

MAHATMA EDUCATION SOCIETY'S
PILLAI COLLEGE OF ARTS, COMMERCE &A SCIENCE

(Autonomous)
NEW PANVEL

PROJECT REPORT ON

“Grammify AI – Transcribe, Correct, Perfect!”

IN PARTIAL FULFILLMENT OF
MASTER OF DATA ANALYTICS

SEMESTER IV – 2024-25

PROJECT GUIDE

Shivapradeep Muthupandi

SUBMITTED BY : Athulkrishna Pramod

ROLL NO : 6856

Mahatma Education Society's
PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)
Re-accredited "A" Grade by NAAC (3rd Cycle)



Project Completion Certificate

THIS IS TO CERTIFY THAT

Athulkrishna Pramod

of **M.Sc. Data Analytics Part - II** has completed the
project titled **Grammify AI** of subject **Natural Language
Processing** under our guidance and supervision during
the academic year 2024-25

Project Guide

Course
Coordinator

Head of the
Department



Introduction

Grammify is an AI-powered application designed to seamlessly convert speech into text while ensuring grammatical accuracy. By leveraging OpenAI's Whisper model for speech-to-text transcription and a T5-based grammar correction model, Grammify provides users with high-quality, error-free text output. Whether you're transcribing an audio recording, dictating notes, or correcting written text, the platform ensures clarity and correctness, making communication more effective and professional.

With an intuitive interface, Grammify allows users to upload audio files for transcription or input text manually for grammar correction. It also supports document uploads, including PDFs and text files, making it a versatile tool for students, professionals, and content creators. By combining cutting-edge AI with user-friendly functionality, Grammify enhances productivity and streamlines the process of generating polished, well-structured text.

Grammify is built to cater to a wide range of users, from students looking to refine their essays to professionals transcribing meetings with accuracy. Its AI-driven approach ensures that even complex sentences are corrected with contextual understanding, improving both grammar and readability. Additionally, with support for text downloads, users can easily save and share their corrected content. Whether for academic, business, or personal use, Grammify empowers users to communicate more effectively with confidence.

Objective

The objective of Grammify is to develop an AI-powered application that seamlessly converts speech into text and enhances its grammatical accuracy. By integrating advanced speech recognition and natural language processing models, the project aims to provide users with a reliable tool for transcription and grammar correction, ensuring clear, professional, and error-free text output.

Key Objectives -

- Accurate speech-to-text transcription using OpenAI's Whisper model.
- AI-powered grammar correction with a T5-based language model.
- Support for both audio and text input, including document uploads (PDFs & text files).
- User-friendly interface for effortless transcription and correction.
- Option to download corrected text for easy access and sharing.
- Enhanced productivity for students, professionals, and content creators.

Grammify aims to bridge the gap between spoken and written communication by providing a smart and efficient solution for transcription and grammar refinement. By leveraging deep learning models, the platform ensures that users receive not just accurate transcriptions but also polished and grammatically sound text. Whether for academic writing, professional documentation, or content creation, Grammify enhances the writing experience by minimizing errors and improving readability, ultimately helping users communicate more effectively.

Methodology

The development of **Grammify** follows a structured approach, integrating state-of-the-art AI models for speech-to-text transcription and grammar correction. The methodology consists of multiple stages, from data acquisition to deployment, ensuring a smooth and efficient workflow.

1. System Design & Architecture

- Define project requirements, including speech-to-text conversion and grammar correction.
- Design a web-based Flask application for seamless user interaction.
- Structure the application into different modules: transcription, text correction, and file processing.

2. Model Selection & Integration

- Utilize **OpenAI's Whisper model** for accurate speech-to-text transcription.
- Implement **T5-based grammar correction model** ([vennify/t5-base-grammar-correction](#)) for refining transcribed and user-input text.
- Leverage **PyTorch** and **Transformers** to load and fine-tune models if needed.

3. Audio Processing & Transcription

- Accept user-uploaded audio files through the Flask interface.
- Convert speech to text using the **Whisper model**, ensuring high accuracy across different accents and noise levels.
- Extract and preprocess text for further grammar correction.

4. Grammar Correction & Text Refinement

- Tokenize and process transcribed text using the **T5 tokenizer**.
- Generate grammatically corrected text using the **T5-based model**.
- Display the refined text while retaining the original for comparison.

