being used to automate event management tasks. An event management chatbot can assist organizers and attendees through 24/7 interactive conversations, reducing human effort and improving efficiency.

This project aims to design and develop a smart event management chatbot that can provide real-time information, handle event-related queries, automate registration, send reminders, and personalize user interaction. The chatbot integrates natural language processing (NLP) techniques and machine learning to understand user inputs and respond contextually, making event management more accessible and efficient.

2. Problem Statement

Event management often relies on manual communication through emails, phone calls, or static web pages. This leads to several issues:

- Attendees struggle to find relevant information quickly (e.g., event schedule, venue, speaker details).
- Organizers spend a lot of time responding to repetitive queries.
- Manual registration and reminder systems are error-prone and time-consuming.

For example:

- “What time does the keynote session start?” → Delayed response from support staff.
- “How can I register for the event?” → Requires manual navigation through multiple webpages.

Objective: Develop an intelligent chatbot that provides instant responses, automates routine tasks, and personalizes user interaction, making event management more efficient and engaging.

3. Objectives

- Design a chatbot interface that interacts with users through natural language.
- Automate common event management functions such as registration, FAQs, and reminders.

- Integrate NLP to understand user queries and provide accurate, contextual answers.
- Support multiple event types (seminars, cultural fests, hackathons, conferences).
- Provide a scalable and user-friendly solution for both organizers and participants.

4. Scope and Applications

Domain: Artificial Intelligence, Natural Language Processing, Event

Management Applications:

- Educational Institutions: Automating registrations and schedules for workshops, seminars, cultural events, and hackathons.
- Corporate Events: Managing conferences, webinars, and meetings.
- Public Events: Handling large-scale queries and announcements efficiently.
- Personal Use: Event planning for private functions such as weddings or parties.

The chatbot can be deployed on websites, mobile apps, or messaging platforms (e.g., WhatsApp, Telegram, Slack). Future extensions include multilingual support and integration with calendar apps.

5. Proposed Methodology

Step 1: Requirement Analysis

Identify event-related functionalities such as registration, FAQ answering, announcements, scheduling, and feedback collection.

Step 2: Data Preparation

- Collect typical event FAQs and conversation patterns.
 - Prepare intents and entities (e.g., “register,” “event date,” “location,” “speaker info”).
- Define chatbot flow using structured conversation data.

Step 3: NLP & Intent Classification

- Use preprocessing (tokenization, lowercasing, stop-word removal).
- Train an intent classification model (e.g., Logistic Regression / SVM) using TF-IDF vectors.
- Use simple NER (Named Entity Recognition) for extracting names, dates, and event details.

Step 4: Chatbot Development

- Build a rule-based + ML hybrid chatbot using frameworks such as Rasa or Flask + Python.

- Implement dialogue flow for registration, FAQ, and reminders.
- Store user data (name, email, event type) in a database.

Step 5: Integration and Deployment

- Integrate chatbot with a website or messaging platform.
- Enable event updates through scheduled messages.
- Allow real-time query handling and registration.

Step 6: Testing and Feedback

- Test chatbot with sample events and user groups.
- Collect feedback to improve accuracy and conversational flow.

6. Expected Outcomes and Advantages

Expected Outcomes:

- An intelligent chatbot capable of understanding and responding to event-related queries.
- Automated registration and reminder functionalities.
- Easy integration with websites or messaging apps.
- Improved communication between organizers and attendees.

Advantages:

- 24/7 Availability: Users get instant responses anytime.
- Reduced Manual Effort: Common queries and registrations are handled automatically.
- Improved User Experience: Quick, personalized answers enhance engagement.
- Scalability: Can handle thousands of users simultaneously without human intervention.
- Cost-Effective: Reduces the need for large support teams.

7. Future Enhancements

- Integrate advanced transformer models (e.g., BERT, Dialogflow CX) for better understanding of complex queries.
- Add multilingual support for wider accessibility.
- Include voice assistant capabilities for hands-free interaction.
- Enable payment integration for paid events.

- Extend chatbot functions to analyze user feedback for event improvement.
- Deploy on multiple platforms including WhatsApp, Facebook Messenger, and custom mobile apps.

8. Conclusion

This project demonstrates that an AI-powered chatbot can effectively automate and streamline event management tasks. By combining NLP-based intent classification with rule based conversation flows, the chatbot provides instant and accurate responses to user queries, handles registrations, and manages reminders. It reduces manual workload, improves participant engagement, and ensures smooth event operations.

The system provides a scalable, lightweight, and cost-effective solution for institutions, companies, and individuals. With future enhancements like transformer integration and multilingual capabilities, the chatbot can evolve into a robust platform for managing events of any scale.

A Mini Project Report on

BookAI

Group Member : Susmit Mahesh Bhogate (Roll No.03)

Rushikesh Vilas Paranjape (Roll No.31)

Suyash Suhas Sakpal (Roll No.41)

1. Introduction

Nowadays, people face a huge amount of text every day – news, research papers, reports, articles, and emails. Reading everything takes a lot of time and effort.

To solve this problem, we developed **BookAI**, a **web-based Natural Language Processing (NLP)** application that takes a large text and creates a short and meaningful summary.

Along with summarization, **BookAI** also supports a **Question-Answer System** that allows users to upload a document and then ask specific questions related to its content. The system automatically analyzes the document and provides accurate answers using NLP and text similarity models.

The summarizer also provides extra features like:

- Finding important **keywords**
- **Sentiment analysis** (shows if the text is positive, negative, or neutral)
- **Named Entity Recognition (NER)** to identify people, places, and organizations
- **Automatic title generation**
- **Text-to-Speech (audio summary)**
- **Downloading the summary as PDF** or receiving it via **email**
- **Question Answering** – User can ask questions based on the uploaded document, and the system generates answers automatically

2. Objectives

The main goals of this project are:

1. To summarize long text into a short, clear version.
 2. To allow users to upload files in formats like `.txt`, `.pdf`, and `.docx`.
 3. To apply **NLP techniques** for keyword extraction, sentiment analysis, and NER.
 4. To provide **audio output** of the summary for better accessibility.
 5. To give an option to **download summary as PDF**.
 6. To **send the summary (PDF + audio)** to the user's email.
 7. To develop a **Question-Answer System** that enables users to ask text-related questions and receive context-based answers instantly.
-

3. Technology Used

Programming Language: Python

Framework: Flask (for web app development)

NLP Libraries:

- **spaCy** → Named Entity Recognition (NER), POS tagging
- **NLTK** → Tokenization, text preprocessing
- **Summa** → Summarization using the TextRank algorithm
- **Transformers (Hugging Face)** → For Question Answering using pre-trained models like BERT or DistilBERT
- **LangDetect** → For language detection

File Handling:

- **PyPDF2** → Extracts text from PDF
- **python-docx** → Extracts text from Word files
- **ReportLab** → Generates downloadable PDF summaries

Email Integration:

- **smtplib + EmailMessage** → Sends summaries and audio files to user email

4. Methodology (Working of the System)

1. **Text Input**
 - User can type or upload a document (PDF, DOCX, or TXT).
2. **Language Detection**
 - System detects the document's language (English, Marathi, etc.).
3. **Text Preprocessing**
 - Tokenization (splitting text into sentences and words)
 - Stopword removal, punctuation cleaning
 - Lemmatization (converting words to their base form)
4. **Summarization**
 - Uses **TextRank Algorithm** to generate extractive summaries.
 - User can select summary length (e.g., 20% of the original text).
5. **Additional Features**
 - **Sentiment Analysis** → Determines text polarity (Positive/Negative/Neutral)
 - **NER (Named Entity Recognition)** → Extracts people, organizations, and locations
 - **Keyword Extraction** → Highlights the most relevant words and phrases
 - **Automatic Title Generation** → Suggests a concise title based on context
6. **Question Answering System**
 - Users can input a **question** related to the uploaded text.
 - The system uses **pre-trained NLP models (e.g., BERT/DistilBERT)** to understand the context of the document.
 - It searches for the most relevant section of text and generates a **contextual answer**.
 - This allows users to interactively retrieve specific information from long documents.

7. Output Display

- Displays summary, keywords, entities, sentiment, and Q&A results on the website.
- Users can **download** the summary as PDF, **listen to** audio, or **receive results by email**.

5. Results

- The system efficiently reduces lengthy documents into short, meaningful summaries.
- The **Question-Answer module** provides quick answers to user queries based on the content of the uploaded document.
- Users can view summaries and Q&A results online, download them as PDFs, or receive them via email.
- The inclusion of keywords, entities, and sentiment offers deeper understanding.
- The **audio summary** supports visually impaired users and those who prefer listening over reading.

6. Applications

- **Education** → Summarizing and answering questions from textbooks, lecture notes, or research papers.
- **Business** → Summarizing corporate reports, policies, and answering queries from large documents.
- **Healthcare** → Quickly extracting and querying information from patient records or research data.
- **News & Media** → Generating short news summaries and answering topic-specific questions.
- **Accessibility** → Providing text-to-speech summaries and interactive Q&A for visually impaired users.

7. Conclusion

The **BookAI** project demonstrates how **Natural Language Processing (NLP)** and **automation** can simplify reading and comprehension.

The newly added **Question-Answer System** enhances the tool's intelligence, enabling users not just to summarize text but also to **interactively ask questions and receive context-based answers**.

This makes **BookAI** highly useful for:

- **Students** – to summarize and query academic material
- **Researchers** – to extract specific insights from long papers
- **Professionals** – to understand lengthy business reports quickly

A Mini Project Report on “College Information Chatbot Using NLP”

Group Members: Srushti Dinesh Redij (39)

Tanvi Rajesh Sagvekar (40)

INTRODUCTION

The project “**College Information Chatbot using Natural Language Processing (NLP)**” demonstrates how Artificial Intelligence can make college communication faster, smarter, and more interactive. Many colleges receive hundreds of repetitive queries daily from students, parents, and visitors about admissions, courses, faculty, fees, and events. Handling all of these manually causes delays and extra work for staff. To overcome this problem, the team designed an intelligent chatbot that automatically understands user questions written in natural language and gives accurate, real-time answers.

