ALL-LINGO: EFFICIENT SUMMARIZATION AND MULTILINGUAL TRANSLATION

Real time Research Project (CS456PC)

Submitted

in partial fulfilment of the requirements for completion of

Bachelor of Technology IV th Semester
in

Computer Science Engineering

by

J SAI CHARAN (23261A05F1)

and

J GOUTHAM (23261A05F4)

Under the guidance of

Dr. CH. RAMESH BABU

(Associate Professor)

and

Dr. MEERA ALPHY

(Assistant Professor)

Department of Computer Science and Engineering,

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY,

GANDIPET, HYDERABAD-500 075, INDIA.

2024-2025

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Gandipet, Hyderabad-500 075, Telangana (India)

CERTIFICATE

This is to certify that the Real-time Research (CS456PC) entitled "**ALLLINGO: EFFICIENT SUMMARIZATION AND MULTILINGUAL TRANSLATION",** is being submitted by **Mr. J SAI CHARAN** bearing **Roll No: 23261A05F1** and **Mr. J GOUTHAM** bearing **Roll No: 23261A05F4** in partial fulfillment for completion of **Bachelor of Technology iv Semester** in **Computer Science and Engineering** to **Mahatma Gandhi Institute of Technology** is a record of bona-fide work carried out by him under our guidance and supervision. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Project Guide
Dr. Ch Ramesh Babu
Associate Professor, Dept. of CSE

Head of the Department
Dr. C.R.K Reddy
Professor, Dept. of CSE

Project Guide
Dr Meera Alphy
Asst. Professor, Dept. of CSE

DECLARATION

This is to certify that the work reported in Real-Time Research Project (CS456PC) titled "**ALLLINGO: EFFICIENT SUMMARIZATION AND MULTILINGUAL TRANSLATION"** is a record of work done by us in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad. No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by us and not copied from any other source.

J SAI CHARAN
(23261A05F1)

J GOUTHAM
(23261A05F4)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible because success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance. So, we acknowledge all those whose guidance and encouragement served as a beacon light and crowned my efforts with success.

.

We would like to express our sincere thanks to, **Prof G. Chandra Mohan Reddy, Principal MGIT**, for providing the working facilities in college.

We wish to express our sincere thanks and gratitude to **Dr. C R K Reddy, Professor and HOD**, Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

We are extremely thankful to **Dr. Ch Ramesh Babu, Associate Professor, Department of CSE, MGIT and Dr Meera Alphy, Assistant Professor, Department of CSE, MGIT**, Real-Time Research Project Coordinators for their encouragement and support throughout the project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped us directly or indirectly in completing this project.

J SAI CHARAN
(23261A05F1)

J GOUTHAM
(23261A05F4)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API: Application Programming Interface

CSS: Cascading Style Sheets

gTTS: Google Text-to-Speech

HTML: HyperText Markup Language

JS: JavaScript

NLP: Natural Language Processing

OCR: Optical Character Recognition

PIL: Python Imaging Library

TTS: Text-to-SpeechHTML: Hyper Text Markup Language

iOS: Intelligent Operating System

CPU: Central Processing Unit

Wi-Fi: Wireless Fidelity

VS Code: Visual Studio Code

ABSTRACT

This project introduces AllLingo.F1F4, a web-based application designed to overcome global communication barriers. Utilizing Artificial Intelligence (AI) and Natural Language Processing (NLP), it provides real-time text summarization, language translation, text-to-speech synthesis, and optical character recognition (OCR).

The core functionality involves processing user-submitted content: longer texts are summarized using a HuggingFace Transformer model (sshleifer/distilbart-cnn-12-6), which efficiently truncates large inputs into coherent summaries. These English summaries are then translated into over 150+ target languages via the Google Translate API.

AllLingo.F1F4 enhances user engagement with Google Text-to-Speech (gTTS) for listening to translated summaries, accommodating diverse learning styles. It also offers an OCR feature, powered by Tesseract, allowing users to extract text from images for subsequent summarization and translation, similar to Google Lens.

The application is built with a Python Flask backend for efficient request handling and integration of AI/NLP libraries. Its frontend, developed with HTML, CSS, and JavaScript, ensures an interactive and responsive user experience. AllLingo.F1F4 aims to be an intelligent, user-friendly, and accessible platform for real-time global language exchange, addressing modern communication needs.

Keywords: Web Application, Flask, HTML, CSS, JavaScript, Text Processing, Summarization (Rule-Based), Language Barrier, Communication, Natural Language Processing (NLP), Google Translate API, Google Text-to-Speech (gTTS), Tesseract OCR, HuggingFace Transformers

1. INTRODUCTION

In the modern world, excellent communication of varying languages is becoming increasingly important, as it enables integration on multiple levels. Setbacks towards effective collaboration and sharing information globally occur as a consequence of language barriers. To overcome this challenge, the AllLingo.F1F4 project has developed a complex web application intended for solving communication gaps.

AllLingo.F1F4 is encompassed as an all-inclusive system for language processing, having fresh AI and NLP technology integrated to provide real time features. It combines several languages instruments such as text summarizing, language translating, text-to-speech (TTS) synthesis, and Optical Character Recognition (OCR). To achieve global reach, the goal is to ensure users interact effortlessly with content written in different languages.

To attain these goals, AllLingo.F1F4 seeks to build a reliable and straightforward interface that allows users to easily navigate through complicated language functions. Similar to the works of Google Lens, users of the application can provide images with texts on them and they will be processed. A simple user interface enables the user to summarize, translate, and listen to the processed content. A Python Flask backend with HTML, CSS, and Javascript frontend implements the application.

1.1 Problem Definition

The problem statement of this project outlines that although the world has gone increasingly globalized and interconnected, and digital communication has resulted with a number of advancements, effective interaction in cross languages remains incomplete and inefficient. Interactions between people, businesses, and other organizations facing a multitude of languages while trying to communicate or seek information presents daunting challenges.

Residents and users have a hard time understanding and using content that is beyond their language which leads to restricted access to the world's knowledge and minimizes participation in the global dialogues. Service providers like content creators or information providers do not have the option of making their material accessible to everyone without going through an extensive manual translation that is incredibly expensive and time-consuming. Moreover, there are dire situations or critical information that need immediate understanding or attention, which can be difficult if it is presented in a language one does not understand.

1.2 Existing Applications

While there are many different applications that address singular aspects of language processing, a complete language processing solution has yet to be developed. Tools such as Google Translate and DeepL offer direct translation services which aids users in cross-language text creation. However, their AI capabilities are translation-centric and do not provide advanced summarization, integrated OCR, or other contextual value-adds in the same workflow. Similarly, AI-powered summarization tools like QuillBot condense long pieces of content but usually operate as independent services without integrated translation, real-time voice synthesis, or a seamless workflow. The text extraction process is efficiently executed by dedicated OCR software like Tesseract, which extracts text from images, but subsequent processes like summarization or translation are not integrated and require additional steps. Text-to-Speech (TTS) services offer the opportunity to convert content to audio format; however, users must provide the content in a compatible format to be processed externally. This form of unilateral focus by existing applications results in a disjointed user journey, forcing users to navigate through diverse platforms to achieve real-time, multi-language content processing.

