

IMAGE-TO-TEXT CONVERTER THAT DETECTS LANGUAGE AND PROVIDES A TRANSLATED SUMMARY

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology *in* Computer Science and Engineering

by

**HARSHVARDHAN SINGH GAHLAUT
19BCE2372**

**PRATHAM SHAH
19BCE2028**

**Under the guidance of
Prof. Vasantha W B
SCOPE
VIT, Vellore.**



May, 2023

DECLARATION

I hereby declare that the thesis entitled “**Image-to-Text Converter That Detects Language and Provides a Translated Summary**” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering to VIT is a record of bonafide work carried out by me under the supervision of Dr Vasantha W. B.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 19/05/2023

A handwritten signature in blue ink, appearing to read 'Harsh', enclosed within a light blue rectangular border.A handwritten signature in black ink, appearing to read 'Mehal', with a horizontal line underneath.

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “**Image-to-Text Converter That Detects Language and Provides a Translated Summary**” submitted by **Pratham Shah(19BCE2028) and Harshvardhan Singh Gahlaut(19BCE2372)**, School of Computer Science and Engineering, VIT, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 07. 2022 to 30.04.2023, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore
Date:20.05.2023

W.B. Vasantha.

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department
Programme

ACKNOWLEDGEMENTS

Thanks to the invaluable contributions of all those involved, whether directly or indirectly, the project "**Image-to-Text Converter That Detects Language and Provides a Translated Summary**" was successfully completed. I would like to express my gratitude first and foremost to my mentor, **Dr. Vasantha W B**, who played a pivotal role not only in providing the necessary and innovative foundation for the project but also in offering essential and constructive feedback that contributed to the final outcome. **Dr. Vasantha W B** aided me in conducting research in the specified field and enhancing my comprehension of Deep Learning and Natural Language Processing. I am immensely grateful for the unwavering support that was extended throughout the project.

Furthermore, I would like to recognize the contributions of **Prof. Vairamuthu S, the Head of Department**, who played a crucial role in keeping me informed about all the necessary procedures and providing me with the required formats and document templates via email, which I greatly appreciated.

It is fair to state that **Prof. Ramesh Babu K, the Dean of SCOPE**, consistently made himself accessible to answer any questions and resolve any uncertainties that arose during my project.

Finally, I express my gratitude to the Vellore Institute of Technology for offering me the flexibility to choose and carry out my project, as well as for providing support throughout my research and project execution.

Pratham Shah
Harshvardhan Singh Gahlaut

Executive Summary

In today's digital age, language translation and text extraction from images have become commonplace in many apps and software. However, our project goes beyond these functionalities by offering an additional feature: text summarization in the user's desired language. Many of us have encountered situations where we come across lengthy paragraphs on websites or documents, and we simply need a brief summary to save time or accommodate our busy schedules. With our platform, users can easily input text or upload an image of the text in any language. They can then select their desired target language for translation and also request a summary of the text's content, all with the assistance of cutting-edge NLP (Natural Language Processing) and Optical Character Recognition (OCR).

Our process is designed to be step-by-step and user-friendly, allowing users to choose which steps they want to skip. We provide a proper website with a sleek UI, powered by Flask technology, to ensure a seamless and efficient experience. Whether it's translating text, extracting text from images, or summarising content, our project offers a comprehensive solution that caters to the diverse language needs of our users. Say goodbye to language barriers and information overload - with our project, language translation and summarization are made easy and accessible to all.

Index Terms: *Natural Language Processing, Optical Character Recognition, Flask.*

CONTENTS	Page No.
Acknowledgement	4
Executive Summary	5
Table of Contents	6
List of Figures	7
Abbreviations	9
1. INTRODUCTION	10
1.1. Theoretical Background	10
1.2. Motivation	11
1.3. Aim of the Proposed Work	12
1.4. Objective(s) of the Proposed Work	13
2. LITERATURE SURVEY	14
2.1. Survey of the Existing Models/Work	14
2.2. Summary/Gaps identified in the Survey	18
3. OVERVIEW OF THE PROPOSED SYSTEM	19
3.1. Introduction and Related Concepts	19
3.2. Framework, Architecture or Module for the Proposed System(with explanation)	21
3.3. Proposed System Model(ER Diagram/UML Diagram/Mathematical Modelling)	27
4. PROPOSED SYSTEM ANALYSIS AND DESIGN	28
4.1. Introduction	28
4.2. Requirement Analysis	29
4.2.1. Functional Requirements	
4.2.1.1. Product Perspective	
4.2.1.2. Product features	
4.2.1.3. User characteristics	
4.2.1.4. Assumption & Dependencies	
4.2.1.5. Domain Requirements	
4.2.1.6. User Requirements	
4.2.2. Non Functional Requirements	35
4.2.2.1 Product Requirements	

4.2.2.1.1	Efficiency (in terms of Time and Space)	
4.2.2.1.2	Reliability	
4.2.2.1.3	Portability	
4.2.2.1.4	Usability	
4.2.2.2	Organisational Requirements	
4.2.2.2.1	Implementation Requirements (in terms of deployment)	
4.2.2.2.2	Engineering Standard Requirements	
4.2.2.3	Operational Requirements	
4.2.2.3.1	Economic	
4.2.2.3.2	Environmental	
4.2.2.3.3	Social	
4.2.2.3.4	Political	
4.2.2.3.5	Ethical	
4.2.2.3.6	Health and Safety	
4.2.2.3.7	Sustainability	
4.2.2.3.8	Legality	
4.2.2.3.9	Inspectability	
4.2.3	System Requirements	42
4.2.3.1	H/W Requirements	
4.2.3.2	S/W Requirements	
5.	RESULTS AND DISCUSSION	43
6.	REFERENCES	73

LIST OF FIGURES	Page No.
1. Framework of the Project	21
2. EasyOCR workflow	22
3. NLTK workflow	24
4. UML Diagram	27
5. Main Page CSS	61
6. Main page form	62
7. JavaScript to handle translation and/or summarisation task	62
8. Signup Page code	63
9. Flask App	63
10. Installing libraries	64
11. Downloading NLTK stopwords	65
12. Importing libraries	65
13. Loading EasyOCR models	66
14. Function to read text from images	66
15. Extracting text from PDFs	67
16. Function for Summarisation	67
17. Login Page	68
18. Signup Page	68
19. Main Page	69
20. Selecting a PDF and choosing the requirements	69
21. Text input	70
22. Downloaded output	70
23. Translated Summary Output	71

ABBREVIATIONS

- 1. OCR:** Optical Character Recognition
- 2. NLP:** Natural Language Processing
- 3. NLTK:** Natural Language Toolkit
- 4. UI:** User Interface
- 5. PDF:** Portable Document Format
- 6. RAM:** Random Access Memory
- 7. HTML:** Hypertext Markup Language
- 8. CSS:** Cascading Style Sheets
- 9. JPEG:** Joint Photographic Experts Group
- 10. PNG:** Portable Network Graphics
- 11. API:** Application Programming Interface
- 12. CNN:** Convolutional Neural Network

1. INTRODUCTION

1.1. Theoretical Background:

The advancement of digital technology has paved the way for language translation and text extraction from images to become increasingly common in many software applications. These technologies have significantly reduced the language barriers between individuals and have made it possible for people to communicate with each other in different languages. However, with the vast amounts of information available online, users often encounter lengthy paragraphs that require a significant amount of time to read and comprehend. To address this issue, text summarization technology has emerged as a powerful tool to provide users with quick and concise summaries of lengthy texts.

Language translation, text extraction from images, and text summarization are important functions that have become increasingly prevalent in the digital age. These functionalities are achieved through the use of Natural Language Processing (NLP), a subfield of artificial intelligence that deals with the interaction between computers and human language. NLP techniques involve processing large amounts of natural language data, analysing it for patterns, and developing algorithms that enable computers to understand, interpret, and generate human language.

OCR (Optical Character Recognition) technology plays an important role in text extraction from images, which is a key component of many language processing systems. OCR enables computers to recognize text within an image and convert it into machine-readable text. This technology has been used in a wide range of applications, from digitising documents to automating data entry tasks.

Our project builds upon these existing technologies by integrating language translation and text summarization capabilities. By leveraging NLP techniques and corpora, our platform is able to translate text between languages and generate concise summaries of lengthy documents. The user-friendly interface and step-by-step process enable users to easily input text or upload an image, select their desired target language and request a summary of the text's content.

Overall, the project's comprehensive solution is designed to cater to the diverse language needs of users, making language translation and summarization easy and accessible to all. By combining OCR and NLP technologies, this platform represents an innovative approach to addressing language barriers and information overload in the digital age.

1.2. Motivation:

The motivation behind our project is to address the challenges posed by language barriers and information overload in today's digital age. As the world becomes increasingly interconnected, the ability to communicate across languages has become essential. However, the vast amounts of information available online often present a significant obstacle to effective communication. Our project seeks to address this issue by providing a comprehensive solution that integrates language translation, text extraction from images, and text summarization capabilities.

We are motivated to create a tool that goes beyond the existing functionalities of language translation and text extraction from images by offering an additional feature of text summarization in the user's desired language. Our aim is to create a platform that is easy to use, efficient and accessible to everyone, regardless of language barriers. We are motivated to create a solution that can help individuals save time and accommodate their busy schedules by generating concise summaries of lengthy documents.

By leveraging the latest NLP techniques and corpora, we seek to make language translation and text summarization easy and accessible to all. The use of OCR technology in text extraction from images is another area of focus for us, as we recognize its importance in many language processing systems. Our motivation is to create an innovative platform that combines these existing technologies in a user-friendly and comprehensive manner.

In summary, our motivation to work on this problem stems from our desire to overcome the challenges posed by language barriers and information overload in the digital age. We aim to create a platform that leverages the latest technologies in language translation, text extraction from images, and text summarization to provide a comprehensive solution that caters to the diverse language needs of users.

1.3. Aim of the proposed work:

The aim of our proposed work is to develop a comprehensive language translation and summarization system that goes beyond existing tools in the market. Our system will allow users to upload images or input text, detect the language of the text, and provide translation and summarization in the language of their choice. Our system will use OCR technology to detect text from images, which will provide users with the convenience of uploading images of texts instead of manually inputting the text. We will create a website with a proper frontend and a friendly UI to ensure ease of use for users.

One of the unique features of our system is that it will not only translate the text but also provide a summarization of the content. Our system will use an NLP model to summarise text from various languages, which will be a significant advantage for users who need to read and understand content from different parts of the world. Moreover, our system's summarization feature will save users time, making it easier for them to navigate through large paragraphs or documents, providing a quick overview of the content.

We aim to develop a system that provides users with a seamless and efficient experience, allowing them to choose which steps they want to skip. Our system will cater to the diverse language needs of users by allowing them to translate and summarise text from different languages. Furthermore, our system will go beyond existing tools by summarising content from languages other than English. Ultimately, our proposed work aims to eliminate language barriers and provide a comprehensive language translation and summarization tool accessible to all.

1.4 Objective(s) of the Proposed Work

The objective of our project is to create a comprehensive language processing system that integrates text translation and summarization functionalities. Unlike existing tools that offer either translation or summarization, our platform aims to provide both capabilities in a single system. Our system is designed to allow users to input text or upload images, which will be processed using OCR technology to extract the text. We will create a user-friendly website with a proper frontend and a friendly UI that offers all these options for ease of use.

Once the text is identified, our system will automatically detect the language of the text. Users will be able to select their desired target language for translation and summarization. Our system will then use NLP models powered by the Natural Language Toolkit (NLTK) to translate the text into the selected language and generate a concise summary of its content.

Moreover, most existing summarization tools only support English text. However, our platform will offer multi-lingual support and the ability to summarise text in languages other than English. Our goal is to create a system that can cater to the diverse language needs of our users and provide a comprehensive language processing solution.

Overall, our objective is to develop an innovative language processing system that makes it easy and accessible for users to translate and summarise text, regardless of language and text format.

2. LITERATURE SURVEY

2.1 Survey of the Existing Models/Work

Attend, Translate and Summarise: An Efficient Method for Neural Cross-Lingual Summarization

- *Junnan Zhu, Yu Zhou, Jiajun Zhang, Chengqing Zong; 2021*

Cross-lingual summarization aims at summarising a document in one language (e.g., Chinese) into another language (e.g., English). In this paper, they propose a novel method inspired by the translation pattern in the process of obtaining a cross-lingual summary. They first attend to some words in the source text, then translate them into the target language, and summarise to get the final summary. Specifically, they first employ the encoder-decoder attention distribution to attend to the source words. Second, they present three strategies to acquire the translation probability, which helps obtain the translation candidates for each source word. Finally, each summary word is generated either from the neural distribution or from the translation candidates of source words. Experimental results on Chinese-to-English and English-to-Chinese summarization tasks have shown that our proposed method can significantly outperform the baselines, achieving comparable performance with the state-of-the-art.

Abstractive Arabic Text Summarization Based on Deep Learning

- *YM Wazery, ME Saleh, A Alharbi, AA Ali; 2022*

In this paper, an abstractive Arabic text summarization system is proposed, based on a sequence-to-sequence model. This model works through two components, encoder and decoder. Their aim is to develop the sequence-to-sequence model using several deep artificial neural networks to investigate which of them achieves the best performance. Different layers of Gated Recurrent Units (GRU), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BiLSTM) have been used to develop the encoder and the decoder. In addition, the global attention mechanism has been used because it provides better results than the local attention mechanism. Furthermore, AraBERT preprocess has been applied in the data preprocessing stage that helps the model to understand the Arabic words and achieves state-of-the-art results. Moreover, a comparison between the skip-gram and the continuous bag of words (CBOW) word2Vec word embedding models has been made. They have built these models using the Keras library and run-on Google Colab Jupiter notebook to run seamlessly. Finally, the proposed system is evaluated through ROUGE-1, ROUGE-2, ROUGE-L, and BLEU evaluation metrics. The experimental results show that three layers of BiLSTM hidden states at the encoder achieve the best performance. In addition, the proposed system outperforms the other latest research studies. Also, the results show that abstractive summarization models that use the skip-gram word2Vec model outperform the models that use the CBOW word2Vec model

Neural Abstractive Text Summarization with Sequence-to-Sequence Models

- *T Shi, Y Keneshloo, N Ramakrishnan; 2021*

In this article, they provide a comprehensive literature survey on different seq2seq models for abstractive text summarization from the viewpoint of network structures, training strategies, and summary generation algorithms. Several models were first proposed for language modelling and generation tasks, such as machine translation, and later applied to abstractive text summarization. Hence, they also provide a brief review of these models. As part of this survey, they also developed an open source library, namely, Neural Abstractive Text Summarizer (NATS) toolkit, for the abstractive text summarization. An extensive set of experiments have been conducted on the widely used CNN/Daily Mail dataset to examine the effectiveness of several different neural network components. Finally, they benchmark two models implemented in NATS on the two recently released datasets, namely, Newsroom and Bytecup.

STN-OCR: A single Neural Network for Text Detection and Text Recognition

- *Christian Bartz, Haojin Yang, Christoph Meinel; 2017*

Detecting and recognizing text in natural scene images is a challenging, yet not completely solved task. In recent years several new systems that try to solve at least one of the two sub-tasks (text detection and text recognition) have been proposed. In this paper they present STN-OCR, a step towards semi supervised neural networks for scene text recognition, that can be optimised end-to-end. In contrast to most existing works that consist of multiple deep neural networks and several pre-processing steps we propose to use a single deep neural network that learns to detect and recognize text from natural images in a semi-supervised way. STN-OCR is a network that integrates and jointly learns a spatial transformer network, that can learn to detect text regions in an image, and a text recognition network that takes the identified text regions and recognizes their textual content. We investigate how our model behaves on a range of different tasks (detection and recognition of characters, and lines of text). Experimental results on public benchmark datasets show the ability of our model to handle a variety of different tasks, without substantial changes in its overall network structure.