SYSTEM OVERVIEW

The chatbot acts as a **virtual assistant** that can be integrated into a college website or student portal. Users simply type questions such as “*What are the courses offered by the Computer Department?*” or “*Who is the principal of the college?*”, and the chatbot instantly provides the correct information.

It uses **NLP techniques** like tokenization, stemming, lemmatization, intent recognition, and entity extraction to interpret the meaning of user inputs. These methods allow the system to understand human language and retrieve the most relevant answer from a structured **database** containing details about courses, departments, faculty, and college events.

METHODOLOGY

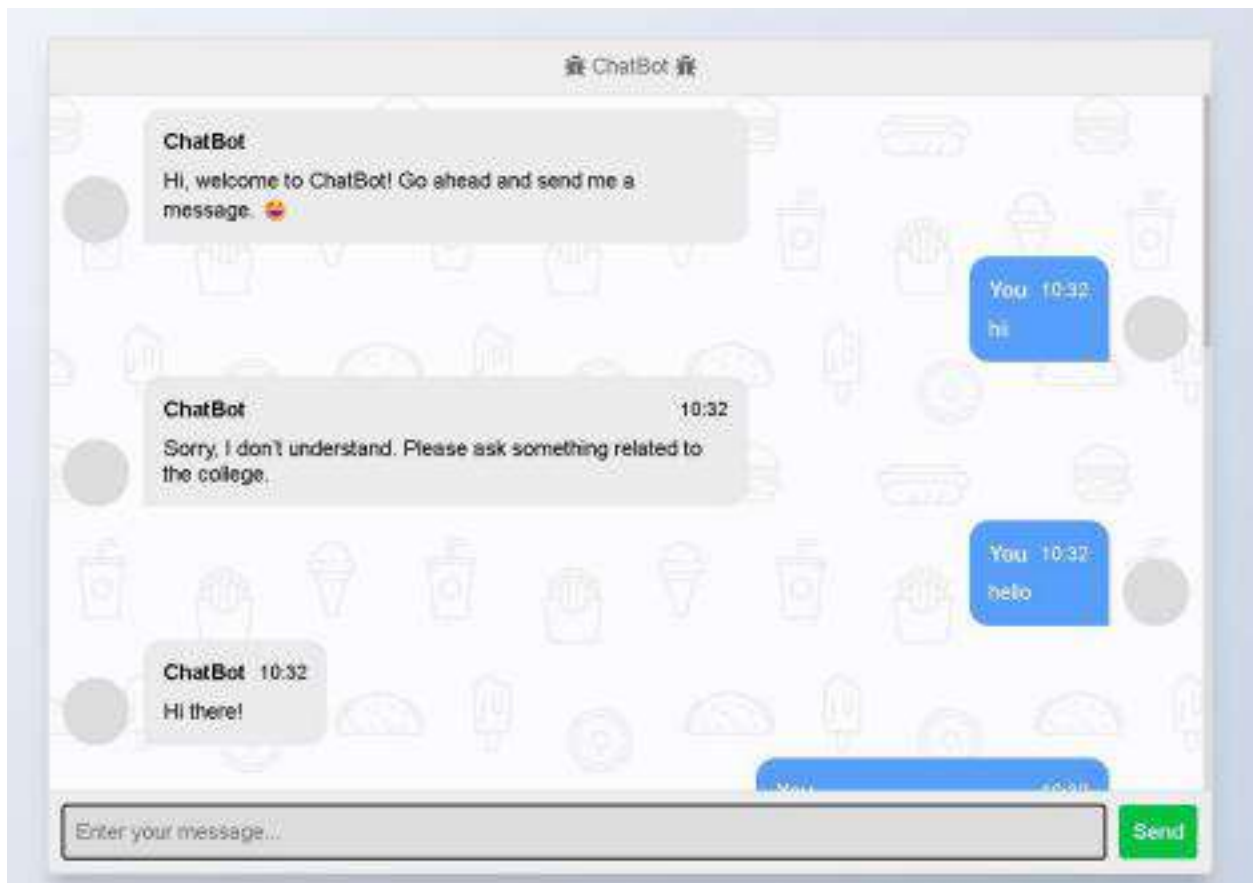
The development process was divided into several stages:

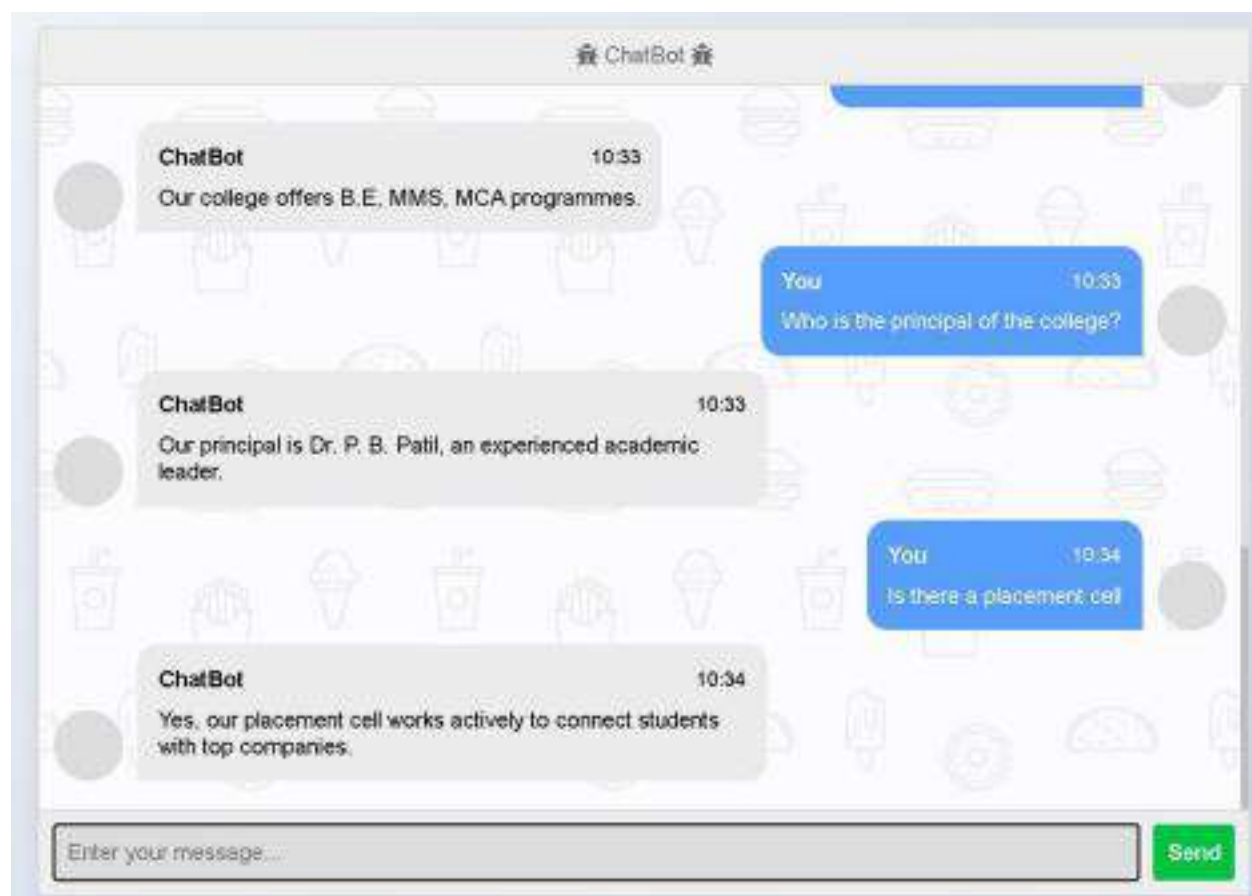
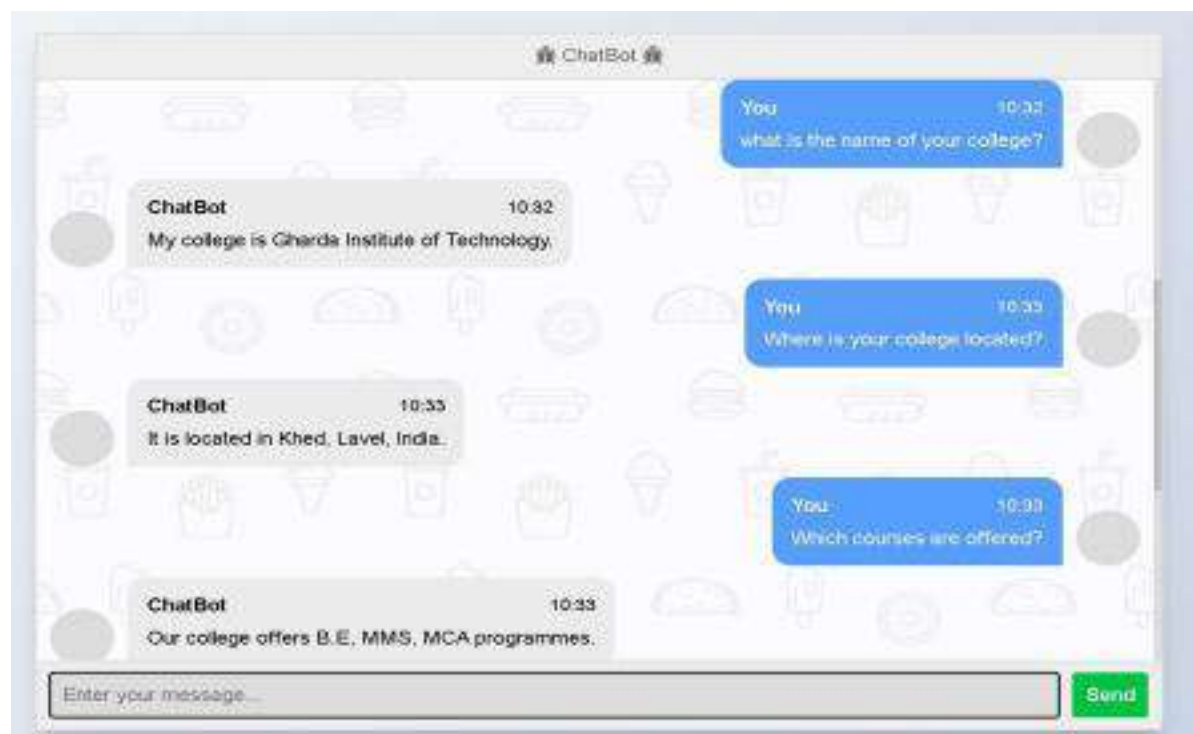
1. **Requirement Analysis** – Identify what kind of queries users usually ask and the functional needs of the chatbot.
2. **Data Collection** – Gather all college information such as course lists, faculty names, departments, and admission procedures and store them in a **JSON or SQLite database**.
3. **NLP Pre-processing** – Use libraries like **NLTK** or **spaCy** to clean and prepare text (tokenization, stopword removal, lemmatization).
4. **Intent Classification & Entity Extraction** – Use simple machine-learning or rule-based logic to recognize the user’s intent (e.g., “admission info”, “faculty info”) and extract key entities like department names or years.
5. **Response Generation** – Match the recognized intent with stored data to produce a clear, conversational answer.