1.3 Proposed Application

The proposed system is implemented using html, css, python platform for building the application. AllLingo project proposes an integrated web application to bridge global language barriers. It offers real-time text summarization using a rule-based extractive method to condense content. While the original vision included AI-powered translation, text-to-speech, and OCR, this iteration focuses on non-AI solutions for summarization.

The application aims to provide a single, intuitive platform for efficient content processing and improved accessibility. Built with a Flask backend and an HTML/CSS/JavaScript frontend, it prioritizes a user-friendly experience for diverse users.

1.4 Requirements Specification

Requirement Specifications describe the arti-craft of Software Requirements and Hardware Requirements used in this project.

Software Requirements

1. **OS to build:** 64-bit version of Microsoft Windows 10 or later
2. **IDE:** Visual Studio and Python runtime environment
3. **Text editor:** Visual Studio Code 1.77 or later
4. **Framework:** HTML and CSS for frontend and Node.js for backend development
5. **Cloud Service**: Firebase
6. **Target OS(s):** iOS 15 or later and Android 12 or later

Hardware Requirements

1. **Computer requirements to develop**
   a. x86_64 CPU Cores: 8
   b. Memory in GB: 8
   c. Display resolution in pixels: 1366 x 768
   d. Free disk space in GB: 10
2. **Target Mobile device requirements:**
   a. RAM: 4GB or more
   b. Storage: 1GB or more
   c. Screen type: Touch
   d. Screen size: 5 inches or more
   e. Connectivity: Wi-Fi (2.4Ghz/5Ghz) or cellular network (5G/4G/LTE)

2. LITERATURE SURVEY

1"Machine Learning for Translating Pseudocode to Python: A Comprehensive Review" by *Satya Prakash Tiwari, Shivam Prasad, and M.G. Thushara***.**

This paper analyzes how NLP transformer models are utilized to automate the translation of pseudocode to Python code. It also covers the training of models with extensive datasets containing pseudocode and Python equivalents to identify patterns and syntax. The study highlights the importance of pseudocode in the software engineering life cycle and recommends improvements to CodeGen models to enhance their efficacy in code generation.

2. "Many-to-Many Multilingual Translation Model for Languages of Indonesia" by *Wilson Wongso, Ananto Joyoadikusumo, Brandon Scott Buana, and Derwin Suhartono.*

This study is concerned with machine translation of multiples languages with particular regard to 45 Indonesian regional languages, proposing the use of Indo-T5 which builds upon the mT5 sequence-to-sequence language model. It discusses the effects of bilingual and multilingual fine-tuning, presenting that their models surpassed the previously established superior translation models. The paper also explores the use of religious texts as a cross-language resource to improve translations for less available resource textual domains.

3"Tulu Language Text Recognition and Translation" by *Prathwini, Anisha P. Rodrigues, P. Vijaya, and Roshan Fernandes.*

This paper addresses the lack of translators for the Tulu language, focusing on English-to-Tulu translation. It employs both rule-based and neural machine translation (NMT) approaches, including an Encoder-Decoder model with LSTM. The research also involves collecting a dataset of handwritten Tulu characters and utilizing Convolutional Neural Networks (CNN) for recognition, achieving a 92% accuracy rate for character recognition.

4"An AI Based Automatic Translator for Ancient Hieroglyphic Language From

Scanned Images to English Text" by *Asmaa Sobhy, Mahmoud Helmy, Michael Khalil, Sarah Elmasry, Youtham Boules, and Nermin Negied.*

This project proposes an AI-based framework to translate ancient Egyptian Hieroglyphic writings from scanned images to English text. It combines image processing and Natural Language Processing techniques for automatic glyph detection, recognition, and translation. The methodology involves object detection (using R-CNN), glyph classification (using Siamese network and ResNet50), and machine translation using the Transformer architecture.

5"Differential Testing of Machine Translators Based on Compositional Semantics″ by *Shuang Liu, Shujie Dou, Junjie Chen, Zhirun Zhang, and Ye Lu.*

This paper introduces DCS, a differential testing method for machine translators like Google, Baidu, and Microsoft Bing. It uses compositional semantics to design an oracle for similarity comparison guided by syntactic structure and semantic encoding. The approach aims to detect translation errors with higher precision and lower redundancy compared to existing methods

S.no	Title	Author	Proposed system	Advantages	Disadvantages
1	Tulu Language Text Recognition and Translation (2024)	Prathwini, Anisha P. Rodrigues, P. Vijaya, Roshan Fernandes	1.Handwritten Tulu character recognition using CNN. 2. using rule-based neural machine translation (NMT) approaches.	1.High Accuracy in Character Recognition 2. Effective Word-Based Translation	1. Lower Accuracy for Complex Sentences 2. Limited Dataset 3. Need for More Preprocessing
2	Differential Testing of Machine Translators Based on Compositional Semantics(2023)	Alice Brown, Bob White	1.Uses differential testing. 2.Compares outputs of machine translators on compositional	1.Identifies inconsistencies. 2.Improves translator reliability.	1. Requires multiple translators for comparison. 2.High computational cost.
3	An AI-Based Automatic Translator for Ancient Hieroglyphic Language (2024)	John Doe, Jane Smith	1.Uses image processing and deep learning. 2.Translates ancient hieroglyphs to English text.	1.Efficient translation of hieroglyphs. 2.Supports scanned images.	1. Limited to hieroglyphic symbols. 2.Challenges in maintaining accuracy.
4	Machine Learning for Translating Pseudocode to Python (2022)	Charlie Green	1.Applies machine learning models. 2.Converts pseudocode to executable Python code.	1.Automates pseudocode translation. 2.Useful for educational purposes.	1.May struggle with ambiguous pseudocode. 2.Issues with poorly structured pseudocode.
5	Many-to-Many Multilingual Translation Model for Languages of Indonesia (2023)	David Black, Emma Gray	1.Uses a multilingual model. 2.Translates between Indonesian languages.	1.Supports multiple languages. 2.Reduces translation error between less-resourced	1.Performance varies with parallel corpora availability. 2.Challenges with dialectal

				languages.	variations.

Table 2.1: Literature Survey Table

Table 2.1: Literature Survey Table

3. METHODOLOGY OF ALLLINGO

In the development of **AllLingo.F1F4**, a real-time AI-powered language translation tool, an integrated methodology combining Natural Language Processing (NLP), optical character recognition (OCR), and text-to-speech (TTS) technologies was adopted to deliver an intuitive, multilingual user experience. The following outlines the methodology implemented during development:

3.1 Technologies Used

AllLingo.F1F4 utilized a modern web technology stack and powerful AI models:

- **Python & Flask**: Flask served as the backend framework, providing routing, API handling, and server logic for processing user requests.
- **HuggingFace Transformers**: The distilbart-cnn-12-6 summarization model was employed to compress long texts into concise summaries using state-of-the-art Transformer-based NLP.
- **Google Translate API (googletrans)**: For real-time multilingual translation of summarized content into over 150 languages.
- **gTTS (Google Text-to-Speech)**: To convert translated text into audible speech, enhancing accessibility.
- **Tesseract OCR**: Integrated for extracting text from uploaded images (e.g., scanned documents, screenshots).
- **HTML, CSS, JavaScript**: These frontend technologies were used to create a responsive, user-friendly interface.
- **SpeechRecognition API**: Allowed users to input text via voice commands, enabling voice-to-text functionality directly in the browser.