5. File Upload & Processing

- Allow users to upload text and PDF files for grammar correction.
- Extract text from PDF documents using **PyPDF2**.
- Process and refine extracted content using the **T5 grammar correction model**.

6. Web Application Development

- Build a Flask-based web interface with **HTML, CSS, and JavaScript** for user-friendly interaction.
- Implement **API endpoints** for handling transcription and text correction requests.
- Enable **file download** functionality for corrected text.

7. Testing & Optimization

- Conduct unit and integration testing to ensure model accuracy and application stability.
- Optimize processing speed and response time for real-time transcription and correction.
- Improve the UI/UX for seamless user experience.

8. Deployment & Maintenance

- Deploy the application using **Flask on a cloud platform** (e.g., AWS, Google Cloud, or Heroku).
- Ensure scalability and security for handling multiple users.
- Provide updates and improvements based on user feedback.

Code & Ouput

Python Code –

- **app.py:**

```
from flask import Flask, request, jsonify, render_template, send_file
import whisper
import torch
from transformers import T5ForConditionalGeneration, T5Tokenizer
import os
from PyPDF2 import PdfReader

app = Flask(__name__)

# Load Whisper model
whisper_model = whisper.load_model("base")

# Load Grammar Correction Model
t5_model =
T5ForConditionalGeneration.from_pretrained("vennify/t5-base-grammar-correction")
t5_tokenizer = T5Tokenizer.from_pretrained("vennify/t5-base-grammar-correction")

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/speech")
def ppeech_page():
    return render_template("speech.html")

@app.route("/text")
def text_page():
    return render_template("text.html")

@app.route("/transcribe", methods=["POST"])
def transcribe_audio():
```

```

audio_file = request.files["audio"]
audio_path = "temp_audio.wav"
audio_file.save(audio_path)

# Transcribe audio
result = whisper_model.transcribe(audio_path)
transcribed_text = result["text"]

# Grammar Correction
input_text = "grammar: " + transcribed_text
input_ids = t5_tokenizer.encode(input_text, return_tensors="pt", max_length=512,
truncation=True)
output_ids = t5_model.generate(input_ids, max_length=512, num_beams=4,
early_stopping=True)
corrected_text = t5_tokenizer.decode(output_ids[0], skip_special_tokens=True)

os.remove(audio_path)

return jsonify({
    "original_text": transcribed_text,
    "transcription": corrected_text
})

@app.route("/correct_text", methods=["POST"])
def correct_text():
    corrected_text = ""

    if "text_input" in request.form:
        text = request.form.get("text_input", "")
        corrected_text = correct_grammar(text)

    elif "file" in request.files:
        file = request.files["file"]
        if file.filename.endswith(".pdf"):
            reader = PdfReader(file)
            text = "\n".join([page.extract_text() or "" for page in reader.pages])
        else:
            text = file.read().decode("utf-8")

```



```

    corrected_text = correct_grammar(text)

# Save corrected text for download
    output_filename = "corrected_text.txt"
    with open(output_filename, "w", encoding="utf-8") as f:
        f.write(corrected_text)

    return jsonify({"corrected_text": corrected_text, "download_url": "/download"})

@app.route("/download")
def download_file():
    return send_file("corrected_text.txt", as_attachment=True)

def correct_grammar(text):
    input_text = "grammar: " + text
    input_ids = t5_tokenizer.encode(input_text, return_tensors="pt", max_length=512,
truncation=True)
    output_ids = t5_model.generate(input_ids, max_length=512, num_beams=4,
early_stopping=True)
    return t5_tokenizer.decode(output_ids[0], skip_special_tokens=True)

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

```

- **index.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Grammar Correction App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
      background-color: #121212;
      color: #f5f5f5;
    }
    h1 { font-size: 28px; }
    p { font-size: 18px; }
    .container { margin-top: 50px; }
    .nav-button {
      display: inline-block;
      font-size: 20px;
      padding: 15px 30px;
      margin: 20px;
      text-decoration: none;
      color: white;
      background-color: #1E88E5;
      border-radius: 8px;
      transition: 0.3s;
    }
    .nav-button:hover {
      background-color: #1565C0;
    }
  </style>
</head>
<body>

  <h1>Welcome to the Grammar Correction App</h1>
  <p>Choose a feature to get started:</p>
```