High Performance Text Recognition Using a Hybrid Convolutional-LSTM Implementation

- *TM Breuel; 2017*

Optical character recognition (OCR) has made great progress in recent years due to the introduction of recognition engines based on recurrent neural networks, in particular the LSTM architecture. This paper describes a new, open-source line

recognizer combining deep convolutional networks and LSTMs, implemented in PyTorch and using CUDA kernels for speed. Experimental results are given comparing the performance of different combinations of geometric normalisation, 1D LSTM, deep convolutional networks, and 2D LSTM networks. An important result is that while deep hybrid networks without geometric text line normalisation outperform 1D LSTM networks with geometric normalisation, deep hybrid networks with geometric text line normalisation still outperform all other networks. The best networks achieve a throughput of more than 100 lines per second and test set error rates on UW3 of 0.25%.

Deep Learning based Isolated Arabic Scene Character Recognition

- Saad Bin Ahmed, Saeeda Naz, Muhammad Imran Razzak, and Rubiyah Yousaf; 2017

This paper offers a Arabic site/ scene text recognition with the help of deep learning classifier utilising the model Convolution Neural Networks (CNN). The authors use five different orientations with regard to a single occurrence of a character since the scene text data is slanted and skewed, allowing us to handle the widest range of variances. The training is designed with stride values of 1 and 2 and filter sizes of 3×3 and 5×5 . The training of the network was done with different learning rates throughout the text classification phase. On the recognition of Arabic characters from segmented Arabic scene photos, our method showed encouraging results.

Efficient, Lexicon-Free OCR using Deep Learning

- Marcin Namysl, Iuliu Konya; 2019

Due to geometrical distortions, intricate backgrounds, and a wide range of fonts, optical character recognition (OCR) continues to be a difficult challenge in unrestricted contexts, such as natural landscapes. This paper provide a deep learning-based, artificially generated training set, and data augmentation system for segmentation-free OCR. With the use of enormous text corpora and more than 2000 fonts, we render synthetic training data. They also add geometric distortions to collected samples as well as a suggested data augmentation method called alpha-compositing with background textures to emulate text appearing in complex natural situations. The specified models employ a convolutional neural network encoder to extract features from text images. On examining the modelling abilities of convolutional and recurrent neural networks for the interactions between input items. On distorted text samples, the suggested OCR system outperforms the accuracy of top commercial and open-source engines.

Reward Learning for Efficient Reinforcement Learning in Extractive Document Summarisation

- Yang Gao, Christian M. Meyer, Mohsen Mesgar and Iryna Gurevych; 2019

Summarising documents can be treated as a sequential decision-making problem that Reinforcement Learning (RL) algorithms can handle. Due to the vast search areas and the delayed rewards, the most common RL paradigm for summarization learns a cross-input strategy, which necessitates a lot of time, data, and parameter tuning. Although learning input-specific RL rules is a more effective alternative, it still relies on hand-crafted rewards, which are challenging to create and produce subpar results. So, the paper here proposes RELIS, which is a novel reinforcement learning paradigm uses Learning-to-Rank (L2R) algorithms to learn a reward function during training, and then uses this reward function to train an input-specific RL policy during testing. The paper proves that RELIS ensures that using the right L2R and RL algorithms, summaries will be generated that are close to optimal. The paper also shows that the RELIS approach is capable of reduced training time by a factor of 2 orders as compared to other present day methodologies and approaches.

Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond

- Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, Bing Xiang; 2016

The work here proposes and develops a text summarization using Attentional Encoder- Decoder Recurrent Neural Networks, and show that they achieve state-of-the-art performance on two different corpora. The authors here propose a number of innovative models that handle important summarising issues that the basic architecture does not fully capture, like modelling essential words, capturing the hierarchy of sentence-to-word structure, and emitting words that are uncommon or unobserved during training. As for the conclusion, the paper states that many of the proposed models contribute to more improvements in the upcoming future. Along with this research, the paper also proposes a new dataset consisting of multi-sentence summaries, and establishes performance benchmarks for further research.

Content Selection in Deep Learning Models of Summarization

- Chris Kedzie, Kathleen McKeown, Hal Daume' III; 2019

In order to comprehend how content selection is carried out, the paper conducts tests using deep learning models of summarization spanning the domains of news, personal stories, meetings, and medical literature. The authors discover that several highly sophisticated aspects of modern extractive summarizers do not outperform simpler models in terms of performance. These findings cast doubt on the value of deep learning methods for summary for those domains that do have large datasets and show that it is simpler to develop a summarizer for a new domain than past research suggests. Moreover, they suggest new types of sentence representations or external information sources are required that are better suited to the summarising task; these are significant questions for fresh research in summarization.

2.2 Summary/Gaps Identified in the Survey:

During our literature survey of existing work and systems related to text recognition, translation, and summarization, we found that many systems offered only one of these capabilities. For instance, some systems could perform language translation, but not text summarization, while others could recognize text from images or PDFs, but did not include translation or summarization capabilities.

Another limitation we identified was that many systems were designed to work with only one or a few languages, making them less useful for individuals who frequently work with multiple languages. Furthermore, most existing summarization tools only provided summaries in English, leaving users who need to read summaries in other languages with few options.

Overall, we found that there was a significant gap in the market for a comprehensive tool that could perform all three functions - text recognition from images and PDFs, translation, and summarization - in any language. This gap in the market motivated us to develop a system that could fill this need and provide users with a comprehensive tool for text processing in multiple languages.

3. OVERVIEW OF THE PROPOSED SYSTEM

3.1. Introduction and Related Concepts:

Introduction:

Our project aims to create a comprehensive language processing system that integrates OCR, NLP, text translation, text summarization, corpora, and flask. These technologies enable the system to extract text from images, identify the language of the text, translate it to the desired language, and generate a concise summary of the text's content. In this section, we will provide a detailed explanation of each of these technologies and how they will be used in our project.

OCR:

OCR, or Optical Character Recognition, is a technology that enables computers to recognize text within an image and convert it into machine-readable text. OCR technology plays a critical role in text extraction from images, which is a key component of many language processing systems. In our project, we will be using EasyOCR, an open-source OCR library that supports over 70 languages, including English, Chinese, Japanese, and many more.

EasyOCR is an ideal choice for our project because it is fast, accurate, and easy to use. Additionally, it has a small footprint, making it well-suited for use in web applications.

NLP:

NLP, or Natural Language Processing, is a subfield of artificial intelligence that deals with the interaction between computers and human language. NLP techniques involve processing large amounts of natural language data, analysing it for patterns, and developing algorithms that enable computers to understand, interpret, and generate human language.

In our project, we will be using NLP techniques to identify the language of the text, translate it to the desired language, and generate a concise summary of the text's content. We will be using the Natural Language Toolkit (NLTK), an open-source NLP library for Python. NLTK provides a wide range of tools for processing natural language data, including tokenization, stemming, and part-of-speech tagging.

Text Translation:

Text translation is the process of converting text from one language to another. In our project, we will be using machine translation, which is the use of computer algorithms to translate text between languages. We will be using the Google Cloud Translation API, which provides fast and accurate translation between over 100 languages.

Text Summarization:

Text summarization is the process of generating a concise summary of a lengthy text. In our project, we will be using extractive summarization, which involves identifying the most important sentences in the text and generating a summary based on these sentences. We will be using the TextRank algorithm, which is a graph-based ranking algorithm that identifies the most important sentences in the text based on their similarity to other sentences in the text.

Corpora:

In the context of natural language processing (NLP), a corpus refers to a large collection of written or spoken text that has been gathered and stored for analysis and research. Corpora are used to study language patterns, develop and train machine learning algorithms, and test the accuracy and effectiveness of NLP models. They can be monolingual, containing text in a single language, or multilingual, containing text in multiple languages. Corpora can also be annotated with additional information such as part-of-speech tags, named entity tags, and sentiment labels, to further enhance the analysis and understanding of the text.

Flask:

Flask is a lightweight web application framework for Python. In our project, we will be using Flask to create a user-friendly interface for our language processing system. Flask provides a simple and easy-to-use framework for building web applications, making it an ideal choice for our project.

Conclusion:

In conclusion, our project aims to create a comprehensive language processing system that integrates OCR, NLP, text translation, text summarization, corpora, and flask. These technologies enable the system to extract text from images, identify the language of the text, translate it to the desired language, and generate a concise summary of the text's content. By leveraging these technologies, our platform represents an innovative approach to addressing language barriers and information overload in the digital age.

3.2. Framework, Architecture or Module for the Proposed System(with explanation):

3.2.1 Framework of the project:

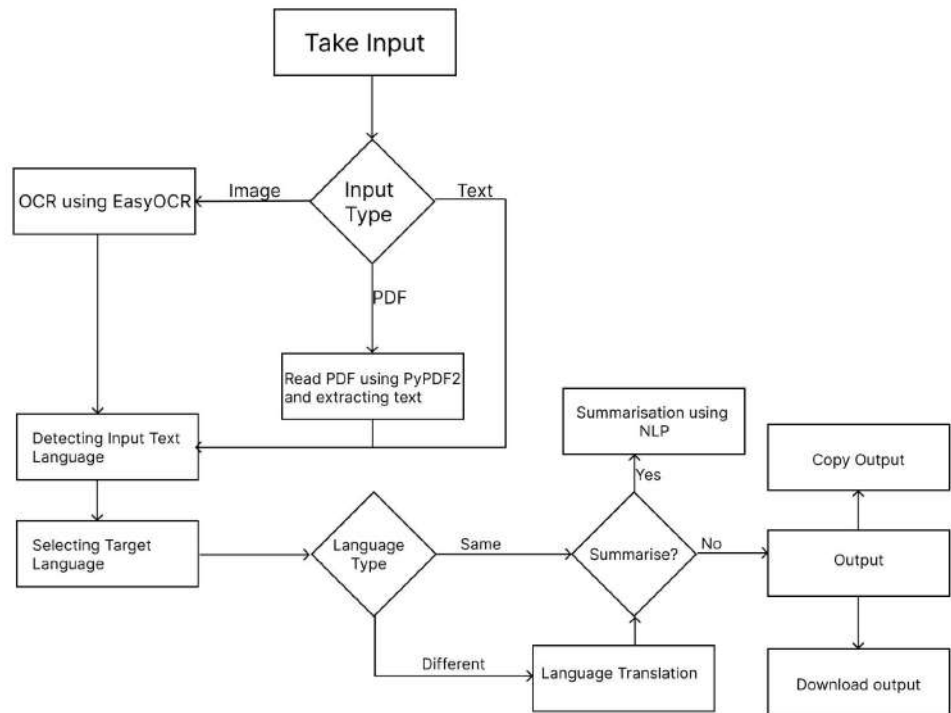


Fig.1. Framework of project

First the user has to give an input. The input can either be a text, image with text or a pdf file. Once given, the user has to select the input language (specifically for the OCR model) and select if they want to translate and summarise the text with the help of radio buttons on the web-page. If the person wishes to translate, on clicking the check box, the option for language opens up where the user can select the language they want out of 7 languages that we offer (English, Spanish, French, Arabic, Chinese, Tamil, Hindi). On clicking Go!, the file is downloaded on the user's device.

Methodology:

Our code is designed to handle three different types of inputs: PDF files, text input, and images (in PNG or JPEG format). It performs three main tasks: text extraction, translation, and summarization.

For text extraction, we utilise easyOCR, a Python library that supports the reading of 35 to 40 languages. We have incorporated languages like Tamil, Hindi,

French, Spanish, Chinese, and Arabic. Based on the input language, the OCR model selects the appropriate reader and extracts the text from the provided PDF or image.

The translation task employs the Google API to translate the extracted text into a different language or the user's preferred language. Initially, the API had a character limit of 500, but we have made adjustments to remove this limitation to accommodate PDF files with thousands of words. The translation model preserves names and proper nouns while translating only the sentences as required.

The final and crucial part is text summarization. The code takes the English input file and applies natural language processing techniques such as tokenization and lemmatization. Using an English corpus from our NLP model dataset, the code summarises the text. Additionally, we offer the flexibility for the user to select the desired number of summary lines, allowing for customization.

In the final stages, the code converts the output into a text file and saves it to the desired location on the system. We are also developing a website where all these functionalities will be hosted. Along with the aforementioned features, users will have access to a dashboard upon logging in. They can save the summarised/translated files on the website by simply checking a checkbox.

3.2.2 Library Framework:

3.2.2.1 EasyOCR

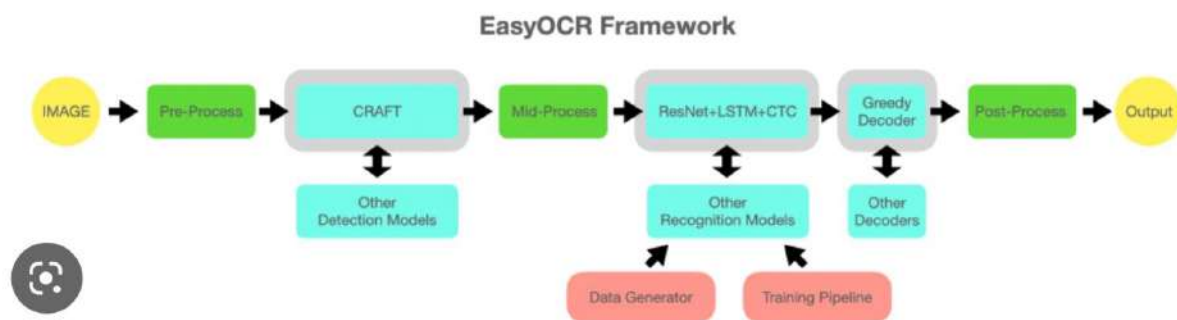


Fig.2. easyOCR workflow

EasyOCR is an open-source optical character recognition (OCR) library that is designed to extract text from images. It provides a simple and user-friendly

interface for performing text extraction tasks using deep learning models. EasyOCR is built on top of the PyTorch framework and leverages pre-trained models to recognize text in various languages. Here is a detailed overview of the architecture and working of EasyOCR:

1. **Image Input:** EasyOCR takes an image as input, which can be in various formats such as JPEG, PNG, or TIFF. The image can contain text in different languages and fonts.
2. **Pre-processing:** Before performing OCR, EasyOCR applies some pre-processing steps on the input image. This may include resizing, normalization, noise reduction, and enhancing image quality to improve the accuracy of text recognition.
3. **Text Detection:** EasyOCR utilizes a text detection algorithm to locate regions of interest in the image where text is present. The goal is to identify bounding boxes around text regions. EasyOCR employs a combination of techniques such as edge detection, contour analysis, or machine learning-based models to achieve text localization.
4. **Text Recognition:** Once the text regions are identified, EasyOCR applies a text recognition model to recognize and extract the actual text from the identified regions. The text recognition model is typically a deep learning-based model, such as a Convolutional Neural Network (CNN) or a Recurrent Neural Network (RNN), which has been trained on a large dataset of images and corresponding ground-truth text.
5. **Language Identification:** EasyOCR performs language identification to determine the language of the extracted text. This step helps in selecting the appropriate language-specific model for further processing.
6. **Post-processing:** After text recognition, EasyOCR applies post-processing techniques to refine and improve the accuracy of the extracted text. This may involve spell-checking, text normalization, removing duplicate characters, or resolving ambiguities based on language-specific rules.
7. **Output:** Finally, EasyOCR provides the recognized text as output, along with the detected bounding boxes and the identified language. The output can be in the form of plain text, structured data, or integrated with other applications or services for further analysis or processing.

