**Vision360 🔬**

**Project Documentation :**

**Overview**

**Vision360** is an AI-powered application designed to assist visually impaired individuals by providing real-time scene understanding, text extraction, and speech conversion. This project leverages advanced technologies such as Generative AI, Optical Character Recognition (OCR), and Text-to-Speech (TTS) to enhance accessibility and improve the quality of life for users.

**Problem Statement**

Visually impaired individuals often face significant challenges in understanding their environment, reading visual content, and performing tasks that rely on sight. There is a pressing 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.

**Features**

* **Visual Insights**: AI-generated descriptions of uploaded images.
* **Text Extraction**: Detect and extract textual content from images.
* **Speech Conversion**: Convert extracted text into speech for audio playback.

**Technologies Used**

* **Streamlit**: For building the web application interface.
* **Pytesseract**: For Optical Character Recognition (OCR).
* **LangChain**: For integrating Generative AI models.
* **Google Generative AI**: For scene understanding and object detection.
* **gTTS (Google Text-to-Speech)**: For converting text to speech.
* **PIL (Python Imaging Library)**: For image processing.

**Installation Requirements**

To set up the project, ensure you have the following Python packages installed:

A screenshot of a computer

Description automatically generated

Additionally, ensure that Tesseract OCR is installed on your system. You can download it from Tesseract OCR.

**Development Environment**

* **IDE**: Visual Studio Code or any preferred Python IDE.
* **Operating System**: Windows, macOS, or Linux.

**End-to-End Project Build**

**Step 1: Set Up the Environment**

1. Create a new directory for your project.
2. Set up a virtual environment (optional but recommended).
3. Install the required packages using the pip commands mentioned above.

**Step 2: Install Tesseract OCR**

1. Download and install Tesseract OCR from the official repository.
2. Set the path to the Tesseract executable in your code.

**Step 3: Create the Application**

1. Create a new Python file (e.g., **app.py**).
2. Copy the provided code into **app.py**.

**Step 4: API Key Configuration**

1. Create a text file named **API Key.txt** in the same directory as your **app.py**.
2. Add your Google API key to this file.

**Setup Google Generative AI API Key**

Follow the link : <https://github.com/bansalkanav/Generative-AI-Scratch-2-Advance-By-ThatAIGuy/blob/main/4.%20GoogleAI%20Walkthrough/1.%20Getting%20Started%20and%20Generating%20API%20Key/getting_started_with_googleai.ipynb>

**Step 5: Run the Application**

1. Open a terminal and navigate to your project directory.
2. Run the Streamlit application using the command:

A black rectangular object with a white border

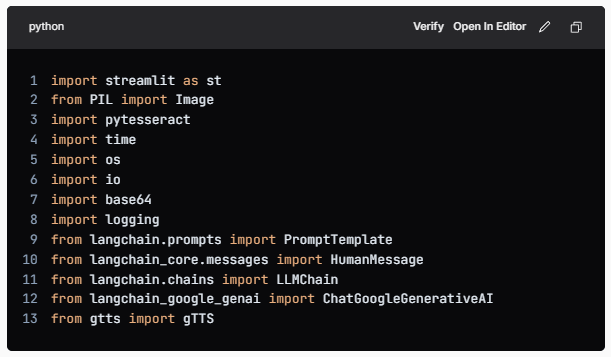
Description automatically generated

**Step 6: Interact with the Application**

1. Open the provided local URL in your web browser.
2. Upload an image and select the desired feature (scene description, text extraction, or speech conversion).

**Code Explanation**

**Importing Libraries**



* **streamlit**: For creating the web application.
* **PIL**: For image processing.
* **pytesseract**: For OCR functionality.
* **time :** For adding delays in streaming responses
* **os, io, base64**: For file handling and data encoding.
* **logging**: For error logging.
* **langchain**: For integrating with Generative AI models.
* **gTTS**: For converting text to speech.

**Setting Up Tesseract OCR**

A black screen with white text

Description automatically generated

* This line sets the path to the Tesseract executable, allowing the application to use OCR capabilities.

**API Key Handling**

**A screen shot of a computer program

Description automatically generated**

* This block reads the API key from a text file and handles the case where the file is not found.

**Error Handling Function**

**A black screen with white text

Description automatically generated**

* A utility function to log errors and display error messages in the Streamlit app.

**Initializing AI Models**

python llm = GoogleGenerativeAI(model="gemini-1.5-pro-latest", google\_api\_key=key) vision\_llm = GoogleGenerativeAI(model="gemini-1.5-pro-latest", google\_api\_key=key)

A screen shot of a computer program

Description automatically generated

* A utility function to log errors and display error messages in the Streamlit app.

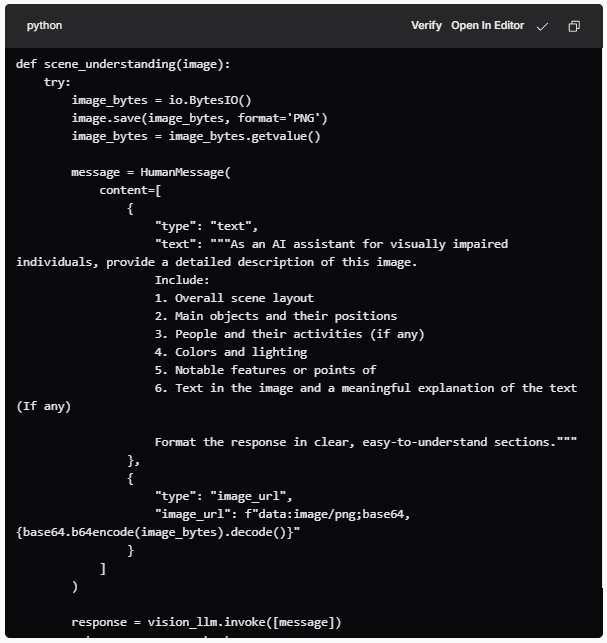
**Streaming Response Data**

**A computer screen with white text

Description automatically generated**

* This function streams the response data word by word, adding a slight delay to simulate a more natural reading pace.

**Scene Understanding Function**

****

* This function processes the uploaded image to generate a detailed description, which is particularly useful for visually impaired users. It converts the image to bytes and sends it to the AI model for analysis.

**Text Extraction Function**

**A screenshot of a computer program

Description automatically generated**

* This function extracts text from the image using OCR and processes it to enhance readability and structure, making it more accessible for users.

**Text-to-Speech Function**

**A screen shot of a computer program

Description automatically generated**

* This function converts the provided text into speech using the gTTS library, enabling audio playback for extracted or generated text.

**Streamlit App Configuration**

**A black and white screen with white text

Description automatically generated**

* Configures the Streamlit app's title, layout, and icon.

**User Interface Elements**

* The application includes a sidebar for information and instructions, a file uploader for image uploads, and buttons for triggering various features (scene description, text extraction, and speech conversion).

**Footer**

**A screen shot of a computer

Description automatically generated**

* The footer provides credits and information about the project, enhancing the professionalism of the application.

**Conclusion**

**Vision360** is a comprehensive solution aimed at improving accessibility for visually impaired individuals. By leveraging cutting-edge AI technologies, the application provides essential functionalities that empower users to interact with their environment more effectively. The project not only addresses a critical need but also showcases the potential of AI in enhancing the quality of life for those with visual impairments.

With features like scene understanding, text extraction, and speech conversion, Vision360 stands as a testament to the capabilities of modern technology in creating inclusive solutions. The project encourages further exploration and development in the field of assistive technologies, paving the way for more innovative applications that can help bridge the gap for individuals with disabilities.

**Raja Sekhar Rapaka**