```

<div class="container">
  <a href="/text" class="nav-button">Text Grammar Correction</a>
  <a href="/speech" class="nav-button">Speech-to-Text & Grammar Correction</a>
</div>

</body>
</html>

```

- speech.html:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Speech-to-Text & Grammar Correction</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
      background-color: #121212;
      color: #f5f5f5;
    }
    .nav-button {
      display: inline-block;
      font-size: 16px;
      padding: 10px 20px;
      margin: 10px;
      text-decoration: none;
      color: white;
      background-color: #1E88E5;
      border-radius: 5px;
      transition: 0.3s;
    }
    .nav-button:hover {
      background-color: #1565C0;
    }
  </style>


```

```

.button-container {
    margin-top: 30px;
}
.record-button {
    font-size: 18px;
    padding: 12px 24px;
    margin: 10px;
    cursor: pointer;
    color: white;
    border: none;
    border-radius: 5px;
    transition: 0.3s;
}
#start {
    background-color: #E53935; /* Red for Start */
}
#start:hover, #start.recording {
    background-color: #C62828;
}
#stop {
    background-color: #43A047; /* Green for Stop */
}
#stop:hover, #stop.stopped {
    background-color: #2E7D32;
}
#status {
    font-size: 18px;
    margin-top: 20px;
    font-weight: bold;
    color: #FFD700; /* Yellow text for status */
}
</style>
</head>
<body>
<h1>Speech-to-Text & Grammar Correction</h1>


<!-- Navigation Buttons -->
<a href="/" class="nav-button"><img alt="Home icon" data-bbox="448 884 471 901" style="vertical-align: middle;"/> Home</a>


```

```
<a href="/text" class="nav-button"> Text Correction</a>
```

```
<!-- Recording Buttons -->
```

```
<div class="button-container">
```

```
  <button id="start" class="record-button"> Start Recording</button>
```

```
  <button id="stop" class="record-button" disabled> Stop Recording</button>
</div>
```

```
<!-- Recording Status -->
```

```
<p id="status">Ready to record</p>
```

```
<p><strong>Original Transcription:</strong></p>
```

```
<p id="original_output"></p>
```

```
<p><strong>Corrected Transcription:</strong></p>
```

```
<p id="corrected_output"></p>
```

```
<script>
```

```
  let mediaRecorder;
```

```
  let audioChunks = [];
```

```
  document.getElementById("start").addEventListener("click", async () => {
    const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
    mediaRecorder = new MediaRecorder(stream);
```

```
    mediaRecorder.ondataavailable = event => {
      audioChunks.push(event.data);
    };
  });
```

```
  mediaRecorder.onstop = async () => {
    const audioBlob = new Blob(audioChunks, { type: "audio/wav" });
    const formData = new FormData();
    formData.append("audio", audioBlob);
```

```
    const response = await fetch("/transcribe", {
      method: "POST",
      body: formData
    });
  });
```

```

        const data = await response.json();
        document.getElementById("original_output").textContent = data.original_text;
// Display raw transcription
        document.getElementById("corrected_output").textContent =
data.transcription; // Display corrected text
    };

    audioChunks = [];
    mediaRecorder.start();

    // Update button styles and status
    document.getElementById("start").classList.add("recording");
    document.getElementById("status").innerText = "🎤 Recording...";
    document.getElementById("status").style.color = "red";

    document.getElementById("start").disabled = true;
    document.getElementById("stop").disabled = false;
});

document.getElementById("stop").addEventListener("click", () => {
    mediaRecorder.stop();

    // Update button styles and status
    document.getElementById("start").classList.remove("recording");
    document.getElementById("status").innerText = "✅ Stopped Recording";
    document.getElementById("status").style.color = "#43A047"; // Green color

    document.getElementById("start").disabled = false;
    document.getElementById("stop").disabled = true;
});
</script>

</body>
</html>

```

- text.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Text Grammar Correction</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
      background-color: #121212;
      color: #f5f5f5;
    }
    h1 {
      margin-bottom: 20px;
    }
    .nav-container {
      margin-bottom: 20px;
    }
    .nav-button {
      display: inline-block;
      font-size: 16px;
      padding: 10px 20px;
      margin: 10px;
      text-decoration: none;
      color: white;
      background-color: #1E88E5;
      border-radius: 5px;
      transition: 0.3s;
    }
    .nav-button:hover {
      background-color: #1565C0;
    }
    textarea {
      width: 80%;
      height: 150px;
```

