**A MAJOR PROJECT REPORT ON**

**ADVANCED BLOOD CELL CLASSIFICATION**

**USING**

**TRANSFER LEARNING**

**Submitted in partial fulfilment of the requirements for the award of the degree**

**BACHELOR OF TECHNOLOGY**

**In**

**ARTIFICIAL INTLLIGENCE & DATASCIENCE**

**Submitted by**

**NP.Ganga devi (223h1a3010)**

**M.Divya (223h1a3008)**

**M.Nandhini (223h1a3007)**

**M.Samatha (223h1a3009)**

**L.Anusha (223h1a3006)**

**Under the esteemed guidance of**

**Mr. D. Mahesh, M.Tech(Ph. D)**

**Assistant Professor & HOD ,Dept of AI&DS**



**DEPARTMENT OF ARTIFICIAL INTLLIGENCE &DATA SCIENCE**

**K.L.M COLLEGE OF ENGINEERING FOR WOMEN**

**(APPROVED BY AICTE AND AFFILIATED TO J.N.T.U. ANANTAPURAMU)**

**KADAPA-516003**

**2024-2025**

**K.L.M COLLEGE OF ENGINEERING FOR WOMEN**

**(APPROVED BY AICTE AND AFFILIATED TO J.N.T.U. ANANTAPURAMU)**

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**CERTIFICATE**

This is to certify that the project report entitled “**ADVANCED BLOOD CELL CASSIFICATION USING TRANSFER LEARNING**” that is being submitted by

**NP.Ganga devi (213H1A0528)**

**M. Divya (213H1A0524)**

**M. Nandhini (213H1A0519)**

**M. Samatha (213H1A0559)**

**L.Anusha (213H1A0502)**

In partial fulfilment of the requirements of the award of the Degree of Bachelor of Technology in ARTIFICIAL INTLLIDENCE&DATASCIENCE to the Jawaharlal Nehru Technological University Anantapur is a record of bonafide work carried out by under my guidance and supervision. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree.

**Project Guide Head of the Department**

**Mr D. Mahesh, M.Tech(Ph.D) Mr D. Mahesh, M.Tech(Ph.D)** Assistant Professor & HOD, Dept of AI&DS Assistant Professor & HOD, Dept of AI&DS

**Submitted for university Examination (Viva Voice) held on : \_\_\_\_\_\_\_**

**Internal Examiner External Examiner**

**VISION AND MISSION OF COLLEGE**

**VISION**

To be an Institution which aims at empowering women through delivering quality Technical Education While Keeping Moral Integrity intact.

**MISSION**

To provide high standard of Technical Education to women students enabling them to face all challenges of their personal and professional life with courage and confidence.

To make women students as global citizens by inculcating critical thinking, effective communication, robust engineering knowledge and ethical behaviour in them.

To provide an environment that has values and encourages knowledge acquisition, gender equality and academic freedom, making this a preferred Institution for knowledge seekers.

To install self confidence among rural students by supplementing with co curricular and extra-curricular activities.

**VISION AND MISSION OF DEPARTMENT OF ARTIFICIAL INTLLIGENCE AND DATASCIENCE**

**VISION**

To generate competent professionals to become part of the industry and research organizations at the National and International levels.

**MISSION**

Providing a strong theoretical and practical background across the computer science discipline with an emphasis on software development.

Imparting the skills necessary to continue education to grow professional. Empowering the women in rural communities with computer education.

Inculcating professional behaviour, strong ethical values, innovative research capabilities and leadership abilities.

**ACKNOWLEDGEMENT**

We are highly indebted to our superiors for the encouragement and help given for the completion of our project.

We wish to express our gratitude to **Dr. K. CHANDRA OBUL REDDY,** **Managing Director, Kandula Group of Institutions and**  **Dr. M.V.RATHNAMMA,** **Principal of K.L.M. College of Engineering for Women,** Kadapa for their valuable suggestions and advice throughout the course.

At the same time, we express a deep sense of reverence in profound gratitude to our **Mr D. Mahesh,** **Assistant Professor** **& HOD of** **artificial intelligence&datascience**  and **Project Guide** for their constant motivation, valuable help throughout the project work and for arranging all the facilities, and resources needed for our project. We also extend our thanks to other faculties for their cooperation during our course. Not to forget, our teaching and non teaching staff and family who had, directly and indirectly, helped and supported us in completing our project.

**Project Associates**

NP.Ganga devi

M.Divya

M. Nandhini

M. Samatha

L. Anusha

**LIST OF SCREENS**

**ABSTRACT**

White blood cells (WBCs) are a vital component of the immune system. The efficient and precise classification of WBCs is crucial for medical professionals to diagnose diseases accurately. This study presents an enhanced convolutional neural network (CNN) for detecting blood cells with the help of various image pre-processing techniques. Various image pre-processing techniques, such as padding, thresholding, erosion, dilation, and masking, are utilized to minimize noise and improve feature enhancement. Additionally, performance is further enhanced by experimenting with various architectural structures and hyperparameters to optimize the proposed model. A comparative evaluation is conducted to compare the performance of the proposed model with three transfer learning models, including Inception V3, MobileNetV2, and DenseNet201.The results indicate that the proposed model outperforms existing models, achieving a testing accuracy of 99.12%, precision of 99%, and F1-score of 99%. In addition, We utilized SHAP (Shapley Additive explanations) and LIME (Local Interpretable Model-agnostic Explanations) techniques in our study to improve the interpretability of the proposed model, providing valuable insights into how the model makes decisions. Furthermore, the proposed model has been further explained using the Grad-CAM and Grad-CAM++ techniques, which is a class-discriminative localization approach, to improve trust and transparency. Grad-CAM++ performed slightly better than Grad-CAM in identifying the predicted area's location. Finally, the most efficient model has been integrated into an end-to-end (E2E) system, accessible through both web and Android platforms for medical professionals to classify blood cell.

Bottom of Form

**CHAPTER-1**

**INTRODUCTION**

INTRODUCTION

Blood cell classification is a critical task in hematology, enabling the diagnosis and monitoring of variour blood related disorders. traditional methods relay on manual examination of blood snears by skilled professionals ,which can be time consuming, labor intensive, and prone to human error. With advancements vision, automated blood cell classification has emerged as promising solution

The main role of transfer learning

Transfer learning, a technique where a pre trained model is fine tuned for a specific task, has shown remarkable success in medical image analysis by leveraging pre trained models,researchers can adapt to the nuances of blood cel classification with limited datasets. This approach not only enhansive labelled data

Accurate blood cell classification can lead to;

Early diseasese detection; Enabling timely intervention and improved patient outcomes.

Reduced manual effort; Automating the classification process cal alleviate the workload of healthcare professionals.

Enhanced research; Facility the study of blood cell morphology and behaviour, leading to a deeper understanding of hematological discorders.