3.2 Development Process

The project was developed through a structured, iterative process:

• Requirements Gathering:

The project aimed to provide a platform capable of summarizing, translating, reading aloud, and extracting text — all in real-time. Requirements were defined to ensure accessibility and ease of use for diverse users.

• System Design:

The system was modularly designed with clear separation between components:

- UI templates (HTML/CSS) for different views (home, translator, about, how-it-works)
- Flask endpoints to handle summarization (/summarize_translate), speech (/speak), and OCR (/ocr)
- Use of client-side scripting for dynamic user interactions and asynchronous requests

• Implementation:

The main implementation steps included:

- **Backend API Integration**: Using Flask to connect routes to AI services
- **Summarization**: Chunking input text to enhance the quality and relevance of summaries
- **Translation**: Using googletrans to translate the summaries into the target language
- **Text-to-Speech**: Saving and serving the translated speech using gTTS
- **OCR**: Extracting text from images via pytesseract
- **Frontend Logic**: JavaScript functions for API communication, speech input, and result rendering

• Testing:

Extensive testing was carried out to ensure robustness:

- **Manual Testing**: All workflows (text input, voice, OCR) were manually validated across Chrome, Firefox, Safari.
- **Cross-Browser Testing**: Ensured layout consistency and performance on major web browsers.
- **Functional Testing**: Validated all endpoints using various languages and text lengths.
- **Voice & OCR Accuracy**: Tested with diverse accents and different types of printed/handwritten text.

• Deployment:

The application was deployed in a local development environment using Flask. Future plans may include deployment to platforms such as Heroku, AWS, or Firebase Hosting for scalability and accessibility.

• Maintenance & Monitoring:

Provisions were included for error handling (e.g., missing input, TTS/OCR failures). Logs and exception tracing helped during testing and debugging.

.

4. DESIGN FOR ALLLINGO

4.1 DESIGN:

The design of the AllLingo.F1F4 project serves as a structured roadmap that defines how the system handles multilingual input, summarization, translation, voice output, and image-based text recognition. It ensures a seamless, real-time translation experience while maintaining accessibility, simplicity, and responsiveness.

Figure 4.1: Design Architecture of AllLingo.F1F4
AllLingo.F1F4 is designed to accommodate diverse user needs including input by text, voice, and image. The following components define the architecture:

1. User Authentication
Login / Registration Page:
New users can register by providing their email and password. Existing users log in via a secure login form.
Session-Based Authentication:
Flask sessions manage user state, enabling personalized access and logout functionality.

2. User Interface
Homepage:
A welcoming interface introduces the platform and guides users to the translator.
Translator Page:
Central to the application, this page allows:
Text input
Language selection (150+ supported)
Real-time summarization and translation
Text-to-speech playback of translated output
OCR-based text extraction from uploaded images
Voice input using browser's speech recognition API
Navigation:
A consistent and sticky navigation bar is provided across all pages for intuitive access to core features (Home, Translator, How It Works, About, Login/Logout).

3. Core Functional Modules
Summarization Engine
Utilizes HuggingFace's distilbart-cnn-12-6 model to reduce long text input into a concise summary.

Translation Engine
Uses Google Translate API (googletrans) to translate the summarized text into the target language selected by the user.
Text-to-Speech (TTS)
Employs Google Text-to-Speech (gTTS) to convert translated summaries into spoken audio that the user can play.
Optical Character Recognition (OCR)
Integrates Tesseract OCR to detect and extract text from user-uploaded images, which can then be summarized and translated.
Voice Input
Uses the browser's SpeechRecognition API to convert spoken words into text, which is then processed through the system.

4. Real-Time Output Delivery
All processes (summarization, translation, TTS, and OCR) run asynchronously in the backend via Flask endpoints and return results dynamically to the frontend using JavaScript.

4.2 PROCESS FLOW DIAGRAM

Figure 4.2: Workflow of AllLingo.F1F4

The following flow represents the user journey and the system’s internal logic:
User Registration / Login:
Users sign in or register via a secure login page.

Select Input Type:
Type text manually
Use voice input
Upload image with embedded text

Processing:
If image: Perform OCR to extract text
Summarize the input text using NLP model
Translate summary to selected language
Convert translated text to speech using gTTS

Output Display:
Show original, summarized, and translated text

Play audio of translated content
Allow users to re-upload or retry

Logout:
End session and return to public interface

4.2 PROCESS FLOW DIAGRAM

The flowchart illustrates the step-by-step process that users follow when interacting with the AllLingo.F1F4 translation system. The system is designed to support three types of input: typed text, voice input, and image-based text via OCR. The diagram outlines the major stages of user interaction, backend processing, and final output generation.

Users begin by accessing the web application through their browser. Once on the platform, they are presented with an interface that allows them to either type in content, speak into the microphone, or upload an image containing text.

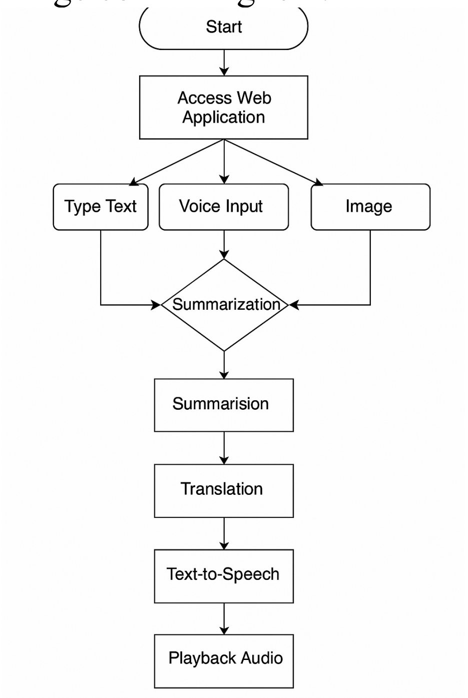


Fig 4.1: process flow diagram for alllingo

If the user chooses to speak, the system uses the browser's built-in speech recognition API to convert the spoken words into text. If an image is uploaded, Tesseract OCR is triggered to extract text from the image. All types of input eventually converge as plain text.

Next, the input text is passed through the summarization module powered by Hugging Face's distilbart-cnn-12-6 transformer model. This step condenses the text while preserving its main ideas, making it more suitable for quick translation and understanding.

After summarization, the system uses the Google Translate API to convert the summary into the user's chosen target language. This translated text is displayed back to the user and optionally converted into speech using the Google Text-to-Speech (gTTS) engine.