```
margin: 20px 0;
background-color: #1e1e1e;
color: white;
border: 1px solid #333;
padding: 10px;
border-radius: 5px;
}
.file-upload {
margin: 15px 0;
}
.btn {
font-size: 18px;
padding: 12px 24px;
margin-top: 15px;
cursor: pointer;
border: none;
border-radius: 5px;
transition: 0.3s;
}

/* Centering Download Button */
.download-container {
display: flex;
justify-content: center;
margin-top: 20px;
}

#correct-btn {
background-color: #1E88E5;
color: white;
}
#correct-btn:hover {
background-color: #1565C0;
}
#download-btn {
background-color: #4CAF50;
color: white;
display: none;
}
```



```

#download-btn:hover {
    background-color: #388E3C;
}
#output-container {
    margin-top: 30px;
    text-align: left;
    width: 80%;
    margin-left: auto;
    margin-right: auto;
    padding: 20px;
    background-color: #1e1e1e;
    border-radius: 5px;
}
#loading {
    margin-top: 15px;
    font-size: 18px;
    font-weight: bold;
    display: none; /* Hidden initially */
}
</style>
</head>
<body>
    <h1>Text Grammar Correction</h1>

    <!-- Navigation Buttons -->
    <div class="nav-container">
        <a href="/" class="nav-button">🏠 Home</a>
        <a href="/speech" class="nav-button">🎤 Speech-to-Text</a>
    </div>

    <!-- Text Input -->
    <textarea id="text_input" placeholder="Type your text here..."></textarea>

    <!-- File Upload -->
    <div class="file-upload">
        <input type="file" id="file_input" accept=".txt, .pdf">
    </div>

```

```
<!-- Correct Button -->
<button id="correct-btn" class="btn" onclick="processText()">Correct Text</button>

<!-- Loading Indicator -->
<p id="loading"> ⌚ Correcting... Please wait</p>

<!-- Output -->
<div id="output-container">
  <p><strong>Corrected Output:</strong></p>
  <p id="corrected_output"></p>
</div>

<!-- Download Button -->
<div class="download-container">
  <button id="download-btn" class="btn" onclick="downloadFile()">Download
Corrected Text</button>
</div>

<script>
  function processText() {
    let formData = new FormData();
    let textInput = document.getElementById("text_input").value;
    let fileInput = document.getElementById("file_input").files[0];
    let correctBtn = document.getElementById("correct-btn");
    let loadingIndicator = document.getElementById("loading");

    if (textInput) {
      formData.append("text_input", textInput);
    } else if (fileInput) {
      formData.append("file", fileInput);
    } else {
      alert("Please enter text or upload a file.");
      return;
    }

    // Show loading indicator and disable button
    correctBtn.disabled = true;
```

```

correctBtn.style.backgroundColor = "#555";
loadingIndicator.style.display = "block";

fetch("/correct_text", {
  method: "POST",
  body: formData
})
.then(response => response.json())
.then(data => {
  document.getElementById("corrected_output").innerText =
data.corrected_text;
  document.getElementById("download-btn").style.display = "block";

  // Hide loading indicator and enable button
  loadingIndicator.style.display = "none";
  correctBtn.disabled = false;
  correctBtn.style.backgroundColor = "#1E88E5";
})
.catch(error => {
  console.error("Error:", error);
  loadingIndicator.style.display = "none";
  correctBtn.disabled = false;
  correctBtn.style.backgroundColor = "#1E88E5";
});
}