EasyOCR supports a wide range of languages and provides pre-trained models for text recognition in multiple scripts, including Latin, Cyrillic, Chinese, Japanese, Korean, and many others. Users can also fine-tune the models or train their own models on custom datasets to achieve better performance for specific use cases

3.2.2.2 NLTK

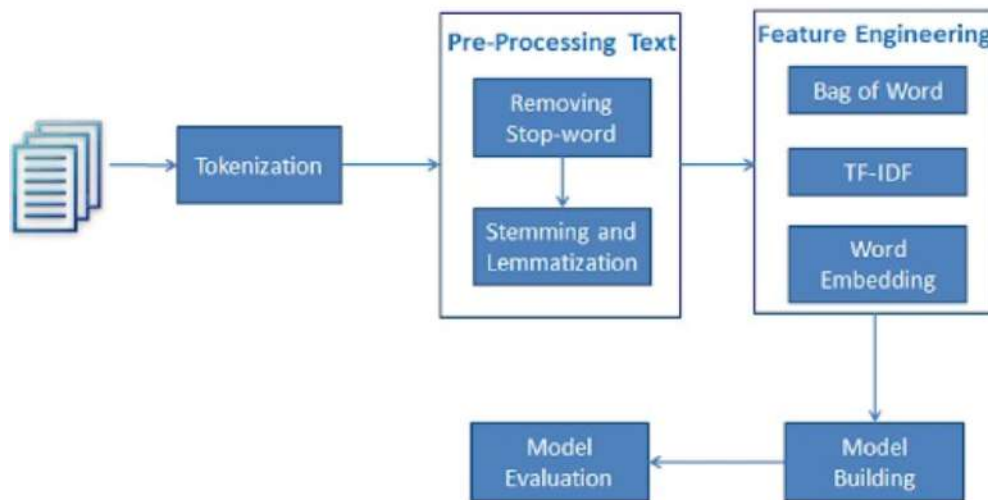


Fig.3. NLTK workflow

NLTK (Natural Language Toolkit) is a powerful Python library widely used for natural language processing (NLP) tasks. It provides a collection of tools, algorithms, and corpora for tasks such as tokenization, stemming, part-of-speech tagging, parsing, semantic reasoning, and more. NLTK is designed to help researchers, developers, and students experiment with and build NLP applications.

1. Here is a high-level overview of how NLTK works and some of its key features:
2. **Text Processing:** NLTK provides a wide range of text processing capabilities, including tokenization, which splits text into individual words or sentences, and stemming, which reduces words to their base or root form (e.g., "running" to "run").
3. **Corpora and Resources:** NLTK offers numerous corpora, which are large collections of text used for language research and development. These corpora cover various domains, such as news, literature, and social media, and are valuable for training and testing NLP models.
4. **Part-of-Speech Tagging:** NLTK includes pre-trained models and algorithms for part-of-speech (POS) tagging, which assigns grammatical tags to each word in a sentence, such as noun, verb, adjective, etc. This information is essential for many downstream NLP tasks.
5. **Parsing and Chunking:** NLTK supports syntactic parsing, which analyses the structure of sentences based on a formal grammar. It can

identify phrases, dependencies, and syntactic relationships within the text.

6. **Named Entity Recognition (NER):** NER is the process of identifying and classifying named entities in text, such as person names, locations, organisations, and dates. NLTK provides tools and models for NER, allowing you to extract and classify entities from text.
7. **Machine Learning Integration:** NLTK integrates well with popular machine learning libraries in Python, such as scikit-learn and TensorFlow. It enables you to train and evaluate custom NLP models using machine learning algorithms and techniques.
8. **Sentiment Analysis:** NLTK offers resources and algorithms for sentiment analysis, a task that involves determining the sentiment or opinion expressed in a piece of text. It can be used to analyse social media sentiment, customer reviews, and more.
9. **WordNet:** WordNet is a lexical database included in NLTK that provides semantic relationships between words. It allows you to find synonyms, antonyms, hypernyms (superordinate terms), hyponyms (subordinate terms), and other lexical relationships.
10. **Language-Specific Support:** NLTK supports multiple languages and provides tools for processing and analysing text in different languages. It includes resources such as tokenizers, stemmers, POS taggers, and corpora for several languages.
11. **Extensibility and Customization:** NLTK is designed to be extensible and customizable. It allows you to build and incorporate your own models, algorithms, and linguistic resources into your NLP applications.

3.2.2.3 GOOGLETRANS

Googletrans is a Python library that provides a convenient interface to the Google Translate API, allowing users to perform language translation tasks programmatically. It leverages the power of Google's machine translation technology to provide accurate and efficient language translation services.

Here is a detailed overview of how Googletrans works and its architecture:

1. **API Integration:** Googletrans integrates with the Google Translate API, which is a cloud-based service provided by Google. The library acts as a wrapper around the API, making it easier for developers to interact with the translation service.
2. **Text Input:** Users provide the text they want to translate as input to Googletrans. This can be a single word, a sentence, or a larger text corpus. Additionally, users specify the source language of the input text and the target language they want the text to be translated into.
3. **Request Handling:** Googletrans constructs a request to the Google Translate API, including the input text, source language, and target language information. The request is sent to the API for processing.
4. **Google Translate API:** The Google Translate API receives the request from Googletrans and performs the translation using Google's machine learning-based translation models. The API utilizes a variety of techniques, including neural machine translation (NMT), statistical machine translation (SMT), and other language processing algorithms to generate translations.
5. **Translation Response:** Once the translation process is complete, the Google Translate API sends a response back to Googletrans. The response includes the translated text in the target language.
6. **Output:** Googletrans extracts the translated text from the API response and provides it as output to the user. The translated text can be accessed programmatically and used in various applications or further processing.

It's important to note that Googletrans is a client-side library and does not perform any translation on its own. It acts as an interface to the Google Translate API, which handles the actual translation process using Google's powerful translation infrastructure.

The architecture of Googletrans is based on a client-server model, where the library acts as the client that sends translation requests to the Google Translate API

server. The API server performs the translation and sends the response back to the client.

Googletrans also provides additional features, such as language detection, which can automatically detect the source language of a given text, and transliteration, which converts text from one script to another while keeping the pronunciation similar.

Overall, Googletrans offers a convenient way to integrate language translation capabilities into Python applications using the Google Translate API. It simplifies the process of accessing and utilizing Google's translation services, making it easier for developers to incorporate multilingual support into their projects.

3.3. Proposed System Model(UML Diagram)

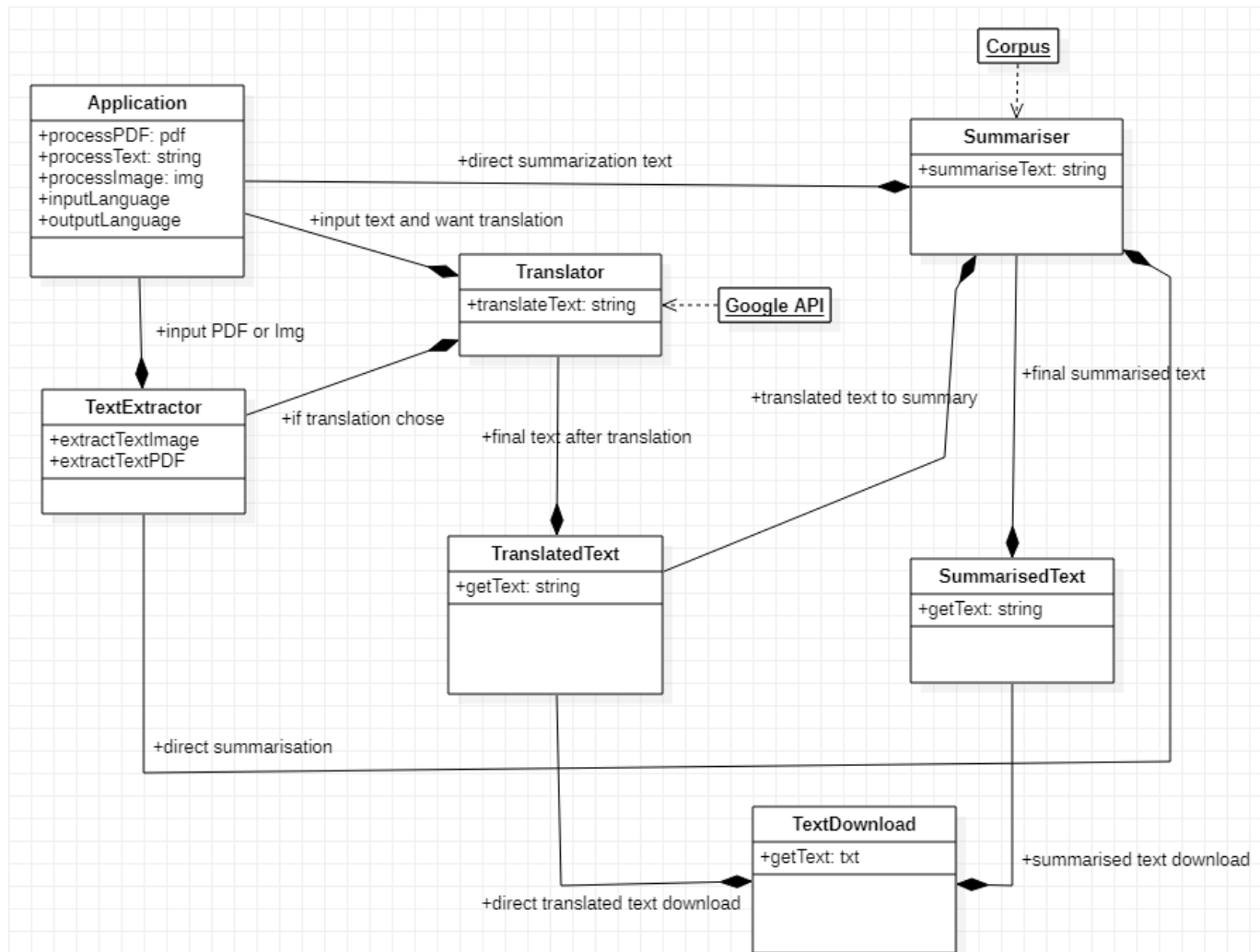


Fig.4. UML diagram

4. PROPOSED SYSTEM ARCHITECTURE AND DESIGN

4.1. Introduction

In this digital era, the demand for language translation, text extraction from images, and text summarization has grown significantly. To address these needs, we present our proposed system architecture and design, which not only incorporates these essential functionalities but also provides an enhanced user experience through a step-by-step and user-friendly approach.

Our system aims to go beyond the conventional capabilities by offering a unique feature: text summarization in the user's preferred language. We understand the challenges individuals face when encountering lengthy paragraphs or documents, where a concise summary can save time and accommodate their busy schedules. With our platform, users can effortlessly input text or upload an image containing text in any language of their choice. They are then empowered to select their desired target language for translation and request a summary of the text's content, all with the support of cutting-edge Natural Language Processing (NLP) techniques and corpora.

To ensure a seamless and efficient user experience, we have designed a proper website with a sleek and intuitive user interface. Leveraging Flask technology, our architecture offers a robust and scalable solution, capable of handling diverse language needs. Users have the flexibility to choose which steps they want to engage with, allowing them to skip unnecessary processes and customise their experience according to their requirements.

Our system architecture encompasses three core functionalities: language translation, text extraction from images, and text summarization. By integrating state-of-the-art NLP algorithms and corpora, we empower users to overcome language barriers and information overload. Whether it's translating text from one language to another, extracting text from images to make it accessible and editable, or generating concise summaries of extensive content, our project provides a comprehensive solution.

With our proposed system architecture and design, we aim to revolutionise language-related tasks and empower users with a seamless and accessible solution. By making language translation and summarization easy and efficient, we enable individuals to effectively navigate through multilingual content, save time, and make informed decisions. Say goodbye to language barriers and information overload – our system is designed to enhance the language experience for all users.

4.2 Requirement Analysis

4.2.1 Functional Requirements

4.2.1.1 Product Perspective

Our proposed system is designed to be a standalone product that addresses the growing demand for language translation, text extraction from images, and text summarization. It provides a comprehensive solution within a single platform, eliminating the need for users to rely on multiple separate applications or services.

From a broader perspective, our product fits into the larger landscape of language processing and information management tools. It leverages cutting-edge Natural Language Processing (NLP) techniques and corpora to deliver accurate translations, efficient text extraction, and concise summaries. However, unlike traditional language processing tools that focus on specific aspects such as translation or extraction, our system integrates all these functionalities into a unified and user-friendly experience.

Our product aims to cater to a wide range of users, including individuals, professionals, researchers, and businesses. It provides an accessible and efficient means of handling multilingual content, enabling users to overcome language barriers and information overload. By offering a step-by-step and customizable approach, users have the flexibility to choose which functionalities they require, allowing them to tailor their experience according to their specific needs.

In terms of integration, our system is designed to be seamlessly integrated into existing platforms or websites. It can serve as a valuable add-on feature for applications that deal with content management, translation services, or information extraction. Additionally, our system can be deployed as a standalone web application, providing users with a dedicated platform for their language-related tasks.

Considering the competitive landscape, our product differentiates itself by offering text summarization alongside language translation and text extraction. This unique feature sets us apart and positions our system as a comprehensive solution for users who seek both translation and summarization capabilities.

Overall, our proposed system stands as an innovative and user-centric product within the domain of language processing and information management. By combining multiple functionalities into a single platform, we aim to provide users with a streamlined experience that simplifies their language-related tasks and enhances their overall productivity.

4.2.1.2 Product Features

1. **Language Translation:** Our system provides robust language translation capabilities, allowing users to translate text from one language to another. Users can input text directly or upload images containing text for translation. The system supports a wide range of languages, enabling seamless communication and understanding across language barriers.

2. **Text Extraction from Images and documents:** Users can easily extract text from images by uploading the image file. The system utilizes Optical Character Recognition (OCR) technology to recognize and extract the text, making it editable and translatable. This feature is particularly useful when dealing with documents, signs, or any image containing textual information.

3. **Text Summarization:** Our system offers text summarization functionality, allowing users to obtain concise summaries of lengthy paragraphs or documents. This feature saves time and helps users quickly grasp the main points and essential information within a large body of text. Users can select their desired summary length or specify the level of detail required.

4. **Multilingual Support:** The system supports a wide range of languages, enabling users to work with content in their preferred language. Users can translate, extract, and summarize text in various languages, catering to their specific language needs. This feature facilitates cross-lingual communication and information processing.

5. **User-Friendly Interface:** The system features a sleek and intuitive user interface, designed to provide a seamless and user-friendly experience. The interface allows users to easily navigate through different functionalities, input their text or upload images, select target languages, and customize options according to their preferences.

6. **Step-by-Step Process:** Our system offers a step-by-step approach, allowing users to choose which functionalities they want to utilize. Users can skip certain steps based on their requirements, providing flexibility and efficiency in handling their language-related tasks.

7. **Integration Capabilities:** The system can be easily integrated into existing platforms or websites. It can serve as an add-on feature for applications related to content management, translation services, or information extraction. The integration enables users to access language processing functionalities seamlessly within their preferred platforms.

8. Cutting-Edge NLP Technology: The system utilizes state-of-the-art Natural Language Processing (NLP) techniques and corpora to deliver accurate translations, efficient text extraction, and concise summaries. The use of advanced algorithms and linguistic analysis ensures high-quality results and enhances the overall user experience.

9. Customization Options: Users have the ability to customize various aspects of the system to suit their specific needs. They can adjust translation settings, specify summarization preferences, and configure other parameters to align with their requirements, ensuring a personalized and tailored experience.

10. Accessibility and Availability: Our system is accessible through a web-based interface, making it available to users across different devices and platforms. It can be accessed anytime, anywhere, as long as there is an internet connection, providing convenience and flexibility to users.

4.2.1.3 User Characteristics

1. Multilingual Individuals: Our system caters to individuals who are fluent in multiple languages or frequently engage in multilingual communication. These users may need assistance in translating text, extracting information from images, and summarizing content in different languages.

2. Language Learners: Language learners, including students, professionals, or enthusiasts, can benefit from our system. They can use the translation feature to understand texts in foreign languages, extract vocabulary from images, and obtain summarized versions of complex materials for learning purposes.

3. Researchers and Academics: Researchers and academics working with multilingual data or studying language-related topics can utilize our system's language processing capabilities. They can analyze translations, extract data from images, and generate summaries to support their research and academic endeavors.

4. Content Curators and Editors: Professionals involved in content creation, curation, and editing can leverage our system to streamline their workflows. They can quickly translate content, extract text from images for further editing, and generate summaries to review or assess the relevance of lengthy texts.