6. **Frontend Design** – Build a simple, attractive chat interface using **HTML, CSS, and JavaScript**.
7. **Backend Integration** – Connect the frontend, NLP engine, and database through a **Flask (Python) web framework**.
8. **Testing & Deployment** – Test multiple queries for accuracy and deploy the chatbot on the college website for real-time use.

PERFORMANCE RESULTS:

- Accuracy: **90–95%** for known queries
- Response Time: **< 2 seconds per query**
- User Satisfaction: **9/10** based on feedback





KEY FEATURES

- **24×7 Availability:** Accessible anytime through a web interface.
- **Fast and Reliable:** Provides instant and accurate responses.
- **User-Friendly:** Simple chat window for natural conversation.
- **Maintainable:** Database can be easily updated without changing code.
- **Scalable:** Future enhancements like voice commands or multi-language support can be added easily.

CONCLUSION AND FUTURE SCOPE

The project successfully achieved its objectives by demonstrating how **NLP and AI** can transform college information systems. The chatbot is capable of understanding user queries, providing quick answers, and maintaining a natural conversational flow.

It has proven to be **accurate, responsive, and easy to use**, making it a practical solution for educational institutions looking to modernize their communication systems.

For future work, the system can be enhanced with **voice-based interactions, multilingual support, and integration with the college's ERP or mobile application**. Such improvements will make the chatbot even more accessible and powerful, paving the way for a fully automated, AI-driven information system in educational environments.

A Mini Project Report on

Email Classification System

By:
Anjali Kisan Chavan
Varada Vikram Jadhav
Gandharva Sameep Latke

1. The Challenge: Information Overload and Delays

In modern organizations, the sheer volume of daily email communications leads to a critical bottleneck. Essential messages—such as server alerts, urgent support issues, or time-sensitive financial reports—are often buried among hundreds of routine updates and announcements.

The Problem: Manual sorting is inefficient and error-prone, resulting in:

- Delayed Response: Critical issues are not routed to the right team (e.g., IT vs. HR) fast enough.
- Wasted Time: Employees spend valuable time manually triaging and forwarding emails.

2. The Solution: Automated Triage with Simulated AI

Our solution is the Email Classification System Prototype. This web-based tool instantly reads the body of an email and automatically assigns it to the correct department or action channel.

The system uses a simple, responsive interface built with HTML and Tailwind CSS. Its core function relies on a Simulated NLP Model—a sophisticated JavaScript dictionary and scoring system—that acts as the AI's "brain."

How the System Works (The Three-Step Process)

1. Input: An employee copies and pastes the email content into the application.
2. Classification: The system scans the text for hundreds of predefined keywords across seven distinct categories(e.g., Finance, Alert, Support, HR).
3. Output & Certainty: The system immediately outputs two pieces of data:
 - Predicted Category: The most relevant action group (e.g., Alert/Urgent).

- Confidence Score: A percentage (e.g., 92%) indicating how certain the system is about its choice. This score helps the user understand if the email is perfectly clear (high score) or a complex, mixed message (moderate score).

3. Key Categories for Automated Routing

The prototype is designed to capture the most common organizational intents:

Category	Typical Content	Business Impact
Alert/Urgent	Server failures, security breaches, immediate action items.	Guarantees instant visibility for IT/Operations.
Finance/Billing	Invoices, budgets, tax deadlines, payment processing.	Ensures accounting deadlines are never missed.
Support/Inquiry	Bug reports, user issues, requests for technical help.	Automatically creates help desk tickets.
HR/Internal Memo	Policy changes, training announcements, benefits info.	Routes internal staff communications correctly.
Task/Project	Meeting requests, deadlines, status updates, deliverables.	Helps project managers prioritize workflow.

4. Results

Email Classification System

Prototype: Classify your email content instantly (Simulated NLP)

Paste Email Body Here:

Greetings from Amazon Web Services,

We are writing to notify you that your AWS Account has GST Invoice(s) available. They can be downloaded from the "Bills" section of the Billing & Cost Management console.

<https://console.aws.amazon.com/billing/home#/bills>

Thank you for using Amazon Web Services.

Sincerely,

Classify Email

Clear Text

Classification Result

Predicted Category:

Finance/Billing

Confidence Score:

100%

Simulated Logic (Keywords Found):

invoice (1), billing (2), account (1)

Email Classification System

Prototype: Classify your email content instantly (Simulated NLP)

Paste Email Body Here:

Hi team,

We have encountered a critical system failure in the authentication module, and an incident report has been logged. We need an immediate status update on the mitigation plan.

Could you please schedule a quick review meeting for the team ASAP to discuss this issue? We must ensure the fix doesn't impact the Q4 deliverable schedule.

Thanks,
Operations Lead

Classify Email

Clear Text

Classification Result

Predicted Category:

Alert/Urgent

Confidence Score:

40%

Simulated Logic (Keywords Found):

critical (1), immediate (1), failure (1), incident (1), meeting (1), status update (1), review (1), deliverable (1), issue (1), fix (1)

5. Conclusion

The AI Email Classification System Prototype successfully validated the concept of automated email triage. By converting hours of manual sorting into an instant, data-driven process, the system offers a clear path to improved efficiency and faster response times for critical communications.

To transition this into a full-scale solution, the logical next step is to replace the keyword simulation with a professional Machine Learning (ML) Model (like a BERT or SVM classifier) hosted on a dedicated backend server. This upgrade will move the system beyond keyword counting to deep semantic understanding, ensuring even greater accuracy and stability in a live environment.

A Mini Project on

Google Playstore Review Analyzer

Group Members :

Aditya Arun Mirgal (Roll no. 20)

Sanika Ravindra Pawar(Roll no.37)

Mahika Mahesh Padval (Roll no.27)

1. Introduction

The **Google Play Store Review Analyzer** is a Natural Language Processing (NLP) based project designed to extract valuable insights from user reviews of mobile applications available on the Google Play Store. With millions of reviews generated daily, manually analyzing user feedback becomes impractical. This project automates the process by applying data preprocessing and topic modeling techniques such as Latent Dirichlet Allocation (LDA) to identify common themes and user sentiments. The insights are further visualized through word clouds and graphs, enabling developers and analysts to better understand user experiences, identify areas for improvement, and enhance overall app quality.

2. Objectives

1. To analyze user reviews of Google Play Store applications using NLP techniques.
2. To perform data preprocessing such as text cleaning, tokenization, and stop-word removal.
3. To apply LDA topic modeling for extracting key themes from the reviews.
4. To generate word clouds and graphical visualizations that highlight key words and trends.
5. To assist app developers and businesses in making data-driven decisions for app

improvement.

3. Technologies Used

Category	Technology / Tool
Programming language	Python
Libraries & Frameworks	NLTK, spaCy, Gensim, Scikit-learn, Pandas, Numpy, Matplotlib, Seaborn, WordCloud
Environment	Google Colab
Dataset source	Google Play Store Dataset (Kaggle)

4. Methodology

The **Google Play Store Review Analyzer** follows a structured workflow to transform raw review text into useful insights.

Step 1: Data Collection

The dataset is obtained from Kaggle, containing app names, ratings, and user reviews.

Step 2: Data Preprocessing

This step involves cleaning the text to remove unwanted elements.

- Lowercasing text
- Removing punctuation, numbers, and special characters
- Tokenization and stop-word removal

- Lemmatization for standardizing words.

Step 3: Topic Modeling (LDA)

Latent Dirichlet Allocation (LDA) is applied to identify hidden patterns and topics in reviews. This helps in grouping reviews that discuss similar issues or features.

Step 4: Visualization

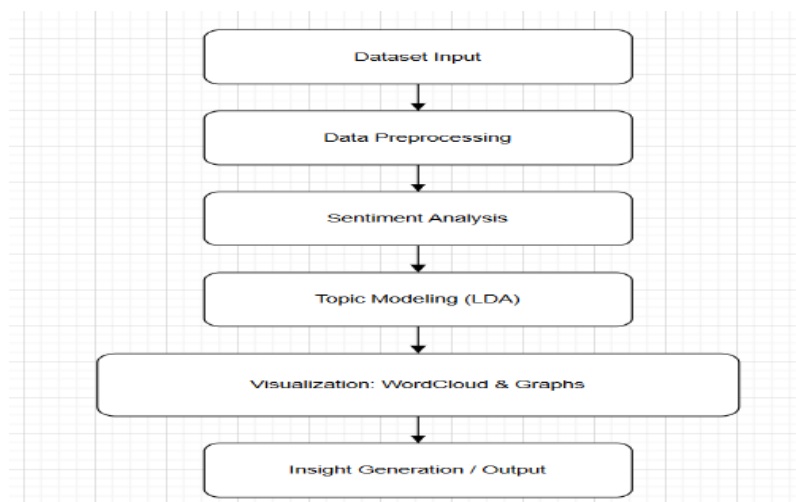
The processed data is visualized using:

- **Word Clouds:** to show the most frequently used words.
- **Bar Graphs/Pie Charts:** to represent sentiment distribution.
- **Topic Graphs:** to display prominent topics discovered through LDA.

Step 5: Insight Generation

The final stage involves interpreting results to identify trends, common complaints, and areas for improvement. These insights support better app management and user engagement strategies.

