

***Project report on the topic:***

***BUILDING AI POWERED SOLUTION FOR ASSISTING***

***VISUALLY IMPAIRED INDIVIDUALS***

***Submitted by:***

**VIGNESH MASANI**

**Building AI Powered Solution for Assisting**

**Visually Impaired Individuals**

**Problem Statement:**

* Visually impaired individuals face significant challenges in understanding their environment, reading visual content, and performing tasks that rely on sight.
* There is a need for an intelligent solution that provides real-time scene understanding, text-to-speech conversion, and object detection to enhance accessibility.

This project aims to develop an AI-powered application that leverages Generative AI to offer these assistive functionalities.

**Objectives:**

1. Build an AI app to aid visually impaired individuals.
2. Generate real-time scene descriptions from images.
3. Extract and vocalize text using OCR and TTS.
4. Detect objects and obstacles for safer navigation.
5. Offer personalized assistance for daily tasks.

**Implementation Requirements:**

**Framework and Tools used**:

* Streamlit: For building the user interface.
* LangChain: To manage interactions with Generative AI models.
* Google Generative AI: For scene understanding and generating descriptive text.
* Tesseract OCR: To extract text from images.
* Pyttsx3 or gTTS: For text-to-speech conversion.

**Functionalities Implemented in this Project:**

**1. Real-Time Scene Understanding:**

* Analyzes the uploaded image to generate detailed scene descriptions.
* Provides users with contextual information about their surroundings.
* Utilizes Google Generative AI for accurate and dynamic text generation.

**2. Text-to-Speech Conversion:**

* Extracts text from images using Tesseract OCR.
* Converts extracted text to audible speech using Pyttsx3 or gTTS.
* Enhances content accessibility by providing audio output directly in the app.

**User Interaction Flow**

1. **Upload Image**: The user accesses the Streamlit application and uploads an image through the file uploader.
2. **Select Functionality**:

* The user selects one or both functionalities:  
  a. **Scene Understanding**  
  b. **Text-to-Speech Conversion**

1. **Processing:**
   * **Scene Understanding**:
     + The uploaded image is processed, and a descriptive text is generated using Google Generative AI.
   * **Text-to-Speech Conversion**:
     + The image is analysed using Tesseract OCR to extract text.
     + The extracted text is converted to speech using Pyttsx3 or gTTS.
2. **Output Delivery**
   * **Scene Description**: Displayed as text on the app interface.
   * **Speech Output**: The app plays the converted text as audio for the user.
3. **Repeat or Exit**
   * The user can upload another image or close the application.

**Outputs:**

1. **Real-Time Scene Understanding**
   * **Text Output**:  
     After uploading an image, the system provides a descriptive output such as:
     + *"A busy street with cars, pedestrians, and street vendors. There is a large tree on the sidewalk, and the sky is overcast."*
2. **Text-to-Speech Conversion**
   * **Audio Output**:  
     The extracted text from the image is converted to speech and played as audio. For example:
     + If the image contains a sign that reads "Caution: Wet Floor", the app will audibly say, *"Caution: Wet Floor."*
3. **Combined Output**
   * For an image showing a "menu" from a restaurant:
     + **Text**: *"The menu includes: Soup of the day, Caesar Salad, Grilled Chicken, and Chocolate Cake."*
     + **Audio**: The app will say, *"The menu includes: Soup of the day, Caesar Salad, Grilled Chicken, and Chocolate Cake."*

**Implementation of Code:**

import streamlit as st

from PIL import Image

import pytesseract

import pyttsx3

import os

import google.generativeai as genai

from langchain\_google\_genai import GoogleGenerativeAI

**Initialize Google Generative AI with API Key**

GEMINI\_API\_KEY = "" # Replace with your valid API key

os.environ["GOOGLE\_API\_KEY"] = GEMINI\_API\_KEY

llm = GoogleGenerativeAI(model="gemini-1.5-pro", api\_key=GEMINI\_API\_KEY)

# Initialize Text-to-Speech engine

engine = pyttsx3.init()

**Streamlit page Application**

st.set\_page\_config(page\_title="SightAssist", layout="wide")

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

st.sidebar.title("🔧 Features")

st.sidebar.markdown("""

- Scene Understanding

- Text-to-Speech

- Object & Obstacle Detection

""")

def extract\_text\_from\_image(image):

"""Extracts text from the given image using OCR."""

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

return text

def text\_to\_speech(text):

"""Converts the given text to speech."""

engine.say(text)

engine.runAndWait()

def generate\_scene\_description(input\_prompt, image\_data):

"""Generates a scene description using Google Generative AI."""

model = genai.GenerativeModel('gemini-1.5-pro')

response = model.generate\_content([input\_prompt, image\_data[0]])

return response.text

**IMPLEMENTATION:**

uploaded\_file = st.file\_uploader("📤 Upload an image...", type=["jpg", "jpeg", "png"])

if uploaded\_file:

image = Image.open(uploaded\_file)

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

# Buttons for functionalities

col1, col2, col3 = st.columns(3)

scene\_button = col1.button("🔍 Describe Scene")

ocr\_button = col2.button("📝 Extract Text")

tts\_button = col3.button("🔊 Text-to-Speech")

# Input Prompt for AI Scene Understanding

input\_prompt = """

You are an AI assistant helping visually impaired individuals by describing the scene in the image. Provide:

1. List of items detected in the image with their purpose.

2. Overall description of the image.

3. Suggestions for actions or precautions for the visually impaired.

"""

# Process based on user interaction

if uploaded\_file:

image\_data = input\_image\_setup(uploaded\_file)

if scene\_button:

with st.spinner("Generating scene description..."):

response = generate\_scene\_description(input\_prompt, image\_data)

st.subheader("Scene Description")

st.write(response)

if ocr\_button:

with st.spinner("Extracting text from image..."):

text = extract\_text\_from\_image(image)

st.subheader("Extracted Text")

st.write(text)

if tts\_button:

with st.spinner("Converting text to speech..."):

text = extract\_text\_from\_image(image)

if text.strip():

text\_to\_speech(text)

st.success("Text-to-Speech Conversion Completed!")

else:

st.warning("No text found in the image.")

**Evaluation Criteria:**

1. **Uniqueness**: The solution should demonstrate creative and innovative use of AI technologies.
2. **Functionality**: At least two core features must be fully implemented and functional.
3. **Accuracy**: The application should provide precise and reliable outputs.
4. **Documentation**: Clear, comprehensive, and easy-to-understand documentation should be provided.

**CONCLUSION**

This AI-powered application successfully leverages advanced technologies to assist visually impaired individuals, enhancing their ability to understand their surroundings and access visual content. By implementing features such as real-time scene understanding and text-to-speech conversion, the application provides valuable support for daily tasks, safety, and accessibility. With its user-friendly interface and reliable functionality, the solution demonstrates the potential of AI to create more inclusive environments for visually impaired individuals.