By harnessing the power of transfer learning,this research seeks to push the boundaries of blood cell classification, ultimately contributing to better healthcare outcomes

Clinical Application:Exploring clinical application of advanced blood cell classificatyion using transfer learning

Precision medicine transfer learning based approaches can contribute to precision medicine by providing accurate and efficient blood cell classification

CHAPTER-2

LITERATURE SURVEY 2.**LITERATURE SURVEY**

Advanced Blood cell classification using transfer learning has showun promising results in improving diagnostic accuracy and efficiency. Here, literature survey highlighting key studies and findings.

* **YOLOv10 Model : Acomparative study demonstrated the effitiveness of YOLOv 10 in real time blood cell detection and classification,outperforming other model like Model like mobile net V2, shuffle net 2, and Dark net in term of precision and recal**
* **Efficientnet B3 Model: Researchers achieved exceptional metrics using Efficient Net B3, including a recall specificitly of 98.54% Precision of 95.86 % F1 score of 97.7% and over all accuracy rate of 97.13% in classifying blast cells.**
* Dark net and mobilenet v2: Another study found Darknet excelled in feature extraction for blood cell classification blast cells.
* Disease diagnosis accurate blood cells classification can aid in diagnosing diseases like leukemia and anemia.
* Active contour incorporating incorporating sophisticated contour based segmentation techniques to enhance accuracy in blast cell nuclei segmentation
* Hybrid CNN models Cobining YOLO based dection with hubrid CNN model for accurate blood cell identification
* Dataset development creating annotated blood cell dataset to support further research and development
* Model optomization continuously optimizing and fine tuning models for improving performance and efficiency.
* Clinical application exploring clinical application of advanced blood cell classification using transfer learning

CHAPTER-3

SYSTEM ANALYSIS

**3.System Analysis**

**Existing System:**  
 Advanced blood cell classification using transfer learning involves analyzing and fine tuning pre trained models for specific tasks. Heres system analysis:

**System components**

* Data collection :gathering blood cell images from various source
* Data preprocessing:Enhacing image quality,normalization,and augmentating data
* Model selection:choosing pre trined model sutabul for blood cell classification
* Transfer learning:Fine tuning pre trained model on blood cell classification
* Classification:prediction blood cell types using the fine tuned model

**System analysis:**  
Accuracy : Evaluating model performance using metrics like accuracy, precision, recall ,and f1 score.

Scalability : Determining the system ability to handle large datasets and various blood cell types.

Robustness : Testing the systems performance on diverse image quality and staining techniques.

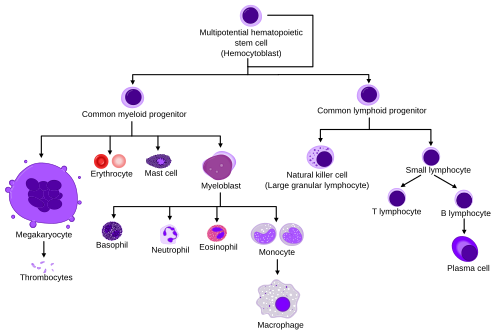
**Advantages:**

* Improved accuracy
* Reduce training time
* Flexibility
* Scalability
* Robustness

**Challenges:**

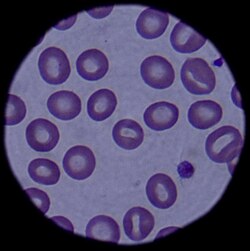
* **Economic:** Uses open-source tools and existing hardware, keeping costs low
* **Technical:** Built with Python and libraries like pyttsx3, speech\_recognition, and json; upgrade-ready
* **Social:** Accessible for users with limited digital skills; supports rural deployment
* **Ethical:** No personal data is stored; aligns with privacy standards and ensures fair access to information

A **blood cell** (also called a **hematopoietic cell**, **hemocyte**, or **hematocyte**) is a [cell](https://en.wikipedia.org/wiki/Cell_(biology)) produced through [hematopoiesis](https://en.wikipedia.org/wiki/Hematopoiesis" \o "Hematopoiesis) and found mainly in the [blood](https://en.wikipedia.org/wiki/Blood). Major types of blood cells include [red blood cells](https://en.wikipedia.org/wiki/Red_blood_cell) (erythrocytes), [white blood cells](https://en.wikipedia.org/wiki/White_blood_cell) (leukocytes), and [platelets](https://en.wikipedia.org/wiki/Platelet) (thrombocytes). Together, these three kinds of blood cells add up to a total 45% of the blood tissue by volume, with the remaining 55% of the volume composed of [plasma](https://en.wikipedia.org/wiki/Blood_plasma), the liquid component of blood.[[1]](https://en.wikipedia.org/wiki/Blood_cell#cite_note-1)

* [](https://en.wikipedia.org/wiki/File:Hematopoiesis_simple.svg)Diagram showing the development of different blood cells from [hematopoietic stem cells](https://en.wikipedia.org/wiki/Hematopoietic_stem_cell) to mature cells.
* A **blood cell** (also called a **hematopoietic cell**, **hemocyte**,

[Red blood cells](https://en.wikipedia.org/wiki/Red_blood_cell) or *erythrocytes* primarily carry [oxygen](https://en.wikipedia.org/wiki/Oxygen) and collect [carbon dioxide](https://en.wikipedia.org/wiki/Carbon_dioxide) through the use of [hemoglobin](https://en.wikipedia.org/wiki/Hemoglobin" \o "Hemoglobin).[[2]](https://en.wikipedia.org/wiki/Blood_cell#cite_note-2) Hemoglobin is an [iron](https://en.wikipedia.org/wiki/Iron)-containing protein that gives red blood cells their color and facilitates transportation of [oxygen](https://en.wikipedia.org/wiki/Oxygen) from the [lungs](https://en.wikipedia.org/wiki/Lung) to tissues and [carbon dioxide](https://en.wikipedia.org/wiki/Carbon_dioxide)

Red and white human blood cells as seen under a microscope using a blue side

[](https://en.wikipedia.org/wiki/File:Blood_cells_090304-F-5951M-108.jpg)

Mature red blood cells are unique among cells in the human body in that they lack a nucleus (although [erythroblasts](https://en.wikipedia.org/wiki/Erythroblasts) do have a nucleus).

The condition of having too few red blood cells is known as [anemia](https://en.wikipedia.org/wiki/Anemia" \o "Anemia), while having too many is [polycythemia](https://en.wikipedia.org/wiki/Polycythemia" \o "Polycythemia).

Erythrocyte sedimentation rate (ESR) is the rate at which RBCs sink to the bottom (when placed in a vertical column after adding an anticoagulant). Normal values of ESR are:

• 3 to 5 mm per hour in males.