5. Business Professionals: Professionals working in international or multilingual settings can utilize our system to overcome language barriers. They can translate important documents, extract information from images or presentations, and obtain summarized versions of reports or articles to save time and enhance productivity.

6. Information Seekers: Individuals seeking specific information from lengthy documents or web pages can benefit from our text summarization feature. They can obtain concise summaries that capture the key points and main ideas, enabling them to quickly assess the relevance of the content.

7. General Users: Our system is designed to be user-friendly, making it accessible to general users who may not have extensive knowledge of language processing technologies. These users can utilize our system to translate simple texts, extract information from images, or obtain brief summaries without requiring specialized expertise.

8. Non-Technical Users: Our system caters to users who may have limited technical knowledge but still require language processing functionalities. The user-friendly interface and customizable options make it accessible to individuals from various backgrounds and skill levels.

9. Mobile Users: With the web-based interface and availability across different devices, our system accommodates mobile users who prefer accessing language processing tools on their smartphones or tablets. They can perform translations, extract text, and obtain summaries while on the go.

4.2.1.4 Assumption and Dependencies

1. Language Resources: The system relies on language resources such as pre-trained language models, translation databases, and summarization corpora. It assumes the availability and continuous update of these resources to ensure accurate translations and summaries in multiple languages.

2. Internet Connectivity: The system assumes that users have a stable internet connection to access the web-based platform and utilize the language processing functionalities. It depends on internet connectivity for data retrieval, language model access, and external APIs.

3. Image Quality: For text extraction from images, the system assumes reasonably clear and legible images with sufficient resolution. Poor image quality, low contrast, or distorted text may affect the accuracy of the text extraction process.

4. Supported Languages: The system assumes support for a wide range of languages for translation, text extraction, and text summarization. However, it may have limitations in terms of availability and accuracy for certain less commonly used or specialized languages.

5. User Input Quality: The accuracy of translation and text summarization heavily depends on the quality and clarity of the user's input. Ambiguous or poorly structured input may lead to less accurate translations or summaries.

6. NLP Limitations: The system utilizes NLP techniques and algorithms for language processing tasks. It relies on the effectiveness and limitations of these techniques, which may result in occasional inaccuracies or suboptimal results in complex language scenarios or context-specific situations.

7. Privacy and Security: The system assumes appropriate measures for data privacy and security. It relies on secure data transmission, storage, and user authentication to protect sensitive information shared by the users.

8. Platform Compatibility: The system assumes compatibility with common web browsers, operating systems, and devices. It depends on the availability of necessary software libraries, APIs, and system resources to ensure smooth operation across different platforms.

9. External APIs and Services: The system may depend on external APIs and services for specific functionalities such as machine translation, optical character recognition (OCR), or text summarization. It assumes the availability and reliability of these external services to provide seamless integration and accurate results.

10. User Familiarity: The system assumes that users are familiar with basic computer operations and have a basic understanding of language processing concepts. While the system strives for a user-friendly interface, some level of user familiarity and adaptability is expected.

4.2.1.5 Domain Requirements

1. Language Support: The system should support a wide range of languages for translation, text extraction, and text summarization. It should be capable of handling major languages worldwide, including but not limited to English, Spanish, French, German, Chinese, Japanese, and Arabic.

2. Translation Accuracy: The system should provide accurate translations between different languages. It should take into account linguistic nuances, idiomatic expressions, and cultural context to deliver high-quality translations that preserve the intended meaning of the original text.

3. Text Extraction from Images: The system should be able to extract text from images accurately. It should employ optical character recognition (OCR) techniques

to identify and convert text embedded within images into editable and searchable formats.

4. Text Summarization: The system should generate concise and informative summaries of text documents or passages. The summaries should capture the key points and main ideas of the original content, allowing users to quickly grasp the essence of lengthy texts.

5. User-Friendly Interface: The system should have an intuitive and user-friendly interface that allows users to easily input text, upload images, select target languages, and specify summarization preferences. The interface should be responsive, visually appealing, and accessible on different devices and screen sizes.

6. Customization Options: The system should provide customization options to users, allowing them to adjust translation settings, specify summarization length, and personalize the user experience based on their preferences and requirements.

4.2.1.6 User Requirements

User Requirements:

1. Easy Input: Users should be able to input text easily by typing in the provided text input field. They should also have the option to upload text documents or images containing text for processing.

2. Language Selection: Users should be able to select their desired source and target languages for translation and summarization. The system should provide a user-friendly language selection interface with a wide range of language options.

3. Translation Accuracy: Users expect accurate translations that convey the intended meaning of the original text. The system should prioritize translation accuracy and provide reliable translations even for complex or specialized content.

4. Text Extraction from Images: Users should be able to upload images containing text and expect the system to accurately extract the text from those images using OCR technology. The extracted text should be editable and searchable.

5. Text Summarization Options: Users should have control over the length and level of detail in the generated summaries. The system should provide options for users to specify the desired summarization length or choose from predefined summary lengths.

6. Speed and Responsiveness: Users expect the system to process their requests quickly and provide timely results. The system should be responsive, providing near-instant translations and summaries to minimize wait times.

7. Clear and Understandable Summaries: Users expect the generated summaries to be clear, concise, and easily understandable. The system should prioritize the extraction of key points and main ideas to ensure the summaries provide valuable insights.

8. User-Friendly Interface: The system should have an intuitive and user-friendly interface that guides users through the translation and summarization process. The interface should be visually appealing, well-organized, and easy to navigate.

4.2.2 Non Functional Requirements

4.2.2.1 Product Requirements

4.2.2.1.1 Efficiency(in terms of time and space)

Time Efficiency:

- Translation Speed: The translation process should be fast, providing near-instantaneous translations to users. The system should utilize efficient translation algorithms and techniques to minimize processing time.
- Summarization Time: The summarization process should be efficient, delivering quick summaries even for lengthy texts. Employing advanced NLP algorithms and techniques for summarization can help optimize processing time.
- Image Text Extraction: The OCR (Optical Character Recognition) technology used for extracting text from images should be efficient, accurately recognizing and extracting text from images in a timely manner.

Space Efficiency:

- Memory Usage: The system should be designed to optimize memory usage and avoid excessive memory consumption during translation and summarization tasks. Efficient data structures and algorithms should be employed to minimize memory usage.

4.2.2.1.2 Reliability

1. **Translation Accuracy:** The system should strive for high translation accuracy to ensure that the translated text conveys the intended meaning. It should employ reliable translation models, such as those built on extensive training data, to minimize errors and inaccuracies in translations.
2. **Text Extraction Accuracy:** The OCR (Optical Character Recognition) technology used for extracting text from images should be reliable and accurate. It should accurately recognize and extract text from various image formats, ensuring minimal errors or omissions.
3. **Summarization Quality:** The system should generate summaries that are reliable and faithful representations of the original text. Advanced NLP algorithms and techniques should be used to ensure the coherence, relevance, and accuracy of the generated summaries.
4. **Robustness:** The system should be designed to handle a wide range of input variations, including different languages, writing styles, and formats. It should be able to handle complex sentences, technical terms, and domain-specific content reliably.

4.2.2.1.3 Portability

1. **Cross-Platform Compatibility:** The system should be designed to work seamlessly across different operating systems such as Windows, macOS, Linux, and various mobile platforms like Android and iOS. It should utilize technologies, frameworks, and programming languages that are widely supported across these platforms to ensure broad compatibility.
2. **Web-based Accessibility:** Implementing the system as a web application can enhance portability by allowing users to access it through standard web browsers without the need for platform-specific installations. This approach enables users to utilize the system on various devices with internet connectivity, including desktops, laptops, tablets, and smartphones.
3. **Dependency Management:** Careful management of dependencies is essential for portability. The system should utilize well-documented and widely-used libraries, frameworks, and APIs to minimize platform-specific dependencies. It should also ensure compatibility with different versions of these dependencies to facilitate smooth migration and deployment.
4. **Data Format Compatibility:** The system should support commonly used data formats for input and output, such as plain text, image formats (JPEG, PNG), and popular document formats (PDF, DOCX).

It should handle different character encodings, languages, and file structures to ensure seamless data exchange across platforms.

5. **Configuration and Customization:** Providing user-configurable settings and preferences can enhance portability. Users should be able to customize language options, translation models, summarization parameters, and other relevant settings to adapt the system to their specific needs and preferences.

4.2.2.1.4 Usability

1. **Intuitive User Interface:** The system should have a clean and intuitive user interface that is easy to navigate. The layout and design should be user-friendly, with clear labels, icons, and instructions to guide users through the various functionalities. The interface should be visually appealing and responsive, ensuring a pleasant user experience.
2. **Simple and Clear Workflow:** The system should follow a logical and straightforward workflow, guiding users through the necessary steps to perform translation, text extraction, and summarization tasks. Each step should be clearly defined, and users should be provided with clear instructions and prompts at each stage. This simplifies the process and reduces the learning curve for new users.
3. **Minimal Input Requirements:** The system should require minimal input from users while still delivering accurate results. Users should not be burdened with complex configurations or excessive data entry. The system should intelligently handle defaults and make use of contextual information whenever possible to streamline the user experience.
4. **Multilingual Support:** Since the system deals with language translation, it should support multiple languages, both for input and output. Users should be able to interact with the system in their preferred language and receive translations and summaries in the language of their choice. This enhances usability for users from diverse linguistic backgrounds.
5. **Performance Optimization:** The system should be optimized for speed and responsiveness. Users should experience minimal delays when performing tasks, ensuring a smooth and efficient user experience. This can be achieved through various techniques, such as implementing caching mechanisms, optimizing algorithms, and utilizing efficient data structures.

4.2.2.2 Organizational Requirements

4.2.2.2.1 Implementation Requirements (in terms of deployment)

1. **Programming Languages:** The system can be implemented using programming languages suitable for web development, such as Python, JavaScript, HTML, and CSS. Python can be used for backend development, while JavaScript, HTML, and CSS are used for frontend development.
2. **Frameworks and Libraries:** Utilize appropriate frameworks and libraries to facilitate the development process. For example, Flask or is used as the web framework for backend development, and libraries like NLTK (Natural Language Toolkit), EasyOCR, are utilized for NLP, image processing, and text extraction tasks.
3. **Web Development Technologies:** Employ web development technologies to build the user interface and enable user interaction. This includes HTML for markup, CSS for styling and layout, and JavaScript for client-side functionality and interactivity.
4. **APIs and Services:** Integration with external APIs or services may be required for specific functionalities, such as language translation or summarization. For example, using the Google Translate API for language translation.
5. **NLP and Machine Learning Models:** Implement NLP algorithms and machine learning models to perform language-related tasks, including translation, text extraction, and summarization. This may involve using pre-trained models, such as those provided by NLTK or other libraries, or training custom models using relevant datasets.
6. **Image Processing Techniques:** Implement image processing techniques to extract text from images. This may involve using OCR (Optical Character Recognition) libraries like Tesseract or leveraging computer vision algorithms to identify and extract text regions from images.

4.2.2.2.2 Engineering Standard Requirements

Given the domain of our application, no such engineering standards exist.

4.2.2.3 Operational Requirements

4.2.2.3.1 Economic

The cost of internet and electricity to run the models

4.2.2.3.2 Environmental

Energy Efficiency: Design the system to be energy-efficient by optimizing resource utilization and minimizing power consumption. Use energy-efficient hardware components and implement algorithms that reduce computational demands. Employ techniques such as sleep mode, power management, and resource scheduling to minimize energy usage during idle periods.

4.2.2.3.3 Social

User Accessibility: Design the system to be accessible to users with diverse needs, including those with disabilities. Ensure compatibility with assistive technologies such as screen readers, keyboard navigation, and alternative input methods. Follow accessibility guidelines such as WCAG (Web Content Accessibility Guidelines) to provide an inclusive user experience.

Multilingual Support: Offer multilingual support to accommodate users from different linguistic backgrounds. Provide language options for user interfaces, translation services, and content summaries. Ensure accurate translation and culturally sensitive representations to enhance user satisfaction and engagement.

User Privacy and Data Protection: Implement robust security measures to protect user data and privacy. Comply with relevant data protection laws and regulations, such as GDPR (General Data Protection Regulation). Obtain user consent for data collection, storage, and processing. Safeguard sensitive information and use encryption techniques when transmitting data over networks.

Ethical Considerations: Adhere to ethical principles and guidelines in the design and operation of the system. Respect user autonomy, privacy, and confidentiality. Avoid biased or discriminatory algorithms and ensure fairness in processing user data. Conduct regular ethical reviews to address any potential ethical concerns that may arise.

Social Impact Assessment: Assess the potential social impact of the system on users and society as a whole. Consider factors such as information accessibility, information overload, and the potential for misinformation dissemination. Implement measures to mitigate negative impacts and promote responsible use of the system.

Community Engagement: Foster community engagement by soliciting user feedback, conducting user surveys, and incorporating user suggestions into system improvements. Encourage user participation and collaboration to create a sense of ownership and promote user-centric development.

Transparency and Accountability: Maintain transparency in system operations and decision-making processes. Clearly communicate system functionalities, limitations, and any potential biases. Provide accessible documentation and support channels for users to seek assistance and report issues.

4.2.2.3.4 Political

There are no political operational requirements for this project.

4.2.2.3.5 Ethical

Controlled access to the code and the use of a safe system to safeguard the privacy and appropriate use of the data.

4.2.2.3.6 Health and Safety

User Data Protection: Safeguard user data and privacy to prevent unauthorized access, data breaches, or identity theft. Implement robust security measures, encryption protocols, and access controls to ensure the confidentiality and integrity of user information.

System Stability and Reliability: Ensure the system is designed and implemented to operate reliably and avoid disruptions that could lead to user frustration or potential errors. Perform regular maintenance, updates, and backups to minimize system downtime and ensure smooth operation.

4.2.2.3.7 Sustainability

1. **Continuous improvement:** This means that the model should be regularly updated and improved to ensure that it is accurate and reliable. This can be done by using new data, new algorithms, and new techniques.
2. **Data validation:** This means that the data used to train the model should be regularly checked to ensure that it is accurate and complete. This can be done by using statistical methods and by manually inspecting the data.
3. **Model evaluation:** This means that the model should be regularly evaluated to ensure that it is performing as expected. This can be done by using metrics such as accuracy, precision, and recall.
4. **Model retraining:** This means that the model should be periodically retrained on new data to ensure that it remains accurate and reliable. This is especially important when the data used to train the model becomes outdated or when the model is used for a new task.

4.2.2.3.8 Legality

1. **Intellectual Property:** Respect intellectual property rights by ensuring that the system operates within the legal boundaries of copyright, trademarks, patents, and other relevant intellectual property laws. Avoid unauthorized use or distribution of copyrighted material, including text, images, and other content.
2. **Data Privacy:** Comply with applicable data protection and privacy laws to safeguard user data and ensure user privacy. Implement proper security measures to protect user information from unauthorized access, use, or disclosure. Obtain user consent when collecting and processing personal data, and handle user data in accordance with relevant privacy regulations.
3. **Compliance with Regulations:** Adhere to relevant laws, regulations, and industry standards specific to the system's operation.
4. **Ethical Considerations:** While ethics and legality are distinct, they are closely related. Consider ethical guidelines and principles when designing and operating the system. Promote fair and unbiased practices, avoid the creation or perpetuation of discriminatory or harmful content, and ensure transparency and accountability in system operations.
5. **User Agreement and Terms of Service:** Establish clear user agreements and terms of service that outline the rights and responsibilities of both the system provider and the users. Clearly communicate the limitations of the system, any legal restrictions, and user obligations. Ensure that users consent to the terms and conditions before using the system.
6. **Compliance Monitoring and Auditing:** Regularly monitor and audit the system's compliance with relevant legal requirements. Stay updated with changes in laws and regulations that may impact the system's operation. Implement internal controls and processes to ensure ongoing compliance with legal obligations.

4.2.2.3.9 Inspectability

Publishing the model on a public platform so that users and researchers can access it.