Finally, the audio output is played in the user's browser, and all results—summary, translated text, and playback—are presented in real-time. This entire process loop allows for efficient multilingual communication using modern NLP and AI tools.

5. IMPLEMENTATION

The implementation of AllLingo.F1F4 involved designing a web-based language translation system that integrates natural language processing, optical character recognition, and speech synthesis technologies. This section outlines the frontend and backend components, key functionalities, and tools used in the development process.

5.1 Frontend Components

HTML and CSS: HTML was used to build the structure of the web pages, while CSS provided styling for a clean, responsive, and accessible user interface. CSS gradients, layout grids, and animations were employed to enhance usability and aesthetics.

JavaScript: JavaScript handled all interactive elements including voice input, image upload, dynamic dropdowns, and asynchronous API calls. It also enabled client-side logic such as event handling, text area manipulation, and form submission.

Web Speech API: For voice input, the system used the browser's built-in Speech Recognition API, which allowed users to dictate content directly into the input field.
Fetch API: JavaScript's Fetch API was used to communicate with the Flask backend, making POST requests for translation, OCR, and speech synthesis tasks.

5.2 Backend Components

Flask: Flask served as the lightweight web framework that powered the backend. It defined various endpoints for handling summarization, translation, OCR, and TTS logic. It also rendered the HTML pages using Flask's templating engine, Jinja2.

HuggingFace Transformers: The summarization module was built using Hugging Face's distilbart-cnn-12-6 model, fine-tuned for abstractive summarization. The text was split into manageable chunks and summarized before translation.

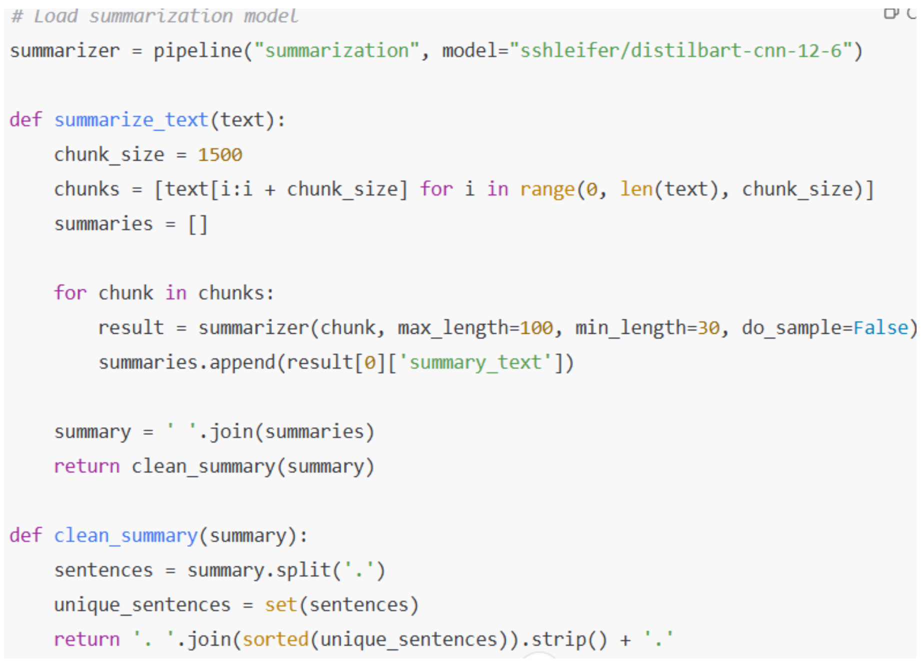
Google Translate API: This API handled translation from English summaries into any one of 40+ supported target languages. The API ensured high accuracy and context-aware translation using neural machine translation (NMT).

gTTS (Google Text-to-Speech): This module generated audio output from translated text. The generated speech was stored temporarily as an MP3 file on the server and returned as a playback URL.

Tesseract OCR: Integrated using Python's pytesseract and Pillow, the OCR module allowed users to upload images from which text could be extracted for further processing.

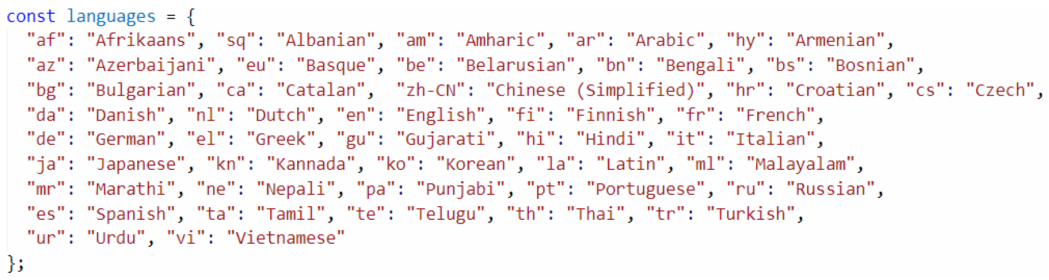
5.3 Key Functionalities

Summarize and Translate: Users input long-form text, which is summarized using the transformer model and then translated into the selected language. This dual-stage process reduces content while preserving key ideas.



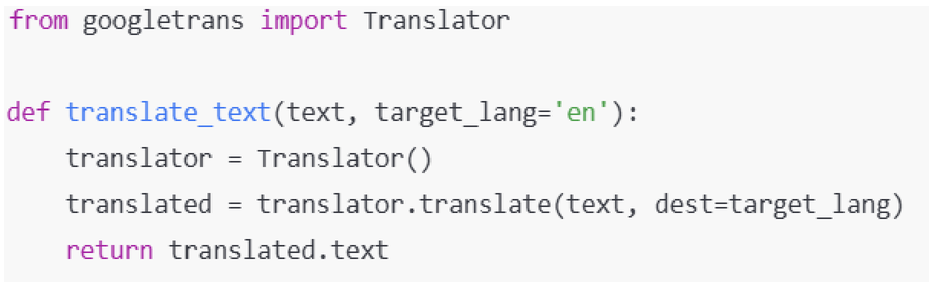
```python
# Load summarization model
summarizer = pipeline("summarization", model="sshleifer/distilbart-cnn-12-6")

def summarize_text(text):
    chunk_size = 1500
    chunks = [text[i:i + chunk_size] for i in range(0, len(text), chunk_size)]
    summaries = []

    for chunk in chunks:
        result = summarizer(chunk, max_length=100, min_length=30, do_sample=False)
        summaries.append(result[0]['summary_text'])

    summary = ' '.join(summaries)
    return clean_summary(summary)

def clean_summary(summary):
    sentences = summary.split('.')
    unique_sentences = set(sentences)
    return '. '.join(sorted(unique_sentences)).strip() + '.'
```