• 4 to 7 mm per hour in females

**CHAPTER-4**

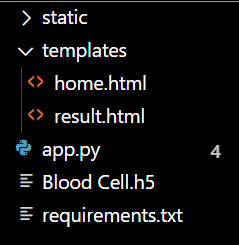
**DATA COLLECTION**

**(AND)**

**PREPARATION**

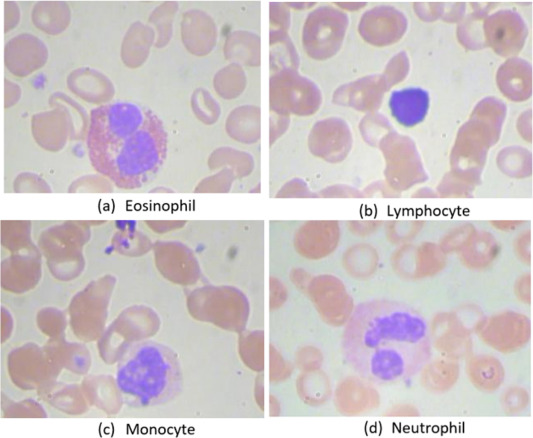
**4.PROJECT STRUCTURE**

**Create the Project folder which contains files as shown below**

****

***4.1 Collect the dataset:***

***White blood cells (WBCs) are a vital component of the immune system. The efficient and precise classification of WBCs is crucial for medical professionals to diagnose diseases accurately. This study presents an enhanced convolutional neural network (CNN) for detecting blood cells with the help of various image pre-processing techniques. Various image pre-processing techniques, such as padding, thresholding, erosion, dilation, and masking, are utilized to minimize noise and improve feature enhancement. Additionally, performance is further enhanced by experimenting with various architectural structures and hyperparameters to optimize the proposed model. A comparative evaluation is conducted to compare the performance of the proposed model with three transfer learning models, including Inception V3, MobileNetV2, and DenseNet201.The results indicate that the proposed model outperforms existing models, achieving a testing accuracy of 99.12%, precision of 99%, and F1-score of 99%. In addition, We utilized SHAP (Shapley Additive explanations) and LIME (Local Interpretable Model-agnostic Explanations) techniques in our study to improve the interpretability of the proposed model, providing valuable insights into how the model makes decisions. Furthermore, the proposed model has been further explained using the Grad-CAM and Grad-CAM++ techniques, which is a class-discriminative localization approach, to improve trust and transparency. Grad-CAM++ performed slightly better than Grad-CAM in identifying the predicted area's location. Finally, the most efficient model has been integrated into an end-to-end (E2E) system, accessible through both web and Android platforms for medical professionals to classify blood cell.***



***The primary contributions of our study are as follows:***

***1.Applied various image pre-processing techniques to prioritize relevant regions and eliminate unnecessary background. This resulted in enhanced precision and scalability in classification, as well as improved computational efficiency for faster training and inference.***

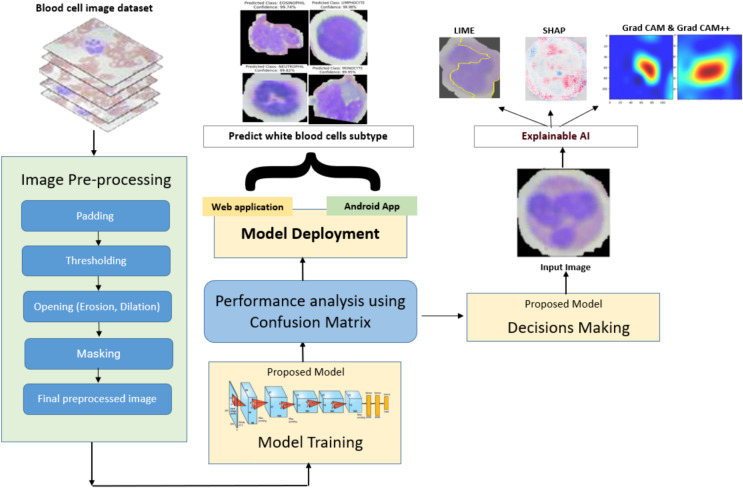
***2.Utilized LIME and SHAP, two widely used XAI approaches, to generate saliency maps and identify the most significant features influencing the model's decision***

***3.Utilized Grad-CAM++ and Grad-CAM to further enhance the interpretability of our proposed approach by generating heatmaps that increased its transparency and provided insights for better understanding.***

***4.Developed a real-time application and interactive web-based tools to aid medical professionals in the early identification of blood***

***DATAVISUALIZATION***

***The methodology utilized to diagnose WBCs consists of four main stages: data collecting, image pre-processing, designing a proposed CNN model, and XAI.******[Fig. 2](https://www.sciencedirect.com/science/article/pii/S2153353924000282" \l "f0010) provides a visual representation of the proposed approach framework. The following sections provide a detailed explanation of each phase.***