5. System Architecture



6. Results

Topic Modelling for '8 Ball Pool' app:

Extracted Insights:-

Topic 1: game, want, back, ball, match, add, make, time...

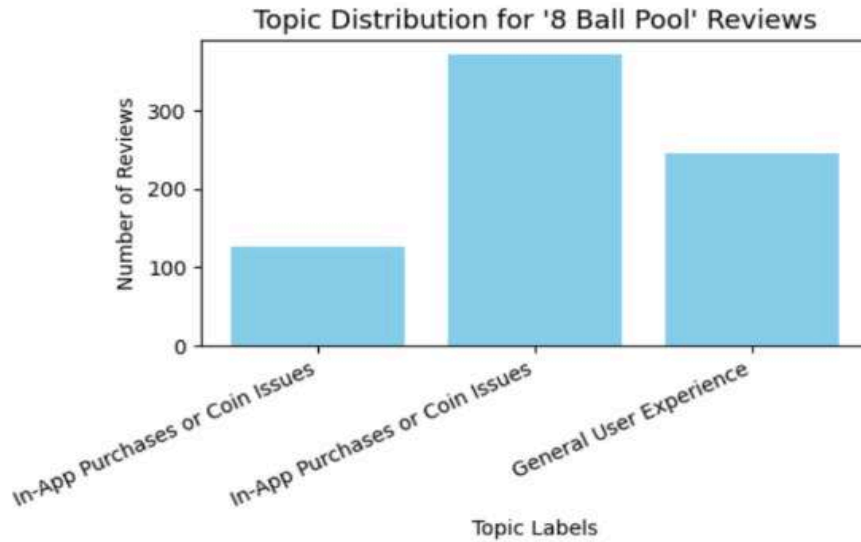
Topic 2: coin, get, star, money, getting, lost, fix, cu...

Topic 3: game, play, money, like, coin, get, pool, back...

Topic 4: game, get, time, club, pool, rigged, ive, crit...

Sr no.	Labels	Keywords
0	In-App Purchases or Coin Issues	option, got, match, tournament, want, back, lost, playing, offline, money
1	In-App Purchases or Coin Issues	coin, star, cue, money, time, different, get, ill, fix, don't
2	General User Experience	ball, run, pool, pot, turn, started,





7. Conclusion

The project “Google Play Store Review Analyzer” demonstrates how NLP can be used to analyze unstructured text data effectively. Through sentiment analysis and topic modeling, the system uncovers valuable insights about user opinions and satisfaction levels. The use of word clouds and visual graphs enhances interpretability, making the analysis more accessible.

Overall, the system provides a data-driven approach for app improvement, helping developers understand user needs and deliver better application experiences.

AI-Study Tool: Important Questions Extractor

Introduction

Students commonly refer to previous years' question papers to identify important and frequently repeated questions for exam preparation. However, manually going through multiple PDFs or scanned papers is inefficient, time-consuming, and prone to human errors. With the growing availability of AI technologies, Natural Language Processing (NLP) and Optical Character Recognition (OCR) provide an effective approach to automate the extraction and analysis of questions.

This project focuses on building a system that extracts text from question paper PDFs, cleans the text using NLP techniques, and identifies repeated questions using TF-IDF-based similarity analysis. The end goal is to help students prioritize their study efforts through a data-driven strategy.

Objectives

- To extract text from uploaded PDF question papers using OCR.
- To preprocess and clean the extracted text by removing noise, formatting errors, and irrelevant segments.
- To apply NLP techniques for segmenting and standardizing question text.
- To identify repeated or similar questions using TF-IDF and frequency analysis.
- To generate a structured list of questions ranked by their repetition count.

Technologies Used

- **Programming Language:** Python
- **Libraries / Tools:**
 - *pdf2image* — PDF to image conversion
 - *PyTesseract* — OCR extraction from scanned pages
 - *re (Regex)* — Cleaning unwanted metadata and symbols
 - *NLTK* — Tokenization and text normalization

- *Scikit-learn (TF-IDF)* — Text vectorization and similarity detection
- **Platform:** Google Colab / Jupyter Notebook / Local System
- **File Support:** PDF (input), TXT (output)

Methodology

1. PDF Upload

- The user selects and uploads multiple question paper files in PDF format.

2. OCR Conversion

- Each page of the PDF is converted to an image.
- PyTesseract extracts text from the images to obtain machine-readable content.

3. Text Cleaning and Question Extraction

- Remove headers, footers, page numbers, symbols, and noise.
- Apply normalization techniques such as lower-casing and whitespace correction.
- Use regex and sentence segmentation to detect question boundaries.

4. TF-IDF Based Repeated Question Detection

- Convert question text into TF-IDF vectors.
- Compute similarity scores to cluster repeated or similar questions.
- Count frequencies and generate a ranked list.

5. Output Generation

- Final list of repeated and important questions is exported as a clean text file for student use.

A Mini Project Report on

“MarketScope:News and Social Media Sentiment Analysis for Stocks”

Group Members: Shreepati Loknath Mishra(22)

Yash Subhash Gujar (09)

Dhiraj Anil Mestri (19)

Introduction

1.1 Background

Stock markets are highly influenced by public sentiment expressed through news and social media. Traditional analysis often overlooks these psychological factors affecting stock trends. **MarketScope** uses **Natural Language Processing (NLP)** to analyze real-time opinions and news articles, providing sentiment insights for smarter investment decisions. This helps bridge the gap between market data and investor behavior.

1.2 Problem Statement

Accurate stock price prediction requires more than just historical and numerical data. Market movements are heavily influenced by investor emotions reflected in news and social media trends. However, capturing and quantifying these sentiments manually is inefficient and unreliable. **MarketScope** addresses this gap by automating sentiment extraction and analysis, providing meaningful sentiment scores as input for next-day stock price prediction models.

1.3 Objectives

- To collect and process financial news and social media data related to specific stocks.
- To generate sentiment scores indicating positive, negative, or neutral market opinions.
- To provide these sentiment insights as input to the stock price prediction system for enhanced accuracy.
- To assist investors and analysts in making informed, data-driven trading decisions.

1.4 Scope

The system focuses on analyzing financial news and social media data related to selected stocks using NLP-based sentiment analysis. It generates sentiment scores that serve as additional input features for a next-day stock price prediction model. The project covers data collection, preprocessing, sentiment computation, and visualization. Future enhancements may include multi-language analysis, sector-specific insights, and integration with live trading dashboards.

1.5 Significance

- Enhances the accuracy of stock price prediction by incorporating real-time market sentiment.
- Automates sentiment extraction, reducing manual effort and human bias in financial analysis.
- Bridges the gap between quantitative stock data and qualitative emotional factors.
- Supports data-driven decision-making for traders, analysts, and financial institutions.

System Design

2.1 System Architecture

The **MarketScope** system follows a modular architecture designed for efficient data flow from raw text input to sentiment output. It begins with the collection of financial news articles and social media posts related to specific stocks.

2.2 Modules

- **Data Collection Module** – Gathers real-time financial news and social media data (e.g., Twitter, Reddit, financial APIs) for selected stocks.
- **Preprocessing Module** – Cleans raw text by removing URLs, symbols, and irrelevant content; performs tokenization, stop-word removal, and lemmatization.
- **NLP Cleaning** – Removes unwanted strings, symbols, and headers.
- **Sentiment Analysis Module** – Uses NLP-based models (such as VADER or transformer-based models) to classify sentiments as positive, negative, or neutral.
- **Score Aggregation Module** – Aggregates individual sentiment scores into a weighted average for each stock based on recent data.
- **Output Module** – Exports processed sentiment results in structured format (CSV/JSON) to be used by the stock price prediction system.

2.3 Data Flow

1. User or system selects target stocks for analysis.
2. The **Data Collection Module** fetches relevant news and social media posts.
3. The **Preprocessing Module** cleans and standardizes the collected text.
4. The **Sentiment Analysis Module** classifies each text segment as positive, negative, or neutral.
5. The **Score Aggregation Module** computes overall sentiment scores for each stock.
6. The **Output Module** exports the sentiment data for integration with the stock price prediction model.

2.4 Tools & Technologies

- **Language:** Python
- **Libraries:** Tweepy, BeautifulSoup, NLTK, VADER, TextBlob, Transformers (Hugging Face), Pandas, NumPy, Matplotlib
- **Platform:** Google Colab / Jupyter / Local IDE
- **Formats:** CSV / JSON (output), API data streams (input)

Implementation

3.1 Pre-processing

- Install and import all required Python libraries and APIs.
- Collect text data from financial news articles and social media posts.
- Clean the data by removing links, hashtags, emojis, and non-alphanumeric characters.
- Perform tokenization, stop-word removal, and lemmatization for text normalization.
- Prepare the cleaned data for sentiment analysis by structuring it into datasets.

3.2 Sentiment Analysis

- Apply NLP-based models such as **VADER**, **TextBlob**, or **Transformer models (BERT/roBERTa)** to classify sentiment polarity.
- Each text entry is evaluated and assigned a sentiment label — *positive*, *negative*, or *neutral*.
- Confidence scores are generated to indicate the model's certainty in each classification.
- Sentiment results are stored in a structured format for further aggregation.
- Ensure consistency by normalizing sentiment outputs across multiple data sources.

3.3 Score Aggregation

- Combine sentiment outputs from various sources such as news and social media.
- Assign weights based on data relevance, recency, and source credibility.
- Calculate an overall sentiment score for each stock using a weighted average method.
- Normalize scores within a defined range (e.g., -1 to +1) to maintain uniformity.
- Store the final sentiment scores in a database or structured file for integration with the stock prediction model.

3.4 Output Generation

- Ensure that the output data is clean, structured, and ready for model ingestion. Visualize sentiment trends using graphs or dashboards for quick interpretation.

Conclusion

MarketScope effectively processes market data to provide key insights, which serve as input for next-day stock price predictions. Its integration into the predictive pipeline improves accuracy and supports more informed investment decisions.

Future Scope

- **Real-time Market Analysis:** Implement real-time data streaming to provide up-to-the-minute market insights and predictions.
- **Automated Trading:** Extend the system to trigger automated buy/sell decisions based on predicted stock movements.
- **Scalability to Multiple Markets:** Expand the system to cover multiple stock exchanges and global markets.