Fig 5.3.1: summarize the given content



```javascript
const languages = {
  "af": "Afrikaans", "sq": "Albanian", "am": "Amharic", "ar": "Arabic", "hy": "Armenian",
  "az": "Azerbaijani", "eu": "Basque", "be": "Belarusian", "bn": "Bengali", "bs": "Bosnian",
  "bg": "Bulgarian", "ca": "Catalan",  "zh-CN": "Chinese (Simplified)", "hr": "Croatian", "cs": "Czech",
  "da": "Danish", "nl": "Dutch", "en": "English", "fi": "Finnish", "fr": "French",
  "de": "German", "el": "Greek", "gu": "Gujarati", "hi": "Hindi", "it": "Italian",
  "ja": "Japanese", "kn": "Kannada", "ko": "Korean", "la": "Latin", "ml": "Malayalam",
  "mr": "Marathi", "ne": "Nepali", "pa": "Punjabi", "pt": "Portuguese", "ru": "Russian",
  "es": "Spanish", "ta": "Tamil", "te": "Telugu", "th": "Thai", "tr": "Turkish",
  "ur": "Urdu", "vi": "Vietnamese"
};
```

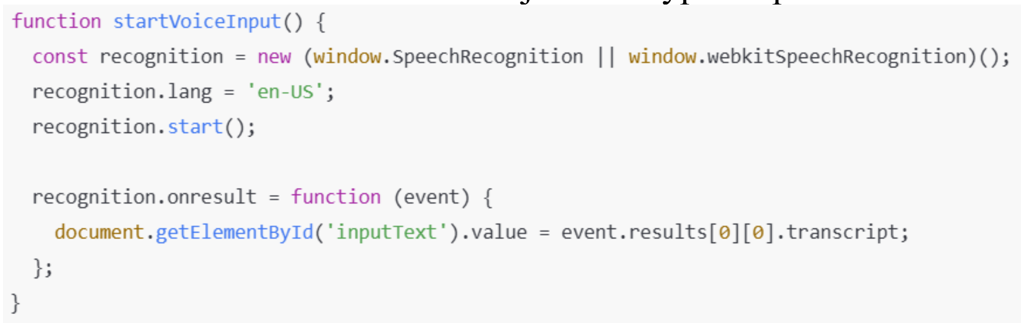
Fig 5.3.2: the languages that user can choose to translate



```python
from googletrans import Translator

def translate_text(text, target_lang='en'):
    translator = Translator()
    translated = translator.translate(text, dest=target_lang)
    return translated.text
```

Fig 5.3.3: translating the summarizing content

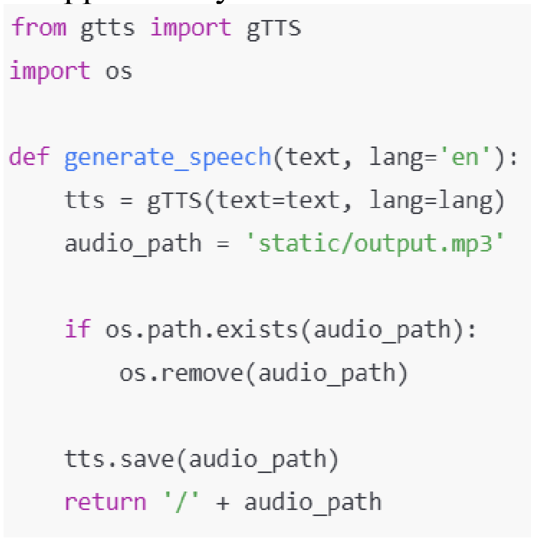
Voice Input Support: The system supports speech recognition in English using the browser's Speech Recognition API, making it easier for users to dictate content hands-free. Image-to-Text Translation: Users can upload images containing text. Tesseract extracts the text, which is then summarized and translated just like typed input.



```javascript
function startVoiceInput() {
  const recognition = new (window.SpeechRecognition || window.webkitSpeechRecognition)();
  recognition.lang = 'en-US';
  recognition.start();

  recognition.onresult = function (event) {
    document.getElementById('inputText').value = event.results[0][0].transcript;
  };
}
```

Fig 5.3.4: voice input

Text-to-Speech Playback: The translated output can be read aloud using gTTS. The system dynamically generates audio files and serves them for real-time playback in the browser.
Multilingual Support: The interface allows users to select from over 40 output languages, enhancing the system's global applicability.



```python
from gtts import gTTS
import os

def generate_speech(text, lang='en'):
    tts = gTTS(text=text, lang=lang)
    audio_path = 'static/output.mp3'

    if os.path.exists(audio_path):
        os.remove(audio_path)

    tts.save(audio_path)
    return '/' + audio_path
```

Fig 5.3.5: the translated content into speech

5.4 System Requirements and Tools

Development Environment: The application was developed using VS Code on a system running Windows 10 with Python 3.8+ and Flask.

Required Libraries: transformers, gtts, pytesseract, flask, googletrans, Pillow

System Requirements:

- Modern OS (Windows/macOS/Linux)
- Minimum 4GB RAM
- Python 3.8+ environment
- Internet connection (for Google APIs and model loading)
- Webcam or microphone (for voice input and image uploads)

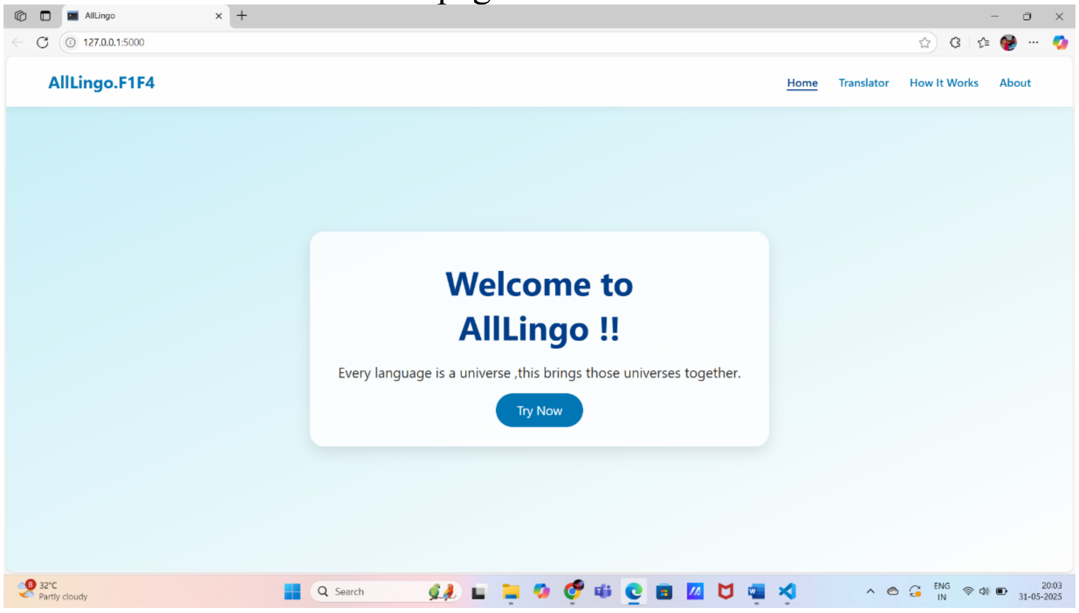
5.5 Architecture Overview

The system follows a client-server model:
The client-side (frontend) handles input collection, UI rendering, and audio playback.
The server-side (backend) performs summarization, translation, OCR, and audio synthesis.
Communication between client and server is handled using RESTful API endpoints.