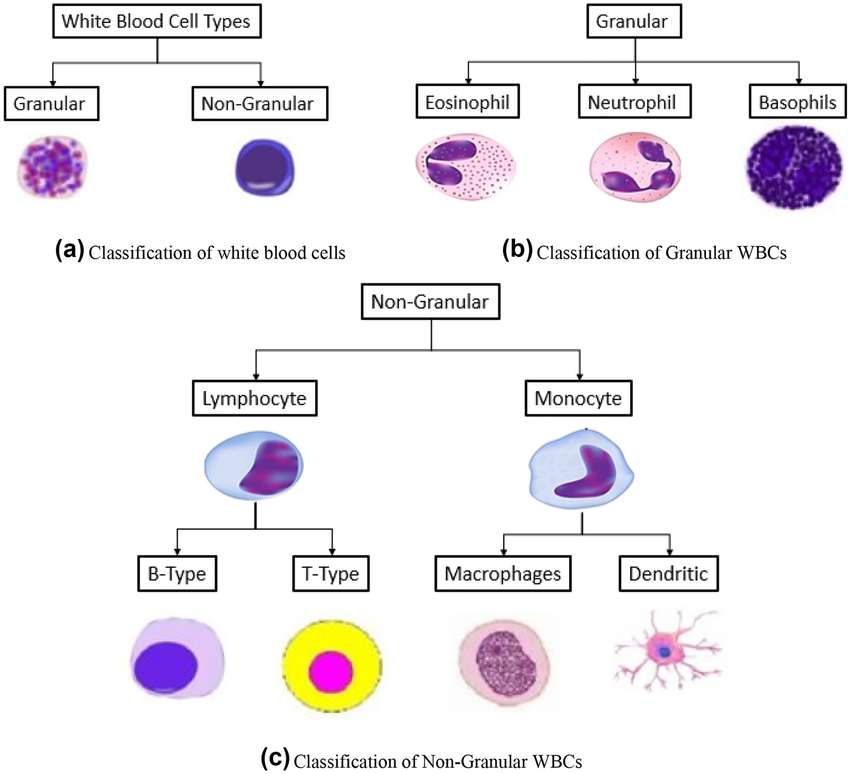


**CHAPTER-5**

**SYSTEM DESIGN**

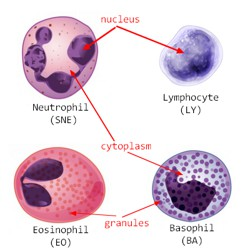
**5.SYSTEM DESIGN**

**5.1 Split data and model building:**

****

**This article was written in collaboration with Mathieu Sarrat and Laleh Ravanbod.**

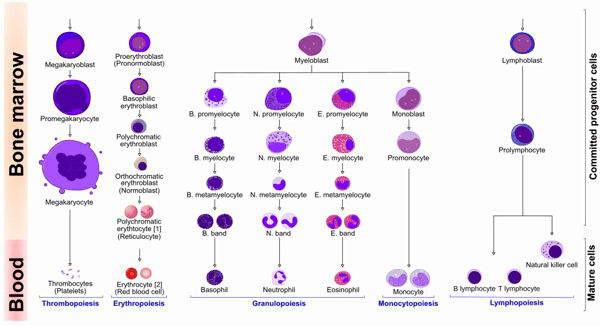
**In this article, we will do a little bit of biology and explore how**[**deep learning**](https://datascientest.com/deep-learning-definition)**can help to classify blood cells. The diagnosis of many pathologies, such as infectious diseases, leukaemia or other haematological disorders rely on the classification of subtypes of white blood cells, a.k.a. leukocytes. Several biological techniques exist to identify leukocytes but the microscopic examination of blood smears is often crucial for the confirmation of diagnosis. Cells are identified by different characteristics: their granules, the number of lobes in the nucleus, the shape of the nucleus, the colour of the cytoplasm.**

***Image***

**This technique however is prone to error, time-consuming and requires experts, which is why computer-aided analysis of blood smears have been developed. Classical leukocyte classification pipelines involve segmentation of the cell from its surrounding, feature extraction and selection, followed by shallow**[**machine learning**](https://datascientest.com/en/machine-learning-what-is-it-and-why-does-it-change-the-world)**classifiers. This type of workflow is extremely difficult to generalise from one laboratory to the other because of the variety of staining, protocols and acquisition systems.**

**This is where deep learning can become handy.**

**Classically, circulating blood cells are split into 5 major subtypes:platelets, red blood cells, granulocytes (basophils, neutrophils, eosinophils), monocytes, and lymphocytes.**

**Peripheral white blood cells and precursors. Image adapted from [A.Rad on wikipedia](https://fr.wikipedia.org/wiki/Fichier:Hematopoiesis_(human)_diagram.png).**

**We have tested here 3 types of deep learning architecture to classify 11 classes of healthy blood cells: the 5 major subtypes and some of their progenitors (neutrophils progenitors).**

* **neutrophils (segmented) – SNE**
* **eosinophils – EO**
* **basophils – BA**
* **lymphocytes – LY**
* **monocytes – MO**
* **platelets – PLATELET**
* **erythroblasts – ERB**
* **immature (metamyelocytes, myelocytes, promyelocytes) and band neutrophils – MMY, – MY, – PMY, and – BN**

**TESTING MODEL &DA**

**CHAPTER-6**

**MODULE IMPLIMENTATION**

**6.MODULE IMPLEMENTATION**

In this chapter, we present the functional decomposition and implementation of each module involved in the development of the Multilingual Intent-Based Voice/Text Chatbot System. The application has been logically divided into three primary modules — User Module, System Module, and Algorithm Module

**6.1 USER MODULE**

This is the **front-end interface** where users interact with the chatbot via **text or voice**.

* + 1. **User Input**:
  + Supports both typed and spoken queries.
  + Uses HTML/CSS frontend, optionally with JavaScript and browser voice APIs.
  + Voice is converted to text using STT libraries or APIs.

**6.1.2 Response Delivery and UI feedback**:

* + Displays real-time responses in a chat interface.
  + Supports multilingual translation back to the user’s language.
  + May include voice output and conversation history in future updates.

**6.2 SYSTEM MODULE**

The System Module contains the core processing pipeline which transforms raw user input into meaningful bot responses. This includes translation, preprocessing, intent matching, response retrieval, and re-translation if required. It is the orchestrator that communicates with all other subsystems and manages the routing logic.

This module handles **core logic and backend processing**, including input validation, translation, intent matching, and response delivery.

**6.2.1 Input Handling and validation**:

* + Validates user input and passes it for translation.
    1. **Multilingual Support(Translate Engine)**:
  + Uses deep\_translator or Google Translate to convert all queries to English.

**6.2.3 Response Retrieval and final translation**:

Fetches the correct answer and translates it back to the original language if needed

**6.2.4 Backend Routing & Flask app**:

Backend built with Flask.

Endpoints handle input and return responses.

Supports deployment via Docker, Heroku, or local servers.

**6.3 ALGORITHM MODULE**

The algorithm module incorporates the core AI/NLP logic that powers the chatbot’s ability to interpret queries and generate intelligent responses.

* + 1. **Fuzzy String Matching Algorithm**:
  + Uses algorithms like Levenshtein Distance and Token Ratios to compute similarity scores between input and stored questions.
    1. **NLP based Preprocessing**:

Cleans and standardizes text with tokenization, stopword removal, lowercasing, and lemmatization.

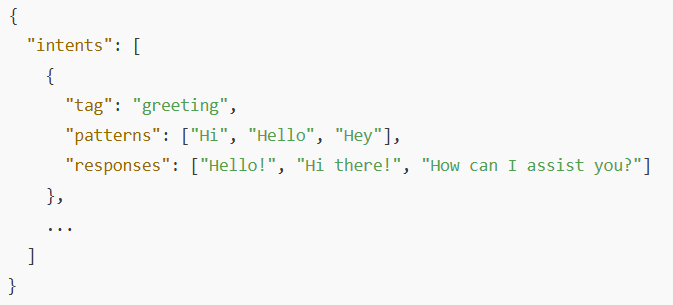
* + 1. **Multilingual Translation logic**:

Converts user queries to English and responses back to the user’s language for multilingual support.

* + 1. **JSON Knowledge Base(intent response storage)**:

Stores predefined questions and answers in a structured format for fast lookup.

**Format:**

****Each pattern is matched against the query, and the corresponding response is selected using probabilistic logic or deterministic scoring.