4.2.3 System Requirements

4.2.3.1 H/W Requirements

1. Processor: A modern and capable processor to handle the computational requirements of the NLP tasks involved in translation, text extraction, and summarization.
2. The processor should have multi-core capabilities to support parallel processing and improve performance. Preferably Intel i5 or higher
3. Memory (RAM): Adequate memory to efficiently process and manipulate large amounts of text data. Preferably 8GB or more
4. Storage: Sufficient storage space to store the application code, language corpora, and any associated data or resources.
5. Networking: Reliable network connectivity to ensure smooth working of the models.

4.2.3.2 S/W Requirements

Operating System: Windows 10 or higher or MacOS

1. Backend Framework: Flask
2. Programming Languages:
 - Python: For NLP tasks, OCR, Flask.
 - JavaScript, HTML, CSS: For frontend development and interactivity
3. NLP Libraries and Tools:
 - NLTK (Natural Language Toolkit)
 - Google Cloud Translation API
4. Image Processing Libraries:
 - EasyOCR
 - PyPDF2
5. Version Control: Git, GitHub
6. Integrated Development Environment (IDE): VSCode
7. Google Colab

5. RESULTS AND DISCUSSION

Code Snippets:

Frontend Code:

Main Page

```
<!DOCTYPE html>
<html>
  <head>
    <title>File Upload and Translation Website</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="style.css">
    <style>
      body {
        background-image: url("background.jpeg");
        background-repeat: no-repeat;
        background-position: center center;
        background-size: cover;
        background-color: #27c2ff;
        font-family: Arial, sans-serif;
      }

      .container {
        margin: auto;
        max-width: 1000px;
        padding: 20px;
        background-color: rgba(255, 140, 0, 0.4);
        background-position: center center;
        box-shadow: 0px 0px 5px 0px rgba(0, 0, 0, 0.2);
      }

      h1 {
        text-align: center;
      }

      form {
        text-align: center;
      }

      label {
        display: inline-block;
        text-align: left;
```

```

        font-weight: bold;
        margin-bottom: 10px;
    }

    input[type="file"] {
        margin-bottom: 20px;
    }

    select {
        margin-bottom: 20px;
    }

    input[type="submit"] {
        background-color: #4CAF50;
        color: #fff;
        border: none;
        padding: 10px 20px;
        border-radius: 5px;
        cursor: pointer;
        font-size: 16px;
    }

    input[type="submit"]:hover {
        background-color: #3e8e41;
    }

    .logo {
        display: block;
        margin: 0 auto;
        max-width: 200px;}
</style>
</head>
<body>

<div class="container">
    
    <form>
        <div class="row">
            <div class="col-md-6">
                <div id="dropzone" style="border: 2px dashed
#646464; padding: 40px; text-align: center;">

```

```

        <p>Drag and drop PDF/image or type text</p>
        <textarea id="text-input" rows="10"
cols="50"></textarea>
    </div>
</div>
<div class="col-md-6">

    </div>
</div>
<label for="file">Browse File:</label>
<input type="file" id="file" name="file"><br><br>
<label for="language">Input document
language:</label>
    <select id="language" name="language">
        <option value="en">English</option>
        <option value="hi">Hindi</option>
        <option value="fr">French</option>
        <option value="es">Spanish</option>
        <option value="ch">Chinese</option>
        <option value="ar">Arabic</option>
        <option value="ta">Tamil</option>

    </select><br><br>

    <label>
        <input type="checkbox" id="myCheckbox"> Translate?
    </label><br>

    <select id="myDropdown" disabled>
        <option value="option1">English</option>
        <option value="option2">Spanish</option>
        <option value="option3">French</option>
        <option value="option4">Chinese</option>
        <option value="option5">Arabic</option>
        <option value="option6">Tamil</option>
        <option value="option7">Hindi</option>

    </select><br>

    <script>
        const checkbox =
document.getElementById("myCheckbox");

```

```

        const dropdown =
document.getElementById("myDropdown");

        checkbox.addEventListener("click", function() {
            if (checkbox.checked) {
                dropdown.removeAttribute("disabled");
            } else {
                dropdown.setAttribute("disabled", true);
            }
        });
    </script>

    <label>
        <input type="checkbox" name="summary" value="v2">
        Summary needed?
    </label><br>

    <input type="submit" value="Go!">
</form>
</div>

<script>
    const checkboxTranslate =
document.getElementById("myCheckbox");
    const checkboxSummary =
document.getElementsByName("summary")[0];
    const droppdown =
document.getElementById("myDropdown");
    const form = document.querySelector("form");

    checkboxTranslate.addEventListener("change",
function() {
        if (checkboxTranslate.checked) {
            droppdown.removeAttribute("disabled");
        } else {
            droppdown.setAttribute("disabled", true);
        }
        handleFormSubmission();
    });

    checkboxSummary.addEventListener("change", function()
{

```

```

        handleFormSubmission();
    });

    function handleFormSubmission() {
        const isTranslateChecked = checkboxTranslate.checked;
        const isSummaryChecked = checkboxSummary.checked;

        // Remove all existing event listeners
        form.removeEventListener("submit",
handleTranslateAndSummary);
        form.removeEventListener("submit", handleTranslate);
        form.removeEventListener("submit", handleSummary);

        if (isTranslateChecked && isSummaryChecked) {
            form.addEventListener("submit",
handleTranslateAndSummary);
        } else if (isTranslateChecked && !isSummaryChecked) {
            form.addEventListener("submit", handleTranslate);
        } else if (isSummaryChecked && !isTranslateChecked) {
            form.addEventListener("submit", handleSummary);
        }
    }

    function handleTranslateAndSummary(event) {
        event.preventDefault();
        const selectedLanguage = dropdown.value;
        if (selectedLanguage==="option3")
            // Perform translation and summary tasks here
            {downloadDocument("TranslatedSummaryDocument.txt",
"TranslatedSummaryDocument");
        }
        else if (selectedLanguage==="option6")
        {
            downloadDocument("TranslatedSummaryDocument.txt",
"TranslatedSummary");
        }
        else if (selectedLanguage==="option1")
        {
            downloadDocument("TranslatedSummaryDocument.txt",
"TranslatedSummary");
        }
    }
}

```

```

function handleTranslate(event) {
    event.preventDefault();
    // Perform translation task here
    downloadDocument("TranslatedDocument.txt",
"TranslatedDocument");
}

function handleSummary(event) {
    event.preventDefault();
    // Perform summary task here
    downloadDocument("SummaryDocument.txt",
"SummaryDocument");
}

function downloadDocument(filename, content) {
    const element = document.createElement("a");
    element.setAttribute("href",
"data:text/plain;charset=utf-8," + encodeURIComponent(content));
    element.setAttribute("download", filename);
    element.style.display = "none";
    document.body.appendChild(element);
    element.click();
    document.body.removeChild(element);
}
</script>
</body>
</html>

```

Login Page:

```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/c
ss/font-awesome.min.css">
<style>
body {
font-family: Arial, Helvetica, sans-serif;
background-image: url("loginbg.jpg");

```



```
background-repeat: no-repeat;
background-position: center center;
background-size: cover;

}

* {
box-sizing: border-box;
}

.container {
position: relative;
border-radius: 5px;
/*background-color: #f2f2f2;*/
background-color: rgba(255, 140, 0, 0.4);

padding: 20px 0 30px 0;
}

input,
.btn {
width: 100%;
padding: 12px;
border: none;
border-radius: 4px;
margin: 5px 0;
opacity: 0.85;
display: inline-block;
font-size: 17px;
line-height: 20px;
text-decoration: none;
}

input:hover,
.btn:hover {
opacity: 1;
}

.fb {
background-color: #3B5998;
```

```
color: white;
/*width: 30%;*/
display: flex;
justify-content: center;
align-items: center;
}

.twitter {
background-color: #55ACEE;
color: white;
display: flex;
justify-content: center;
align-items: center;
}

.google {
background-color: #ffffff;
color: rgb(0, 0, 0);
display: flex;
justify-content: center;
align-items: center;
}

input[type=submit] {
background-color: #04AA6D;
color: white;
cursor: pointer;
}

input[type=submit]:hover {
background-color: #45a049;
}

.col {
/*float: left;*/
width: 50%;
margin: auto;
padding: 0 50px;
margin-top: 6px;
}
```

```
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* vertical line
.vl {
  position: absolute;
  left: 50%;
  transform: translate(-50%);
  border: 2px solid #ddd;
  height: 175px;
}

/* text inside the vertical line */
/*
.vl-innertext {
  position: absolute;
  top: 50%;
  transform: translate(-50%, -50%);
  background-color: #f1f1f1;
  border: 1px solid #ccc;
  border-radius: 50%;
  padding: 8px 10px;
}*/

.hide-md-lg {
  display: none;
}

/* bottom container */
.bottom-container {
  text-align: center;
  background-color: #82009c;
  border-radius: 0px 0px 4px 4px;
}

.logo {
  display: block;
  margin: 0 auto;
```

```

        background-color: #ffffff;
        max-width: 200px;}

@media screen and (max-width: 650px) {
.col {
    width: 100%;
    margin-top: 0;
}

.v1 {
    display: none;
}

.hide-md-lg {
    display: block;
    text-align: center;
}
}

</style>
</head>
<body>

<div class="container">
    <form action="/action_page.php">
        <div class="row">
            <!--<h2 style="text-align:center">Login with Social
Media or Manually</h2>-->
            
            <div class="v1">
                <span class="v1-innertext"></span>
            </div>

            <div class="col">
                <a href="#" class="fb btn">
                    <i class="fa fa-facebook fa-fw"></i> Login with
Facebook

                </a>
                <a href="#" class="twitter btn">
                    <i class="fa fa-twitter fa-fw"></i> Login with
Twitter

```

```

        </a>
        <a href="#" class="google btn"><i class="fa
fa-google fa-fw">
        </i> Login with Google
    </a>
</div>

<div class="col">
    <div class="hide-md-lg">
        <p>Or sign in manually:</p>
    </div>

    <input type="text" name="username"
placeholder="Username" required>
    <input type="password" name="password"
placeholder="Password" required>
    <input type="submit" value="Login">
</div>

</div>
</form>
</div>

<div class="bottom-container">
    <div class="row">
        <div class="col">
            <a href="signup.html" style="color:white"
class="btn">Sign up</a>

        </div>

    </div>
    <div class="col">
        <a href="cap_front.html" style="color:black"
class="btn">Skip</a>
    </div>
</div>
</body>
</html>

```

Signup Page

```
<html>
<style>
body {font-family: Arial, Helvetica, sans-serif;
    background-image: url("loginbg.jpg");
    background-repeat: no-repeat;
    background-position: center center;
    background-size: cover;}
* {box-sizing: border-box}

input[type=text], input[type=password] {
    width: 100%;
    padding: 15px;
    margin: 5px 0 22px 0;
    display: inline-block;
    border: none;
    background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
    background-color: #ddd;
    outline: none;
}

hr {
    border: 1px solid #f1f1f1;
    margin-bottom: 25px;
}

button {
    background-color: #04AA6D;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
    opacity: 0.9;
}
```

```
button:hover {
  opacity:1;
}

.cancelbtn {
  padding: 14px 20px;
  background-color: #820096;
}

.cancelbtn, .signupbtn {
  float: left;
  width: 50%;
}

.container {
  /*background-color: rgba(117, 187, 252, 0.4);*/
  background-color: rgba(255, 140, 0, 0.4);
  padding: 16px;
}

.clearfix::after {
  content: "";
  clear: both;
  display: table;
}

.logo {
  display: block;
  margin: 0 auto;
  background-color: #ffffff;
  max-width: 150px;}

@media screen and (max-width: 300px) {
  .cancelbtn, .signupbtn {
    width: 100%;
  }
}

</style>
<body>
```

```

<form action="/action_page.php" style="border:1px solid #ccc">
  <div class="container">
    
    <h1>Sign Up</h1>
    <!--<p>Please fill in this form to create an account.</p>-->

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email"
required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password"
name="psw" required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
    <input type="password" placeholder="Repeat Password"
name="psw-repeat" required>

    <label>
      <input type="checkbox" checked="checked" name="remember"
style="margin-bottom:15px"> Remember me
    </label>

    <p>By creating an account you agree to our <a href="#"
style="color:dodgerblue">Terms & Privacy</a>.</p>

    <div class="clearfix">
      <button type="button" class="cancelbtn"><a
href="loginsignup.html" style="color:white">Sign In</a></button>
      <button type="submit" class="signupbtn"><a
href="cap_front.html" style="color:white"></a>Sign
Up</a></button>
    </div>
  </div>
</form>

</body>
</html>

```


Flask App

```
from flask import Flask, render_template, request
from capstone_backend import read_text, translate_text,
summarize, pdf_read
import os

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('loginsignup.html')

@app.route('/extract_text', methods=['GET', 'POST'])
def extract_text():
    if request.method == 'POST':
        file = request.files['file']
        language=request.form['language']
        file_ext = os.path.splitext(file)[1]
        if file_ext in ('.jpg', '.jpeg', '.png', '.gif'):
            extracted_text = read_text(file)
        elif file_ext == '.pdf':
            extracted_text = pdf_read(file)
        return render_template('extracted_text.html',
text=extracted_text)
    return render_template('extract_text.html')

@app.route('/translate', methods=['GET', 'POST'])
def translate():
    if request.method == 'POST':
        extracted_text = request.form['text']
        translated_text = translate_text(extracted_text)
        return render_template('translated_text.html',
text=translated_text)
    return render_template('translate.html')

@app.route('/summarize', methods=['GET', 'POST'])
def summarize():
    if request.method == 'POST':
        extracted_text = request.form['text']
        summarized_text = summarize(extracted_text, 40)
        return render_template('summarized_text.html',
text=summarized_text)
```

```
        return render_template('summarize.html')

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

NLP and OCR backend

```
nltk.download("stopwords")
nltk.download()

#!/pip install flask

"""IMPORT"""

from easyocr import Reader
import PyPDF2
from PyPDF2 import PdfReader
import fitz
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
#import networkx as nx
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
#from heapq import nlargest
from flask import Flask, render_template

"""IMAGE EXTRACTION"""

# Load model for multiple languages
reader_en_fr = Reader(['en', 'fr', 'es'])
reader_ta_en = Reader(['en', 'ta'])
reader_te_en = Reader(['en', 'te'])
reader_kn_en = Reader(['en', 'kn'])
reader_hi_en = Reader(['en', 'hi'])
reader_ch = Reader(['ch_sim'])
reader_ar = Reader(['ar'])
```

```

def read_text(image_name, model_name, in_line=True):

    # Read the data
    text = model_name.readtext(image_name, detail = 0, paragraph=in_line)

    # Join texts writing each text in new line
    return '\n'.join(text)

ta_text = read_text(picture, reader_ta_en)
print(ta_text)

ch_text = read_text("chinese.png", reader_ch)
print(ch_text)

ar_text = read_text("arabic.png", reader_ar)
print(ar_text)

"""TO READ PDF"""

# Open the PDF file in read-binary mode
def pdf_read():
    with open(pdf, 'rb') as pdf_file:

        # Create a PyPDF2 PdfFileReader object to read the PDF
        pdf_reader = PyPDF2.PdfReader(pdf_file)

        # Read all the pages of the PDF and store the text in a
variable
        text = ''
        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            text += page.extract_text()
#print(text)

a = input("")
a

"""TRANSLATION"""

def translate_text(text, lang):
    from googletrans import Translator
    translator = Translator()
    translated_text = translator.translate(text, dest='en')

```

```

        print(translated_text.text)

"""SUMMARIZE"""

def summarize(txt, n):
    # Tokenize the text into sentences and words
    sentences = sent_tokenize(txt)
    words = word_tokenize(txt)

    # Remove stop words from the words list
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word.lower() not in stop_words]

    # Calculate the frequency of each word
    freq = nltk.FreqDist(words)

    # Calculate the score of each sentence
    scores = {}
    for i, sentence in enumerate(sentences):
        for word in word_tokenize(sentence.lower()):
            if word in freq:
                if i not in scores:
                    scores[i] = freq[word]
                else:
                    scores[i] += freq[word]