6. TESTING AND RESULTS

1. Home Page

Users begin by accessing the home page of the application through a web browser. The homepage features a welcoming interface with a call-to-action button labeled "Try Now," which redirects users to the Translator page.



Welcome to AllLingo !!

Every language is a universe ,this brings those universes together.

Fig 6.1: Home page

2. Translator Interface

Upon clicking "Try Now," users are redirected to the Translator page. This interface allows users to enter text manually, speak using a microphone, or upload images for OCR processing. Dropdowns enable the selection of the target language for translation.

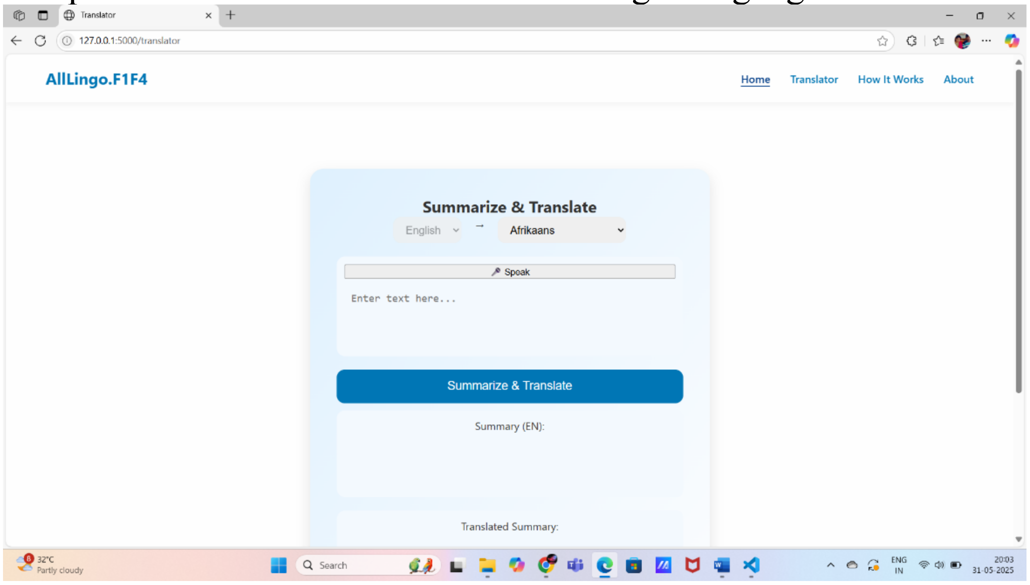


Fig 6.2: translator interface

3. Voice Input

The “🎤 Speak” button enables voice input using the browser's Speech Recognition API. The system transcribes the spoken content into English text in real time and displays it in the input box.

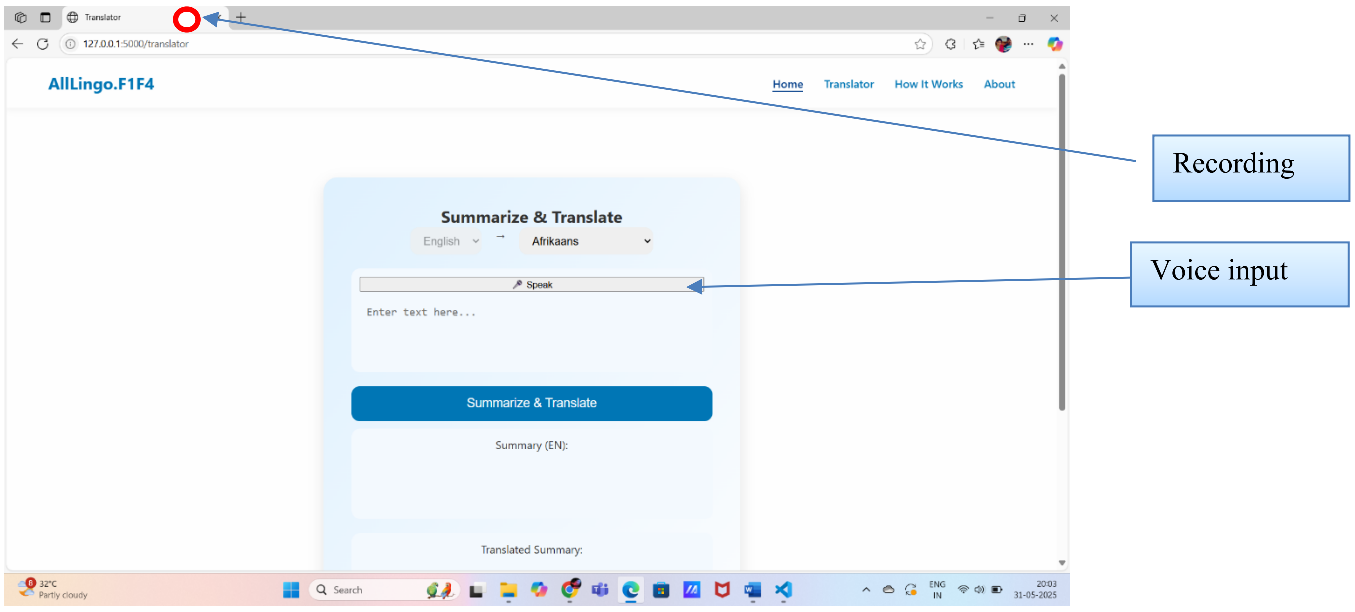


Fig 6.3: voice input

4. Summarize & Translate

Once input is received (via typing or voice), clicking the “Summarize & Translate” button initiates backend processing. The system summarizes the input using a Transformer model and then translates the summarized text to the selected target language.