**CHAPTER-7**

**SOURCE CODE**

**7.SOURCE CODE**

**App.py**

from flask import Flask, render\_template, request, jsonify

import json

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from deep\_translator import GoogleTranslator

from fuzzywuzzy import process

import re

import os

# Download necessary NLTK data

nltk.download("punkt")

nltk.download("stopwords")

app = Flask(\_name\_)

app.secret\_key = "your\_very\_secure\_secret\_key\_12345!@#$%"

# Load chatbot data

try:

with open("college\_data.json", "r", encoding="utf-8") as f:

category\_responses = json.load(f)

except (FileNotFoundError, json.JSONDecodeError):

category\_responses = {}

# Preload stopwords

STOP\_WORDS = set(stopwords.words("english"))

def preprocess\_text(text):

"""Remove stopwords, tokenize, and clean user query."""

words = word\_tokenize(text.lower())

filtered\_words = [word for word in words if word.isalnum() and word not in STOP\_WORDS]

return " ".join(filtered\_words)

def clean\_text(text):

"""Remove special characters from text."""

return re.sub(r"[^\w\s]", "", text)

def translate\_text(text, source\_lang, target\_lang):

"""Translate text using GoogleTranslator with exception handling."""

cleaned\_text = clean\_text(text)

try:

return GoogleTranslator(source=source\_lang, target=target\_lang).translate(cleaned\_text)

except Exception as e:

print(f"Translation failed: {e}")

return text # Return original text if translation fails

@app.route("/")

def home():

return render\_template("index1.html", chat\_history=[])

@app.route("/chat", methods=["POST"])

def chat():

data = request.json

user\_query = data.get("query", "").strip()

user\_lang = data.get("language", "en")

if not user\_query:

return jsonify({"response": "Please enter a valid query."})

# Translate user query to English

processed\_query = translate\_text(user\_query, user\_lang, "en")

processed\_query = preprocess\_text(processed\_query)

# Find best match response with fuzzy matching

best\_match, score = process.extractOne(processed\_query, category\_responses.keys(), score\_cutoff=40)

if best\_match:

response = category\_responses.get(best\_match, "Sorry, I couldn't find relevant information.")

else:

response = "Sorry, I couldn't find the information. Please try rephrasing."

# Translate response back to user's language

response\_translated = translate\_text(response, "en", user\_lang)

return jsonify({"response": response\_translated})

if \_name\_ == "\_main\_":

app.run(debug=True)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>College Chatbot</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style1.css') }}">

</head>

<body>

<div class="chat-container">

<h2>College Chatbot</h2>

<div class="chat-box" id="chat-box"></div>

<!-- Language Selection -->

<div class="language-container">

<label for="language">Choose Language:</label>

<select id="language">

<option value="en">English</option>

<option value="hi">Hindi</option>

<option value="te">Telugu</option>

<option value="ta">Tamil</option>

</select>

</div>

<!-- Chat Input -->

<form class="input-area">

<input type="text" id="query" placeholder="Type or speak..." required autofocus>

<button type="submit">Send</button>

<button type="button" class="mic-button" onclick="startSpeechRecognition()">🎙</button>

</form>

</div>

<script>

document.querySelector(".input-area").addEventListener("submit", function(event) {

event.preventDefault();

let userQuery = document.getElementById("query").value.trim();

let userLang = document.getElementById("language").value;

if (!userQuery) return;

displayMessage(userQuery, "user-message");

fetch("/chat", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({ query: userQuery, language: userLang })

})

.then(response => response.json())

.then(data => {

let botResponse = data.response || "Sorry, I couldn't understand.";

displayMessage(botResponse, "bot-message");

speakResponse(botResponse, userLang);

})

.catch(error => {

console.error("Error:", error);

displayMessage("An error occurred. Please try again.", "bot-message");

});

document.getElementById("query").value = "";

document.getElementById("query").focus();

});

function displayMessage(message, className) {

let chatBox = document.getElementById("chat-box");

let messageDiv = document.createElement("div");

messageDiv.className = "message " + className;

messageDiv.textContent = message;

chatBox.appendChild(messageDiv);

chatBox.scrollTop = chatBox.scrollHeight;

}

function startSpeechRecognition() {

if (!("SpeechRecognition" in window || "webkitSpeechRecognition" in window)) {

alert("Speech recognition is not supported in this browser.");

return;

}

const recognition = new (window.SpeechRecognition || window.webkitSpeechRecognition)();

let selectedLang = document.getElementById("language").value;

let langMapping = { "en": "en-US", "hi": "hi-IN", "te": "te-IN", "ta": "ta-IN" };

recognition.lang = langMapping[selectedLang] || "en-US";

recognition.start();

recognition.onresult = function(event) {

document.getElementById("query").value = event.results[0][0].transcript;

};

recognition.onerror = function(event) {

console.error("Speech recognition error:", event.error);

displayMessage("Speech recognition failed. Try again.", "bot-message");

};

}

function speakResponse(text, language) {

if (!window.speechSynthesis) {

console.warn("Speech synthesis not supported in this browser.");

return;

}

const synth = window.speechSynthesis;

const utterance = new SpeechSynthesisUtterance(text);

const langMapping = { "en": "en-US", "hi": "hi-IN", "te": "te-IN", "ta": "ta-IN" };

utterance.lang = langMapping[language] || "en-US";

let voices = synth.getVoices();

let femaleVoices = voices.filter(voice => voice.lang === utterance.lang &&

(voice.name.toLowerCase().includes("female") || voice.name.toLowerCase().includes("woman") || voice.name.toLowerCase().includes("zira") || voice.name.toLowerCase().includes("samantha"))

);

if (femaleVoices.length > 0) {

utterance.voice = femaleVoices[0]; // Pick the first female voice

} else {

console.warn("No female voice found, using default.");

}

synth.speak(utterance);

}

// Ensure voices load before selecting one

window.speechSynthesis.onvoiceschanged = function() {

console.log("Voices updated.");

};

</script>

</body>

</html>

/\* General Styles \*/

body {

font-family: 'Poppins', sans-serif;

background: linear-gradient(135deg, #11cb2d, #2575fc);

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

/\* Chat Container \*/

.chat-container {

width: 400px;

max-width: 90%;

background: white;

border-radius: 15px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);

padding: 20px;

display: flex;

flex-direction: column;

}

/\* Chat Box \*/

.chat-box {

height: 300px;

overflow-y: auto;

padding: 10px;

background: #f1f1f1;

border-radius: 10px;

scroll-behavior: smooth;

display: flex;

flex-direction: column;

}

/\* Scrollbar Styling \*/

.chat-box::-webkit-scrollbar {

width: 6px;

}

.chat-box::-webkit-scrollbar-thumb {

background-color: rgba(0, 0, 0, 0.3);

border-radius: 10px;

}

.chat-box::-webkit-scrollbar-track {

background: transparent;

}

/\* Messages \*/

.message {

padding: 10px;

border-radius: 10px;

margin: 5px 0;

max-width: 70%;

animation: fadeIn 0.3s ease-in-out;

word-wrap: break-word;

}

.user-message {

background: #007bff;

color: white;

text-align: right;

margin-left: auto;

}

.bot-message {

background: #f1f1f1;

color: black;

text-align: left;

margin-right: auto;

}

/\* Input Area \*/

.input-area {

display: flex;

gap: 10px;

margin-top: 10px;