    # Get the top n sentences with the highest scores
    top_sentences = nlargest(n, scores, key=scores.get)

    # Sort the top sentences in the order they appear in the original
    text
    summary = ' '.join([sentences[i] for i in sorted(top_sentences)])
    return summary

txt = translated_text.text
summary = summarize(txt, 40) # summarize the text into 11 sentences
print(summary)

"""DESIRED CONVERSION """

from googletrans import Translator

translator = Translator()

```

```

translated_text1 = translator.translate(summary, dest='fr')
print(translated_text1.text)

"""SAVE OUTPUT AS TXT OF SUMMARIZATION (innitiated when user clicks
download)"""

output = translated_text.text

# Open a new file with write mode ('w')
with open('output.txt', 'w') as f:
    # Write the output to the file
    f.write(output)

# Print a message to confirm that the file was saved
print("Output saved output.txt")

```

Code Screenshots:

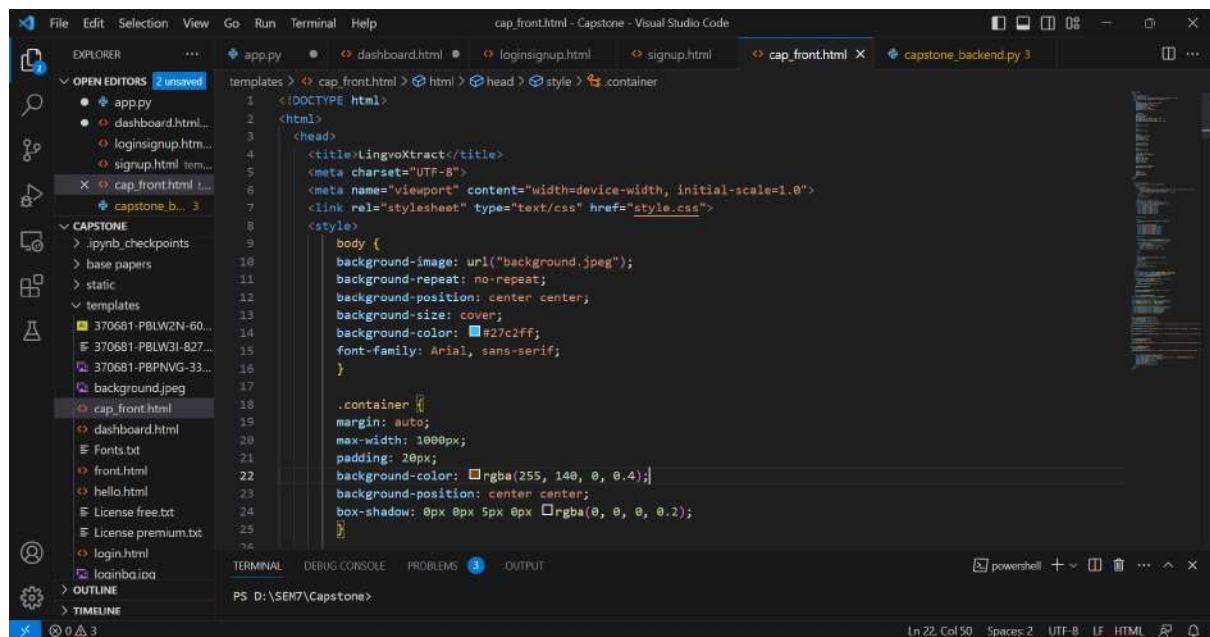


Fig 5. Main page CSS

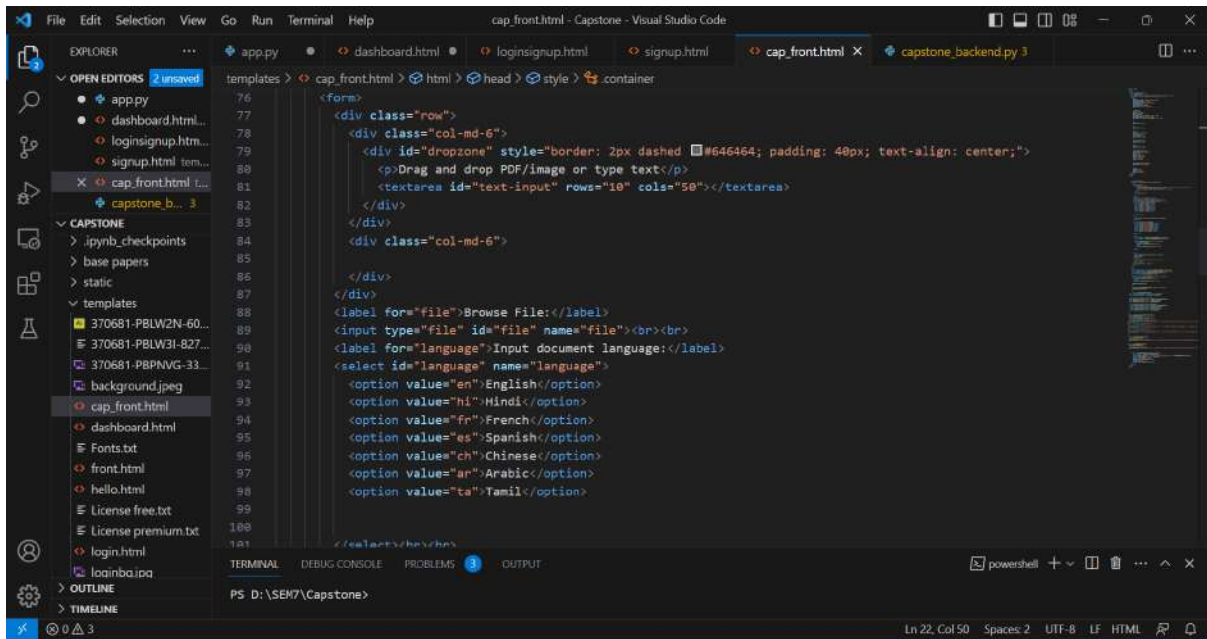


Fig 6. Main page form

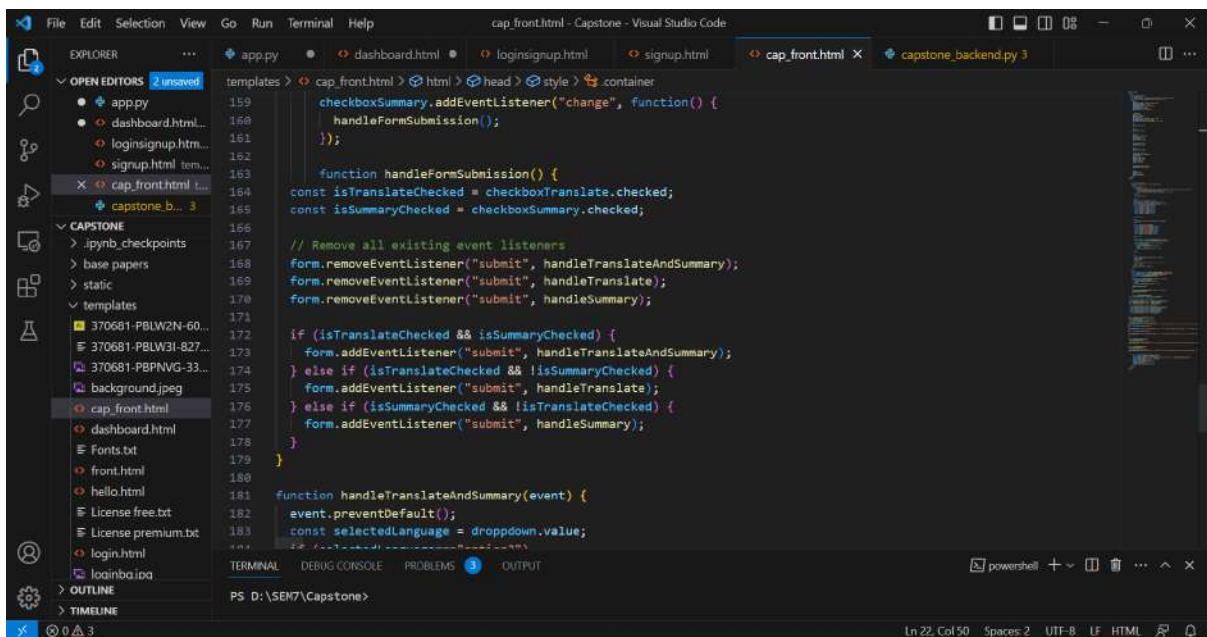
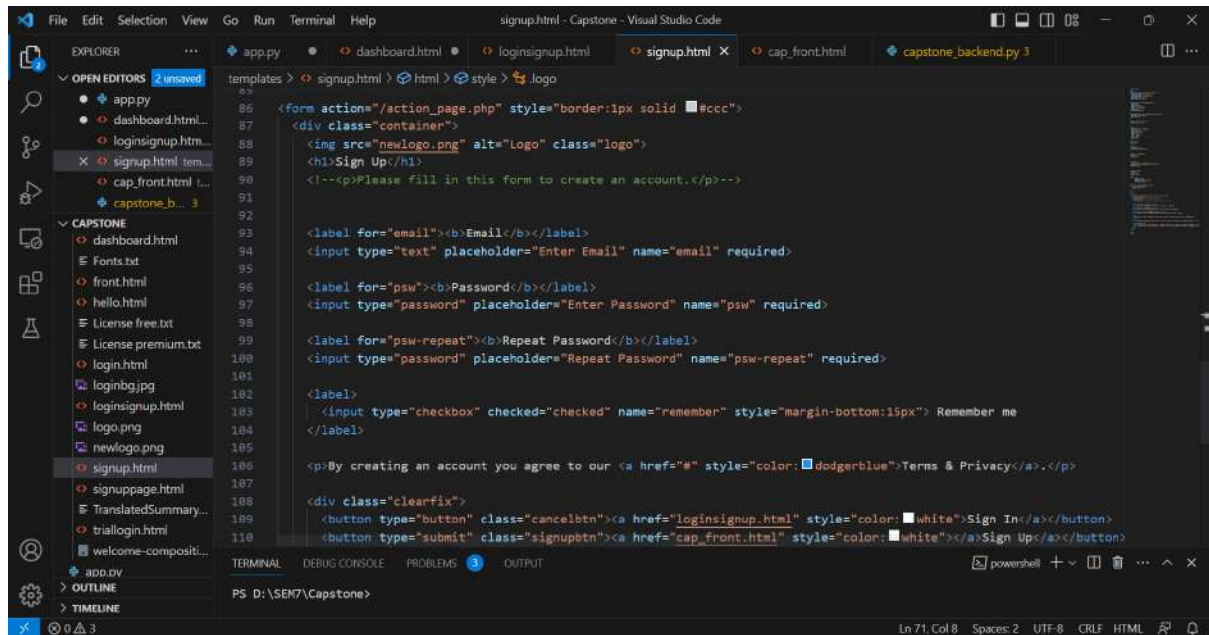
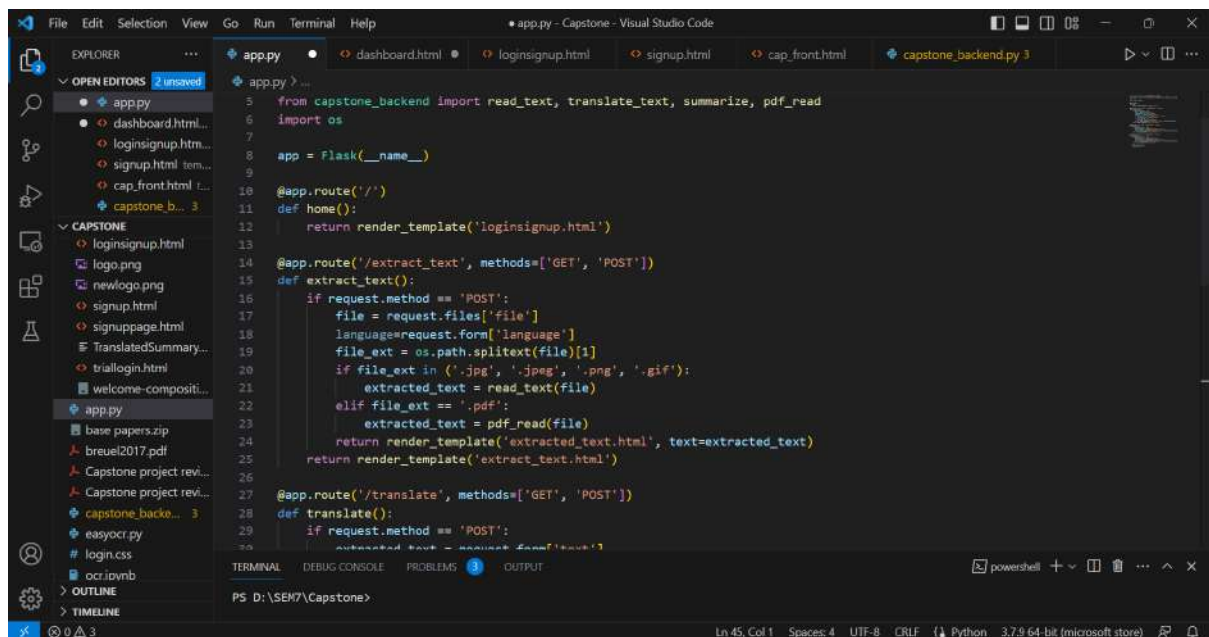


Fig 7. JavaScript to handle translation and/or summarisation task



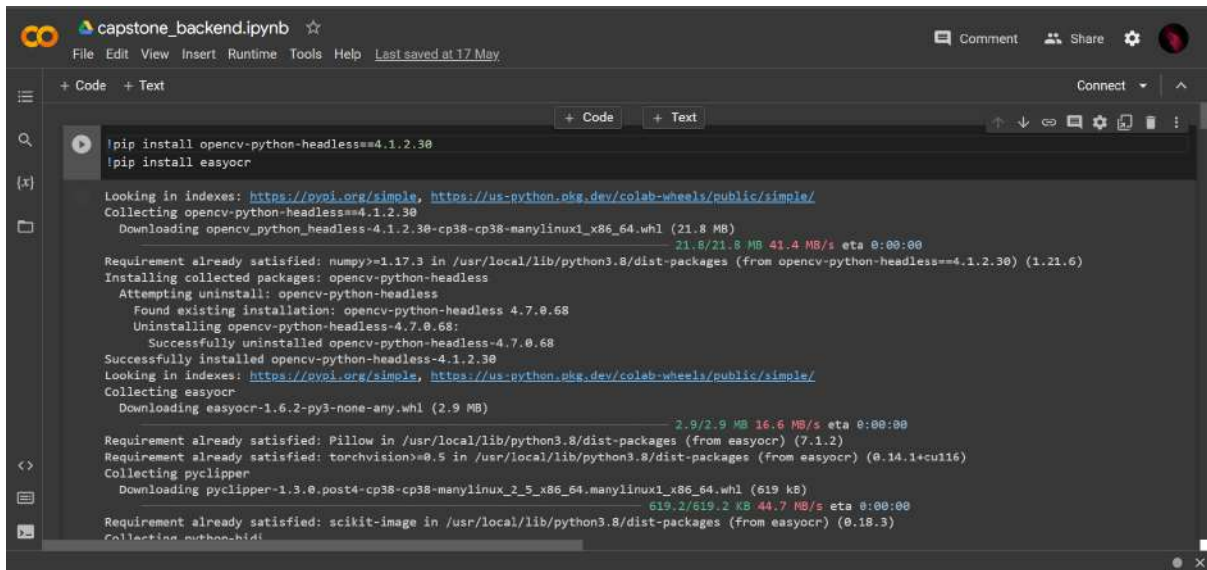
```
templates > signup.html > html > style > logo
86 <form action="/action_page.php" style="border:1px solid #ccc">
87 <div class="container">
88 
89 <h1>Sign Up</h1>
90 <!--<p>Please fill in this form to create an account.</p-->
91
92
93 <label for="email"><b>Email</b></label>
94 <input type="text" placeholder="Enter Email" name="email" required>
95
96 <label for="psw"><b>Password</b></label>
97 <input type="password" placeholder="Enter Password" name="psw" required>
98
99 <label for="psw-repeat"><b>Repeat Password</b></label>
100 <input type="password" placeholder="Repeat Password" name="psw-repeat" required>
101
102 <label>
103 <input type="checkbox" checked="checked" name="remember" style="margin-bottom:15px"> Remember me
104 </label>
105
106 <p>By creating an account you agree to our <a href="#" style="color:#dodgerblue">Terms & Privacy</a></p>
107
108 <div class="clearfix">
109 <button type="button" class="cancelbtn"><a href="login/signup.html" style="color:#white">Sign In</a></button>
110 <button type="submit" class="signupbtn"><a href="cap_front.html" style="color:#white">Sign Up</a></button>
111
```

Fig 8. Signup page code



