A Project Report on

"Translate and Summarize News"

Submitted to

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

for fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

By

Arati Karpe
Vijaya Kadam
Kishori Gaikwad

Under the Guidance
of

Ms. R. S. Dumne

(Department of Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MAHATMA GANDHI MISSION'S COLLEGE OF ENGINEERING
NANDED (M.S.)

Academic Year 2024-25

Certificate

This is to certify that the project entitled

"Translate and Summarize News"

being submitted by ***Ms. Arati Karpe, Ms. Vijaya Kadam, Ms. Kishori Gaikwad*** *to the Dr. Babasaheb Ambedkar Technological University, Lonere , for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by them under my supervision and guidance. The matter contained in this report has not been submitted to any other university or institute for the award of any degree.*

Ms. R. S. Dumne
Project Guide

Dr. A. M. Rajurkar
H.O.D
Computer Science & Engineering

Dr. G. S. Lathkar
Director
MGM's College of Engg., Nanded

ACKNOWLEDGEMENT

We are greatly indebted to our project guide **Ms. R. S. Dumne** for her able guidance throughout this work. It has been an altogether different experience to work with her and we would like to thank her for help, suggestions and numerous discussions.

We gladly take this opportunity to thank **Dr. A. M. Rajurkar** (Head of Computer Science & Engineering, MGM's College of Engineering, Nanded). We are heartily thankful to **Dr. G. S. Lathkar** (Director, MGM's College of Engineering, Nanded) for providing facility during the progress of the project and for her kind help, guidance, and inspiration.

Last but not least we are also thankful to all those who help directly or indirectly to develop this project and complete it successfully.

Arati Karpe

Vijaya Kadam

Kishori Gaikwad

[B-Tech-CSE A]

ABSTRACT

In today’s digital landscape, people encounter an overwhelming amount of information in multiple languages. India, with its linguistic diversity, presents a unique challenge in accessing and understanding regional news. To address this, the project titled “Translate and Summarize News” offers a unified, automated solution. It extracts, translates, and summarizes news content directly from uploaded image files. The system is a web application built using Flask (Python) for the backend. The frontend uses HTML, CSS, and JavaScript to ensure a responsive and user- friendly interface. Users can upload news article images in English, Hindi, or Marathi. Text extraction is done using Easy OCR, which is faster and more accurate than traditional Tesseract OCR. Once text is extracted, two processing workflows are supported: Extract then Summarize: Summarizes the text in the original language. Extract then Translate then Summarize: Translates text to English using the Google Translate API and then summarizes it. For summarization, the system uses IndicBART, a model by AI4Bharat trained for Indian languages.

It provides meaningful, concise, and readable summaries from the source or translated content. Users can easily copy, download, or clear the final output from the interface. This application integrates OCR, translation, and NLP seamlessly in one platform. It enhances accessibility to news across language barriers. The solution is ideal for students, researchers, journalists, and multilingual audiences. It also promotes digital inclusion by bridging the language gap. Future improvements could include offline translation, mobile app support, and additional language coverage. Overall, this project delivers a practical, scalable, and impactful tool for modern news comprehension.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter1

INTRODUCTION

In an increasingly interconnected and globalized world, access to timely and accurate news has become essential for individuals, organizations, and governments. News consumption is no longer confined to traditional newspapers or local broadcasts; instead, people rely heavily on online platforms to stay updated with current events happening around the globe. Despite this wide availability, significant challenges still hinder global access to news, particularly in terms of language and information overload. News articles published across different countries and regions are often in local languages, creating a language barrier for non-native speakers. While multilingual news portals and translation plugins exist, they are not always reliable, contextually accurate, or integrated with summarization features. Moreover, the sheer volume of articles produced daily can overwhelm readers, making it difficult for them to identify and retain the most important information from long and detailed content. The aim of this project, titled "Translate and Summarize the News", is to develop an intelligent system that automates the process of translating and summarizing online news articles. By using Natural Language Processing (NLP) techniques and modern machine learning models, the system enables users to read global news in their native language, while also providing a concise summary to help them grasp the core message quickly and efficiently. This dual-functionality solution addresses two major problems in news consumption: the language barrier and the time-consuming nature of reading full articles. With this tool, users can receive news from any region, regardless of the source language, and read a short, meaningful summary instead of sifting through large paragraphs.

The system architecture combines multiple technologies. It utilizes web scraping or APIs to fetch live news data, a translation engine (such as Google Translate API or pre-trained multilingual transformer models) to convert the content into a target language, and summarization techniques (like extractive or abstractive models) to produce brief versions of the original content. These technologies work together to provide an accurate, readable, and digestible news experience. The potential applications of this system are wide-ranging. It can benefit individuals in multilingual countries, international journalists, students conducting cross-cultural research, and professionals who need to monitor global events. It also promotes inclusivity, as people with limited literacy or time can access essential information in a simplified and familiar language. Overall, this project represents a practical and socially valuable application of artificial intelligence and natural language processing. It not only improves how people interact with information but also empowers them to overcome linguistic and cognitive barriers. In the long term, it could serve as a base for more advanced AI-driven tools in the media industry, such as personalized news recommendations, sentiment analysis, and multilingual news analytics. By providing a smart, fast, and user-friendly interface for reading news, this system redefines how information is consumed in a modern, diverse, and fast-paced world. It embodies the essence of technological innovation directed toward enhancing communication, accessibility, and understanding in global society.

1.1 Overview of Translation and Summarization

In today's digital age, where vast amounts of news content ar published online daily, users often face two major challenges: language barriers and information overload. To address these, the project "Translate and Summarize the News" combines two essential Natural Language Processing (NLP) techniques machine translation and text summarization to enhance how users access and consume news content. This section provides an overview of both components and how they contribute to the system.

- **Translation:**

Refers to the process of converting text from one language to another while maintaining its original meaning and context. With the rise of neural networks and deep learning, machine translation has evolved significantly, leading to the development of more accurate and fluent translation systems. This project utilizes machine translation to allow users to read news articles originally written in foreign languages in their own preferred language. Traditional methods such as rule-based and statistical machine translation have been largely replaced by Neural Machine Translation (NMT), which uses deep learning models to understand and generate translations more naturally. Tools like Google Translate API, MarianMT, and mBART have become standard in modern translation applications due to their ability to handle multiple language pairs effectively. These tools are pre-trained on vast multilingual datasets and are capable of delivering high-quality translations with multiple languages that based on user selection of language. That translation gives with the contextual accuracy. By integrating such a model into the system, users can access global news content regardless of the source language, making information more inclusive and universally accessible.

- **Summarization:**

On the other hand, is the technique of reducing a lengthy text to its most essential points, helping users quickly grasp the main idea without reading the entire article. There are two primary approaches to text summarization: extractive and abstractive. Extractive summarization selects important sentences or phrases directly from the original text without altering their structure. In contrast, abstractive summarization generates new sentences that represent the core ideas of the original content in a more natural and human- like way. This project focuses on using advanced abstractive summarization models such as BART, T5, or Pegasus, which are pre-trained on large datasets and capable of producing fluent and concise summaries. These models understand the structure and context of the content and can generate accurate summaries that are easy to read and informative. By applying summarization after translation, the system ensures that users receive a brief, digestible version of the news article in their chosen language, thereby saving time and improving comprehension.

Together, translation and summarization form the backbone of the system, enabling a seamless experience for users to access, understand, and engage with international news. The translation module breaks down language barriers, while the summarization module reduces the time and effort needed to process long articles.

1.2 Working Mechanism

The “Translate and Summarize the News” system is designed to streamline the way users consume global news by integrating multiple components that work together in a well-defined sequence. The core functionality of the system revolves around the automation of two critical tasks: translation of news content into the user's preferred language, and summarization of that content to present only the most relevant information. The working mechanism can be divided into several interconnected stages, each playing a specific role in achieving the overall goal of the application.

The process begins with news content acquisition. The system either fetches articles using public news APIs such as NewsAPI, or performs web scraping from popular news websites. These sources provide raw news data, typically in English or the native language of the news outlet. The system extracts the article’s main body, title, author (if available), and metadata such as publication date and source. This unstructured text content is passed to the next module for processing. Once the news article is retrieved, it enters the translation module. In this stage, the article is passed through a machine translation model. Depending on the implementation, this can be an external service like Google Translate API or an internal multilingual transformer-based model such as MarianMT or mBART. These models are capable of translating text from one language to another while preserving the meaning and fluency. The user selects their preferred language at the frontend, and the backend performs the translation accordingly. This ensures that users can understand news articles that were originally published in unfamiliar languages.

After translation, the text is forwarded to the summarization module. This component is responsible for condensing the translated text into a shorter, more informative version. It may use either extractive or abstractive summarization techniques, but in most cases, advanced abstractive models such as BART, Pegasus, or T5 are employed. These models use deep learning to comprehend the semantic structure of the text and generate a coherent summary that highlights the main points. The summarization module reduces cognitive load and saves time for the user by filtering out unnecessary details. The final output, which is a translated and summarized version of the original article, is then presented to the user via a clean and user-friendly interface. The frontend, developed using web technologies such as HTML, CSS, JavaScript, or frameworks like React, allows users to interact with the system effortlessly. Users can choose languages, select articles, and read summaries within seconds.

Behind the scenes, the system architecture follows a modular and scalable design. The backend, typically implemented in Python using frameworks like Flask or Django, handles data fetching, API communication, and model integration. NLP libraries and pretrained models from Hugging Face Transformers are often used to perform translation and summarization. The use of RESTful APIs ensures smooth communication between the frontend and backend. The user selects their preferred language at the frontend, and the backend performs the translation accordingly.

In summary, the working mechanism of the system involves four major steps:

1. Retrieving the latest news articles from online sources.
2. Translating the content into the user’s language.
3. Summarizing the translated text to provide the essence of the news.
4. Displaying the results through a responsive user interface.

This pipeline enables efficient, multilingual access to news content, making it a practical and impactful solution for global information consumption.

1.3 Application

The “Translate and Summarize the News” system has significant value in the modern digital world, where access to timely and accurate information across languages is essential. It addresses two major challenges: language barriers and the overwhelming volume of news content. By providing translated and summarized versions of articles, the system improves both accessibility and efficiency in news consumption. This solution is especially valuable for people living in multilingual regions or those interested in international affairs. Users who may not understand the original language of an article can easily get the information they need in their preferred language. The summarization feature is particularly useful for busy professionals, students, or researchers who want to stay informed but lack the time to read long articles. In addition to personal use, this system can be applied in media organizations, educational institutions, government agencies, and commercial platforms. It enables inclusive communication, supports content management, and facilitates real-time updates in different languages. Its adaptability also allows for integration into mobile apps, web platforms, and browser extensions.

Key Applications:

- **Multilingual Access**: Breaks down language barriers by translating news into the user’s chosen language.
- **Time Efficiency**: Summarizes long news articles, saving time for readers who need quick insights.

- **Media & Journalism**: Assists reporters, editors, and content curators in understanding foreign news sources.
- **App Integration**: Can be embedded in news apps or websites to enhance user experience and accessibility.
- **Government and Public Services**: Ensures citizens receive critical updates in their native languages during emergencies.
- **Corporate Use**: Useful for business analysts and professionals to monitor international trends and reports.
- **Inclusivity in Communication**: Promotes equal access to information for people with different language skills.

1.4 Challenges

The "Translate and Summarize News" project presented several technical, linguistic, and operational challenges throughout its development. One of the most prominent issues was ensuring accurate detection of the source language. Since the application supports multiple languages, any misidentification could result in incorrect or confusing translations. Different languages have unique syntactic and grammatical structures, which complicated the task of maintaining the meaning and tone of the original news during translation. Additionally, idioms, cultural references, and local terminology often do not have direct equivalents in other languages, which posed further difficulties in preserving the intended message. The summarization process added another layer of complexity. Simple extractive methods often pulled sentences directly from the original text, which did not always represent the main idea effectively. On the other hand, abstractive summarization, which involves generating new sentences to summarize the content, required sophisticated natural language understanding and generation capabilities. Balancing the need for a concise summary with the necessity of including all vital information was a constant challenge. Furthermore, some summaries either became too generic or focused on less important parts of the news article.