}

/\* Input Field \*/

input[type="text"] {

flex: 1;

padding: 10px;

border-radius: 5px;

border: 1px solid #ccc;

font-size: 14px;

}

/\* Buttons \*/

button {

padding: 10px 15px;

background: #6b0aea;

color: white

border: none;

border-radius: 5px;

cursor: pointer;

transition: background 0.3s ease-in-out;

}

button:hover {

background: #4a089c;

}

/\* Mic Button \*/

.mic-button {

background: #28a745;

}

.mic-button:hover {

background: #1e7e34;

}

/\* Language Dropdown \*/

.language-container {

margin-bottom: 10px;

text-align: center;

}

select {

padding: 8px;

border-radius: 5px;

border: 1px solid #ccc;

font-size: 14px;

}

/\* Animations \*/

@keyframes fadeIn {

from {

opacity: 0;

transform: translateY(5px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

**CHAPTER-8**

**SYSTEM TESTING**

### **8.SYSTEM TESTING**

System testing ensures the chatbot functions as expected across all modules and user scenarios. It validated the system's ability to handle **multilingual input**, process **natural language queries**, perform **accurate intent matching**, and deliver **real-time response.**

**8.1 Types of Testing**

System testing ensured the chatbot’s **accuracy, reliability, and multilingual capability** through various testing methods.

#### **8.1.1 Unit Testing**

Tested individual components like the **translation module**, **preprocessing**, **fuzzy matcher**, and **response generator** using tools like **PyTest**. Helped detect early logic errors and validate language handling and similarity scoring.

#### **8.1.2 Integration Testing**

Verified **smooth interaction** between modules such as translation, preprocessing, and response generation. Ensured proper **data flow** and resolved issues like encoding mismatches and response errors.

#### **8.1.3 Manual Testing**

Evaluated real user scenarios such as:

* Queries in **multiple languages** (Hindi, Tamil, English,Telugu)
* Fallback handling for **ambiguous inputs**
* Correct **translation and response display**

#### **8.2 Testing Methodology**

Followed a **bottom-up approach**, from unit testing to full system validation. Maintained logs for:

* Test cases and expected outcomes
* Pass/fail status
* Bugs and resolutions  
  Included **regression testing** after system updates.

#### **8.3 Features Tested**

* Text/voice input validation
* Language detection and translation
* NLP preprocessing and JSON-based response retrieval
* Fuzzy intent matching and Multilingual output and UI behavior

**CHAPTER-9**

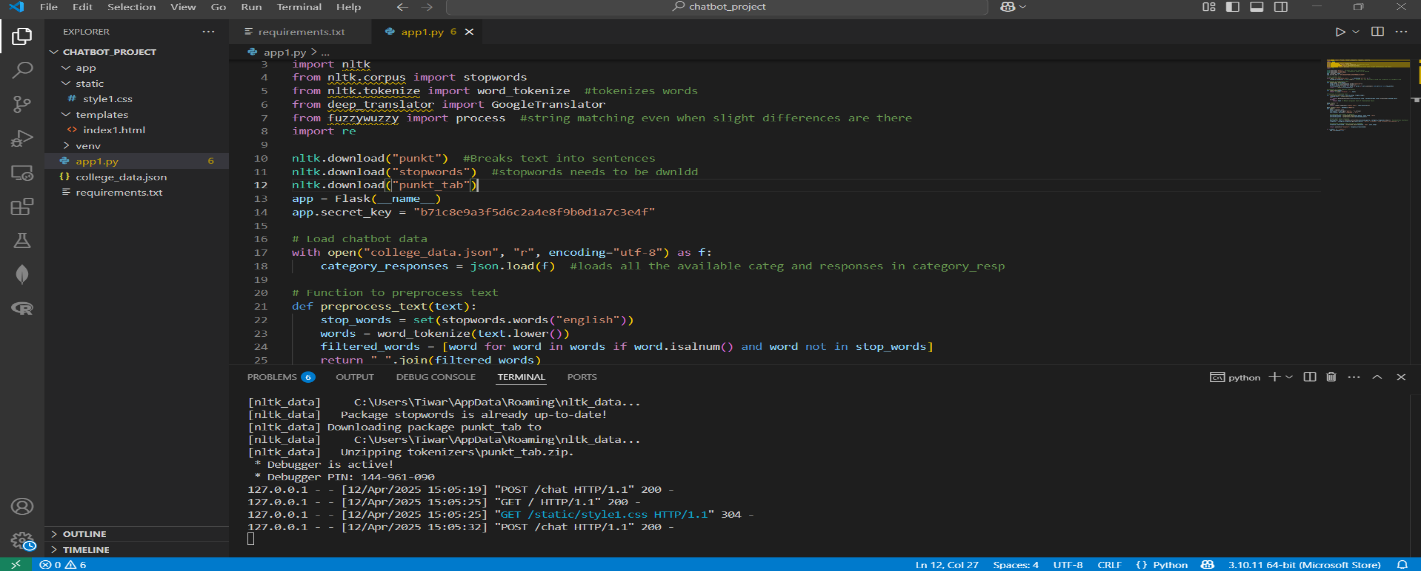
**OUTPUT SCREENSHOTS**

**9**.**OUTPUT SCREENSHOTS**

This section showcases key visual outputs of the **Multilingual Intent-Based Chatbot System**, highlighting both frontend interactions and backend operations.

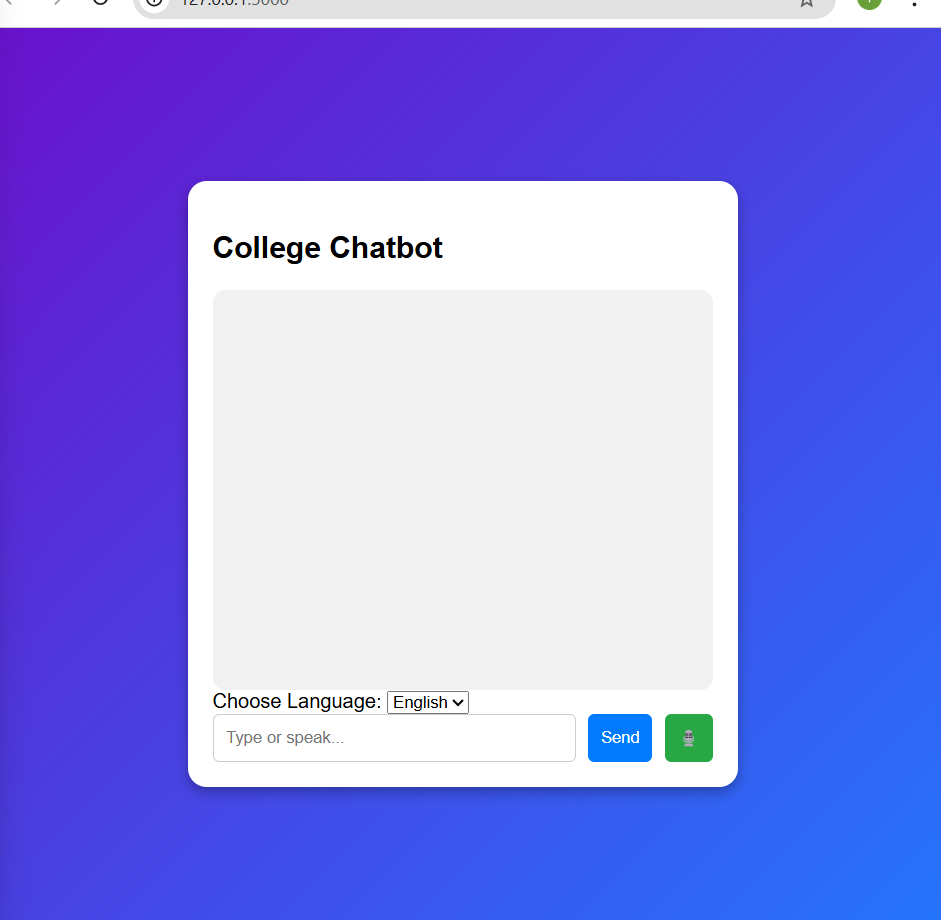
* **Figure 1: Backend Console in Visual Studio Code**  
  Displays the execution of app1.py, which runs the **Flask backend**, handles user requests, and processes input with **NLTK**.  
  The terminal logs confirm:
  + Successful **POST requests**
  + Active **route handling**
  + Proper communication between the **client and server**