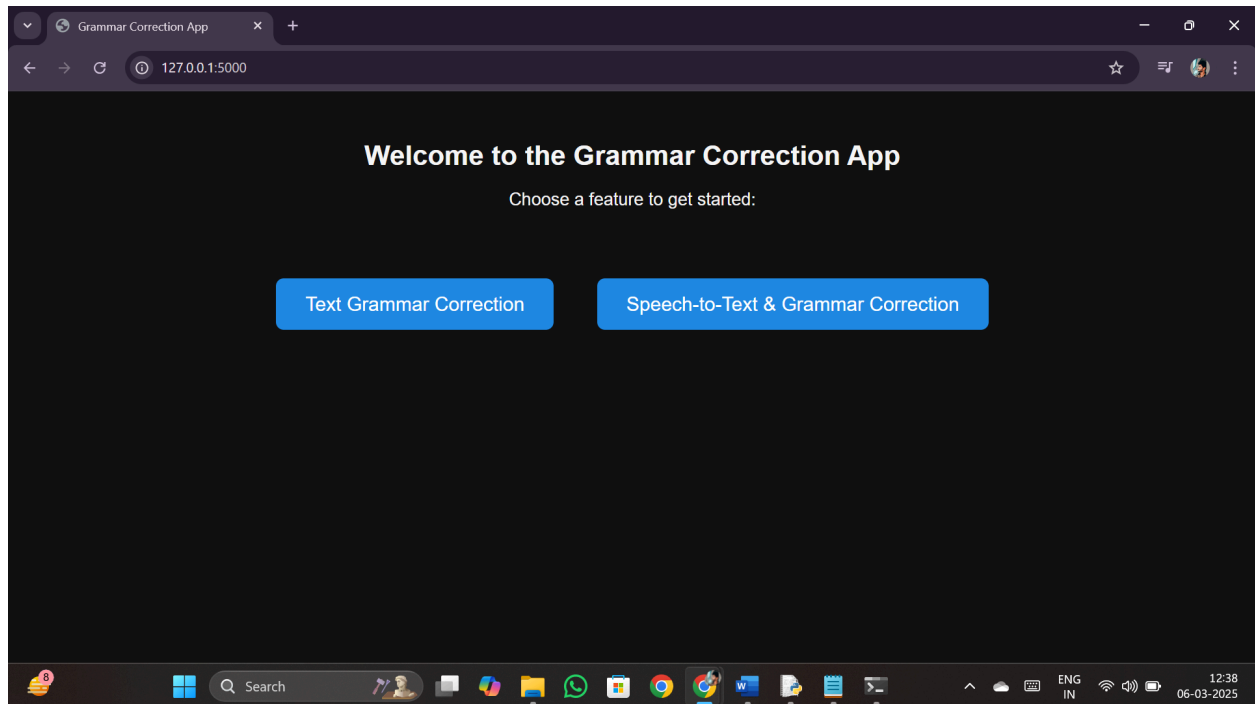
function downloadFile() {
  window.location.href = "/download";
}
</script>