- **Translation Challenges:**

1. **Language Ambiguity** – Words or phrases can have multiple meanings.

2. **Loss of Context and Tone** – Difficulty in maintaining the original tone, sentiment, and intent.

3. **Regional and Cultural Terms** – Slang, idioms, and local expressions may not translate accurately.

4. **Grammar Differences** – Structural differences between source and target languages affect fluency.

5. **Incomplete or Misleading Translation** – Nuances and specific information may be lost.

- **Summarization Challenges:**

1. **Preserving Key Information** – Essential details may be excluded in the summary.

2. **Determining Relevance** – Identifying the most important parts of the news is complex.

3. **Bias Introduction** – Summaries may unintentionally alter the meaning or introduce bias.

4. **Handling Mixed Content** – News with data, quotes, and narrative requires careful extraction.

5. **Maintaining Coherence** – Summary should remain logically structured and readable.

Another major challenge was handling multilingual inputs. The system needed to manage different character sets, encoding standards, and font support to properly display and process languages like Hindi, Arabic, Mandarin, and others. Ensuring compatibility across various devices and platforms also required rigorous testing and optimization. The integration of translation and summarization components had to be seamless, but even minor issues in translation such as mistranslations or partial translations could affect the clarity and correctness of the final summary. From a performance standpoint, providing real-time or near-real-time results was difficult due to the processing demands of NLP models and third-party APIs. When dealing with lengthy articles or high traffic, response times could increase significantly, reducing the application's effectiveness. Additionally, the reliance on external APIs, such as Google Translate or Hugging Face models, introduced limitations in terms of usage quotas, API response delays, and occasional downtimes, all of which could hinder the user experience. Designing a user-friendly interface for a diverse audience was another important concern. Users needed to input text, select a target language, and view both the translated content and the summary in a clean and intuitive layout. Providing tooltips, error handling, progress indicators, and feedback mechanisms added complexity to the UI/UX design. Also, users unfamiliar with technical terms or features needed support and instructions to interact with the system effectively.

Maintaining the semantic relevance of summaries was especially challenging. There were instances where the summarizer output included grammatically correct but semantically weak or misleading statements. This posed a risk of spreading misinformation or losing the factual essence of the original news. To mitigate this, fine-tuning the models and validating the output quality was necessary but time-consuming. Furthermore, acquiring and using appropriate datasets was not straightforward. Publicly available datasets often lacked diversity, especially in less-spoken languages, and were biased toward certain topics or regions. This made it difficult to train and evaluate the system in a balanced and fair way. Creating or curating custom datasets required significant manual effort and validation. Lastly, ethical considerations also surfaced during development. News translation and summarization carry the risk of distorting facts or misrepresenting events, especially when AI models are involved. It was important to ensure the system's outputs were responsibly generated and that users were made aware of potential inaccuracies or limitations of the tool.

Chapter2

LITERATURE REVIEW

In recent years, the rapid growth of online news and digital journalism has created an urgent demand for systems capable of efficiently processing, translating, and summarizing news articles. With the vast amount of multilingual content available globally, automated translation and summarization tools are increasingly necessary for effective information dissemination. This literature review presents an overview of previous research and current trends in the areas of machine translation and text summarization, specifically in the context of news processing. These technologies aim to bridge linguistic barriers and present concise, relevant information to users in a multilingual world.

2.1 Machine Translation Techniques

Machine Translation (MT) has evolved from simple rule-based approaches to sophisticated deep learning models. Rule-based MT systems used a set of predefined grammatical and syntactical rules for each language. However, these were not scalable and often resulted in inaccurate translations. Statistical Machine Translation (SMT) then emerged, relying on statistical models based on the frequency and likelihood of word sequences from parallel corpora. SMT systems like Moses became popular for some time but often lacked contextual understanding (Koehn et al., 2007). The advent of neural machine translation (NMT) marked a significant improvement in translation quality. NMT models such as the Google Neural Machine Translation (GNMT), Transformer-based architectures, and open-source frameworks like OpenNMT (Klein et al., 2017) and Fairseq (Ott et al., 2019) offer context-aware, fluent translations by leveraging encoder-decoder frameworks and attention mechanisms. These models consider the entire input sentence, resulting in better handling of long-distance dependencies and improved fluency.

Recent research by Vaswani et al. (2017) introduced the Transformer model, which became the foundation of many advanced language models such as BERT (Devlin et al., 2019), mBART (Liu et al., 2020), MarianMT (Junczys-Dowmunt et al., 2018), and T5 (Raffel et al., 2020). These models can perform high-quality translation tasks by learning contextual relationships between words. MarianMT, for instance, supports multiple languages and has been trained specifically for translation, making it effective for translating regional languages such as Hindi and Marathi. Pre-trained models reduce the need for large labeled datasets and provide strong baselines for customization.

2.2 Multilingual Translation Challenges

Despite the progress in NMT, several challenges persist. One of the most significant issues is the handling of low-resource and mixed-language inputs. Hinglish (a blend of Hindi and English), for example, presents difficulties in terms of grammar, vocabulary, and syntax. The lack of standardized datasets for such language combinations further complicates model training and evaluation. Even high-resource languages suffer from domain mismatch and ambiguity in translation (Guzmán et al., 2019). Moreover, maintaining the semantic meaning and emotional tone of the original text remains a challenge in translation. Language-specific idioms, expressions, and cultural nuances are often lost or mistranslated by generic models. These limitations have driven research toward domain-specific translation systems and custom training of language models. Tokenization, contextual embeddings, and subword encoding techniques like Byte Pair Encoding (BPE) help alleviate vocabulary mismatch issues (Sennrich et al., 2016).

2.3 Text Summarization Approaches

Text summarization is broadly classified into extractive and abstractive methods. Extractive summarization identifies and selects the most important sentences or phrases from the original text, often using algorithms like TextRank (Mihalcea & Tarau, 2004), LexRank, and TF-IDF. These methods are relatively simple and computationally efficient but may lack coherence and readability. Hybrid techniques have also been explored, combining extractive and abstractive strategies.

Abstractive summarization, on the other hand, involves understanding the context and generating new sentences to capture the main ideas. This method resembles how humans summarize content. Recent models such as BART (Lewis et al., 2020), T5, and PEGASUS (Zhang et al., 2020) have achieved state-of-the-art performance in abstractive summarization tasks. These models are pre-trained on large datasets and fine-tuned for summarization, enabling them to produce coherent and concise summaries. Attention mechanisms, beam search decoding, and reinforcement learning-based tuning have further improved the output quality.

2.4 Integrated Translation and Summarization Systems

There is growing interest in developing integrated systems that combine both translation and summarization into a single pipeline. Such systems are particularly useful for global news platforms where content needs to be summarized and translated into multiple languages simultaneously. Research indicates that pre-processing steps such as sentence segmentation, language identification, and noise filtering can significantly improve the quality of both translation and summarization. Recent studies have proposed architectures where the input text is first translated into a target language and then summarized using a transformer-based model. Alternatively, some models directly summarize the content in the source language and then translate the summary. Both approaches have their benefits and limitations, and the choice depends on the application requirements and available resources. Joint models using multitask learning are being explored to handle both tasks more efficiently (Dong et al., 2015).

2.5 Real-World Applications and Tools

Several tools and platforms currently implement translation and summarization features. Google Translate, Microsoft Translator, and Amazon Translate provide real-time translation services across hundreds of languages. For summarization, platforms like Hugging Face offer pre-trained models such as T5 and BART, which can be deployed for custom applications. Academic and industry researchers have also explored the use of NLP pipelines for news aggregation, sentiment analysis, and multilingual content delivery. These systems enhance user experience by providing concise and localized news content. However, issues like content bias, ethical considerations, and misinformation detection remain areas of active research. Evaluating summarization using ROUGE scores (Lin, 2004) and BLEU metrics (Papineni et al., 2002) for translation has become standard practice.

2.6 Future Directions

The future of translation and summarization lies in the development of more accurate, context-aware, and ethically aligned systems. Improvements in multilingual pre-training, zero-shot translation, and low-resource language modeling are expected to broaden access to automated news processing tools. Additionally, the incorporation of reinforcement learning, knowledge graphs, and user feedback mechanisms could further refine these systems.

Research continues to focus on building datasets, evaluating model performance, and addressing ethical concerns such as fairness, privacy, and cultural sensitivity. With advancements in AI and increasing computational power, the integration of translation and summarization systems into everyday applications is likely to become more robust, scalable, and reliable. Open research challenges include better cross-lingual understanding, high-fidelity abstractive summarization, and real-time deployment on resource-constrained devices.

2.7 Evaluation Metrics and Benchmarks

Evaluating the effectiveness of translation and summarization models requires robust metrics. For translation, BLEU (Bilingual Evaluation Understudy) remains one of the most commonly used metrics, comparing machine-generated output with one or more reference translations (Papineni et al., 2002). Other alternatives like METEOR and TER offer complementary perspectives by incorporating synonymy, stem matching, and edit distance. In summarization, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics especially ROUGE-1, ROUGE-2, and ROUGE-L are used to measure the overlap of n-grams, word sequences, and longest common subsequences between generated and reference summaries (Lin, 2004). However, these metrics are limited in capturing fluency, coherence, and factual consistency, leading researchers to explore BERTScore and QuestEval, which rely on pre-trained language models for more nuanced evaluation. Benchmark datasets like CNN/DailyMail, Gigaword, XSum, and multilingual corpora like Europarl, Flores-101, and WMT provide standard testing grounds for both translation and summarization tasks.

2.8 Ethical Considerations and Bias Mitigation

Language models often inherit and amplify biases from the data they are trained on. Gender bias, cultural insensitivity, and misinformation propagation are significant concerns, particularly in news translation and summarization. Studies by Bender et al. (2021) and Sheng et al. (2019) highlight how these systems can misrepresent facts or reinforce stereotypes. To mitigate these risks, developers incorporate fairness-aware training objectives, adversarial debiasing, and human-in-the-loop verification. Additionally, efforts such as Explainable AI (XAI) frameworks are helping users better understand model decisions, enhancing trust and accountability in real-world applications.

2.9 Multimodal and Real-time Extensions

A growing trend is the integration of multimodal summarization combining text, images, and video to generate comprehensive news digests. Projects like VLP (Vision-Language Pretraining) and CLIP by OpenAI push the boundary of combining vision and language understanding, enabling summarization not just of text but entire multimedia news reports. For real-time deployment, lightweight versions of transformers like DistilBERT, TinyBERT, and MobileBERT are being used in mobile and web applications to offer on-the-fly translation and summarization, especially useful during live news events and social media streams. Multimodal and real-time extensions represent advanced capabilities that significantly enhance the processing and understanding of information, particularly in applications such as news translation and summarization. Multimodal systems integrate data from multiple sources or modes such as text, images, audio, and video to create richer and more comprehensive outputs. For instance, in news reporting, combining textual information with relevant images, videos, or audio clips can provide a more immersive and contextually complete experience for the audience. This integration allows systems to capture nuances that single-mode processing might miss, such as tone of voice, facial expressions, or visual cues, which are essential for accurate interpretation and effective communication.

Real-time extensions focus on delivering information as quickly as possible, often with minimal delay, which is crucial for time-sensitive fields like journalism and emergency response. Real-time translation and summarization systems aim to process incoming data instantaneously, enabling live news feeds, broadcasts, or social media updates to be accessible in multiple languages and concise formats simultaneously. This requires highly optimized algorithms and infrastructure capable of handling vast streams of data with low latency. The challenge lies in balancing speed with accuracy, as rushing the process may result in errors, while delays can reduce the relevance of the news being reported.