- **Portfolio Optimization:** Use MarketScope outputs to suggest optimized investment portfolios based on predicted trends.

References

- Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (1998). *Forecasting with artificial neural networks: The state of the art*. International Journal of Forecasting, 14(1), 35–62. Link
- Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural Computation, 9(8), 1735–1780. Link
- Yahoo Finance. *Stock Market Data*. <https://finance.yahoo.com/>
Quandl. *Financial and Economic Data*. <https://www.quandl.com>

Results:


NLP Project
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

```

print("Section 6 Complete!")

```

MULTIPLE STOCKS ANALYSIS

```

Fetching data for AAPL...
[✓] Fetched Yahoo Finance data.
Error fetching Finviz news for AAPL: 'NoneType' object has no attribute 'text'
[✓] Fetched 5 articles from Finviz.
[✓] Fetched 6 simulated tweets.
[ ] Total articles for AAPL: 11
Fetching data for TSLA...
[✓] Fetched Yahoo Finance data.
Error fetching Finviz news for TSLA: 'NoneType' object has no attribute 'text'
[✓] Fetched 5 articles from Finviz.
[✓] Fetched 3 simulated tweets.
[ ] Total articles for TSLA: 8
Fetching data for MSFT...
[✓] Fetched Yahoo Finance data.
Error fetching Finviz news for MSFT: 'NoneType' object has no attribute 'text'
[✓] Fetched 5 articles from Finviz.
[✓] Fetched 1 simulated tweets.
[ ] Total articles for MSFT: 6
Fetching data for AMZN...
[✓] Fetched Yahoo Finance data.
Error fetching Finviz news for AMZN: 'NoneType' object has no attribute 'text'
[✓] Fetched 5 articles from Finviz.
[✓] Fetched 3 simulated tweets.
[ ] Total articles for AMZN: 8
Fetching data for NVDA...
[✓] Fetched Yahoo Finance data.
Error fetching Finviz news for NVDA: 'NoneType' object has no attribute 'text'
[✓] Fetched 5 articles from Finviz.
[✓] Fetched 3 simulated tweets.
[ ] Total articles for NVDA: 8

```


NLP Project
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

```

Finnviz: Negative (-0.133)
Finnviz: Neutral (0.000)
Finnviz: Negative (-0.500)
Finnviz: Neutral (0.000)
Finnviz: Neutral (0.000)
Twitter (Simulated): Positive (0.500)
Twitter (Simulated): Positive (0.114)
Twitter (Simulated): Neutral (0.000)
[✓] NVDA: Neutral (-0.002)

```

SENTIMENT RANKINGS:

	Symbol	Sentiment_Score	Sentiment_Label	Articles	Confidence
3	AMZN	0.154	Positive	8	0.187
1	TSLA	0.107	Positive	8	0.313
4	NVDA	-0.002	Neutral	8	0.156
2	MSFT	-0.077	Neutral	6	0.256
0	AAPL	-0.108	Negative	11	0.219

Results saved to: stock_sentiment_results.csv

[✓] Section 6 Complete!

Resume Parser using NLP

Team Members:

- 1. Divya Nitin Pawar (Roll No. 36)**
- 2. Gauri Rajendra Sonawane (Roll No. 47)**
- 3. Vaishnavi Rajendra Ugave (Roll No. 48)**

Abstract

The Resume Parser using NLP is a smart system that automatically reads resumes and extracts important details like name, skills, education, and experience.

It uses Natural Language Processing (NLP) and tools like SpaCy, PyPDF2, and Regex in Python.

The system takes resumes in PDF or Word format, converts them to text, and then finds useful information using language models and pattern matching.

This helps companies quickly sort and understand many resumes without doing it manually.

Introduction

With the increasing use of digital recruitment platforms, companies receive hundreds of resumes daily in various formats like PDF and DOCX. Manually scanning and analyzing each resume to extract relevant details is time-consuming and error-prone.

The Resume Parser using NLP project aims to automate this process by using Natural Language Processing (NLP) to extract key candidate details such as name, email, phone number, education, experience, and skills from unstructured text. This ensures faster, more consistent, and accurate resume screening.

Problem Statement

Recruiters often struggle to analyze large volumes of resumes presented in different layouts and styles. Manual screening is repetitive, slow, and inconsistent, often leading to delays in hiring or missed talent opportunities.

This project addresses the problem by creating an automated system that reads resumes in PDF and DOCX formats, extracts key information using NLP and pattern-matching techniques, and presents the data in a structured form like JSON. The result is a faster, more reliable, and efficient resume review process.

Objectives

- To develop an **automated resume parsing system** using NLP.
- To extract important details like **Name, Email, Phone, Education, Experience, and Skills**.
- To use **SpaCy** for Named Entity Recognition (NER) and linguistic analysis.
- To employ **PyPDF2** and **python-docx** for reading files in different formats.
- To apply **Regex** for identifying contact and numeric patterns.
- To structure and display extracted data in a clear JSON format.

Technology Stack

- **Language:** Python
- **Framework:** Flask (for backend)
- **Libraries:** SpaCy, PyPDF2, python-docx, Regex (re)
- **Frontend:** HTML, CSS, JavaScript
- **Data Format:** JSON
- **IDE:** Visual Studio Code

System Architecture & Working

The system follows a client–server architecture:

- **Frontend (Client):** A simple web interface built with HTML, CSS, and JavaScript that allows users to upload resumes and view extracted results.
- **Backend (Server):** A Flask application in Python that extracts text using PyPDF2 or python-docx, performs NLP processing with SpaCy, and identifies key entities using Regex patterns.

- **Data Flow:**

1. User uploads a resume file.
2. The backend extracts and cleans text.
3. SpaCy performs tokenization and NER.
4. Regex detects Email, Phone, and numeric patterns.
5. Extracted information (Name, Skills, Education, etc.) is structured into JSON.
6. The frontend displays results to the user.

This architecture ensures modularity, speed, and ease of extension for future features like resume ranking or candidate matching.

Outcome

- Automated extraction of essential resume information.
- User-friendly, responsive web interface.
- Structured output in JSON format.
- Reduced manual screening time by more than 80%.

Future Scope

- Add candidate ranking based on job relevance.
- Integrate with HR portals or ATS systems.
- Support for multiple languages and formats.
- Use Machine Learning models to improve accuracy.

Conclusion

The Resume Parser using NLP project effectively automates resume reading and analysis using Natural Language Processing.

By integrating SpaCy, PyPDF2, python-docx, and Regex, the system accurately extracts key candidate details and presents them in a structured, machine-readable format.

This solution greatly reduces manual effort for recruiters, increases efficiency, and ensures consistent data extraction.

A Mini Project Report on

NLP-Based Resume Ranker

Group Member : Abhishek Mahesh Ambavale (Roll No.01)

Pratik Dilip Ghanekar (Roll No.07)

Vipulya Bhikaji Kamble (Roll No.14)

Sarvesh Sunil Parthe (Roll No.33)

1. Introduction

In today's competitive job market, recruiters often receive hundreds of resumes for a single job opening. Manually screening each resume is time-consuming and prone to human bias.

To overcome this problem, Natural Language Processing (NLP) techniques can be used to automatically analyze and rank resumes based on a given job description.

The NLP-based Resume Ranker is a web application that helps recruiters or HR professionals to quickly identify the most suitable candidates. It compares the content of uploaded resumes with the provided job description and ranks them according to their similarity. This system provides a smart, efficient, and unbiased way to shortlist candidates.

2. Objectives

The main objectives of this project are:

1. To develop an automated system that ranks resumes according to a job description using NLP techniques.

2. To reduce manual effort and time required for resume screening.
3. To implement a web-based interface for uploading resumes and viewing ranked results.
4. To use TF-IDF and cosine similarity for accurate textual comparison between resumes and job descriptions.
5. To improve ranking accuracy by including keyword overlap and text preprocessing methods.

3. Technology Used

- **Programming Language** : Python
- **Web Framework** : Flask
- **Frontend** : HTML, CSS, JavaScript
- **Libraries Used** : scikit-learn , pdfplumber, werkzeug, re, json, os.
- **Text Analysis Technique** : TF-IDF (Term Frequency-Inverse Document Frequency)
- **Similarity Measure** : Cosine Similarity
- **Storage** : JSON file (for storing extracted resume data)

4. Methodology (Working of the System)

The system works in several steps:

Step 1: Uploading Resumes

- ➔ The user uploads multiple PDF resumes through the web interface.
- ➔ Each resume file is stored on the server, and its text content is extracted using the

pdfplumber library.

Step 2: Preprocessing

- ➔ Extracted text is cleaned by:
 - Converting to lowercase
 - Removing punctuation, numbers, and special characters
 - Removing extra spaces

Step 3: Job Description Input

- ➔ The recruiter or user provides a job description in a text area on the website.

Step 4: Feature Extraction (TF-IDF)

- ➔ Both the job description and all resumes are converted into TF-IDF vectors.
- ➔ TF-IDF helps in identifying important words in each document by reducing the impact of common words.

Step 5: Similarity Calculation

- ➔ Cosine similarity is calculated between the job description and each resume vector.
- ➔ Additionally, keyword overlap is computed to check how many key terms from the job description appear in each resume.

Step 6: Scoring and Ranking

- ➔ A combined score is calculated using:
$$\text{Final Score} = 0.75 \times \text{Cosine Similarity} + 0.25 \times \text{Keyword Overlap}$$
$$\text{Final Score} = 0.75 \times \text{Cosine Similarity} + 0.25 \times \text{Keyword Overlap}$$
- ➔ Resumes are ranked in descending order of their scores.

Step 7: Displaying Results

→ The system displays a ranked list of resumes with:

- ◆ Filename
- ◆ Match percentage
- ◆ Progress bar visualization

5. Applications

- **Human Resource (HR) Departments** : Used by HR professionals to automate the resume screening process, reducing manual effort and time spent on candidate shortlisting.
- **Recruitment Agencies** :Helps staffing firms to quickly match resumes with client job descriptions and provide relevant candidate profiles.
- **Job Portals & Career Platforms**: Can be integrated into job search websites to help job seekers match their resumes to job listings and improve their resumes accordingly.