This screenshot verifies that the **backend is fully operational** and correctly integrated with the chatbot interface.

 **Fig 9.1 Code Execution**

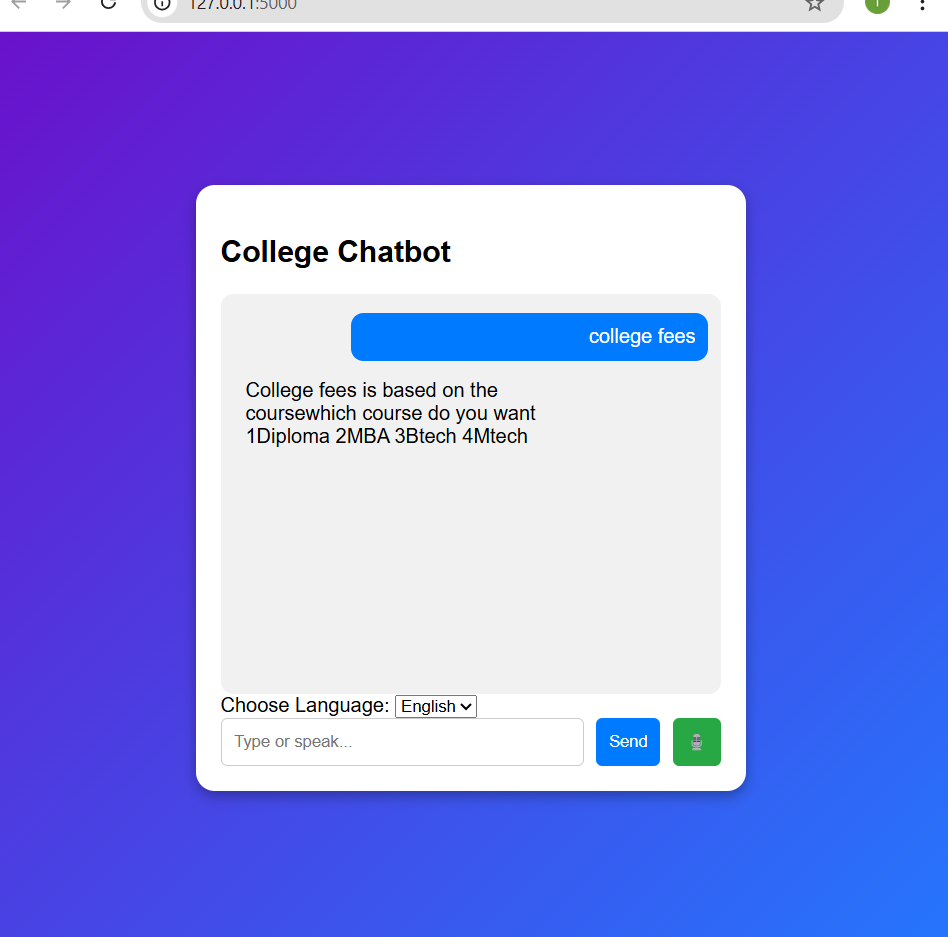
**Figure 2: Initial Chatbot Interface**

The image below shows the homepage of the chatbot when the server is initially launched via Flask at 127.0.0.1:5000. The interface is minimal and user-friendly, designed with accessibility in mind. Users are presented with a chat window, a language selection dropdown, a text input field, and buttons to either submit text or activate voice input. The clean layout allows users to begin interaction immediately without unnecessary distractions.