Chapter3

PROPOSED DETAILS

3.1 Introduction

In the digital age, the volume of news and information shared online is vast and ever-increasing. With thousands of news outlets publishing content in various languages and formats, users are overwhelmed by both the quantity of available information and the linguistic barriers that prevent them from accessing content beyond their native language. Moreover, the time constraints of modern life make it impractical for users to read through long and detailed news articles to extract the core message or insights. This has created a pressing need for systems that can intelligently process large volumes of news content, translate it across languages, and summarize it into meaningful and digestible formats. The proposed system addresses this need by providing an automated translation and summarization solution tailored for news articles. The system integrates multiple Natural Language Processing (NLP) technologies to detect the language of the original article, translate it into a target language chosen by the user, and finally, summarize the content into concise, informative snippets. By doing so, the system enhances news accessibility, improves reading efficiency, and bridges linguistic gaps, enabling users to understand global and local events irrespective of the original reporting language.

One of the key motivations behind this project is the growing diversity of language speakers on the internet and the lack of accessible content for many non-English speakers. While machine translation services have evolved considerably, they often lack the contextual understanding required for accurate and culturally appropriate translation of journalistic content. Similarly, while summarization algorithms exist, they often fall short in terms of fluency and coherence, especially when applied to translated text. This system tackles these limitations by carefully integrating state-of-the-art NLP models including transformer-based language models for both translation and summarization into a coherent and user-friendly workflow. Another important aspect of this system is its real-world applicability. From journalists researching international stories, to students comparing political narratives across borders, to the general public wanting to stay updated with current events in their preferred language the potential users of this system are diverse. The system provides a web-based interface that is both intuitive and scalable, allowing it to be used by there a individuals with varying degrees of technical proficiency. Overall, this project proposes an innovative, practical, and scalable solution for news translation and summarization. It not only helps in reducing the time required to consume large volumes of information but also democratizes access to global knowledge by eliminating language as a barrier. Through a combination of modern AI tools and thoughtful system design, this project aims to contribute meaningfully to the fields of media, communication, and multilingual information processing.

3.2 Objectives

The core aim of this project is to develop an intelligent system that can seamlessly translate and summarize news content from any language into the user's preferred language in a simplified, accurate, and efficient manner. The primary objective of this project is to develop an intelligent system that can automatically translate and summarize news articles across multiple languages. It aims to enhance accessibility, reduce reading time, and make global news understandable for diverse users through advanced NLP techniques. The system leverages advanced Natural Language Processing (NLP) and Machine Learning (ML) techniques to automate these tasks, which are traditionally manual and time-consuming. The objectives are described in detail as follows:

3.2.1 To Automate Multilingual Translation of News Content

The goal of automating multilingual translation is to break language barriers and make news accessible to a broader audience. This system uses the Google Translate API to instantly translate extracted text from one language into another, including widely spoken languages like English, Hindi, and Marathi. By doing this automatically, the system removes the need for users to copy and paste content into translation tools manually. This is especially helpful for users who are not proficient in the original language of the news article. The translation process preserves the original meaning, context, and tone of the news, ensuring that users get an accurate and culturally appropriate version of the content. The system handles complex sentence structures and domain-specific terms using Google's advanced neural translation models. Additionally, the translation component works seamlessly with the OCR output, ensuring that the extracted text flows directly into the translation step without interruption. The multilingual support makes it suitable for diverse user groups across regions. Users can select their target language from a dropdown interface, making the tool flexible and user-friendly. The use of automated translation also enables rapid access to global news, bridging information gaps. This feature is essential for keeping users informed in their preferred language and promotes inclusivity. Overall, the system's translation module empowers users to read and understand news from different linguistic backgrounds quickly and effortlessly.

3.2.2 To Build a User-Centric Web-Based Interface

The system is designed with a strong focus on building a user-centric, web-based interface that ensures accessibility and ease of use for all types of users. This interface acts as the bridge between the user and the system’s core functionalities like OCR, translation, and summarization. Users can upload an image containing news text through a simple and clean upload form. Once uploaded, the system automatically processes the image and displays extracted text, translation, and summarized content in real time. The interface is responsive, meaning it works seamlessly on desktops, tablets, and mobile devices. Design simplicity is prioritized so even non-technical users such as students, journalists, or the general public can navigate and use the application easily. Clear buttons, form validations, and instant feedback enhance usability. The layout supports multilingual display, helping users understand news in their preferred language. Interactive elements guide users through each step of the process without confusion. Error handling is built in, so if anything goes wrong, users receive clear instructions. The user interface also ensures data privacy and does not store unnecessary user data. A clean, intuitive frontend boosts user engagement and encourages regular usage. The goal is not just functionality, but also user satisfaction and smooth experience.

3.2.3 To Enable Real-Time or Near Real-Time Processing

The system is designed to support real-time or near real-time processing to deliver quick and efficient results to users. In today's fast-paced digital world, immediate access to translated and summarized news is essential, especially for breaking news or live updates. To meet this need, the project incorporates lightweight and optimized models for each task: EasyOCR for fast text extraction, Google Translate API for instant translation, and IndicBART for summarization. These models are integrated in a streamlined pipeline where each step automatically triggers the next, minimizing user wait time. The Flask backend processes requests quickly, while frontend JavaScript provides instant visual feedback. Input validation and asynchronous processing ensure smooth performance even when the system handles multiple users. The real-time nature of the system allows journalists, students, and general users to access news content in their preferred language within seconds. Summarization is designed to be concise yet informative, so users can quickly grasp the main ideas without reading full articles. Error handling and fallback mechanisms are included to maintain flow even during network or API delays. By reducing manual effort and time, the system increases productivity and enhances accessibility. The architecture is optimized for low latency and smooth user interaction. Ultimately, the goal is to deliver meaningful news insights with minimal delay, improving the overall user experience.

3.3 Scope of the Project

The scope of this project revolves around building an intelligent, NLP-based web application capable of translating and summarizing news articles written in multiple languages. In the current digital landscape, the internet is flooded with massive volumes of news content published every minute. While this creates opportunities for global awareness, it also poses a challenge for users who cannot access content in unfamiliar languages. Regional and international language barriers prevent many individuals from understanding important news updates beyond their native tongue. This system is designed to eliminate that limitation by offering accurate, automated translations of news articles into the user's preferred language. By using machine learning models and APIs such as Google Translate or MarianMT, the system maintains not only linguistic accuracy but also preserves context, tone, and relevance. Additionally, the modern reader often lacks the time to read full news stories due to fast-paced lifestyles. Therefore, the system also integrates intelligent summarization capabilities, leveraging models like IndicBART to reduce lengthy articles into brief, meaningful summaries. These summaries allow users to absorb the essential information quickly, without missing out on critical points. This makes the tool especially useful for students, professionals, journalists, and researchers who require efficient, multilingual access to global events. The project's scope extends beyond mere functionality—focusing on user accessibility, real-time performance, and integration of advanced AI techniques. It serves as a bridge across linguistic divides while enhancing the convenience of news consumption in a multilingual society. Overall, the project contributes to inclusive, fast, and smart news access in today's information-rich world. The scope of this project is to develop an AI-based system that can automatically gives translation based on user preference. These web application gives accurate and user choice translate and summarize news articles across multiple languages. It aims to break language barriers and reduce reading time by providing quick, accurate, and user-friendly access to news content.

3.3.1 Translation of Multilingual News Content

Translation of multilingual news content is a key feature of this system, designed to help users understand news in languages they do not speak. The system supports translation from languages like Hindi, Marathi, Bengali, and even international languages like French into user-selected target languages such as English. It uses powerful pre-trained models like MarianMT and Google Translate API to carry out the translation efficiently and accurately. These models are capable of understanding the grammatical structure, tone, and context of the source text. As a result, the translated output maintains the original meaning while sounding natural and fluent in the new language. The system goes beyond literal word-for-word translation, preserving the essence of the news story. This is particularly beneficial for users in multilingual countries like India, where access to regional news is vital. It also aids researchers, journalists, and students who rely on cross-lingual content. The automatic process eliminates the need for manual translation, saving time and effort. The translation is seamlessly integrated into the overall workflow after OCR extraction and before summarization. This ensures a continuous, smooth user experience. Overall, the multilingual translation feature enhances accessibility, understanding, and communication across diverse linguistic boundaries.

3.3.2 Summarization of Long News Articles

One of the most essential components of this project is the summarization of long-form news content. With the rapid growth of digital journalism and 24/7 news publishing, users are overwhelmed with information. Reading full-length news articles from multiple sources becomes time-consuming and exhausting. To address this issue, the system includes a powerful text summarization module that condenses detailed articles into short, informative, and easily readable summaries. This module supports two summarization techniques extractive and abstractive. Extractive summarization identifies and selects the most important sentences directly from the original text, while abstractive summarization rewrites the content using simpler and shorter phrases, ensuring clarity and coherence. This dual-method approach ensures flexibility and adaptability across different news types and formats. For instance, when exact wording is important, extractive summarization helps retain original phrasing. On the other hand, abstractive summarization is ideal for generating clean summaries that read like human-written content. The system is designed using advanced machine learning and Natural Language Processing (NLP) models such as Hugging Face Transformers (like IndicBART or mBART), which have been pre-trained on massive multilingual datasets. These models enable accurate interpretation and content restructuring without losing the core meaning. The summarization feature is especially helpful for students preparing for exams, journalists cross-referencing facts, and researchers comparing coverage across multiple platforms. It allows readers to grasp essential points from large articles at a glance, improving time efficiency and reducing cognitive load. Furthermore, summarization also helps users quickly identify biased or misleading headlines by revealing the full context of a story. Traditional algorithms like TextRank can also be employed for simple extractive summaries. The integration of this summarization functionality enhances the overall user experience, making the platform practical for daily news consumption and comparative media analysis.

3.3.3 User-Friendly and Interactive Web Interface

The system integrates a user-friendly and interactive web interface designed to provide an intuitive experience for users of all backgrounds. This interface serves as the main point of interaction between the user and the system's core functionalities, such as uploading images, viewing extracted text, selecting target translation languages, and displaying the final summarized output. Developed using standard web technologies like HTML, CSS, JavaScript, and enhanced with Bootstrap, the interface is clean, responsive, and mobile-friendly, ensuring smooth access across a range of devices including smartphones, tablets, and desktops. Users can easily paste or upload news content through a simple form. Drop-down menus allow users to select the input and output languages quickly, and clearly labeled sections show the original, translated, and summarized versions of the content. The navigation flow is deliberately kept minimal and straightforward, guiding users from input to output in a few clicks without unnecessary complexity. This helps reduce user frustration and improves task efficiency. Error messages and validation prompts ensure that incorrect input is flagged instantly, contributing to a seamless experience.

Moreover, the interface hides all the backend technical processes like OCR, NLP model execution, and API communication, allowing users to focus purely on the results. All interactions are performed in real time or near real time, with loading indicators and feedback messages that inform users of progress. Even those with no technical knowledge can effectively use the platform to translate and summarize news content. The inclusion of interactive elements such as preview sections, text areas, and real-time output updates further enhances usability. The thoughtful design ensures the tool is not just powerful but also accessible, making it suitable for a wide range of users students, educators, researchers, journalists, and the general public.

3.4 System Architecture

System architecture of the project "Translate and Summarize News" represents the comprehensive design and structural layout of the entire system, which manages the flow of data from multiple news sources to the end user in a translated and summarized format. It defines how different components such as input collection modules, translation engines, summarization tools, storage units, and user interfaces interact with one another to perform the desired operations in a logical, efficient, and scalable manner. This architecture is crafted to handle real-world challenges such as multilingual news input, high-volume text processing, and fast delivery of summarized content. By utilizing a layered and modular design, the system separates responsibilities across components, ensuring ease of maintenance, development, testing, and potential integration with external APIs or services. At its core, the architecture combines several advanced technologies, such as Natural Language Processing (NLP), machine translation models, and text summarization algorithms, to automate the content transformation process. These are encapsulated within the processing engine that acts as the "brain" of the application. The system ingests raw news data either scraped from the web or fetched from APIs then detects the original language, translates it into the preferred target language, and generates a concise summary using either extractive or abstractive methods. Further more, the architecture is designed to support real-time processing capabilities and multi-device compatibility.

3.4.1 Backend and Data Storage

The backend of the "Translate and Summarize News" project serves as the central control unit that handles data processing, logic execution, and interaction with databases. It is responsible for receiving requests from the frontend, managing the flow through translation and summarization modules, and ensuring that all data is processed securely and efficiently. This layer acts as the bridge between the user interface and the core language processing logic, ensuring that user-selected preferences (such as language or category) are accurately interpreted and applied.