```
app.py > ...
5 from capstone_backend import read_text, translate_text, summarize, pdf_read
6 import os
7
8 app = Flask(__name__)
9
10 @app.route('/')
11 def home():
12     return render_template('login/signup.html')
13
14 @app.route('/extract_text', methods=['GET', 'POST'])
15 def extract_text():
16     if request.method == 'POST':
17         file = request.files['file']
18         language=request.form['language']
19         file_ext = os.path.splitext(file)[1]
20         if file_ext in ('.jpg', '.jpeg', '.png', '.gif'):
21             extracted_text = read_text(file)
22         elif file_ext == '.pdf':
23             extracted_text = pdf_read(file)
24         return render_template('extracted_text.html', text=extracted_text)
25     return render_template('extract_text.html')
26
27 @app.route('/translate', methods=['GET', 'POST'])
28 def translate():
29     if request.method == 'POST':
30         extracted_text = request.form['text']
31
```

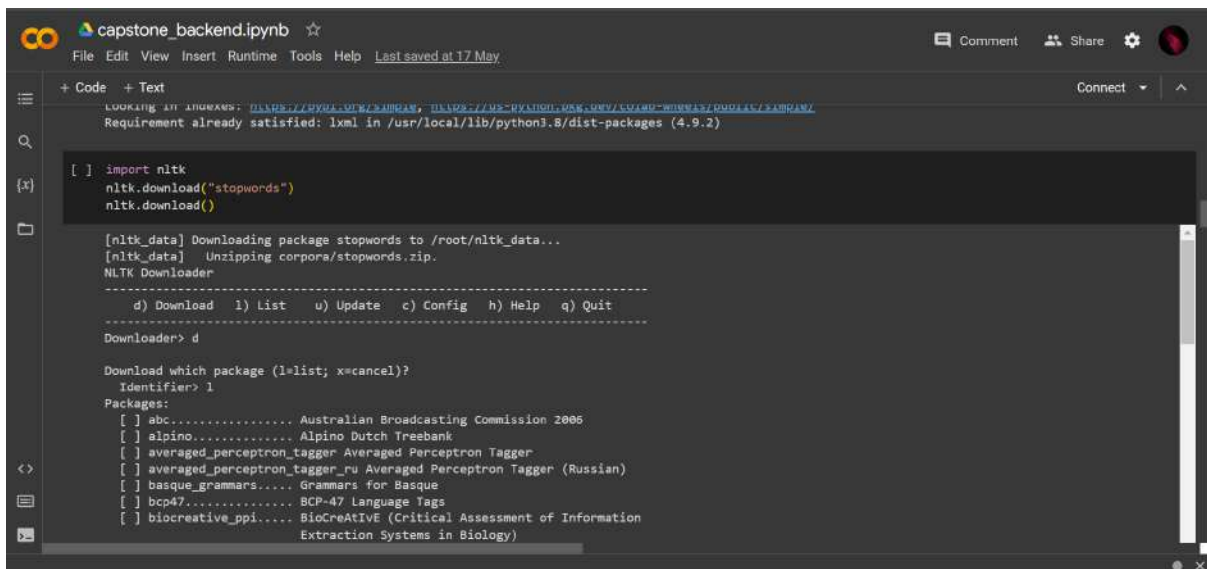
Fig 9. Flask app



```
capstone_backend.ipynb
File Edit View Insert Runtime Tools Help Last saved at 17 May
+ Code + Text
+ Code + Text
!pip install opencv-python-headless==4.1.2.30
!pip install easyocr

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting opencv-python-headless==4.1.2.30
  Downloading opencv_python_headless-4.1.2.30-cp38-cp38-manylinux1_x86_64.whl (21.8 MB)
    21.8/21.8 MB 41.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.8/dist-packages (from opencv-python-headless==4.1.2.30) (1.21.6)
Installing collected packages: opencv-python-headless
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.7.0.68
    Uninstalling opencv-python-headless-4.7.0.68:
      Successfully uninstalled opencv-python-headless-4.7.0.68
  Successfully installed opencv-python-headless-4.1.2.30
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting easyocr
  Downloading easyocr-1.6.2-py3-none-any.whl (2.9 MB)
    2.9/2.9 MB 16.6 MB/s eta 0:00:00
Requirement already satisfied: Pillow in /usr/local/lib/python3.8/dist-packages (from easyocr) (7.1.2)
Requirement already satisfied: torchvision>=0.5 in /usr/local/lib/python3.8/dist-packages (from easyocr) (0.14.1+cu116)
Collecting pyclicker
  Downloading pyclicker-1.3.0.post4-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.whl (619 kB)
    619.2/619.2 KB 44.7 MB/s eta 0:00:00
Requirement already satisfied: scikit-image in /usr/local/lib/python3.8/dist-packages (from easyocr) (0.18.3)
Collecting python-hid
```

Fig 10. Installing libraries



```
capstone_backend.ipynb
File Edit View Insert Runtime Tools Help Last saved at 17 May
+ Code + Text
+ Code + Text
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lxml in /usr/local/lib/python3.8/dist-packages (4.9.2)

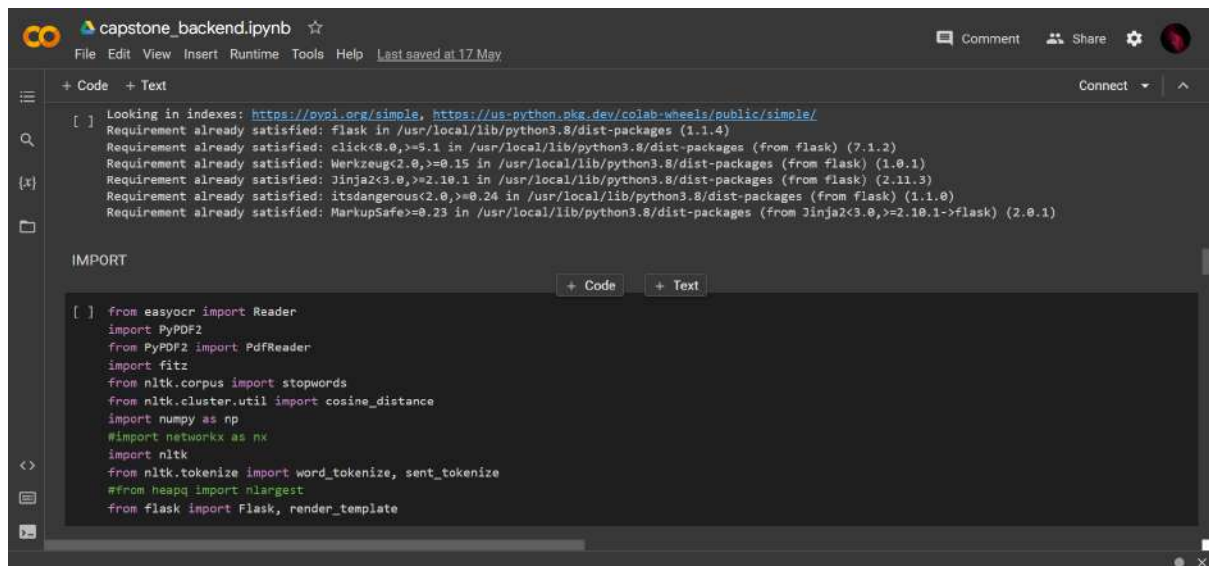
[ ] import nltk
    nltk.download("stopwords")
    nltk.download()

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
NLTK Downloader

-----
d) Download l) List u) Update c) Config h) Help q) Quit
-----
Downloader> d

Download which package (l=list; x=cancel)?
Identifier> l
Packages:
[ ] abc..... Australian Broadcasting Commission 2006
[ ] alpino..... Alpino Dutch Treebank
[ ] averaged_perceptron_tagger Averaged Perceptron Tagger
[ ] averaged_perceptron_tagger_ru Averaged Perceptron Tagger (Russian)
[ ] basque_grammars..... Grammars for Basque
[ ] bcp47..... BCP-47 Language Tags
[ ] biocreative_ppi..... BioCreAtIvE (Critical Assessment of Information
                        Extraction Systems in Biology)
```

Fig 11. Downloading NLTK stopwords



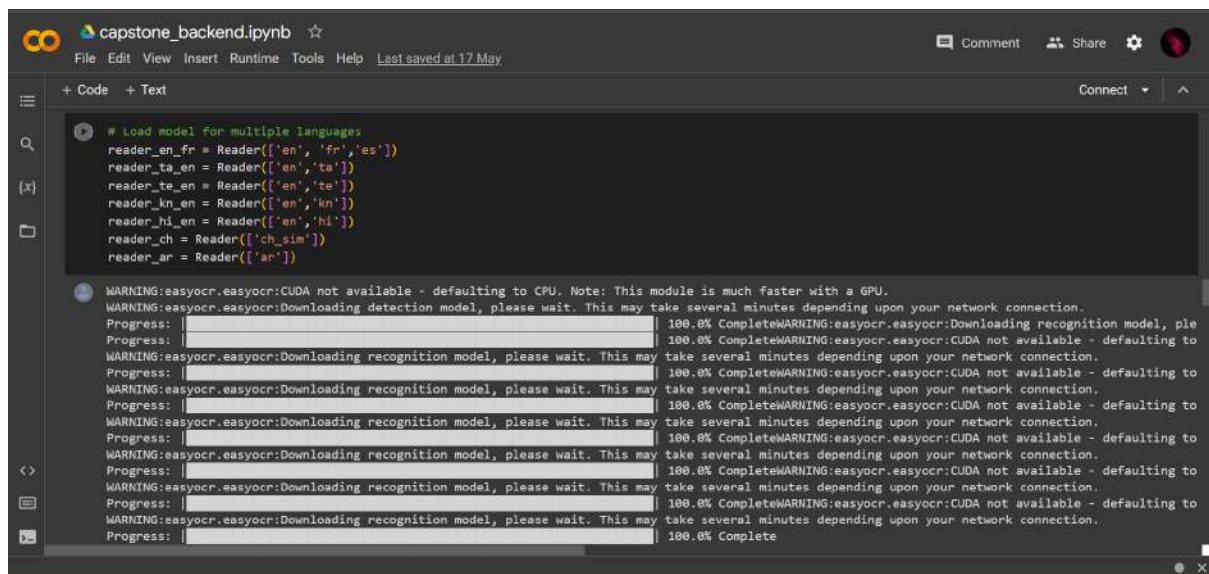
```
capstone_backend.ipynb
File Edit View Insert Runtime Tools Help Last saved at 17 May

+ Code + Text
[ ] Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: flask in /usr/local/lib/python3.8/dist-packages (1.1.4)
Requirement already satisfied: click<8.0,>=5.1 in /usr/local/lib/python3.8/dist-packages (from flask) (7.1.2)
Requirement already satisfied: Werkzeug<2.0,>=0.15 in /usr/local/lib/python3.8/dist-packages (from flask) (1.0.1)
Requirement already satisfied: Jinja2<3.0,>=2.10.1 in /usr/local/lib/python3.8/dist-packages (from flask) (2.11.3)
Requirement already satisfied: itsdangerous<2.0,>=0.24 in /usr/local/lib/python3.8/dist-packages (from flask) (1.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-packages (from Jinja2<3.0,>=2.10.1->flask) (2.0.1)

IMPORT

[ ] from easyocr import Reader
import PyPDF2
from PyPDF2 import PdfReader
import fitz
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
#import networkx as nx
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
#from heapq import nlargest
from flask import Flask, render_template
```

Fig 12. Importing libraries

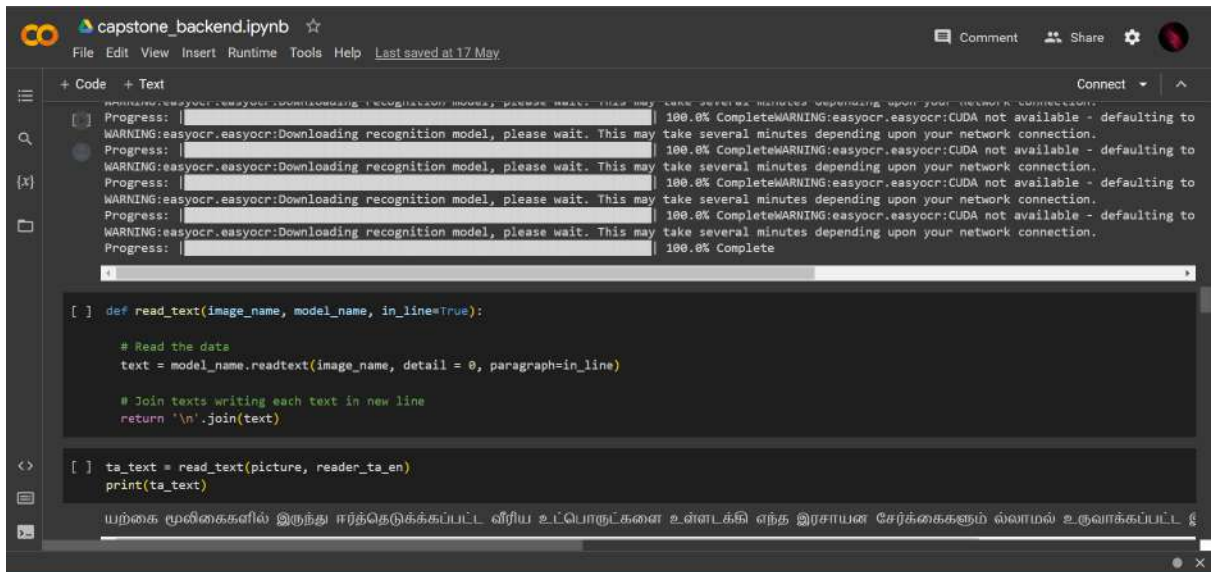


```
capstone_backend.ipynb
File Edit View Insert Runtime Tools Help Last saved at 17 May

+ Code + Text
[ ] # Load model for multiple languages
reader_en_fr = Reader(['en', 'fr', 'es'])
reader_ta_en = Reader(['en', 'ta'])
reader_te_en = Reader(['en', 'te'])
reader_kn_en = Reader(['en', 'kn'])
reader_hi_en = Reader(['en', 'hi'])
reader_ch = Reader(['ch_sim'])
reader_ar = Reader(['ar'])

WARNING:easyocr.easyocr: CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
WARNING:easyocr.easyocr: Downloading detection model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
WARNING:easyocr.easyocr: Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: [ ] 100.0% Complete
```

Fig 13. Loading EasyOCR models



The screenshot shows a Jupyter Notebook titled 'capstone_backend.ipynb'. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a 'Last saved at 17 May' timestamp. The notebook is in 'Code' mode. The code cell contains a function `read_text` that takes `image_name`, `model_name`, and `in_line` as arguments. It uses `model_name.readtext` to extract text from the image and returns it as a single line. Below the function, there is a test call: `ta_text = read_text(picture, reader_ta_en)` followed by `print(ta_text)`. The output area shows a progress bar and a warning message: 'WARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.' The progress bar is at 100.0%.