</body>
</html>

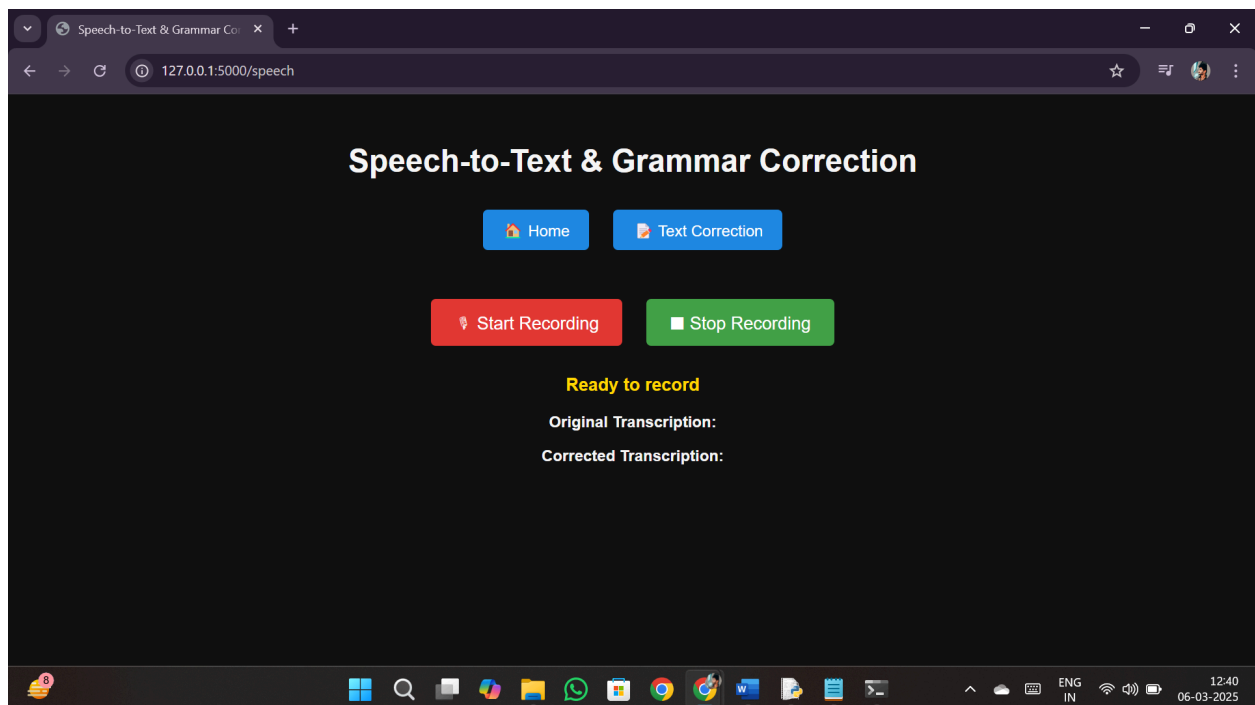
```

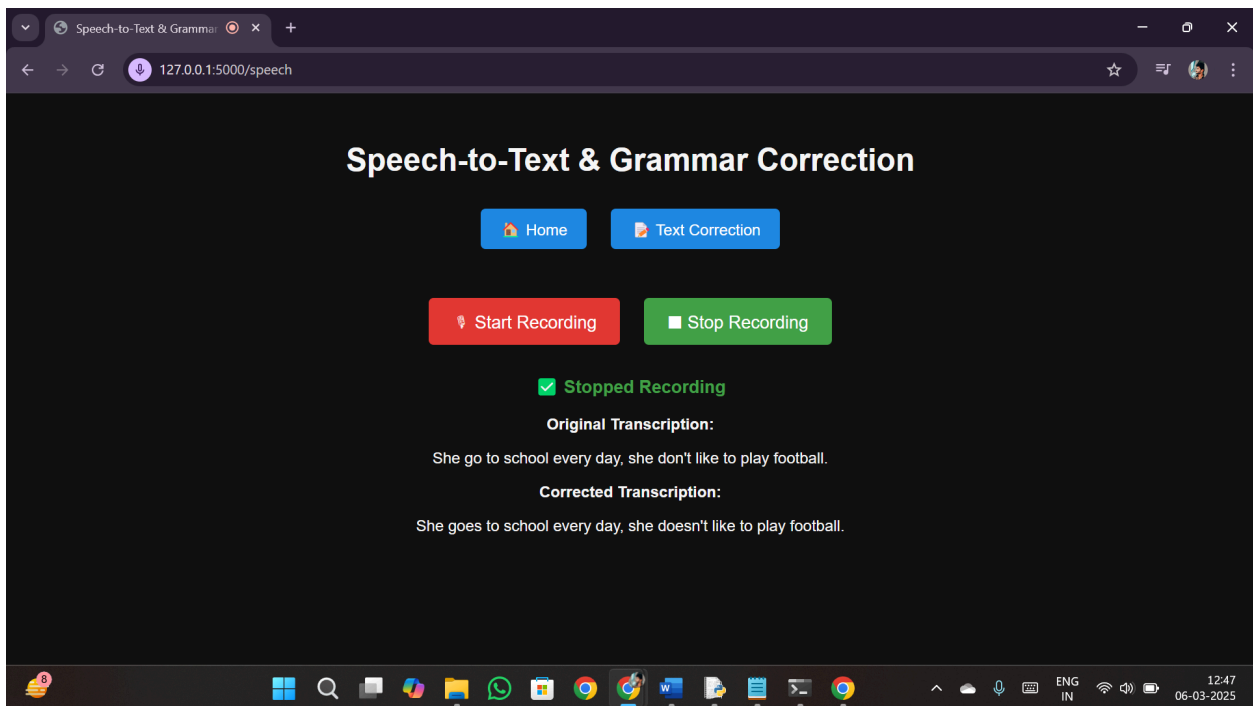