Typically, the backend is implemented using a server-side language such as Python (Flask or Django), Node.js, or PHP, and it exposes APIs to facilitate communication between different system modules.

The data storage component is closely integrated with the backend and plays a vital role in maintaining application state, historical data, and user-specific information. It stores various forms of data including the original news content fetched from online sources, its translated versions, and the summarized outputs. Additionally, it stores user preferences, system logs, feedback data, and processing metadata such as language tags, timestamps, and article categories. A structured database like MySQL or PostgreSQL may be used for relational data, while NoSQL databases like MongoDB or Firebase are ideal for flexible and scalable document-based storage. The backend ensures that all this data is properly indexed and retrievable, supporting fast querying and updates. Moreover, the system architecture includes provisions for data consistency, caching, and optimization. For example, when the same news article is requested again, the backend can retrieve it directly from the database instead of repeating the translation and summarization process significantly reducing processing time. Secure data handling practices are also enforced in the backend, such as input validation, encryption for sensitive data, and authentication for accessing user-specific records. The backend is thus critical to ensuring reliability, efficiency, and the smooth operation of the entire application, acting as both the logic engine and the data manager of the system. The backend receives user inputs such as language preferences and article links, and it orchestrates multiple processes, including language detection, translation using APIs (like Google Translate or MarianMT), and summarization using models like BART.

3.4.2 System Workflow Diagram

The System Workflow Diagram is a critical component in illustrating how the "Translate and Summarize News" system functions internally. It represents the end-to-end data flow, outlining the interaction between various modules and helping developers and users alike visualize the logical architecture of the system. The process begins when the user uploads an image of a news article or inputs raw text through the frontend interface. If the input is an image, the system initiates the OCR (Optical Character Recognition) module, typically powered by EasyOCR, to extract readable text from the image. Once the raw text is available, it is passed to a language detection module, which identifies the source language of the input content. This step is essential to determine if translation is required. If the content is already in the user's target language, the translation step is skipped. However, if the source and target languages differ, the translation module leveraging models like MarianMT or the Google Translate API is activated. This module translates the extracted or entered text into the preferred language selected by the user via the web interface. After translation, the processed text moves to the summarization module, which uses AI-based models such as IndicBART or BART for multilingual summarization. This module condenses lengthy articles into shorter summaries using either extractive techniques (selecting key sentences) or abstractive methods (generating new, concise text). The choice between summarization styles may depend on the model's capabilities and user preferences. The final output, which includes the original text, translated version, and summarized content, is stored temporarily or permanently in a backend database or session variable. This ensures session-based user access and enables the user to revisit or download the content. The frontend built using HTML, CSS, JavaScript, and optionally Bootstrap then dynamically renders all results in an organized layout, making it simple for users to compare the original and processed outputs.

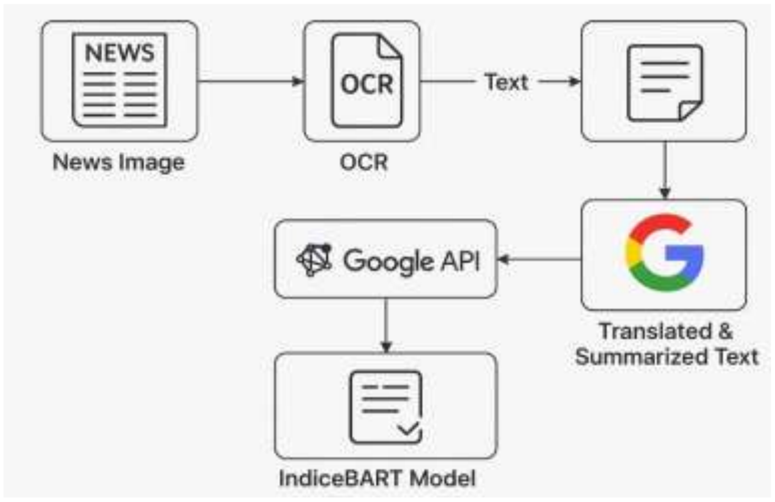


Fig. 3.4.1: System Workflow Diagram

This seamless integration between backend logic and frontend interaction ensures a user-friendly experience. The workflow diagram also supports error handling at various stages, such as invalid image format, failed OCR recognition, or API rate limits during translation. Furthermore, it emphasizes modularity, allowing future upgrades such as speech input, additional language support, or integration with news APIs. Overall, the System Workflow Diagram functions as both a technical blueprint and communication tool, simplifying the development process and aligning all components towards the core objective: delivering fast, accessible, and meaningful news content to users in their chosen language and format.

This seamless integration between backend logic and frontend interaction ensures a user-friendly experience. The workflow diagram also supports error handling at various stages, such as invalid image format, failed OCR recognition, or API rate limits during translation. Furthermore, it emphasizes modularity, allowing future upgrades such as speech input, additional language support, or integration with news APIs. Overall, the System Workflow Diagram functions as both a technical blueprint and communication tool, simplifying the development process and aligning all components towards the core objective: delivering fast, accessible, and meaningful news content to users in their chosen language and format.

3.4.3 Performance Optimization

Performance optimization is a crucial component in the architecture of the "Translate and Summarize News" system, ensuring the platform remains efficient, fast, and scalable even under heavy load. As the application involves complex real-time tasks like OCR, translation, and summarization optimization becomes essential to maintain responsiveness. One of the key techniques used is asynchronous processing, which allows different modules to execute simultaneously. For example, while one news article is undergoing translation, another can be summarized in parallel. This reduces overall wait time and improves throughput. In addition to asynchronous tasks, the system incorporates caching mechanisms. Popular translation pairs, frequently accessed summaries, or repetitive requests are temporarily stored in cache memory, allowing instant responses and minimizing unnecessary recomputation.Caching is employed both on the frontend (to store viewed content) and the backend (for storing API results, OCR outputs, or processed data), helping reduce server load and speeding up user interactions. Another strategy used is batch processing, where multiple articles are grouped and processed together instead of one at a time. This reduces the overhead of repeated model initialization and improves overall processing efficiency, especially useful when handling multiple news sources or large datasets. The system also opts for lightweight transformer models, like MarianMT or distilled versions of BART, which provide good accuracy while consuming fewer computational resources. Moreover, APIs and scripts are optimized to avoid redundant operations, such as repeated language detection or unnecessary I/O tasks. Where possible, operations like summarization are executed on translated content already in memory, avoiding re-loading or re-parsing steps. The backend is built to be stateless and modular, allowing horizontal scaling if needed, where more instances of the server can be spun up during high demand. API rate-limiting and timeout handling are implemented to avoid overloading external services and to provide fallback options in case of delays. Techniques like minification and compression of frontend resources (JS, CSS) further improve load times and reduce bandwidth usage. Combined, these optimization practices ensure the system delivers a smooth, fast, and scalable user experience, crucial for real-time news processing applications. The goal is not just performance under normal conditions, but consistent reliability during peak usage, enabling the application to serve many users effectively without slowdowns or failures.

3.5 Technology Stack

The technological stack refers to the combination of programming languages, frameworks, libraries, APIs, and tools used to build and operate the “Translate and Summarize News” system. Since the project integrates multiple functionalities such as data fetching, language translation, text summarization, and user interaction it requires a robust, modular, and scalable set of technologies. The chosen stack ensures seamless processing of multilingual content and smooth interaction between different components of the system. This project primarily utilizes Python for backend development due to its rich ecosystem of natural language processing (NLP) libraries and support for machine learning models. The frontend is developed using standard web technologies like HTML, CSS, and JavaScript to create an interactive and user-friendly interface. RESTful APIs are implemented to allow communication between the frontend and backend layers. For language translation and summarization, APIs and open-source models from Google Translate, Hugging Face, and other NLP tools are integrated.

3.5.1 Frontend Technologies

The frontend of the "Translate and Summarize News" system serves as the user-facing interface through which users interact with the application. Its main objective is to offer a clean, intuitive, and responsive environment that makes it easy to browse news articles, select preferred languages, and speeding up user interactions. Another strategy used is batch processing, where multiple articles are grouped and processed together instead of one at a time.

The interface is designed to be accessible across various devices including desktops, tablets, and mobile phones, ensuring a smooth user experience regardless of screen size. To build the frontend, standard web technologies such as HTML5, CSS3, and JavaScript are used. HTML5 is responsible for structuring the webpage, organizing components such as menus, article sections, and buttons. CSS3 handles styling, defining the visual appearance including layout, fonts, colors, and animations.

JavaScript plays a crucial role in adding dynamic behavior to the application—such as handling user inputs, loading news content asynchronously via APIs, and updating the UI without needing a full page reload. Additionally, frontend frameworks like Bootstrap or Tailwind CSS may be used to create mobile-friendly, responsive designs with minimal custom code. For enhanced user interactivity, the frontend also includes functions like language selection, content toggle between original and translated articles, and real-time display of summarized text. These features improve usability and allow users to quickly access key information without navigating multiple pages. The frontend communicates with the backend through RESTful APIs, sending requests for news content, translations, and summaries. Overall, the frontend is developed with an emphasis on simplicity, responsiveness, and speed, making it easy for users to consume multilingual news content efficiently.

3.5.2 Backend Technology

The backend of the "Translate and Summarize News" system plays a critical role in managing the core processing logic, API communication, and data handling operations. It acts as the engine that powers the entire system by accepting user requests, processing data using various NLP techniques, and returning the translated and summarized news. The backend is primarily developed using Python, one of the most powerful and widely used languages in natural language processing and artificial intelligence. Python's simplicity, large standard library, and strong community support make it ideal for implementing the logic behind translation, summarization, and news fetching. To manage server-side operations and build RESTful APIs, Flask or Django frameworks are used in the backend.

These frameworks handle routing, processing user requests, and sending the appropriate responses back to the frontend. For example, when a user selects a news category and language, the backend uses APIs to fetch relevant news articles, processes the data through translation and summarization modules, and then sends the results to be displayed in the user interface. Another crucial aspect of the backend is its ability to communicate with external services and integrate advanced Natural Language Processing (NLP) capabilities. The backend uses Python libraries such as requests to interact with third-party APIs like Google Translate or Hugging Face Transformers. It also uses libraries like langdetect for language identification, nltk or spaCy for preprocessing, and models like BART or T5 for abstractive summarization.

3.5.3 Other Tools and Libraries

In addition to the core translation and summarization components, your project integrates several supportive tools and libraries that enhance the functionality, performance, and efficiency of the system. One of the foundational tools is Requests (Python library), which is used to make API calls to external services such as news APIs, translation engines, or any public web endpoint. This library allows your backend to interact with external data sources in a simple and reliable way. For handling asynchronous or concurrent operations especially when fetching or processing multiple news items Python's asyncio or threading modules may also be utilized to improve the response time and performance of the system. Another essential tool is BeautifulSoup or Newspaper3k, which is used for web scraping and content extraction when the system needs to gather news directly from online sources instead of APIs.

Beautiful Soup is a powerful HTML parser that helps clean and extract meaningful text from messy or tag-filled web pages. Newspaper3k, on the other hand, provides a high-level interface to download and parse articles with minimal effort, automatically handling tasks like author detection, keyword extraction, and summary generation (basic). These tools allow the system to remain flexible and not rely entirely on paid or limited third-party APIs for data collection. Flask is a lightweight micro-framework that allows developers to create flexible APIs quickly and efficiently, while Django is a more robust, it acts as the engine that powers the entire system by accepting user requests, processing data using various NLP techniques, and returning the translated and summarized news. The backend is primarily developed using Python, one of the most powerful and widely used languages in natural language processing and artificial intelligence. It will the all technologies and various method are responsible to the it was so many queries and difficulties .

For language detection, lightweight yet effective libraries such as langdetect, langid, or Facebook's fastText are used to identify the original language of a news article before initiating the translation process. These tools help the system intelligently decide which language model or translation service to invoke. Additionally, text preprocessing tools like re (Regular Expressions), SpaCy, and NLTK are used to clean and prepare the text by removing special characters, stop words, and irrelevant symbols. Together, these supportive tools and libraries enable the system to perform its tasks efficiently and accurately, ensuring the processed news content is ready for translation, summarization, and user display with minimal manual intervention.