**Fig 9.2 Chatbot Interface**

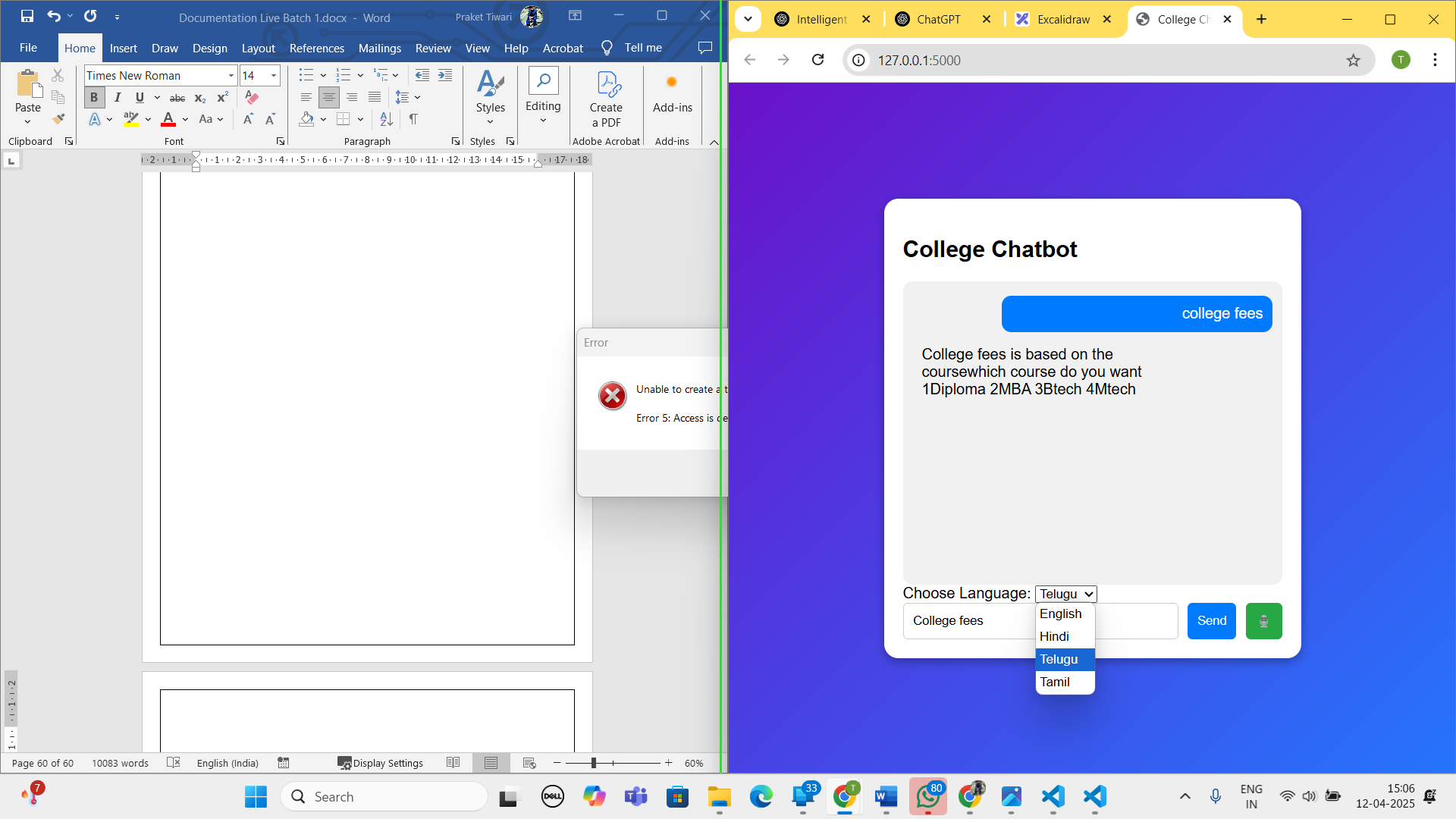
**Figure 3: Chat Interaction with a Query on College Fees**

This screenshot captures a live interaction where the user enters the phrase "college fees". The chatbot successfully identifies the intent and responds with a clarification, asking the user to specify the course. The response is rendered dynamically in the chat window, indicating successful backend processing, intent matching, and content retrieval from the knowledge base. This interaction demonstrates the primary functional behavior of the chatbot—accepting a natural language input and returning a context-aware resp 

**Fig 9.3 Chatbot Interaction**

**Figure 4: Language Selection Functionality**

This image highlights the chatbot’s multilingual capability. The user is shown selecting Telugu from the language dropdown. This feature enables users to communicate in their native language. Internally, the input is translated into English for processing and the response is translated back into the selected language before being displayed. This two-way translation ensures broader accessibility and demonstrates the system's adaptability to regional contexts.



**Fig 9.4 Chatbot Language Selection**

These output screenshots provide visual confirmation of the chatbot’s operational flow, from user input to backend processing and dynamic response generation. Each image supports the claim that the chatbot system is functionally complete, user-oriented, and capable of multilingual interaction across diverse contexts. The interface is responsive, the system logic is sound, and the backend processes user input with high accuracy and consistency.

**CHAPTER-10**

**CONCLUSION**

**and**

**FUTURE ENHANCEMENT**

**10.CONCLUSION AND FUTURE ENHANCEMENTS**

### **10.1 Conclusion**

The Multilingual Intent-Based Chatbot successfully integrates **language translation**, **fuzzy logic intent recognition**, and **structured response generation** within a modular, user-friendly framework. Key strengths include:

* **Multilingual support** for inclusive communication
* **Robust intent matching** tolerant of phrasing and spelling errors
* **Simple UI** with voice/text interaction
* **Lightweight backend** built on Flask
* **Thorough testing** to ensure reliability

The project meets its goals and offers a scalable foundation for advanced features like **AI-based intent detection, cloud deployment**, and **context-aware interaction**. With continued refinement, this chatbot can evolve into a full-featured, intelligent virtual assistant for domains like education, customer service, and public support.

### **10.2 Future Enhancements**

The current chatbot system provides solid functionality with multilingual support, fuzzy intent matching, and a static knowledge base. However, several future upgrades are proposed to improve performance, scalability, and user experience:

* **Advanced Intent Detection**: Replace fuzzy matching with deep learning models like BERT or RoBERTa for better semantic understanding and accuracy.
* **Dynamic Knowledge Base**: Migrate from static JSON to a real-time database (SQL/NoSQL) for easier updates and potential self-learning features.
* **Text-to-Speech (TTS)**: Enable voice-based responses for improved accessibility, especially for visually impaired users.
* **Enhanced UI**: Add features like typing animations, conversation history, responsive design, and avatar-based interactions to boost engagement.
* **Sentiment Analysis**: Incorporate emotion recognition to adjust tone and improve interaction in sensitive or personal contexts.
* **Admin Dashboard**: Build a web-based control panel for non-technical administrators to manage intents, track activity, and view analytics.
* **Cloud Deployment**: Migrate to platforms like AWS or Google Cloud for scalability, better performance, and multi-department access.
* **Multi-turn Conversations**: Introduce context retention to allow continuous, natural dialogue and follow-up queries.

**CHAPTER-11**

**REFERENCE**

**11.REFERENCES**

1. Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, *"DeepFace: Closing the Gap to Human-Level Performance in Face Verification,"* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
2. Joy Buolamwini and Timnit Gebru, *"Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification,"* Proceedings of the 1st Conference on Fairness, Accountability, and Transparency, 2018.
3. Google Cloud, *"Cloud Translation API Documentation."* [Online]. Available: https://cloud.google.com/translate
4. Matthew Honnibal and Ines Montani, *"spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,"* To appear, 2017.
5. Jason Brownlee, *"Introduction to Text Cleaning with Python,"* Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/text-cleaning-for-machine-learning/
6. NLTK Project, *"Natural Language Toolkit Documentation,"* [Online]. Available: <https://www.nltk.org/>
7. Python Software Foundation, *"Python Language Reference, version 3.x,"* [Online]. Available: <https://www.python.org/>
8. Flask Documentation, *"Flask Web Framework,"* [Online]. Available: https://flask.palletsprojects.com/
9. RapidFuzz Library, *"RapidFuzz: Rapid String Matching in Python,"* [Online]. Available: https://maxbachmann.github.io/RapidFuzz/
10. Deep Translator Documentation, *"deep\_translator — Python translation library,"* [Online]. Available: <https://pypi.org/project/deep-translator/>
11. Tesseract OCR, *"Tesseract Open Source OCR Engine,"* [Online]. Available: <https://github.com/tesseract-ocr/>
12. K. Kowsalya, *"Artificial Intelligence Chatbot System Using Natural Language Processing,"* International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 11, pp. 3204–3208, 2019.