# AI-Powered Solution for Assisting Visually Impaired Individuals

Project Report

Developed by: [SAHIL LOTIYA]

Date: [30/11/2024]

## Problem Statement

This project aims to leverage Generative AI to assist visually impaired individuals in perceiving and interacting with their surroundings. Visually impaired individuals often face challenges in understanding their environment, reading visual content, and performing tasks that rely on sight.

There is a need for an intelligent, adaptable, and user-friendly solution that provides:  
- Real-time scene understanding.  
- Text-to-speech conversion for reading visual content.  
- Object and obstacle detection for safe navigation.  
- Personalized assistance for daily tasks.

## Objective

The objective of this project is to build an AI-powered application using Streamlit with at least two core functionalities:  
1. Real-time scene understanding.  
2. Text-to-speech conversion for visual content.

## Features Implemented

### 1. Real-Time Scene Understanding

This feature uses Google Generative AI to describe scenes from uploaded images, enabling users to understand their surroundings.

### 2. Text-to-Speech Conversion

This feature uses PyTesseract for Optical Character Recognition (OCR) to extract text from images and gTTS to convert the extracted text into speech.

## Technology Stack

The project utilizes the following technologies:

* Framework: Streamlit for UI/UX.
* APIs: Google Generative AI for scene understanding.
* Libraries:
* - PyTesseract for OCR.
* - gTTS for text-to-speech conversion.
* - PIL for image handling.

## Implementation Details

The codebase is structured for modularity, with dedicated functions for each feature. Custom styles have been applied for better UI, and error handling mechanisms ensure a smooth user experience. Integration with Google Generative AI provides real-time scene analysis.

## Evaluation Criteria

The project is evaluated based on the following criteria:  
1. Uniqueness of the implementation.  
2. Accurate and functional real-time scene understanding.  
3. Seamless text-to-speech conversion.  
4. Quality of documentation.

## Conclusion

This project demonstrates the potential of AI-powered solutions in assisting visually impaired individuals. Future enhancements could include object detection, obstacle recognition, and personalized task assistance.

## Code:-

import streamlit as st

import google.generativeai as genai

from PIL import Image

import pytesseract

from gtts import gTTS

import io

import base64

# --- CONFIGURATIONS ---

# Configure Tesseract OCR

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Configure the Generative AI API

genai.configure(api\_key="AIzaSyAMjhGfJB86nvfUlLyEjLJMHYEUEPR0kaE")  # Replace with your actual API key

# Initialize the LLM model

llm = genai.GenerativeModel("models/gemini-1.5-flash")

# --- CUSTOM STYLES ---

def apply\_custom\_styles():

    st.markdown(

        """

        <style>

            .main {

                background-color: #f8f9fa;

                padding: 20px;

                border-radius: 10px;

                box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);

            }

            .sidebar .sidebar-content {

                background-color: #e9ecef;

                padding: 15px;

                border-radius: 10px;

            }

            h1 {

                color: #007bff;

            }

            h2, h3 {

                color: #343a40;

            }

            .stButton button {

                background-color: #007bff;

                color: white;

                border-radius: 5px;

            }

        </style>

        """,

        unsafe\_allow\_html=True,

    )

# --- FUNCTIONS ---

def describe\_scene(image):

    """Use Generative AI to describe the scene in the image."""

    try:

        # Convert image to Base64

        image\_bytes = io.BytesIO()

        image.save(image\_bytes, format="JPEG")

        image\_data = base64.b64encode(image\_bytes.getvalue()).decode("utf-8")

        # Create content: Combine text prompt with inline Base64 image data

        prompt = "Describe this scene in detail for a visually impaired individual."

        content = f"{prompt}\n[BASE64\_IMAGE\_START]{image\_data}[BASE64\_IMAGE\_END]"

        # Send request to Generative AI API

        chat = llm.start\_chat()

        response = chat.send\_message(content=content)

        # Extract the text content from the response

        if hasattr(response, 'text'):

            return response.text

        else:

            return "The response does not contain text content."

    except Exception as e:

        return f"Error: {e}"

def text\_to\_speech\_conversion(image):

    """Extract text from the image and convert it to speech."""

    try:

        # Extract text using OCR

        text = pytesseract.image\_to\_string(image)

        if not text.strip():

            return None, "No readable text found in the image."

        # Convert text to speech

        tts = gTTS(text=text, lang="en")

        audio\_file = "output.mp3"

        tts.save(audio\_file)

        return audio\_file, text

    except Exception as e:

        return None, f"Error during text-to-speech conversion: {e}"

# --- MAIN APP ---

def main():

    apply\_custom\_styles()

    st.title("👁️ AI Assistant for Visually Impaired Individuals")

    st.write(

        "This app leverages AI to provide detailed scene descriptions and text-to-speech features."

    )

    # Sidebar for Feature Selection

    with st.sidebar:

        st.header("🔧 Features")

        scene\_understanding = st.checkbox("Describe the Scene in Detail")

        text\_to\_speech = st.checkbox("Text-to-Speech Conversion for Visual Content")

        st.markdown("---")

        st.write("Choose features and upload an image for analysis.")

    # Image Upload Section

    uploaded\_image = st.file\_uploader(

        "📤 Upload an Image to Analyze (JPEG/PNG)", type=["jpg", "png", "jpeg"]

    )

    if uploaded\_image:

        # Display the uploaded image

        image = Image.open(uploaded\_image)

        st.image(image, caption="Uploaded Image", use\_container\_width=True)

        if st.button("🚀 Start Analysis"):

            # Scene Understanding

            if scene\_understanding:

                st.subheader("📸 Scene Understanding")

                with st.spinner("Analyzing the scene..."):

                    result = describe\_scene(image)

                    if result.startswith("Error"):

                        st.error(result)

                    else:

                        st.success("Scene Description:")

                        st.write(result)

            # Text-to-Speech Conversion

            if text\_to\_speech:

                st.subheader("🔊 Text-to-Speech Conversion")

                with st.spinner("Extracting and converting text..."):

                    audio\_file, text\_result = text\_to\_speech\_conversion(image)

                    if audio\_file:

                        st.success("Extracted Text:")

                        st.write(text\_result)

                        st.audio(audio\_file, format="audio/mp3")

                    else:

                        st.warning(text\_result)

    else:

        st.info("📷 Please upload an image to get started.")

    # Footer

    st.caption("Built with ❤️ using Streamlit, Google Generative AI, and PyTesseract.")

# Run the App

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

    main()