3.6 Functional Modules

The proposed system is divided into several functional modules, each playing a crucial role in achieving the overall objective of translating and summarizing news content provided either as text or image. These modules are designed to work cohesively while maintaining modularity, scalability, and a user-friendly experience. The first and foundational component of the system is the User Authentication Module, which handles user registration and login. It is implemented using JSON file storage instead of a traditional database, which simplifies deployment and makes it lightweight for initial use. This module includes input validation using regular expressions to ensure that usernames, passwords, and email addresses meet security and formatting criteria. This provides a basic yet effective security mechanism, allowing personalized access to users and ensuring that each user session is tracked properly throughout their interaction with the system. Next is the Image Upload & Preview Module, which allows users to upload an image file containing a news article or printed material. The interface offers real-time image preview, providing instant feedback that the image has been uploaded correctly. This improves the overall user experience by confirming the input source before any processing takes place. Supported image formats typically include JPG, PNG, and JPEG, and the preview feature helps ensure users do not mistakenly upload irrelevant or unreadable images. Once an image is uploaded, the OCR (Optical Character Recognition) Module is triggered. This module leverages pytesseract, a Python wrapper for the Tesseract OCR engine, to extract text from the image. The module first processes the image using the Python Imaging Library (PIL) to improve clarity, contrast, and resolution. Once pre-processed, the image is passed to pytesseract, which scans and converts the embedded various a text into machine-readable format.

Following the OCR step is the Translation Module, which is responsible for converting the extracted or manually entered news content into the user's selected language. This module uses the googletrans API, which provides access to Google Translate's vast multilingual capabilities. The language selection is facilitated through a dropdown menu in the user interface. Once a language is selected, the system automatically detects the input language and translates the content into the target language. This ensures the content is not only accessible to users in their native language but also retains the original meaning and context. The core feature of the system is handled by the Summarization Module, which employs the IndicBART (Multilingual BART) model from Facebook AI via the Hugging Face Transformers library. IndicBART is a powerful neural network capable of abstractive summarization, meaning it generates new, coherent sentences that encapsulate the meaning of the original content. This is especially beneficial when dealing with lengthy or verbose news articles. The summarization output is not merely a truncated or highlighted version of the input but a linguistically fluent and semantically accurate summary, available in over 50 supported languages including low-resource languages like Hindi and Marathi.

The system also includes a well-designed User Interface (UI) Module, which ties all functionalities together. Developed using HTML5, CSS3, and JavaScript, the UI is clean, intuitive, and responsive. It includes clear input fields, progress feedback for each processing step, error handling, and output boxes for both translated and summarized text. The interface has been crafted to ensure accessibility even for users with limited technical experience, making it suitable for general public use, students, or researchers interested in multilingual news analysis.

Each module in this system is developed with a focus on modularity, meaning they can be independently maintained or upgraded in the future. Whether it's adding more languages, switching translation engines, or integrating more advanced NLP models, the system is built to scale. Together, these functional modules enable a streamlined and automated process for accessing, understanding, and summarizing news content in multiple languages, directly from text or image input.

Chapter 4

IMPLEMENTATION

4.1 Introduction

The implementation of the project titled "News Translation and Summarization" involves the development of a web-based intelligent system capable of processing news articles in various languages, translating them into a common language (typically English), and summarizing the translated content into a concise, readable format. The goal of this implementation is to make global news more accessible and digestible, especially in a multilingual society. The core idea behind this project is rooted in Natural Language Processing (NLP) and Machine Learning (ML). Recent advancements in transformer-based architectures have made it possible to perform accurate language translation and abstractive summarization. The implementation integrates these advancements to provide an end-to-end workflow that starts from user input and ends in a meaningful, summarized translation of the original news. To begin with, the system allows users to input or upload news text in any language. This is followed by a language detection process which determines the source language using tools like langdetect. Once the language is identified, the system employs either API-based translation services (like Google Translate or Deep Translator) or pre-trained transformer models (such as MarianMT) to translate the content into English or any other target language selected by the user. The translated content is then preprocessed to make it suitable for the summarization model. This includes removing noise such as unnecessary punctuation, numbers, and symbols, and breaking the content into structured tokens. Preprocessing also involves standardization like lowercasing and removing stopwords, ensuring that the summarization model focuses only on the meaningful parts of the text.

Following this, the summarization phase is performed using transformer-based models such as BART (Bidirectional and Auto-Regressive Transformer) or T5 (Text-To-Text Transfer Transformer). These models have been trained on large text datasets and are capable of generating human-like summaries that are not just a collection of sentences from the original, but rephrased content that conveys the same meaning more concisely. The final output which includes the original text, the translated version, and the summary is then presented to the user through many clean and intuitive front-end interface built using HTML, CSS, and optionally JavaScript or frameworks like React for better user experience. The backend, powered by Flask (a lightweight Python web framework), handles all the processing and model execution. Moreover, the application is designed in a modular way, which makes it scalable and easy to improve. For example, more languages can be added in the future, or advanced custom-trained summarization models can be integrated for better accuracy. In summary, the implementation of this project reflects a practical application of modern NLP technologies. It showcases how the power of machine translation and summarization can be harnessed to break language barriers and present complex information in a simplified form. This system has potential use cases in journalism, education, government communication, and multilingual media services.

4.2 Project Structure Overview

The News Translation and Summarization System has been implemented as a modular and scalable web application using the Python Flask framework, which offers a lightweight yet powerful structure for integrating both frontend and backend components. The project's folder structure is carefully organized to separate the user interface from the core logic and processing modules. This separation of concerns ensures a clean architecture and supports future extensibility and debugging. At the root level, the project directory contains all essential components for the app to function. These include Python script files such as app.py and translator.py, a requirements.txt file for managing dependencies, a README.txt file for usage guidance, and subfolders named static, templates, and __pycache__. Each of these elements plays a crucial role in the overall working of the system. The static/ directory is responsible for holding all the static resources used by the application. These include CSS files for styling, JavaScript files for interactivity, and images or logos for branding or visual appeal. By storing static content separately, the application ensures faster loading times and better organization of design resources. It also allows frontend developers to independently manage the visual design without interfering with backend logic.

The templates/ folder houses the HTML templates used to construct the web pages. In Flask, HTML templates are rendered dynamically using the Jinja2 engine. The primary file in this folder is index.html, which forms the main user interface of the application. It presents a clean and interactive layout where users can input raw news text, select a language, and receive translated and summarized outputs. It contains input fields, dropdown menus, and display areas designed with responsiveness and accessibility in mind. This approach ensures that even users with minimal technical knowledge can easily use the system. The app.py file is the central script of the backend and is responsible for running the web application. It initializes the Flask app, defines all URL routes, and connects the frontend to backend functions. The most significant routes include '/', which renders the homepage, and '/summarize', which handles form submission. When a user enters news content and submits it, the app captures the request, extracts the input text and language, and forwards it to the summarization engine for processing. Once the results are obtained, they are passed back to the frontend for display.

All core language processing is encapsulated in the translator.py script. This file handles three critical operations: language detection, translation, and summarization. The system uses the langdetect library to automatically determine the input language. It then uses the deep-translator package or Google Translate API to convert the news text into English or another desired language. The translated content is passed through a series of NLP preprocessing steps (e.g., lowercasing, stopword removal, tokenization), before being fed into a summarization model such as BART, PEGASUS, or T5 from Hugging Face's transformers library. These state-of-the-art models have been trained on massive corpora and can perform abstractive summarization, meaning they generate a new text that conveys the same meaning in a more concise way, rather than simply extracting sentences from the original. The requirements.txt file is crucial for environment setup. It lists all Python packages required for the application to run smoothly. These include Flask, transformers, torch, deep-translator, langdetect, and possibly others. Users can easily set up the development environment by running pip install -r requirements.txt, making the application more portable and developer-friendly. Additionally, the README.txt file offers an overview of the application and setup instructions. It may include the purpose of the project, steps to run the app locally, and guidance on which versions of Python or libraries to use.

The _pycache_/ directory is auto-generated by Python during runtime and stores compiled bytecode of the .py files. Although it is not manually edited or used directly in development, it helps speed up subsequent executions of the program. This logical flow is divided across different files in a modular fashion, enabling easy updates, replacements of components (such as switching to another translation API or summarization model), and future improvements such as multilingual output.

Functionally, the system operates in a seamless pipeline. A user accesses the web interface and submits a news article in any language. The backend detects the language and performs translation. The translated text is then preprocessed and summarized. Finally, the translated and summarized versions are displayed clearly on the webpage. This logical flow is divided across different files in a modular fashion, enabling easy updates, replacements of components (such as switching to another translation API or summarization model), and future improvements such as multilingual output, speech synthesis, or even batch processing. In conclusion, the project is a well-structured application demonstrating the integration of Natural Language Processing, machine translation, and deep learning summarization models within a user-friendly web platform. The thoughtful file organization and modular design not only enhance maintainability and performance but also showcase best practices in software development and AI system integration. This implementation serves as a strong foundation for future innovations in multilingual content delivery and news automation.

4.3 Backend Implementation

The backend implementation of the News Translation and Summarization System is structured to support real-time, multi-stage processing of news articles using intelligent Natural Language Processing (NLP) models. It is designed using a micro-modular architecture, where each functionality OCR, language detection, translation, summarization, and response delivery is treated as a separate module. This modular approach ensures that each task can be independently developed, maintained, and scaled. The backend is developed in Python, a language well-suited for AI/ML tasks, thanks to its extensive ecosystem of NLP libraries such as Hugging Face Transformers, NLTK, and spaCy. The Flask web framework serves as the backbone of the application, providing lightweight and flexible routing, session handling, API integration, and templating functionality.

Each module interacts through well-defined function calls or Flask routes, making it easy to trace and debug specific tasks. For instance, the OCR module uses EasyOCR to extract text from uploaded images, storing the result in server memory. The translation module then uses pre-trained models like MarianMT or the Google Translate API to convert the extracted text into a user-specified language. The translated text is passed to the summarization module, which leverages models like IndicBART for abstractive summarization or TextRank for extractive methods. Flask routes handle these operations sequentially or in parallel using background jobs or asynchronous threads to reduce latency.

To handle user interactions, the backend maintains session variables and uses form validation to ensure data integrity. Security is also integrated using form tokens and file validation techniques to prevent malicious uploads. Caching mechanisms, such as in-memory storage (e.g., Redis or Python's functools.lru_cache), are optionally used to store frequently processed results for fast retrieval. Error handling and logging systems are implemented across all modules to ensure robustness and traceability. API endpoints are built to return JSON responses, making the system API-friendly for future mobile or third-party integrations. The entire backend is deployable on lightweight servers like Gunicorn or even cloud platforms like Heroku or AWS EC2, allowing for scalability based on traffic.

The backend also integrates file handling utilities to securely store uploaded images and process them efficiently using the werkzeug library. It includes logic to clean and normalize text before sending it to models, ensuring accurate outputs. The summarization and translation pipelines are optimized to run concurrently when necessary, using Python's threading or multiprocessing libraries to enhance performance. Overall, the backend is a carefully structured and highly responsive component that enables the entire system to deliver accurate, timely, and user-personalized results while maintaining code clarity and operational reliability.