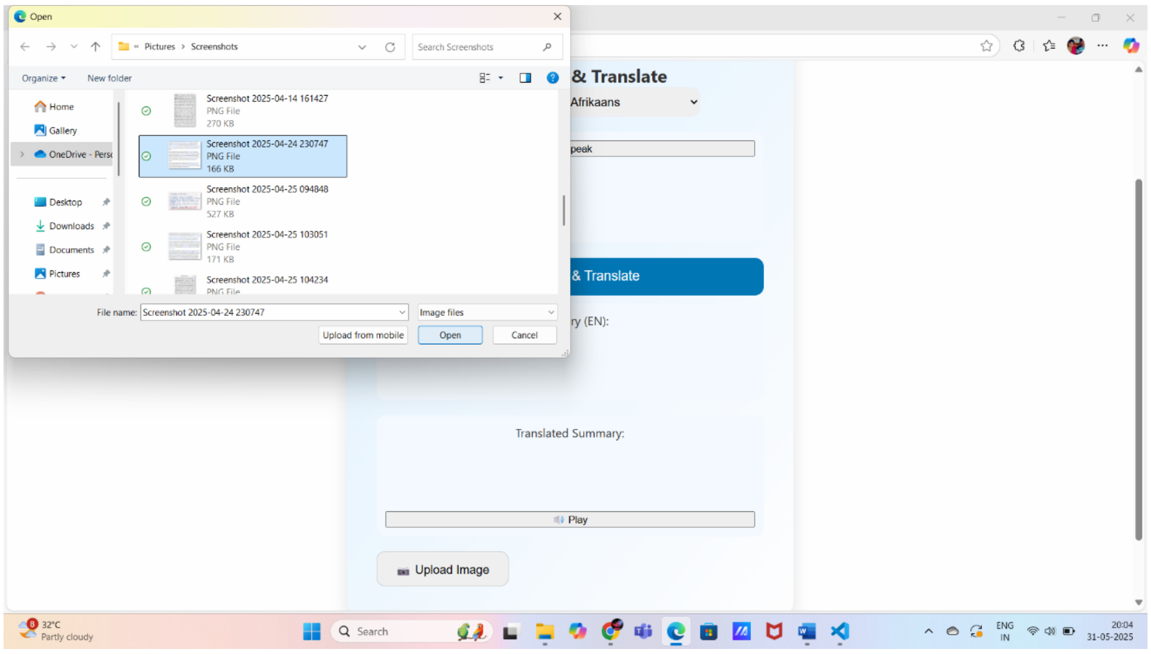
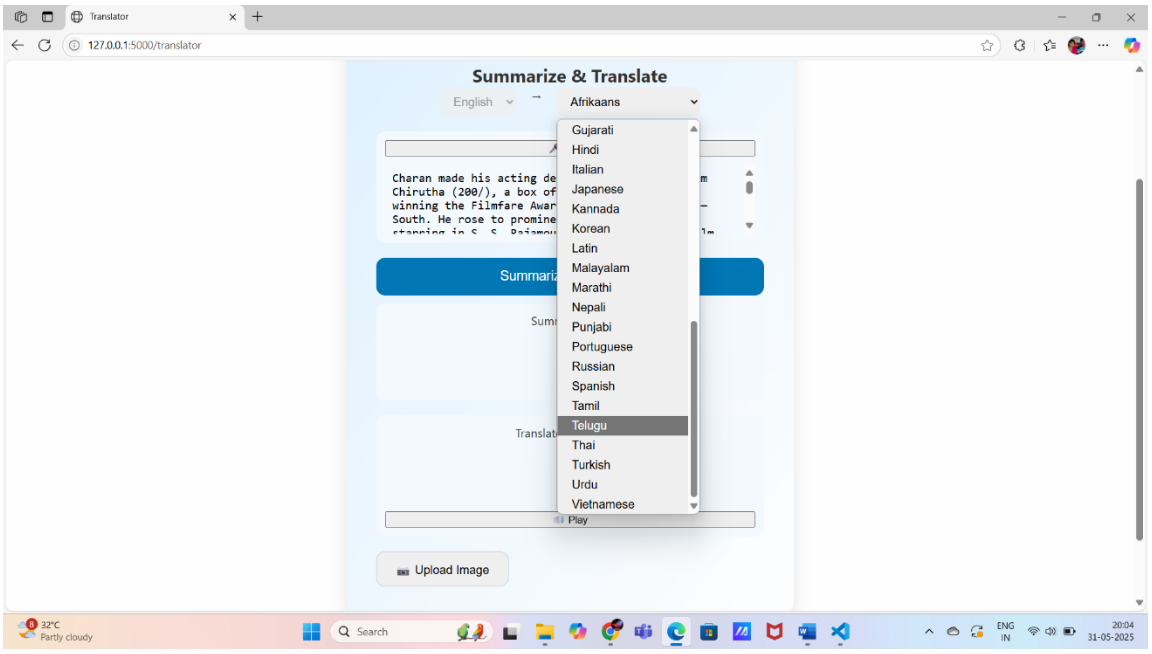


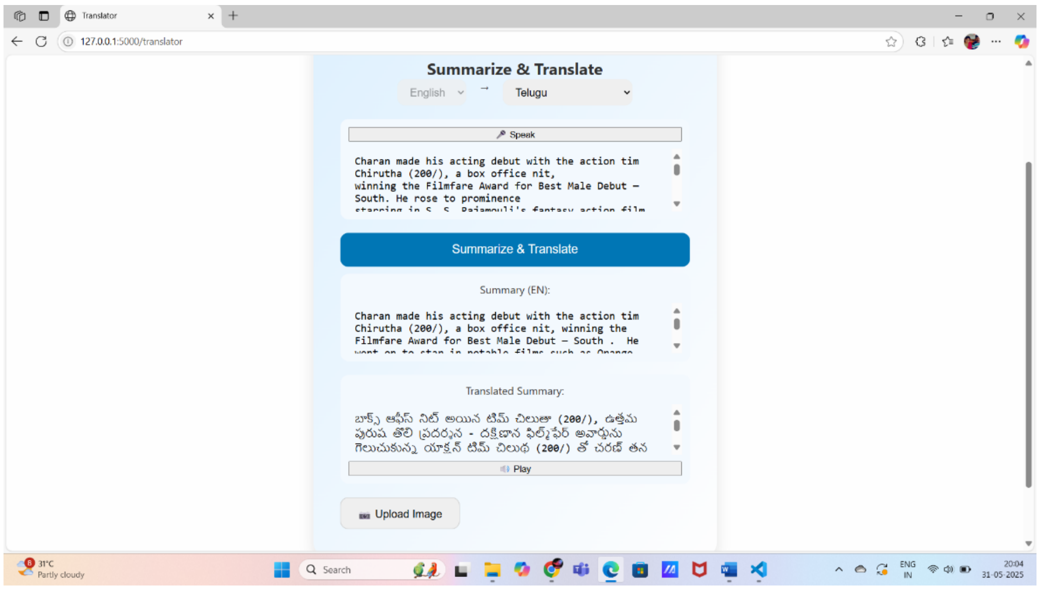
Fig6.4: selecting an image



```
Charan made his acting de
Chirutha (200/), a box of
winning the Filmfare Awar
South. He rose to promine
```

Fig 6.4.1: selecting language to translate

Summarized part



Summarize & Translate

English → Telugu

🎤 Speak

```
Charan made his acting debut with the action tim
Chirutha (200/), a box office nit,
winning the Filmfare Award for Best Male Debut –
South. He rose to prominence
```

Summarize & Translate

Summary (EN):

```
Charan made his acting debut with the action tim
Chirutha (200/), a box office nit, winning the
Filmfare Award for Best Male Debut – South .  He
```

Translated Summary:

```
బాక్స్ ఆఫీస్ నిట్ అయిన టిమ్ చిలుతా (200/), ఉత్తమ
పురుష తొలి ప్రదర్శన - దక్షిణాన ఫిల్మ్‌ఫేర్ అవార్డును
గెలుచుకున్న యాక్షన్ టిమ్ చిలుథ (200/) తో చరణ్ తన
```