6. Conclusion

The NLP-Based Resume Ranker provides an efficient and intelligent solution to automate resume screening using Natural Language Processing. By analyzing the similarity between resumes and a job description using TF-IDF and cosine similarity techniques, the system can rank candidates based on relevance. This saves time, improves consistency in hiring, and reduces human bias in the recruitment process.

Smart Chat-to-Notes Converter for Student Group Discussions

Team Members

1. Tanvi Pramod Mahadik (Roll No. 18)
2. Prathamesh Ravindra Pabe (Roll No. 26)
3. Nikita Suryaprakash Pandey (Roll No. 30)

Abstract

The **Smart Chat-to-Notes Converter** is an intelligent system that automatically summarizes lengthy project discussions or chat conversations into concise, meaningful notes. It uses **Natural Language Processing (NLP)** and deep learning models like **BART (facebook/bart-large-cnn)** to extract the most relevant information. Users can either type directly into a chat interface or upload text files. The system then cleans, processes, and summarizes the input, providing a clear overview of important points and decisions. This helps teams save time, improve communication, and focus on essential details without reading through long discussions.

Introduction

In academic and professional environments, meetings and group discussions generate large amounts of textual data. Manually reviewing this data to identify key points is both time-consuming and inefficient. The **Chat-to-Notes Converter** project addresses this issue by leveraging **NLP** to automatically generate summaries from chat logs or project transcripts. It enhances team productivity and ensures that no critical information is missed during analysis.

Problem Statement

Teams often struggle to extract meaningful insights from lengthy project discussions or chat histories. Manually summarizing this information can lead to oversight, reduced efficiency, and information overload. The proposed system solves this problem by developing an **AI-powered summarizer** that reads large amounts of text, identifies essential sentences, and generates concise summaries. It works for both **chat inputs** and **uploaded files**, making it ideal for team collaboration platforms.

Objectives

- To develop an **automated summarization system** using NLP.
- To summarize both **chat messages** and **uploaded text files**.
- To implement **BART Transformer model** for abstractive summarization.

- To use **SpaCy** for preprocessing and text cleaning.
- To provide a **real-time, user-friendly web interface** for interaction.

Technology Stack

- **Language:** Python
- **Framework:** Flask
- **Frontend:** HTML, CSS, JavaScript
- **Libraries:** SpaCy, Hugging Face Transformers (BART), SentenceTransformer
- **Architecture:** Client–Server Model
- **IDE:** Visual Studio Code

System Architecture & Working

The system follows a client–server structure with separate **frontend** and **backend** modules.

Frontend (Client):

- Built with HTML, CSS, and JavaScript.
- Provides a chat-based interface for users to type or upload files.
- Displays summarized output dynamically.

Backend (Server):

- Developed in Python using Flask.
- Handles file uploads and chat messages.
- Performs **text cleaning** using SpaCy (removing symbols, URLs, and extra spaces).
- Generates summaries using the **BART model** for abstractive summarization.
- Optionally applies **SentenceTransformer** for contextual understanding.

Data Flow:

1. User enters chat text or uploads a file.

2. Flask backend processes and cleans the text.
3. NLP model generates a summary.
4. Summary is sent back and displayed in the frontend chat interface.

Outcome

- Automatic summarization of long project discussions.
- Easy-to-use chat interface for interaction.
- Real-time summarization with accurate context retention.
- Saves significant time and enhances team decision-making.

Future Scope

- Integration with project management and collaboration tools.
- Support for **multilingual summarization**.
- Inclusion of **voice-to-text summarization** for spoken discussions.
- Enhanced analytics to identify frequently discussed topics.

Conclusion

The **Smart Chat-to-Notes Converter** successfully demonstrates the application of NLP in simplifying team communication. By using advanced models like **BART**, the system produces concise, high-quality summaries from chat data and files. This reduces manual effort, improves productivity, and ensures that important decisions are easily accessible. The project highlights the potential of **AI-driven automation** in educational and professional teamwork environments.

A Mini Project Report on

Smart Text Summarizer

Group Member : Pranita Pandurang Kadam (Roll No.12)

Gauri Sushil Kate (Roll No.15)

Saniya Suresh Rajeshirke (Roll No.38)

1. Introduction

Nowadays, people face a huge amount of text every day – news, research papers, reports, articles, and emails. Reading everything takes a lot of time and effort.

To solve this problem, we developed a **Smart Text Summarizer**. It is a **web-based application** that takes a large text and creates a short and meaningful summary. The summarizer also provides extra features like:

- Finding **important keywords**
- **Sentiment analysis** (shows if the text is positive, negative, or neutral)
- **Named Entity Recognition (NER)** to identify people, places, and organizations
- **Automatic title generation**
- **Text-to-Speech** (audio summary)
- Downloading the summary as **PDF** or receiving it via **email**

2. Objectives

The main goals of this project are:

1. To summarize long text into a short, clear version.
2. To allow users to upload files in formats like **.txt, .pdf, and .docx**.
3. To apply **NLP techniques** for keyword extraction, sentiment analysis, and NER.
4. To provide **audio output** of the summary for better accessibility.
5. To give an option to **download summary as PDF**.
6. To send the summary (PDF + audio) to the user's email.

3. Technology Used

- **Programming:** Python
- **Framework:** Flask (for web app)

● NLP Libraries:

- spaCy → Named Entity Recognition, POS tagging
- NLTK → Tokenization and text segmentation
- TextBlob → Sentiment analysis
- Summa → Summarization using TextRank algorithm

● Speech/Audio: gTTS (Google Text-to-Speech)

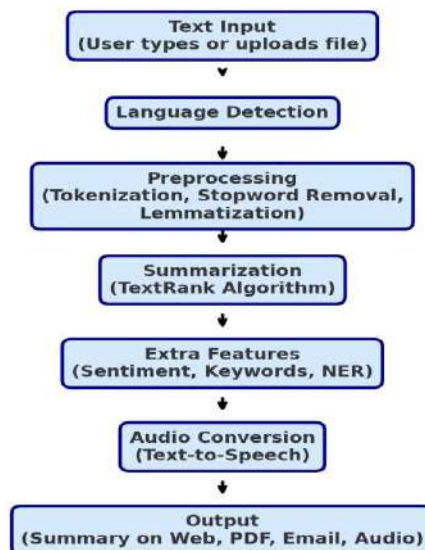
● File Handling:

- PyPDF2 → Extracts text from PDF
- python-docx → Extracts text from Word files
- ReportLab → Creates downloadable PDF summaries

● Email Integration: smtplib + EmailMessage (for sending summaries via email)

4. Methodology (Working of the System)

Smart Text Summarizer - Workflow Diagram



1. Text Input

- User can type text or upload a document (PDF, DOCX, TXT).

2. Language Detection

- The system detects the language of the text (English, Marathi, etc.).

3. Text Preprocessing

- Splits the text into sentences and words (tokenization).
- Removes stopwords, punctuation, and unnecessary characters.
- Lemmatization: converts words to their base form.

4. Summarization

- Uses the **TextRank algorithm** to generate a short extractive summary.
- User can set how long the summary should be (example: 20% of original text).

5. Additional Features

- **Sentiment Analysis** → Positive, Negative, or Neutral text.
- **NER (Named Entity Recognition)** → finds names of people, places, organizations.
- **Keyword Extraction** → highlights important words.

6. Audio Conversion

- The summary is converted into audio using gTTS.

7. Output

- Summary, keywords, entities, and sentiment are shown on the website.
- Users can **download PDF**, **listen to audio**, or **get results by email**.

5. Results

- The system reduces large documents into short, meaningful summaries.
- Users can view the summary online, download it, or receive it via email.
- Keywords, entities, and sentiment give extra useful information.
- Audio feature makes it helpful for visually impaired users or those who prefer listening.

6. Applications

- **Education** – Summarizing textbooks and research papers
- **Business** – Summarizing reports and emails
- **Healthcare** – Making short versions of patient records or medical studies
- **News & Media** – Creating short news summaries
- **Accessibility** – Audio summaries for visually impaired people

7. Conclusion

The Smart Text Summarizer project shows how **NLP and automation** can make reading and understanding information easier.

This tool is useful for:

- Students → to quickly get summaries of textbooks or notes
- Researchers → to understand research papers faster
- Professionals → to read reports and emails quickly
- Visually impaired people → to listen to audio summaries

By combining summarization, keyword extraction, NER, sentiment analysis, and audio features,

A Mini Project Report on

“Text Categorization”

Group Members: Anas Nazeer Mirkar (Roll No. 21)
Saurav Sudhakar Nalawade (Roll No. 25)
Sohail Rijwan Solkar (Roll No. 46)

1. Introduction

With the rapid growth of digital media, enormous volumes of textual data are generated daily through social media, reviews, and online content. Processing and understanding this unstructured data manually is impractical.

Text Categorization, also known as **Text Classification**, is an essential Natural Language Processing (NLP) technique that automatically assigns predefined categories or sentiments (such as positive or negative) to text documents. In this project, we implement a **Sentiment Analysis model** that classifies movie reviews from the IMDB dataset into **positive** or **negative** sentiments using **Deep Learning** (LSTM neural network).

2. Problem Definition

The challenge is to develop a model that can:

- Understand the context and semantics of natural language text.
- Categorize unseen movie reviews accurately as **positive** or **negative**.
- Handle variations in vocabulary, sentence length, and expression.

Traditional machine learning methods like Naïve Bayes or SVM rely heavily on handcrafted features. Instead, this project uses an **LSTM (Long Short-Term Memory)** network, which can capture long-term dependencies and contextual information in sequences, making it ideal for text data.

3. Objectives

- To preprocess and prepare textual data for machine learning models.
- To build and train an **LSTM-based neural network** for sentiment classification.
- To evaluate the model's accuracy and generalization on unseen data.

4. **Proposed Solution:** The proposed solution involves building a **Sequential LSTM Model** using the **IMDB movie reviews dataset** provided by Keras.

Steps followed:

Step1: Load IMDB dataset $\rightarrow (X_{train}, y_{train}), (X_{test}, y_{test})$

Step2: Pad all sequences to fixed length (200).

Step 3: Build Sequential Model.