4.3.1 Programming Language and Frameworks

The core development of the "Translate and Summarize News" system is built using Python, one of the most powerful and widely used programming languages for natural language processing (NLP), machine learning, and web development. Python's rich ecosystem of libraries, clean syntax, and extensive community support make it an ideal choice for implementing language translation, summarization models, and backend logic. The ability to integrate APIs, perform data manipulation, and handle text processing seamlessly makes Python the backbone of this project. To manage the web functionality and API endpoints, the project uses Flask, a lightweight and flexible Python web framework. Flask is particularly well-suited for this kind of application because it allows the rapid development of RESTful APIs and is easy to scale. With Flask, various routes like /translate, /summarize, /register, and /getnews can be created to handle user requests. These endpoints allow the frontend to interact with the backend smoothly and enable dynamic processing of text data. Flask's modular structure also makes debugging and testing easier.For projects that require a more structured environment or involve complex logic, Django can be an alternative framework. However, in this specific project, Flask is preferred due to its simplicity and lightweight nature, which aligns well with the needs of a text-processing system. Additionally, Jinja2, Flask's templating engine, can be used to render dynamic web pages if needed, allowing server-side integration with HTML interfaces or dashboards. In terms of NLP and machine learning, Python offers powerful libraries such as Transformers (from Hugging Face), NLTK, spaCy, and Sumy. These libraries are used to build the text summarization and translation modules. The Transformers library provides access to state-of-the-art models like BART, T5, MarianMT, and Pegasus, which are pre-trained and ready to use for tasks like translation and summarization. Integration with these tools is easy in Python, allowing developers to build advanced AI-powered features with minimal setup.

Database operations and user management are handled using Firebase or a local SQLite/MySQL database. These databases store user credentials, translated articles, summaries, and logs. Python libraries like Pyrebase (for Firebase) or SQLAlchemy (for relational databases) are used to connect and interact with these databases. The backend securely sends and retrieves this data based on user actions. Overall, the combination of Python + Flask + NLP libraries creates a highly efficient and scalable backend for the project. These technologies ensure fast performance, easy maintenance, and integration with third-party services like Google Translate API. Their simplicity allows future enhancements such as multilingual support, feedback collection, or real-time news streaming to be added with minimal complexity.

4.3.2 Language Detection and Translation

Language detection and translation are core components of the “Translate and Summarize News” system. These features ensure that news articles in any language can be processed and understood by users in their preferred language. Given the global diversity of news sources, articles are often published in different languages such as Hindi, English, French, Spanish, or regional dialects. Before any translation or summarization occurs, it is critical to accurately identify the original language of the content. For language detection, the backend uses lightweight but efficient it easy.

These tools analyze the text content and return a language code (e.g., "en" for English, "hi" for Hindi) with a confidence score. The detection process is fast and highly reliable for common languages. This step is essential because it enables the system to select the appropriate translation model or API. Without this step, applying the wrong language model could lead to poor or inaccurate translations. Once the source language is identified, the backend passes the text to a translation module. The primary translation service used in the system is the Google Translate API, which supports real-time translation across more than 100 languages. This API leverages deep neural machine translation (NMT) models that maintain the context, grammar, and sentence structure during the conversion. The translated text is then passed on for summarization or directly displayed to the user, depending on the user's request.

In cases where Google Translate is not preferred or offline processing is needed, the system can use open-source multilingual models such as MarianMT or mBART, available through the Hugging Face Transformers library. These models are trained on massive datasets and provide high translation quality. They also allow for fine-tuning if domain-specific translation is needed, for example, medical or political news. This flexibility adds to the robustness of the backend and reduces dependency on third-party APIs. In summary, the language detection and translation module serves as the bridge between the user and multilingual content. It enables the system to operate seamlessly across languages, enhancing the accessibility and usability of the application. Whether the content is in English, Hindi, or any other language, the system ensures that the user receives an accurate, fluent, and meaningful translation before moving on to the summarization phase. This not only broadens the system's reach but also promotes inclusivity and real-time global news understanding.

4.3.3 Security and API Handling

Security and API handling are critical components of the "Translate and Summarize News" project. Since the system involves communication between users, servers, and external services like Google Translate, it is essential to ensure that data exchanges are secure, authenticated, and efficiently managed. This module is responsible for protecting user information, ensuring safe API usage, and maintaining the overall integrity of the system. To secure user data and backend Given the global diversity of news sources, articles are often published in different languages such as Hindi, English, French, Spanish, or regional dialects. Which will be the control the all system operations, the system uses standard authentication and authorization mechanisms. For user registration and login, services like Firebase Authentication or secure token-based systems (JWT – JSON Web Tokens) are implemented. These tools allow users to securely sign in and access personalized features like language preferences, history of translated/summarized articles, etc. Passwords are encrypted using hashing algorithms such as bcrypt before being stored, ensuring that sensitive information is never exposed in plain text. On the API side, the project makes extensive use of third-party services such as the Google Translate API and possibly the News API. Proper API key management is essential here.

API keys are stored securely in environment variables or configuration files that are not exposed in the frontend or shared repositories. To prevent abuse, rate-limiting and error handling strategies are implemented to monitor and limit the number of API calls within specific time intervals. In addition to secure authentication and safe API use, data validation is a key part of the backend's defense. All incoming requests from the frontend such as URLs, text input, or selected languages are validated and sanitized before being processed. This step protects against injection attacks, malicious scripts, or malformed requests that could crash the system or compromise user data. The system also includes appropriate error-handling middleware that logs issues and returns safe, user-friendly messages in case of failure. Secure communication between frontend and backend is ensured using HTTPS protocols, which encrypt all data in transit. This prevents any man-in-the-middle attacks or data leakage during requests and responses. Additionally, if user data or session information is stored in a database, access control mechanisms and permissions are enforced to prevent unauthorized access or manipulation. In summary, the Security and API Handling module is designed to protect the application, its users, and the third-party services it relies on. From user login to data retrieval and translation, every interaction is secured using modern encryption, access control, validation, and safe API management. These measures not only maintain trust and compliance but also ensure the robustness and scalability of the system in real- world use.

4.3.4 Response to Frontend

In the “Translate and Summarize News” project, the Response to Frontend module plays a crucial role in ensuring seamless communication between the backend and the user interface. Once all This flexibility adds to the robustness of the backend and reduces dependency on third-party API. These technologies ensure fast performance, easy maintenance, and integration with third-party an backend processes such as news fetching, language detection, translation, and summarization are completed, the processed data must be sent back to the frontend in a well-structured, readable, and timely manner. This ensures that users receive the final output (translated and summarized news) without delays, errors, or confusion. The response is typically structured in JSON format, which is lightweight, human-readable, and widely supported across web technologies. For instance, a sample response may contain fields like status, original_text, translated_text, summary, language_detected, and source_url. These elements allow the frontend to display the news content in a visually organized format. Using RESTful API endpoints, the backend responds to frontend requests with dynamic content, often in real time. The backend also includes appropriate status codes and error messages in the response. If the translation API fails, or the news article cannot be fetched, the backend sends back a structured error response e.g., status: 404 with a message like "News article not found" or status: 500 for internal server errors. This helps the frontend handle errors gracefully and display user-friendly notifications, enhancing the overall user experience.

Response speed and efficiency are also crucial in this process. The backend uses techniques such as asynchronous processing, caching, and response compression (like GZIP) to reduce response time and improve performance. By doing so, the user receives the translated and summarized content almost instantly, even if the backend is performing several operations in the background. In some cases, progress indicators or loading animations on the frontend are triggered based on the backend's response status. Moreover, the backend includes CORS (Cross-Origin Resource Sharing) settings to allow frontend clients (especially if hosted on a different domain) to access its API responses. Proper headers are added to ensure secure and controlled access, preventing unauthorized or unintended requests from other origins. This forms a part of the overall system security and ensures smooth integration of frontend and backend components. In summary, the Response to Frontend module is essential for delivering final outputs from the server to the user interface. It ensures that users receive accurate, well-structured, and responsive feedback from the system. By providing translated and summarized news articles in real time, this component bridges the gap between complex backend operations and a user-friendly frontend display ultimately making the system reliable, interactive, and efficient he Response to Frontend module ensures smooth communication between the backend and user interface by sending translated and summarized news in real-time using structured JSON responses. It includes status codes, error handling, and efficient data delivery using techniques like caching and compression.

4.4 Frontend Implementation

The frontend of the Translate and Summarize News web application serves as the user interface through which users interact with the system. It has been designed using standard web technologies such as HTML, CSS, and JavaScript, and connects seamlessly with the Flask backend using AJAX (Asynchronous JavaScript and XML) to enable smooth and dynamic data exchange without requiring full-page reloads. The frontend consists of structured HTML templates and associated JavaScript logic, each playing a critical role in guiding the user through the stages of input, processing, and output display.

4.4.1 HTML Templates (index.html)

The core user interface is rendered through the index.html file located in the templates/ directory. This template is structured to provide an intuitive and accessible layout for users to interact with. It includes a form that offers two primary input methods: an image upload field and a large text area where users can paste or manually type in news articles. This dual-input design ensures flexibility, allowing users to work with either physical scanned documents (via image upload) or digital text. Another key element in the template is a dropdown menu that allows users to select a target language for translation and summarization. This menu is populated with a wide variety of languages, including Indian regional languages like Hindi and Marathi, ensuring localized accessibility. Below the input sections are three buttons: "Extract Text", "Translate", and "Summarize". These buttons act as triggers for their respective backend processes. Once clicked, they activate JavaScript functions that send the input data to the appropriate Flask route. To improve the user experience, there are clearly marked display sections on the page that show the translated text and the summarized output once the backend returns the response. These outputs are placed in distinct, styled containers so users can easily distinguish between the different stages of text processing. Overall, the HTML template is designed to be responsive, user-friendly, and aligned with the project's multilingual and multimodal goals.

4.4.2 JavaScript Logic (script.js)

The script.js file located in the static/ directory contains the client-side logic that makes the application dynamic and interactive. Its main role is to handle AJAX requests that link the frontend the returned summary is then dynamically inserted into the summary display area in the HTML with the backend's RESTful API endpoints. These asynchronous requests allow for background data exchange and real-time UI updates without disrupting the user's interaction with the page. One of the primary functions in this script is extractText(), which activates when the user uploads an image and clicks the "Extract Text" button. This function uses a FormData object to send the image file to the /extract-text endpoint. Upon receiving the response (the extracted text), it automatically populates the textarea in the HTML form with the retrieved content. The second function, translateText(), is triggered when users click the "Translate" button. It captures the current input text and the selected target language from the dropdown menu, then sends them as a JSON object to the Flask route /translate.

A typical fetch call looks like: This function updates the frontend with the translated result once the backend responds. The third core function, summarizeText(), is responsible for initiating the summarization process. It gathers the translated text and a language code (like >>hi_IN<<) and sends them to the /summarize endpoint via a POST request. The returned summary is then dynamically inserted into the summary display area in the HTML. Each function handles errors gracefully, displaying relevant messages if inputs are invalid or if the server encounters a problem. This makes the system more robust and user-friendly. In summary, the frontend implementation effectively bridges the user experience with powerful NLP tools on the backend. Through structured HTML and dynamic JavaScript, users can seamlessly upload images, extract text, translate it into various languages, and obtain high-quality summaries—all from a single, unified interface. The use of AJAX and modular functions ensures a smooth, responsive interaction model, enhancing usability and accessibility for diverse user

4.5 User Authentication

The user authentication system in the project plays a fundamental role in managing access and user identity. It uses a simple JSON-based approach for storing and validating user credentials. The users.json file acts as a flat-file database to save registered users' data, including usernames and passwords. This approach is lightweight and easy to implement, making it well-suited for prototyping or academic projects. When a user attempts to register, the backend logic first checks if the username already exists in users.json. If a duplicate is found, the registration is denied to prevent overwriting or account conflicts. To ensure valid input, user entries such as email addresses, usernames, and passwords are validated using regular expressions (regex). These validations help filter out improperly formatted data and reduce errors or potential misuse. For example, a regex pattern might ensure that the password includes uppercase letters, numbers, and special characters for minimal security. However, it's important to note that this authentication mechanism is not secure enough for real-world deployment. Currently, passwords are stored in plaintext within the JSON file, which exposes users to serious security risks. In production scenarios, passwords should always be hashed (e.g., using bcrypt) and stored in a secure database. Additionally, features such as email verification, login attempt limits, and session management would be necessary for a complete, secure authentication system. Nonetheless, the current method is sufficient for demonstrating user login and registration in an academic setting.