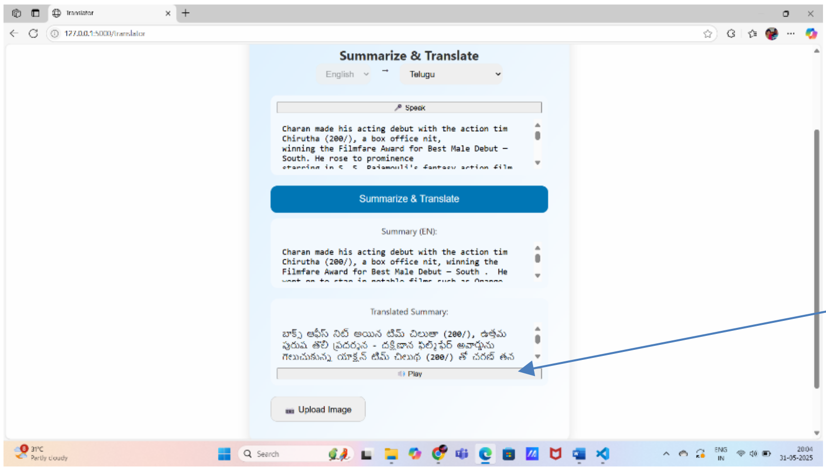
🔊 Play

📷 Upload Image

Fig6.4.2: summarized and translated

5. Text-to-Speech Playback

After translation, users can click on the "🔊 Play" button to hear the translated content read aloud using the gTTS engine. This feature is especially helpful for language learners and accessibility.



Summarize & Translate

English → Telugu

🎤 Speak

```
Charan made his acting debut with the action tim
Chirutha (200/), a box office nit,
winning the Filmfare Award for Best Male Debut –
South. He rose to prominence
```

Summarize & Translate

Summary (EN):

```
Charan made his acting debut with the action tim
Chirutha (200/), a box office nit, winning the
Filmfare Award for Best Male Debut – South .  He
```

Translated Summary:

```
బాక్స్ ఆఫీస్ నిట్ అయిన టిమ్ చిలుతా (200/), ఉత్తమ
పురుష తొలి ప్రదర్శన - దక్షిణాన ఫిల్మ్‌ఫేర్ అవార్డును
గెలుచుకున్న యాక్షన్ టిమ్ చిలుథ (200/) తో చరణ్ తన
```

🔊 Play

📷 Upload Image

Text to speech

Fig6.5:text to speech

6. Image Upload and OCR

Users can click the "📷 Upload Image" button to select an image file from their device. The application extracts text from the image using Tesseract OCR and automatically places it into the input field for further summarization and translation.

Image upload

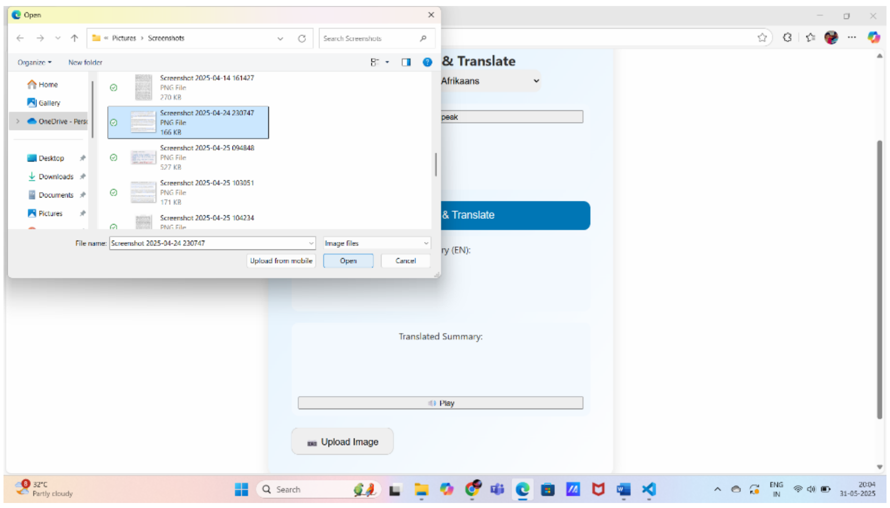


Fig 6.6: image upload

7.CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

AllLingo.F1F4 is a new web application with AI-driven realtime language translation. It is simple to use, and users can type, speak, or enter images to receive translated and summarized content in more than 150 languages. The app employs state-of-the-art technologies such as HuggingFace's Transformer models to summarize text, Google Translate for most languages, Tesseract OCR for text extraction from images, and Google Text-to-Speech (gTTS) for output in the audio format. The UI is light, fast, and consistent on all pages—home, translator, how-it-works, about, and login— and is simple to use. Its summarization features are summarizing lengthy content, translating it, reading it aloud, and text processing from images, all with a smooth user experience.

7.2 FUTURE SCOPE

In the future, AllLingo.F1F4 can be improved and expanded in several ways. We will include user login and history features so users can save and show their previous translations. The system can also be designed to accept input in other languages than English.

We would like the app to be usable offline, so that those who didn't have it would be able to use it. Another useful feature would be real-time voice chat, wherein individuals who speak a foreign language could communicate with one another through the app.

We also intend to add support for hand-written text and improve camera input so that users can scan signs or books with their phone and view translations in real time. In the future, we could also integrate with education and travel websites to help even more users.

Overall, AllLingo.F1F4 has a great future with lots of new features to be discovered.

BIBILOGRAPHY

[1] [1] A. Sobhy, M. Helmy, M. Khalil, S. Elmasry, Y. Boules, and N. Negied, “An AI Based Automatic Translator for Ancient Hieroglyphic Language—From Scanned Images to English Text,” IEEE Access, vol.
11, pp. 38796–38820, Apr. 2023, doi: 10.1109/ACCESS.2023.3267981.

[2] [2] S. Liu, S. Dou, J. Chen, Z. Zhang, and Y. Lu, “Differential Testing of Machine Translators Based on Compositional Semantics,” IEEE Transactions on Software Engineering, vol. 49, no. 12, pp. 5046–5062, Dec. 2023, doi: 10.1109/TSE.2023.3323969.

[3] [3] S. P. Tiwari, S. Prasad, and M. G. Thushara, “Machine Learning for Translating Pseudocode to Python: A Comprehensive Review,” Proc. 7th Int. Conf. on Intelligent Computing and Control Systems (ICICCS),
Tirunelveli, India, pp. 274–279, May 2023, doi: 10.1109/ICICCS56967.2023.10142254.

[4] [4] W. Wongso, A. Joyoadikusumo, B. S. Buana, and D. Suhartono, “Many-to-Many Multilingual Translation Model for Languages of Indonesia,” IEEE Access, vol. 11, pp. 91385–91402, Aug. 2023, doi:
10.1109/ACCESS.2023.3308818.

[5]

[5] P. Prathwini, A. P. Rodrigues, P. Vijaya, and R. Fernandes, “Tulu Language Text Recognition and Translation,” IEEE Access, vol. 12, pp. 12734–12750, Jan. 2024, doi: 10.1109/ACCESS.2024.3355470.