Embedding $\rightarrow$ LSTM $\rightarrow$ Dense (Sigmoid)

Step4: Compile Model (Adam optimizer, Binary Cross-Entropy).

Step5: Train model with 80-20 validation split.

Step6: Evaluate test accuracy and visualize results.

Step 7: Predict sentiment for new reviews entered by the user.

5. Conclusion

This project successfully demonstrates **Text Categorization** using **Deep Learning (LSTM)** for sentiment analysis on the IMDB dataset. The model achieves high accuracy and generalizes well to unseen data. It effectively learns contextual word relationships and captures sentiment from text sequences. Future improvements can include:

- Using **Bidirectional LSTMs** or **Transformer-based models** (like BERT).
- Expanding the dataset and performing **hyperparameter tuning**.
- Deploying the model as a web API or chatbot for real-time sentiment analysis.

A Mini Project Report on
“Toxic Comment Classifier using NLP”

Group Members: Shreyash Prakash Parit (Roll No. 32)

Viraj Vinod Panchal (Roll No. 29)

1. Introduction

This project allows users to enter a comment or sentence to determine whether it is Clean, Offensive, or Toxic. Manual moderation can be slow and inconsistent, so the system uses Natural Language Processing (NLP) and machine learning to automatically classify the input. A simple Tkinter GUI enables users to test comments easily, showcasing the practical use of NLP for interactive text classification.

2. Problem Definition

Toxic language can harm digital communication by causing harassment and reducing user engagement. Manual moderation is slow and inconsistent, especially with large volumes of comments. This project develops an NLP-based system that classifies user comments entered through a Tkinter GUI. The problem can be summarized in two main challenges:

1. User Input Processing: Handling comments entered via the GUI and preparing them for analysis.
2. Automated Classification: Preprocessing text, extracting features, and classifying comments as Clean, Offensive, or Toxic for fast and consistent detection.

3. Objectives

The objective of this project is to develop an NLP-based system that classifies user-entered comments into Clean, Offensive, or Toxic categories. It focuses on applying key NLP techniques such as text cleaning, stopwords removal, and lemmatization to prepare data for analysis. The project also emphasizes feature extraction using TF-IDF vectorization, which converts processed text into numerical representations suitable for machine learning. A Logistic Regression model is trained on this extracted data to

perform classification. Finally, a Tkinter-based GUI allows users to input comments and instantly view NLP-driven predictions.

4. Proposed Solution

The proposed solution follows a structured NLP pipeline to classify user-entered comments. Each comment is preprocessed through tokenization, lowercasing, stopword removal, and lemmatization. The cleaned text then undergoes feature extraction using TF-IDF, converting it into numerical vectors for analysis. A trained Logistic Regression model uses these features to predict whether the comment is Clean, Offensive, or Toxic, and the result is displayed through a Tkinter GUI for real-time interaction.

5. Technology / Platform Used

- Python: Core programming language.
- NLTK: Text preprocessing (stopword removal, lemmatization).
- Scikit-learn: TF-IDF feature extraction and Logistic Regression.
- Tkinter: GUI for input and predictions.
- Pandas & NumPy: Data handling and numerical operations.
- Matplotlib & Seaborn: Visualization (e.g., confusion matrix).

6. Workflow

- User Input → Comment entered by user.
- Text Preprocessing → Cleaning, stopword removal, lemmatization.
- Feature Extraction → Convert text to TF-IDF numerical features.
- Classification → Logistic Regression predicts the comment category.
- Prediction Output → GUI displays Clean, Offensive, or Toxic result.

7. Conclusion

This mini-project demonstrates the use of NLP to classify user-entered comments. By combining preprocessing, TF-IDF feature extraction, and Logistic Regression, the system identifies comments as Clean, Offensive, or Toxic. The Tkinter GUI allows real-time testing, making the tool interactive and showcasing the practical application of NLP for automated text classification.

Translator Chatbot: German to English

Group Members: Alisha Firoj Sayyad (44)
Shrutika Shrinivas Vetkar (49)
Uniza Ayub Khadpolkar (16)

Abstract

Our project is a **Translator Chatbot**, a web-based tool that translates text from **German to English** in real time. It uses **artificial intelligence techniques** to understand and process language, making translation accurate and fast.

The application is simple to use: you type or paste German text into the website, and the chatbot immediately gives the English translation. The system also handles the technical processing in the background so the user sees results instantly.

This project shows how technology can help **people communicate across languages**. In the future, it can be expanded to translate multiple languages or even speak translations aloud.

Introduction

Language is a key tool for communication. People from different countries often face language barriers, which can make understanding each other difficult. Our project solves this by automatically translating **German text into English**.

The system is built to be **easy to use**, with a clean web interface. Users do not need to know any programming or technical details—they simply type the text and get the translation.

We used a **pre-trained translation model** that understands German grammar and sentence meaning, allowing the system to produce accurate and readable English translations. The project also includes interactive features like a **quiz for learning German vocabulary**, making the system educational as well as practical.

Objective

- Make a **user-friendly website** for translating German to English.
- Translate text **quickly and accurately** using AI techniques.
- Ensure the system is **interactive and easy to use** for anyone.

- Handle multiple user requests and maintain stable performance.
 - Demonstrate how modern technology can help **bridge language gaps**.
-

How It Works (Simple Explanation)

Our system works in **five easy steps**:

1. **User Input**
 - You type a word or sentence in German into the website.
2. **Sending the Text to the System**
 - When you click the “Translate” button, the website sends your text to the system in the background.
3. **Translation Process**
 - The system uses AI to understand the German words and generate the correct English translation.
 - It works like a smart assistant that reads, understands, and rewrites your text in English.
4. **Returning the Result**
 - The system sends the English translation back to the website instantly.
5. **Displaying the Translation**
 - The translated English text appears below your original input on the page.
 - No extra steps or technical knowledge are needed.

The system also handles errors safely—for example, if you don’t enter any text, it asks you to provide input. Even if the internet connection is slow, the system manages the request without crashing.

Technology Used (Explained Simply)

- **Python:** The main programming language used to create the translation system.
- **Flask:** A tool that helps run the website and connect the user interface to the translation system.
- **Translation Library:** A pre-built tool that uses AI to understand German and translate it to English.
- **Web Design Tools:** HTML, CSS, and JavaScript are used to make the website look clean and user-friendly.

All these tools work together to make the system **fast, reliable, and easy for anyone to use**.

Testing and Results

We tested the system with simple words and phrases:

- “Hallo” → “Hello”
- “Guten Morgen” → “Good Morning”
- “Wie geht’s?” → “How are you?”
- “Danke” → “Thank you”

The chatbot translated each input correctly and instantly. Even when multiple users typed text at the same time, the system handled all requests efficiently.

Conclusion

The Translator Chatbot shows how **artificial intelligence and web technology** can help people communicate across languages. It provides **fast, accurate, and interactive translations** from German to English.

Anyone can use it easily—no technical knowledge is required. The project also demonstrates the practical use of modern technology to **solve real-world problems** like language barriers. In the future, we plan to add **more languages** and **speech translation features** to make it even more useful.

A Mini Project report on

“SEO & Business Growth Assistant”

Team Members:

1. Omkar Dattatray Waghalkar	50
2. Rahul Hiratsingh Sisode	45
3. Sujal Vijay Patil	35

Abstract

This project presents an AI-powered SEO and Business Growth Assistant that offers real-time strategic guidance through Natural Language Processing and modern web technologies. Built with React, Tailwind CSS, Supabase, PostgreSQL, Lovable AI Gateway, and Gemini 2.5 Flash, it provides intelligent, conversational recommendations for digital marketing and business growth. Unlike traditional tools, the assistant enables natural, real-time interaction with session history and data persistence, making strategy planning faster and more accessible.

Introduction

In today’s digital age, businesses rely on online visibility and customer engagement to stay competitive. Search Engine Optimization (SEO) and digital marketing are key to improving brand reach and growth, yet many lack access to real-time, expert guidance.

This project introduces an AI-powered SEO & Business Growth Assistant — a smart chat system that provides instant, data-driven recommendations for marketing strategies, keyword planning, and content optimization. Built with modern technologies like React, Tailwind CSS, Supabase, and Gemini 2.5 Flash, the assistant delivers personalized insights, real-time responses, and session-based interaction for effective business strategy support.

Problem Statement

Many startups and small businesses lack access to expert SEO and business guidance. Existing tools provide data but fail to offer interactive, AI-driven insights that are easy to apply. This project aims to build an intelligent, conversational platform that delivers real-time, personalized SEO and business growth recommendations through an AI-powered chat assistant.

Objectives

The main objectives of this project are:

1. To develop an AI-powered chat assistant that provides SEO and business growth guidance.
2. To integrate a database for storing chat sessions and message history.
3. To enable real-time streaming responses using AI models.
4. To implement a modern and responsive frontend interface using React and Tailwind.
5. To use Supabase and Edge Functions for backend processing and AI requests.

Technology Stack

- Frontend: React 18, Tailwind CSS, Shadcn/UI, TypeScript
- Backend: Supabase, PostgreSQL, Edge Functions (Deno)
- AI Integration: Google Gemini 2.5 Flash via Lovable AI Gateway
- Programming Tools: VS Code, npm, Supabase CLI
- Hosting & Database: Supabase Cloud
- Libraries/Frameworks: React Query, Radix UI, Lucide React

System Architecture & Working

The AI-Powered SEO and Business Growth Assistant follows a **client-server architecture** combining conversational AI with modern web technologies.

1. Frontend (Client):

Developed using React, Tailwind CSS, and Shadcn/UI, it provides a clean chat interface where users can send queries and view AI-generated responses in real time.

2. Backend (Server):

Built with Supabase, PostgreSQL, and Edge Functions (Deno). It manages user sessions, stores chat history, and connects to the AI model through the Lovable AI Gateway.

• Data Flow:

1. User enters a query in the chat interface.
2. The message is stored in Supabase and sent to the Edge Function.
3. The function calls the AI model for response generation.
4. The response is streamed in real time and saved in the database.