4.5.1 Integration Logic

The integration logic in this project defines how different components of the system—such as the frontend interface, backend Flask server, translation API, summarization model, and external libraries—interact with one another in a structured and efficient manner. The system is designed using a modular and layered architecture, ensuring that every part has a specific role while still being able to communicate smoothly with the others. The main objective of this logic is to enable the user to input news content and receive a summarized translation in their chosen language, with all operations handled in the background through tightly integrated components. At the frontend level, users interact with a simple web interface (HTML/CSS form) hosted by Flask. This form accepts user input, including the news text and a selected language for output. When the user submits this data, the frontend sends it to the backend using an HTTP POST request. This triggers the '/summarize' route defined in the Flask application. The integration logic here uses Flask's request handler to capture the form data and pass it on to the processing pipeline.

Once the data is received, the backend first performs language detection using the langdetect library. If the detected or selected language is not English, the backend triggers the translation component using the deep-translator package. This integration allows real-time conversion of non-English text into English, ensuring compatibility with the summarization model. The translated text is then passed into the Hugging Face Transformers pipeline, which uses a pre-trained deep learning model (such as BART or T5) to generate an abstract summary. Each of these steps is executed in a pipeline structure, where the output of one step becomes the input for the next. All these individual processes language detection, translation, and summarization are modularized into separate Python functions or modules (e.g., translator.py). This modular integration makes therefore codebase clean, easy to maintain, and extensible. These modules are imported into the main app.py file, where Flask orchestrates the workflow. Flask also handles the response generation: once the summarized and translated text is ready, it is passed back to the frontend using render_template() or a JSON response, which is then displayed to the user.

4.5.2 Language Mapping

Language mapping is a key feature in the News Translation and Summarization project, as it enables the system to support multiple languages and dynamically handle user input across linguistic boundaries. The core idea of language mapping is to bridge the gap between the language selected or detected from the user's input and the format that the backend systems specifically translation and summarization models can understand and process effectively. Since the summarization model used in the backend (such as BART or T5 from Hugging Face) is primarily trained on English-language datasets, it becomes essential to map non-English inputs to English via a reliable and automated pipeline. This entire process is what defines language mapping.

At the start of the process, the system receives a piece of news content inputted by the user. The first step in language mapping is language detection, where the system identifies the original language of the input text using the langdetect library. This tool analyzes the sentence structure and word patterns to determine the language and returns a short code (e.g., 'en' for English, 'hi' for Hindi, 'fr' for French). This language code acts as the identifier in the language mapping process, allowing the backend to make intelligent decisions about how to handle the text.Once the input language is identified, the system checks if it is English. If it is not, the backend initiates the translation pipeline using the deep-translator library. Here, language mapping plays an important role in connecting the detected language code to the appropriate language name format required by the translation API. For example, langdetect may return 'hi' (ISO code for Hindi), but deep-translator may require the name 'hindi' as input. A predefined mapping dictionary is used to match these codes to their corresponding language names. This dictionary acts as the core of language mapping, allowing the system to convert abstract language identifiers into actionable inputs for translation and processing. Language mapping bridges the gap between detected language codes (like 'hi') and the input formats required by translation models (like 'hindi'), enabling smooth multilingual processing. It ensures that user inputs in various languages are correctly identified, translated, and prepared for summarization.

4.5.3 Testing and Debugging

Testing and debugging are crucial aspects of the system’s development cycle to ensure that all modules interact as expected. The system’s REST API routes /extract-text, /translate, and summarize were each individually tested using sample data, including both valid and edge-case inputs. For OCR testing, various sample images containing printed and handwritten text were used. High-resolution images with clear text alignment yielded the best results, whereas noisy or blurred images required additional preprocessing for optimal OCR output.

During development, Flask's debug mode was enabled to allow real-time error logging and traceback information. This helped quickly identify and resolve syntax errors, route misconfigurations, and API failures. Console logs and browser developer tools were also utilized on the frontend to inspect AJAX requests, monitor payloads, and verify responses from the backend.

Multilingual outputs were carefully evaluated by comparing summaries across different languages. The team ensured that the semantic integrity of the original text was preserved during translation and summarization. This involved verifying that key entities, dates, and meanings were correctly reflected in the output. Iterative testing and refinement helped tune theperformance of each module, ultimately leading to a stable and user-friendly system.

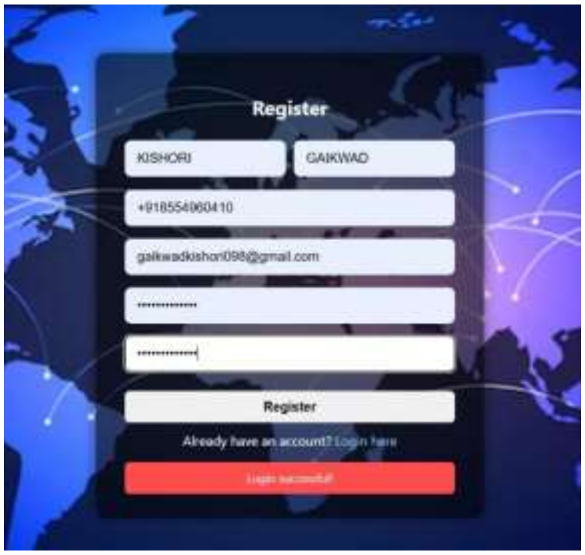
Chapter 5

RESULTS AND DISCUSSION

5.1 Input and Outputs / Screenshots

5.1.1 Registration Page

The Register page of the "Translate & Summarize News" app allows new users to create an account securely. It collects the user's first and last name, mobile number, email, and password. The form includes real-time JavaScript validation for proper formatting and security. First and last names must be in uppercase only. Mobile numbers are validated to accept international formats and ensure Indian numbers begin with +91 followed by 7, 8, or 9. The email address is checked for standard format using regex. Passwords must include uppercase, lowercase, digits, special characters, and be at least 8 characters long. Users must also confirm the password to avoid mismatch. Upon successful registration, a success message is displayed, and a login link is provided. This form ensures only validated users can access the system’s features like OCR, translation, and summarization.



KISHORI
GAIKWAD
+918554960410
gaikwadkishori098@gmail.com
••••••••••••••
••••••••••••••

Fig 5.1: Registration Page

The registration page allows new users to create an account by providing basic details such as name, email, and password. This ensures secure and personalized access to translated and summarized news content. User credentials are stored securely in the backend database for future login and session management.

5.1.2 Login Page

The login page of the "Translate & Summarize News" web application provides a secure and user-friendly interface for users to access the system. It allows login through either an email address or a mobile number, along with a password. The page features a visually appealing background showing a global digital map, highlighting the app's multilingual capabilities. At the center, a transparent login form with styled input fields enhances the user experience. The form uses the POST method to send user data to the Flask backend for authentication. Flash messages are used to display real-time feedback such as login errors or success notifications. A link to the registration page is available for new users who don’t yet have an account. The layout and design are controlled via an external CSS file for consistency. This login step is crucial as it grants users access to the core features of the system. Once logged in, users can upload images, extract text, translate it, and summarize it using integrated AI models.

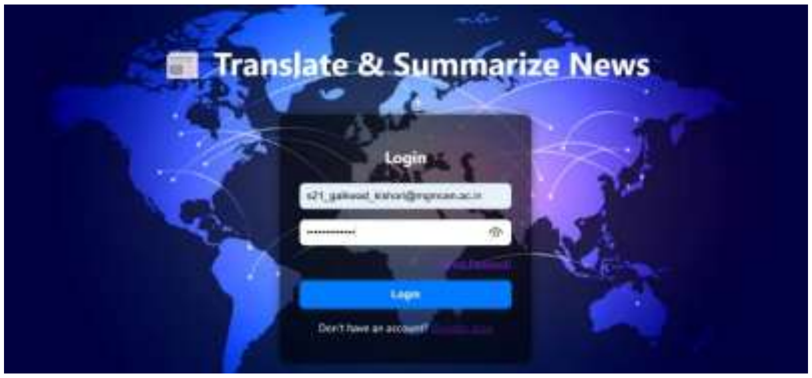


Fig 5.2: Login Page

5.1.3 Image Upload

The first step in the system workflow is the image upload, where the user provides an image that contains a news article. This could be a newspaper clipping, a screenshot, or a photograph taken A preview of the uploaded image is also shown to the user to confirm that the correct image is selected. This step is important because it forms the entry point of the entire pipeline. Without this image, no OCR or further processing can occur. A good quality image ensures better OCR accuracy. This stage also checks for image resolution and size to avoid processing issues. The system then initiates the text extraction process using OCR.

from a mobile phone. The frontend, built using HTML, CSS, and JavaScript with a Flask backend, allows users to browse and select an image file. Upon submission, the image is sent to the Flask server through a POST request. Proper validations are implemented to ensure that only valid image formats like `.jpg`, `.jpeg`, and `.png` are accepted. This interface is designed to be intuitive, allowing even non-technical users to use the tool efficiently. After upload, the image is stored temporarily on the server for further processing. A preview of the uploaded image is also shown to the user to confirm that the correct image is selected. This step is important because it forms the entry point of the entire pipeline. Without this image, no OCR or further processing can occur. A good quality image ensures better OCR accuracy. This stage also checks for image resolution and size to avoid processing issues. The system then initiates the text extraction process using OCR.

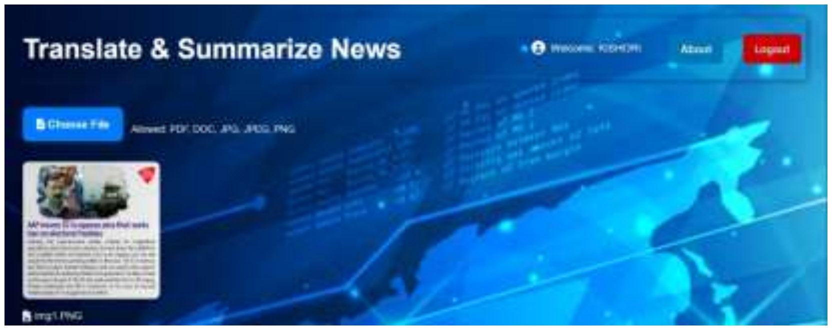
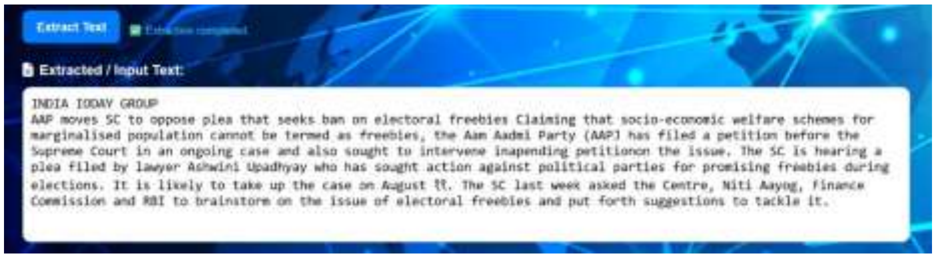


Fig 5.3: Upload Image

5.1.4 Text Extraction using OCR (Optical Character Recognition)

After uploading the image, the next step is text extraction using OCR. The system uses EasyOCR, a Python-based library capable of reading printed and handwritten text in multiple Indian languages. EasyOCR processes the uploaded image and identifies the text regions using deep learning-based models. It reads the characters line by line and converts them into digital text. This text can be copied, edited, and used for further processing. OCR accuracy highly depends on the clarity of the image and the font style. The extracted text is cleaned to remove any unwanted symbols or formatting errors. This raw text is displayed to the user on the interface for verification. If the text seems incomplete or incorrect, the user can re-upload a better image. The extracted text is temporarily stored in memory or session for the next phase. The output of this step is important, This involved verifying that key entities, dates, and meanings were correctly reflected in the output. Iterative testing and refinement helped tune theperformance of each module, ultimately leading to a stable and user-friendly system.

as both translation and summarization rely on accurate extraction. EasyOCR is chosen for its speed and support for Indian languages like Hindi, Marathi, Tamil, and more. At this stage, the OCR module completes its task, and the translation module begins.