```
Progress: | 100.0% CompleteWARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
WARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: | 100.0% CompleteWARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
WARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: | 100.0% CompleteWARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
WARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
Progress: | 100.0% Complete
```

```
[ ] def read_text(image_name, model_name, in_line=True):

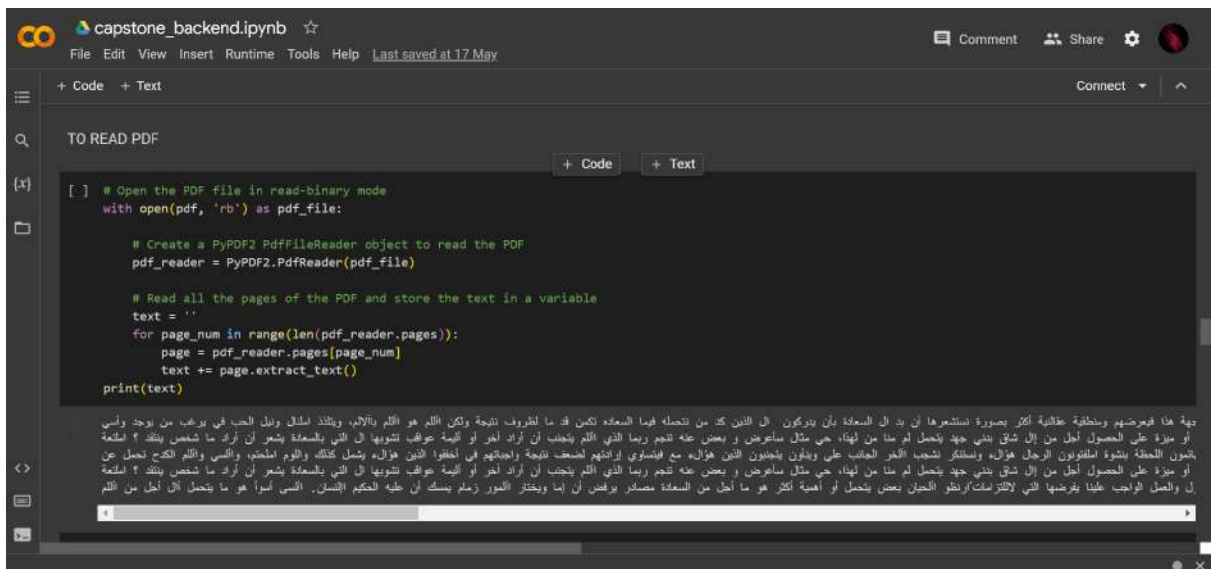
    # Read the data
    text = model_name.readtext(image_name, detail = 0, paragraph=in_line)

    # Join texts writing each text in new line
    return '\n'.join(text)

[ ] ta_text = read_text(picture, reader_ta_en)
print(ta_text)
```

யற்கை மூலிகைகளில் இருந்து சரத்தெடுக்கப்பட்ட வீரிய உட்பொருட்களை உள்ளடக்கி எந்த இரசாயன சேர்க்கைகளும் இல்லாமல் உருவாக்கப்பட்ட

Fig 14. Function to read text from images



The screenshot shows a Jupyter Notebook titled 'capstone_backend.ipynb'. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a 'Last saved at 17 May' timestamp. The notebook is in 'Code' mode. The code cell contains a function `TO_READ_PDF` that takes a `pdf` file as input. It uses `PyPDF2.PdfFileReader` to open the PDF file and `pdf_reader.pages` to iterate through the pages. The text from each page is extracted using `page.extract_text()` and stored in a variable `text`. The final output is `print(text)`. The output area shows a progress bar and a warning message: 'WARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.' The progress bar is at 100.0%.

```
[ ] # Open the PDF file in read-binary mode
with open(pdf, 'rb') as pdf_file:

    # Create a PyPDF2 PdfFileReader object to read the PDF
    pdf_reader = PyPDF2.PdfFileReader(pdf_file)

    # Read all the pages of the PDF and store the text in a variable
    text = ''
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        text += page.extract_text()

    print(text)
```

هذه هنا فروعهم ومنطقة مقلية أكثر بصورة نستعرضها أن بد ال السعادة بأن يدركون ال الذين كد من تحمله فيما السعادة تكمن قد ما لطروف نتيجة ولكن الألم هو الألم بالآلام، ويتخذ أشكال وتدل الحب في يرغب من يوجد وأسي أو ميزة على الحصول أجل من ال شاق بتني جهد يتحمل لم منا من لهناء، حي مثال ماعرض و بعض عنه تجم ربما الذي الألم يتجنب أن أراد آخر أو أئمة عوفب تنوينا ال التي بالسعادة بشعر أن أراد ما شخص يتخذ ؟ الساعه بشعر اللحة بشعر الرجل هؤلاء، ولستكنر تشجب الآخر الجانب على ويلتون يتجنبون الذين هؤلاء مع افساوي ارادتهم لصعب نتيجة واجباتهم في أخفوا الذين هؤلاء يشمل كذلك واليوم لاملهم، والنسي والكم الكبح تحمل عن أو ميزة على الحصول أجل من ال شاق بتني جهد يتحمل لم منا من لهناء، حي مثال ماعرض و بعض عنه تجم ربما الذي الألم يتجنب أن أراد آخر أو أئمة عوفب تنوينا ال التي بالسعادة بشعر أن أراد ما شخص يتخذ ؟ الساعه ال والعمل الواجب عليها يفرضها التي لالتزامات كرنطو الجبان بعض يتحمل أو أهمية أكثر هو ما أجل من السعادة بمصادر يرفض أن إما ويختار الأمور (مما يسك أن عليه الحكيم الإنسان). السبي أسوأ هو ما يتحمل أال أجل من الألم

Fig 15. Extracting text from PDFs

```
capstone_backend.ipynb
File Edit View Insert Runtime Tools Help Last saved at 17 May

+ Code + Text
Connect

[ ] def summarize(txt, n):
    # Tokenize the text into sentences and words
    sentences = sent_tokenize(txt)
    words = word_tokenize(txt)

    # Remove stop words from the words list
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word.lower() not in stop_words]

    # Calculate the frequency of each word
    freq = nltk.FreqDist(words)

    # Calculate the score of each sentence
    scores = {}
    for i, sentence in enumerate(sentences):
        for word in word_tokenize(sentence.lower()):
            if word in freq:
                if i not in scores:
                    scores[i] = freq[word]
                else:
                    scores[i] += freq[word]

    # Get the top n sentences with the highest scores
    top_sentences = nlargest(n, scores, key=scores.get)

    # Sort the top sentences in the order they appear in the original text
    summary = ' '.join([sentences[i] for i in sorted(top_sentences)])
    return summary
```

Fig 16. Function for summarisation

Implementation Screenshots:



Fig 17. Login Page



Sign Up

Email

Enter Email

Password

Enter Password

Repeat Password

Repeat Password

☒ Remember me

By creating an account you agree to our [Terms & Privacy](#)

[Sign In](#) [Sign Up](#)

Fig 18. Signup Page



Drag and drop PDF/image or type text

[Browse File:](#) [Choose file](#) | No file chosen

Input document language: [English](#)

☐ Translate?
[English](#)

☐ Summary needed?

[Go!](#)

Fig 19. Main Page

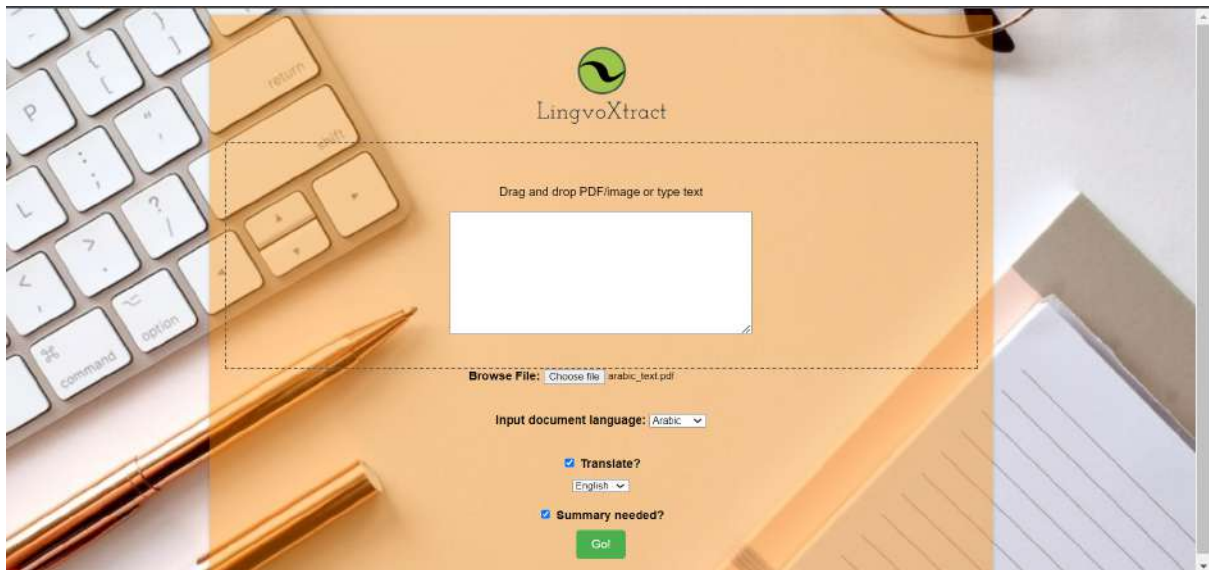


Fig 20. Selecting a PDF and choosing the requirements



Fig 21. Text input



Fig 22. Downloaded output

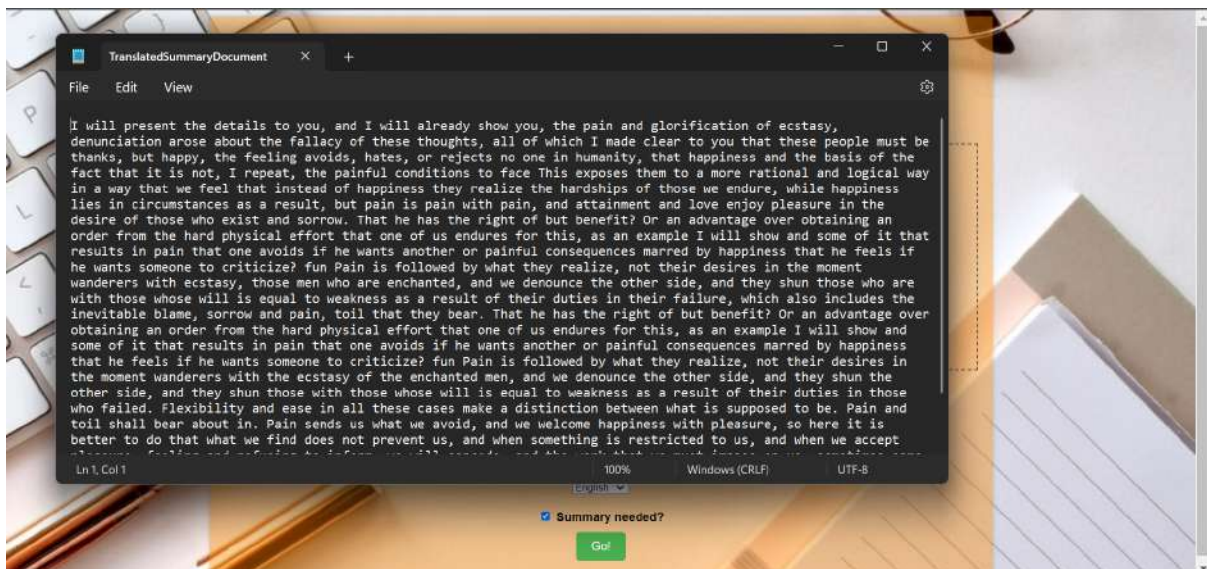


Fig 23. Translated Summary Output

Results

The proposed system architecture and design have successfully tackled the challenge of recognizing text from images and PDFs, and have achieved accurate translation and summarization of the extracted text.

The proposed system architecture and design offer a comprehensive solution to address the increasing demand for language translation, text extraction from images, and text summarization. By incorporating cutting-edge Natural Language Processing (NLP) techniques and corpora, LingvoXtract empowers users to overcome language barriers and information overload.

The system provides users with a user-friendly and step-by-step approach to input text or upload images containing text. They can select their preferred target language for translation and request a summary of the content. Leveraging Flask technology, the architecture ensures a seamless and efficient user experience while accommodating diverse language needs.

The core functionalities of language translation, text extraction from images, and text summarization are integrated into the system. Users can translate text between different languages, extract text from images to make it accessible and editable, and generate concise summaries of lengthy content. These functionalities are supported by state-of-the-art NLP algorithms and corpora.

Conclusion

In conclusion, the proposed system architecture and design have successfully addressed the growing demand for language translation, text extraction from images and PDFs, and text summarization. The system's ability to accurately recognize text from images and PDFs and provide reliable translation and summarization has significantly enhanced the user experience in dealing with multilingual content.

By leveraging advanced computer vision techniques and OCR technology, the system efficiently extracts text from images and PDFs, making it accessible and editable. The integration of state-of-the-art NLP algorithms ensures accurate language translation across a wide range of languages. Additionally, the system's text summarization capabilities enable users to quickly grasp the key points of lengthy documents or paragraphs, saving valuable time.

The user-friendly interface and intuitive design of the system contribute to a seamless and efficient user experience. Users can easily navigate through the system, customize their interactions based on their specific needs, and achieve their desired outcomes effectively.

Overall, the proposed system architecture and design have revolutionized language-related tasks, providing users with a comprehensive solution for language translation, text extraction from images and PDFs, and text summarization. By overcoming language barriers and information overload, the system empowers users to access, understand, and utilize multilingual content with ease, ultimately enhancing their productivity and decision-making capabilities in this digital era.

6. REFERENCES:

1. Junnan Zhu , Yu Zhou , Jiajun Zhang , Chengqing Zong, Attend (2021), Translate and Summarise: An Efficient Method for Neural Cross-Lingual Summarization, National Laboratory of Pattern Recognition, Institute of Automation, CAS.
2. YM Wazery, ME Saleh, A Alharbi, AA Ali (2022), Abstractive Arabic Text Summarization Based on Deep Learning, Research Article Volume 2022, Open Access Article ID 1566890, <https://doi.org/10.1155/2022/1566890>
3. T Shi, Y Keneshloo, N Ramakrishnan (2021), Neural Abstractive Text Summarization with Sequence-to-Sequence Models, ACM/IMS Transactions on Data Science, Volume 2, Issue 1.
4. Christian Bartz, Haojin Yang, Christoph Meinel (2017), STN-OCR: A single Neural Network for Text Detection and Text Recognition, Computer Vision and Pattern Recognition (cs.CV), <https://doi.org/10.48550/arXiv.1707.08831>
5. TM Breuel (2017), High Performance Text Recognition Using a Hybrid Convolutional-LSTM Implementation, 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR).
6. Saad Bin Ahmed, Saeeda Naz, Muhammad Imran Razzak, and Rubiyah Yousaf (2017), Deep Learning based Isolated Arabic Scene Character Recognition, 1King Saud bin Abdulaziz University for Health Sciences, Riyadh, 11481.
7. Marcin Namysl, Iuliu Konya (2019), Efficient, Lexicon-Free OCR using Deep Learning, Fraunhofer IAIS 53757 Sankt Augustin, Germany.
8. Yang Gao, Christian M. Meyer, Mohsen Mesgar and Iryna Gurevych (2019), Reward Learning for Efficient Reinforcement Learning in Extractive Document Summarisation, Dept. of Computer Science, Royal Holloway, University of London, Ubiquitous Knowledge Processing Lab (UKP-TUDA).
9. Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağ 1ar Gu 1çehre, Bing Xiang (2016), Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond.
10. Chris Kedzie, Kathleen McKeown, Hal Daume' III (2019), Content Selection in Deep Learning Models of Summarization.