5. The chat history is displayed for future reference.

Outcome

- Successfully developed an AI-powered SEO & Business Growth Assistant that provides real-time, conversational guidance for improving digital marketing strategies, keyword optimization, and business expansion.
- Integrated Gemini 2.5 Flash AI model via Lovable AI Gateway for intelligent, context-aware response generation.
- Implemented a responsive and interactive web interface using React 18, TypeScript, Tailwind CSS, and Shadcn/UI, ensuring smooth user experience.
- Enabled real-time chat streaming and message persistence using Supabase (PostgreSQL + Edge Functions) for backend operations.

Future Scope

- User Authentication and Personalization – Add login functionality to allow personalized user dashboards, chat history tracking, and customized SEO recommendations.
- Analytics-Based SEO Suggestions – Integrate data analytics tools to provide keyword trend analysis, traffic predictions, and ranking performance insights.
- Voice and Chat Hybrid Interaction – Implement voice input/output to make the assistant more interactive and accessible for users with different preferences.
- Platform Integrations – Connect the system with platforms like WordPress, Shopify, and HubSpot for automated SEO optimization and content management.

Conclusion

The AI-Powered SEO & Business Growth Assistant effectively combines conversational AI, modern frontend tools, and cloud-based backend systems to provide intelligent business guidance. It delivers real-time, AI-generated suggestions for SEO, marketing, and growth strategies. Built with React, Tailwind CSS, Supabase, PostgreSQL, and Gemini 2.5 Flash, the system ensures fast performance, smooth interaction, and reliable data storage, making it a practical and user-friendly solution for digital business support.

A Mini Project Report on

“AUTO-SPELL FIX”

Group Members: Bendkhale Atharv Mahendra (02)
Deorukhakar Aditi Sanjay (06)
Kadu Raj Manohar (13)

INTRODUCTION

In today’s digital world, written accuracy is crucial. Traditional spell checkers often miss contextual or domain-specific errors. *Auto-Spell Fix* uses NLP techniques to correct spelling and grammar intelligently while preserving technical terms. It supports text, PDF, and DOCX formats with a simple web interface for students, professionals, and content creators.

LITERATURE SURVEY

1. **Historical Systems:** Early rule-based methods (Peterson, 1980) used dictionary lookups and edit distance but lacked context awareness.
2. **Statistical Models:** Noisy channel and n-gram models improved contextual correction (Kernighan et al., 1990).
3. **Neural Models:** Transformer-based systems like BERT and GPT enhanced contextual grammar correction.
4. **Hybrid Approaches:** Tools like SymSpell and TextBlob combine speed with accuracy using mixed methods.
5. **Limitations:** Existing tools struggle with technical terms, acronyms, and domain-specific contexts.

PROBLEM DEFINITION

Writers frequently face spelling and grammar errors, especially in technical content. There is a need for an intelligent, adaptive system that ensures accurate, context-aware corrections while supporting multiple document formats.

Objectives:

- Develop an NLP-based hybrid correction system.
- Combine SymSpell and TextBlob for efficient contextual correction.
- Enable multi-format document input and real-time feedback.

- Preserve domain-specific and technical terminology.

PROPOSED SOLUTION

The system uses advanced NLP to deliver accurate and context-aware corrections. It integrates:

- **Frontend:** User-friendly web interface for text input and uploads.
- **Document Processing:** Handles PDF, DOCX, and TXT using PyPDF2 and python-docx.
- **Hybrid Engine:** Combines SymSpell for spelling correction, TextBlob for grammar, and custom rules for contractions and term preservation.
- **Backend (Flask):** Manages correction, storage, and secure communication.
- **Results Display:** Shows original vs. corrected text with basic statistics.

METHODOLOGY

- Designed modular Flask-based web architecture.
- Used Python NLP libraries (SymSpellPy, TextBlob, Autocorrect).
- Implemented hybrid correction flow: normalization → spelling → grammar → term preservation.
- Added user interface for real-time correction and file upload.
- Tested accuracy, speed, and performance using varied document samples

HARDWARE & SOFTWARE USED

Hardware: Intel i3/i5 processor, 4–8GB RAM, Internet connection.

Software: Windows/Linux, Python 3.8+, Flask, HTML/CSS/JS, SymSpellPy, TextBlob, PyPDF2, python-docx, VS Code.

EXPERIMENTATION & RESULTS

Setup: Python with Flask, NLP libraries (SymSpellPy, TextBlob), and file handlers.

Input: Text or file uploads with spelling/grammar errors.

Output: Corrected text, statistics, and download option.

Performance:

- Spelling accuracy: 85–90%
- Grammar accuracy: 80–85%
- Technical term preservation: 95%

- Response time: <3 seconds for 10-page files

Observations:

Hybrid architecture gave better accuracy and preserved domain terms effectively.

Results :

The screenshot displays the 'Advanced Spell Checker' web application interface. The header is purple with the title 'Advanced Spell Checker' and a subtitle 'Upload PDF, DOCX, or TXT files or type text directly'. The main content area is divided into two columns. The left column, titled 'Type Text', contains a text input field with the placeholder 'Type or paste your text here...' and a 'Check Spelling' button. The right column, titled 'Upload File', contains a file upload area with a folder icon, a 'Click to upload file' link, and a 'Process File' button. Below the upload area, a note states: 'Note: For PDFs, only text-based PDFs are supported (not scanned images)'. The bottom section of the interface shows the 'Original Text' and 'Corrected Text' fields, both containing the word 'Sentecen'. At the bottom, three statistics are displayed: '1 Words', '8 Characters', and '1 Sentences'.

Advanced Spell Checker
Upload PDF, DOCX, or TXT files or type text directly

Type Text
Enter your text:
Type or paste your text here...
Check Spelling

Upload File
Upload PDF, DOCX, or TXT:
Click to upload file
Supports: PDF, DOCX, TXT (Max 16MB)
Process File
Note: For PDFs, only text-based PDFs are supported (not scanned images)

Original Text
Sentecen

Corrected Text
Sentecen

1 Words
8 Characters
1 Sentences

File "input.txt" processed successfully!

Type Text

Enter your text:

She dont know hows to code properly.
I am lerning NLP and python.
This PDF has spelling mistaks.

Check Spelling

Upload File

Upload PDF, DOCX, or TXT:

Click to upload file

Supports: PDF, DOCX, TXT (Max 16MB)

Process File

Note: For PDFs, only text-based PDFs are supported (not scanned images)

File: input.txt

Original Text

She dont know hows to code properly.
I am lerning NLP and python.
This PDF has spelling mistaks.

Corrected Text

She don't know how's to code properly.
I am learning NLP and python.
This PDF has spelling mistake.

18 Words

98 Characters

3 Sentences

CONCLUSION

Auto-Spell Fix successfully integrates NLP methods for efficient and accurate error correction. The hybrid SymSpell–TextBlob approach improves accuracy while maintaining technical terms. Its multi-format support, fast performance, and web-based interface make it practical for real-world use by students and professionals.

REFERENCES

1. Jurafsky & Martin, *Speech and Language Processing*, Pearson, 2023.
2. TextBlob Documentation (2023)
3. SymSpellPy (GitHub, 2023)
4. Flask Framework Docs (2024)
5. PyPDF2 & python-docx Libraries (2023)

Piggy Bank Chatbot Project Report

1. Introduction

In today's fast-paced world, many individuals struggle to manage their finances effectively, save systematically, or achieve their financial goals. Traditional saving methods often lack guidance, personalization, and engagement, making it difficult for users to stay motivated and disciplined.

To address this challenge, the Piggy Bank Chatbot project provides an intelligent, interactive solution that helps users save money systematically and achieve their financial goals. The system leverages a rule-based AI engine and a pre-trained chatbot to provide personalized saving suggestions, guide users in creating virtual piggy banks for specific goals, and simulate a banking experience using dummy funds.

Users can track their progress, ask questions such as "How much is in my piggy bank?" or "Where should I save next?" and receive real-time guidance, all through a conversational interface.

The project is developed using Python (Flask/FastAPI) for backend processing, React.js or Flutter for the frontend, and GPT-like chatbot technology for interactive user support. By combining AI-driven insights with a simulated banking environment, this system encourages disciplined saving, enhances financial literacy, and provides a safe, engaging, and user-friendly platform for financial goal management.

2. Objective

- To provide users with a smart and interactive chatbot that guides them in saving money effectively.
 - To enable users to create and manage virtual piggy banks for specific financial goals.
 - To offer personalized saving suggestions based on user income and spending patterns using AI-driven logic.
 - To simulate a banking experience with dummy funds, allowing users to practice savings without financial risk.
 - To develop a user-friendly interface for easy tracking of balances, goals, and progress.
 - To promote financial literacy and disciplined saving habits through real-time guidance and interactive support.
 - To validate the effectiveness of AI-assisted saving strategies before implementing real money transactions.
-

3. Technology Used

1. **Python:** Core language for chatbot logic, AI suggestions, and backend processing.

2. **Flask / FastAPI:** Lightweight web frameworks for backend management, user request handling, and real-time responses.
 3. **Rule-Based AI / NLP:** Enables the chatbot to understand user queries and provide relevant saving advice.
 4. **React.js / Flutter:** Frontend development for responsive and interactive interface.
 5. **Database (Firebase / SQLite / JSON):** Stores user profiles, piggy bank details, dummy balances, and transaction logs.
 6. **Chatbot Integration (GPT API / Rasa):** Provides conversational support and engagement.
 7. **Optional Blockchain Ledger (Lite):** Secure transaction logging in JSON, with potential for smart contract upgrades.
-

4. Methodology

- **User Input:** Users provide salary, financial goals, and queries related to savings.
- **Data Processing:** Extracts details such as goal amounts, target dates, and current balances for accurate recommendations.
- **Rule-Based AI / Chatbot Processing:** Interprets user queries and guides users in managing virtual savings.
- **Savings Suggestion Engine:** Calculates personalized saving plans (e.g., allocating percentages to emergency funds or home goals).
- **Piggy Bank Management:** Users can create, edit, or delete virtual piggy banks; progress is tracked in real time.
- **Dummy Bank Ledger:** Records all transactions to ensure transparency and safe experimentation.
- **Frontend Display:** Responsive interface shows balances, piggy bank statuses, AI suggestions, and chatbot interactions.
- **Backend Processing (Flask / FastAPI):** Handles requests, processes recommendations, and updates piggy bank data.