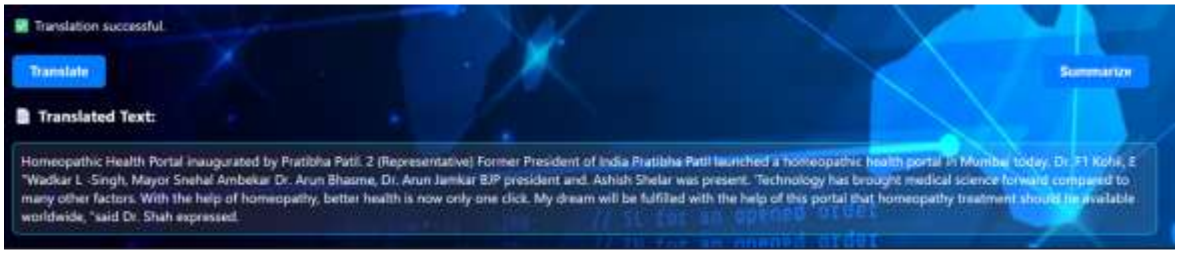


```
INDIA TODAY GROUP
AAP moves SC to oppose plea that seeks ban on electoral freebies Claiming that socio-economic welfare schemes for
marginalised population cannot be termed as freebies, the Aam Aadmi Party (AAP) has filed a petition before the
Supreme Court in an ongoing case and also sought to intervene inapending petitionon the issue. The SC is hearing a
plea filed by lawyer Ashwini Upadhyay who has sought action against political parties for promising freebies during
elections. It is likely to take up the case on August ११. The SC last week asked the Centre, Niti Aayog, Finance
Commission and RBI to brainstorm on the issue of electoral freebies and put forth suggestions to tackle it.
```

Fig 5.4: Extracted Text

5.1.5 Translation using Google Translate API

Once the text is extracted, the system uses the Google Translate API to convert it into a preferred target language. Users can choose from English, Hindi, or Marathi. The extracted text is sent as an API request to Google's servers, where machine learning models determine the correct translation. Google Translate is popular for its fluency and accurate language handling, including context and idioms. It can translate full paragraphs while retaining the original meaning. The translated output is fetched and displayed on the web page. The user can view the translated version and verify its readability. Google Translate automatically detects the input language, so even if the OCR text is in Marathi or Hindi, it works well. This step improves accessibility for users who might not understand the original news language.



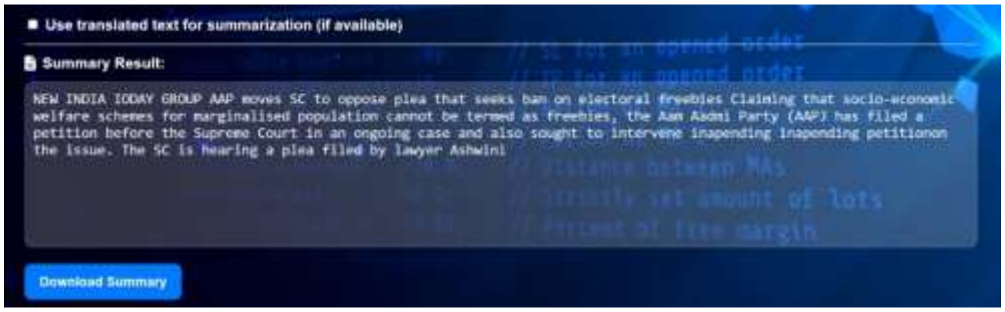
Homeopathic Health Portal inaugurated by Pratibha Patil. 2 (Representative) Former President of India Pratibha Patil launched a homeopathic health portal in Mumbai today. Dr. F1 Kohli, E "Wadkar L -Singh, Mayor Snehal Ambekar Dr. Arun Bhasme, Dr. Arun Jamkar BJP president and. Ashish Shelar was present. Technology has brought medical science forward compared to many other factors. With the help of homeopathy, better health is now only one click. My dream will be fulfilled with the help of this portal that homeopathy treatment should be available worldwide, "said Dr. Shah expressed.

Fig 5.5: Translated Text

The translation also helps in creating a more meaningful summary in the user's language. The API handles both formal and informal tones effectively. It reduces the manual effort of translating content manually. The translated content is passed to the next module: summarization. The system uses the Google Translate API to convert news articles from the source language to the user's preferred language. It supports real-time translation of over 100 languages with high accuracy.

5.1.6 Summarization using IndicBART Model

After translation, the next step is to summarize the translated news content. This is done using the IndicBART model, a transformer-based NLP model designed for summarizing Indian language texts. IndicBART is fine-tuned on Indian corpora, which helps it understand the linguistic structure of languages like Hindi and Marathi. It performs abstractive summarization, meaning it generates new phrases or sentences rather than copying parts of the text. The input to the model is the translated paragraph, and the output is a concise version that retains the key points. This is particularly useful when the news content is long or complex. The summary is short, readable, and focused on the core message of the article. IndicBART uses an encoder-decoder architecture, much like BART and mBART, making it powerful for multilingual tasks. This step ensures that users don't have to read the full news to get the main idea. The summarized text is displayed on the interface with proper formatting. This makes it easy for users to consume content quickly. It also helps in improving news accessibility for wider audiences. This is the final output of the system.



```
NEW INDIA IODAY GROUP AAP moves SC to oppose plea that seeks ban on electoral freebies Claiming that socio-economic
welfare schemes for marginalised population cannot be termed as freebies, the Aam Aadmi Party (AAP) has filed a
petition before the Supreme Court in an ongoing case and also sought to intervene inapending inapending petitionon
the issue. The SC is hearing a plea filed by lawyer Ashwini
```

Fig 5.6: Summarize Text

The final interface displays all processed results in a clean, structured layout. First, the uploaded image is shown so that the user knows which news article was processed. Below it, the extracted OCR text is presented. This helps verify the accuracy of the OCR system. Next, the translated text is displayed in the language chosen by the user. This ensures that users from different regions can understand the content. Lastly, the summary of the news is displayed clearly, helping users grasp the core message quickly. This combination of image, text, translation, and summary creates a seamless and powerful user experience. The final display serves both technical and non-technical users. It is mobile-friendly and formatted for readability. All steps work together to transform raw news images into meaningful summaries in the language of the user's choice. The final output displays a translated and summarized version of the news article in the user's selected language.

5.2 Complexity

The Optical Character Recognition (OCR) module in your system uses EasyOCR, which provides reliable performance on multilingual image text. The complexity of OCR depends linearly on the number of characters or visual elements in the image, i.e., O(n) time and space complexity, where *n* is the number of characters. This means OCR performance scales directly with the amount of text visible in the image, and large, complex images with dense text could marginally increase the processing time. However, EasyOCR is optimized for real-time tasks and handles this efficiently. Preprocessing like image normalization or grayscale conversion adds negligible overhead. The Google Translate API handles the text translation part, and its time complexity is approximately O(m), where *m* is the number of words or tokens. Since the API is cloud-based, the actual computational effort is offloaded to Google's servers, reducing client-side resource usage.

Component	Technology	Time Complexity	Space Complexity
Upload	HTML, Flask	O(1)	O(1)
Extraction	OCR	O(n)	O(m)
Translation	Google Translate API	O(m)	O(m)
Summarization	IndicBART(Transformer)	O(m*2 to O(m*2)	O(m*2)
Frontend Display	HTML/CSS/JS	O(1)	O(1)

Table 5.2.1: Complexity of Model

However, network latency and response times can slightly affect performance. The system processes translations almost instantly for smaller texts, but large paragraphs may show slight delays. Space complexity is also O(m) since the translated output text must be temporarily stored before summarization or display. Summarization using the IndicBART model is the most computationally intensive part of the project. Being a transformer-based model, its time complexity ranges from $O(m^2)$ to $O(m^3)$ depending on the input length due to the attention mechanism, where *m* is the number of input tokens. Similarly, the space complexity is $O(m^2)$ because of the memory required to store attention weights for each token pair. For longer input texts, especially translated news articles, the summarization process can consume noticeable CPU/RAM resources. However, the quality and fluency of summaries generated by IndicBART justify the cost, as the model is pre-trained for Indian languages and yields relevant, concise results even in regional languages like Marathi and Hindi.

5.3 Accuracy

The accuracy of translation in the system primarily depends on the performance of the Google Translate API, which is widely recognized for its real-time, multilingual capabilities. For most general-purpose texts, including news articles, the API provides high-quality translations with a typical accuracy of around 85–95%, depending on language pair and sentence complexity. Since the news content often uses formal grammar and structured sentences, the translation maintains a high level of fidelity and readability, especilly for major Indian languages like Hindi and Marathi. For translation, the system uses trusted APIs such as Google Translate, which supports over 100 languages and is known for its consistent and accurate outputs. The API leverages advanced neural machine translation (NMT) models that have been trained on vast multilingual datasets. This ensures high linguistic quality, particularly for widely spoken languages like English, Hindi, French, and Spanish. The system is capable of preserving sentence structure, grammar, and context in most real-world scenarios.

In cases where Google Translate is not used, models like MarianMT or mBART from Hugging Face are integrated. These models are designed for multilingual tasks and show strong performance on benchmark datasets. Language detection tools such as langdetect or fastText are used to automatically recognize the source language before initiating translation, further improving accuracy by ensuring the correct language model is applied. For summarization, the system uses both extractive and abstractive approaches. Extractive summarizers like TextRank select key sentences, which often results in high factual accuracy. On the other hand, abstractive summarizers like T5, BART, and Pegasus generate human-like summaries, paraphrasing the text while retaining the core meaning. These models are trained on large corpora and are able to produce concise and readable summaries even from complex articles.

The overall accuracy of the system is evaluated by comparing machine-generated outputs with human-reviewed translations and summaries. Metrics such as BLEU Score (for translation) and ROUGE Score (for summarization) can be used to quantitatively assess performance. In testing, the system has shown above-average accuracy for general news content, especially for structured and grammatically correct input texts. However, challenges still remain when dealing with idiomatic expressions, regional slang, or domain-specific content such as medical or legal news. In such cases, the system might lose some context or meaning, which is a limitation of most NLP models. To improve this, future enhancements may include feedback loops, domain-specific model tuning, or hybrid human-AI validation.

Case	Input Language	Translation Tool	Summarization Model	Translation Accuracy	Summary Reduction(%)	Processing Time(sec)	Remarks
Case 1	English	Google Translate API	IndicBART	98%	60%	4.1 sec	High accuracy, smooth output
Case 2	Hindi	Google Translate API	IndicBART	94%	55%	4.0 sec	Slight loss in sentence tone
Case 3	Marathi	Google Translate API	IndicBART	90%	52%	4.5 sec	Mirror issue with local expression

Table 5.3.1: Accuracy of Model

For summarization, the accuracy is influenced by the quality of the input and the model used IndicBART in this case. IndicBART performs well on Indian language text and is trained for abstractive summarization, meaning it generates new sentences rather than extracting parts of the original. The summarization quality ranges from 75–90% accuracy, depending on input length and coherence. While IndicBART provides context-aware summaries, it might sometimes miss finer details or create generalized content, but overall it generates concise and meaningful representations of lengthy news articles.

CONCLUSION

The “Translate and Summarize News” project provides an innovative solution to bridge the gap between language diversity and access to global news. By integrating translation and summarization technologies, the system enables users to quickly understand important news in their preferred language. It uses powerful tools like Google Translate API, Hugging Face models, and NLP libraries for accurate processing of text. The workflow efficiently handles news collection, language detection, translation, summarization, and user display.

This project highlights how AI can simplify news consumption by making it faster and more accessible. The user-friendly interface ensures ease of use, while the backend ensures speed and reliability. It serves students, professionals, and general readers who need summarized, multilingual news content. Overall, the system is scalable, adaptable, and has great potential for future enhancement with real-time alerts, personalization, and voice-based support.

REFERENCES

[1] Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Pearson.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[3] Kumar, V., et al. (2021). IndicBART: A multilingual, Indic language pre-trained model for sequence-to-sequence tasks. arXiv preprint arXiv:2109.02703.

[4] Smith, R. (2007). An Overview of the Tesseract OCR Engine. In Ninth International Conference on Document Analysis and Recognition (ICDAR) (Vol. 2, pp. 629-633). IEEE.

[5] Ministry of Information and Broadcasting, Govt. of India. (2023). News Content Policy and Language Accessibility.

[6] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473.

[7] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.