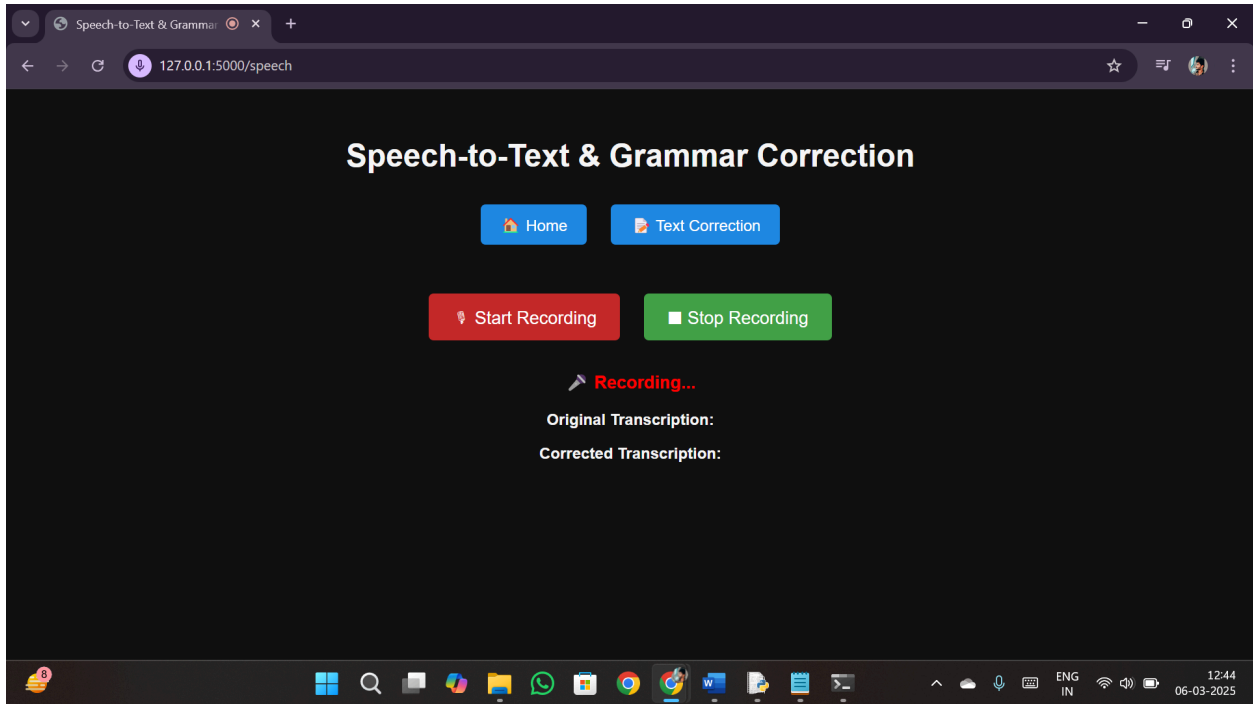
Output –

1.Home Page:

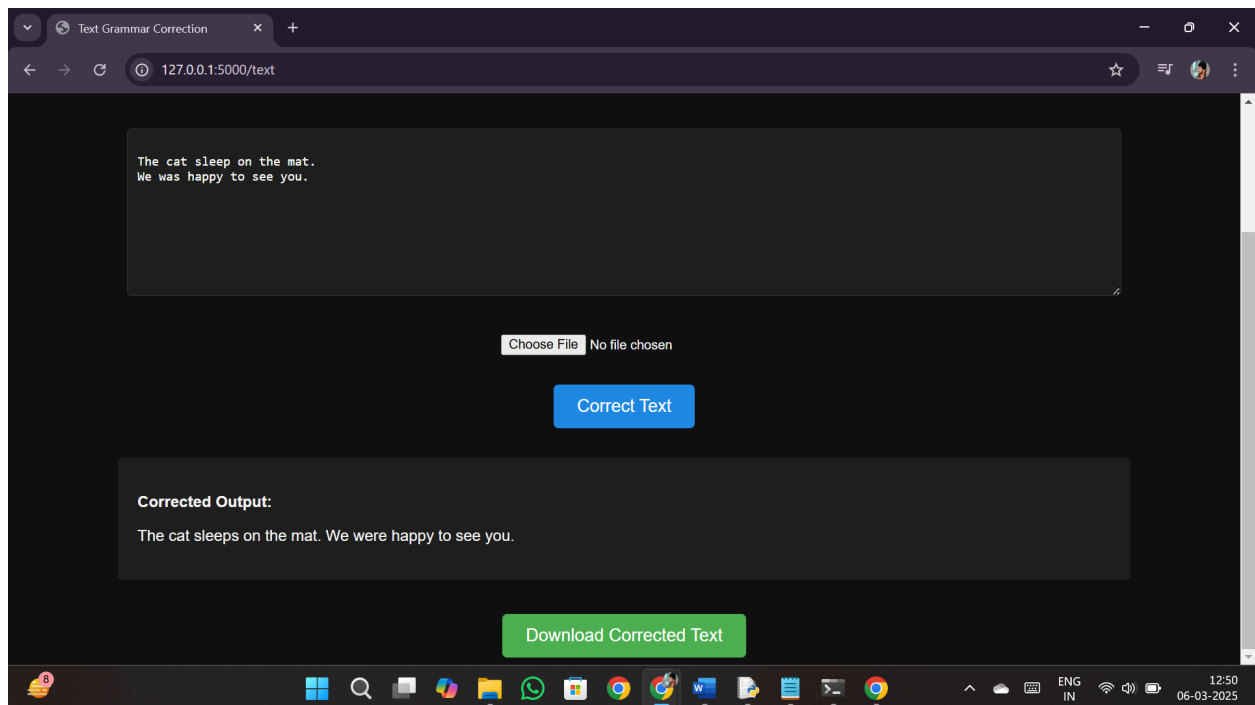
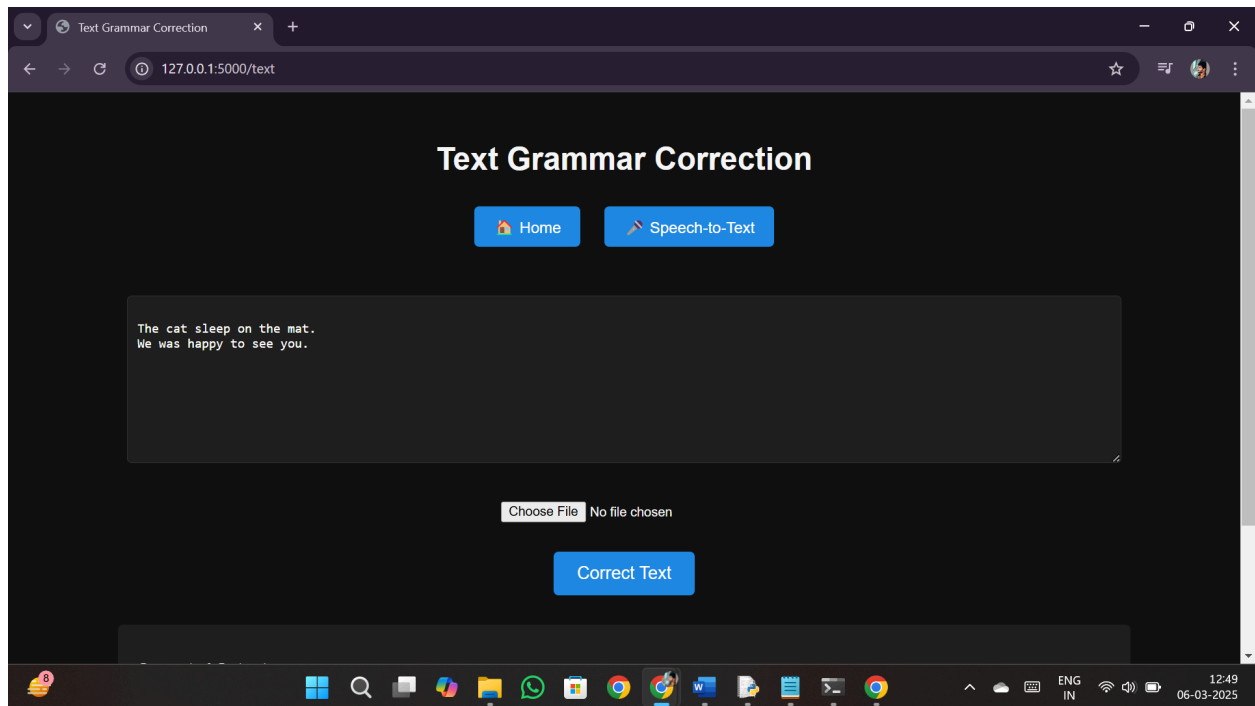


2.Speech to text and grammar correction page:





3.Text Grammar Correction:



Using PDF file:

