INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

AI Powered Solution for Assisting Visually Impaired Individuals

Omkar Arun Shinde
ID : IN9240411

About me

I am currently pursuing a B.Tech degree in Computer Engineering at Pimpri Chinchwad College of Engineering, Pune. With a strong foundation in mathematics, having been selected for the Indian National Mathematical Olympiad (INMO) in 2020, I have always been deeply passionate about problem-solving and analytical thinking. This love for mathematics naturally led me to explore data science, where I can combine my analytical skills with programming to uncover patterns, insights, and solutions in complex datasets.

I am particularly drawn to data science because it offers a perfect blend of my interests in statistics, machine learning, and data-driven decision-making. The field's ability to transform raw data into actionable knowledge fascinates me, and I love the challenge of making sense of numbers, finding trends, and predicting outcomes.

About me

Beyond academics, I am also passionate about poetry, novel reading, and acting, all of which help me cultivate a creative and critical mindset. These interests have shaped my approach to data science, allowing me to think outside the box and approach problems with a fresh perspective.
This is my first internship, and I am currently enjoying every moment of it. I am gaining a wealth of knowledge, particularly in exploratory data analysis, Python, and machine learning. It has been an enriching experience, providing me with hands-on learning opportunities and a deeper understanding of the data science field.

https://www.linkedin.com/in/omkararunshinde/

https://github.com/Omkar-Shinde12/

INNOMATICS RESEARCH LABS

Agenda

- Business Problem and Use case domain understanding(If Required)
- Objective of the Project
- Work Flow

- **Work Flow**

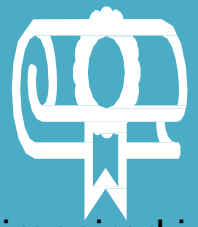
a. *Environment Setup*
b. *Core Functionalities*
c. *User Interaction*
d. *Output Delivery*

Conclusion (Key finding overall)

Your Experience/Challenges working on this project.

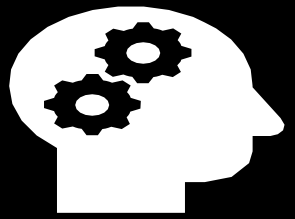
INNOMATICS RESEARCH LABS

Business Problem Statement



Visually impaired individuals face significant challenges in navigating their environments, detecting obstacles, and performing tasks such as reading text or identifying objects. The problem lies in creating an intelligent, real-time system to assist visually impaired people in understanding their surroundings, recognizing objects, and receiving actionable insights.

Develop a user-friendly AI-powered tool leveraging computer vision and NLP to provide real-time scene descriptions and assistance for visually impaired individuals. Prioritize accessibility, real-time performance, and privacy while continuously improving based on user feedback and technological advancements. Promote the system through collaborations with accessibility-focused organizations.



To empower visually impaired individuals by providing real-time, AI-driven assistance for understanding their surroundings, improving accessibility, independence, and quality of life through advanced technology.

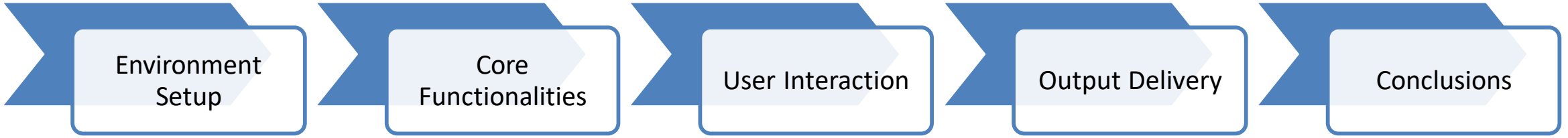
Objective

The primary objective of this project is to develop a robust AI-based vision assistance system that can:

1. Detect and highlight objects in images or videos.
2. Extract text from images.
3. Provide detailed descriptions of scenes and objects.
4. Offer voice-based assistance for a hands-free user experience.
5. Assist in daily tasks by analyzing the environment and identifying potential obstacles or objects.

This application aims to enhance the independence and quality of life of visually impaired users.

Workflow:-



INNOMATICS RESEARCH LABS

Overview of Workflow

The application consists of the following steps:

1. Environment Setup

•Import necessary libraries and modules such as streamlit, torch, pytesseract, cv2, and google.generativeai.

•Configure API keys using dotenv to ensure secure access to external services.

•Initialize pre-trained models like fasterrcnn_resnet50_fpn for object detection.

2. Core Functionalities

•**Text Extraction from Images**: Use pytesseract to extract readable text.

•**Object Detection**: Use a pre-trained Faster R-CNN model for detecting objects in images and drawing bounding boxes.

•**Scene Description**: Leverage a generative AI model to analyze and describe the scene.

•**Text-to-Speech Conversion**: Use pyttsx3 to convert extracted text or scene descriptions into speech for auditory feedback.

3. User Interaction

•Users can upload images or initiate video capture.

•The app processes the input, performs specified tasks, and returns results in the form of annotated images, text, or speech.

4. Output Delivery

•Annotated images are displayed with detected objects and their labels.

•Descriptions are provided for scenes, objects, and text content.

•Voice-based assistance delivers the outputs audibly.

Environment Setup

1. Import Libraries and Modules

The libraries and modules used in the project are essential for various functionalities, including object detection, text recognition, image processing, and natural language processing.

VisionAssist.py | Langchain.ipynb

C: > Users > shind > Python_Coding > API > VisionAssist.py > ...

```python
import streamlit as st
import google.generativeai as genai
from PIL import Image, ImageDraw
import pytesseract
import pyttsx3
import torch
from torchvision import transforms
from torchvision.models.detection import fasterrcnn_resnet50_fpn
from dotenv import load_dotenv
import cv2
import os
import uuid
import time
from io import BytesIO

```

2. Configuring API Keys with dotenv

Environment variables are used to store sensitive information like API keys securely.

How dotenv Works:

Place your keys in a .env file:

The load_dotenv() function reads these values and makes them accessible via os.getenv.

VisionAssist.py | Langchain.ipynb

C: > Users > shind > Python_Coding > API > VisionAssist.py > ...

```python
# Loading the environment variables
load_dotenv()

# Loading API key
with open(r"C:\Users\shind\Python_Coding\API\keys\API_key.txt") as f:
    api_key = f.read()

genai.configure(api_key=api_key)
```

INNOMATICS RESEARCH LABS

Environment Setup

3. Pre-trained Model Initialization

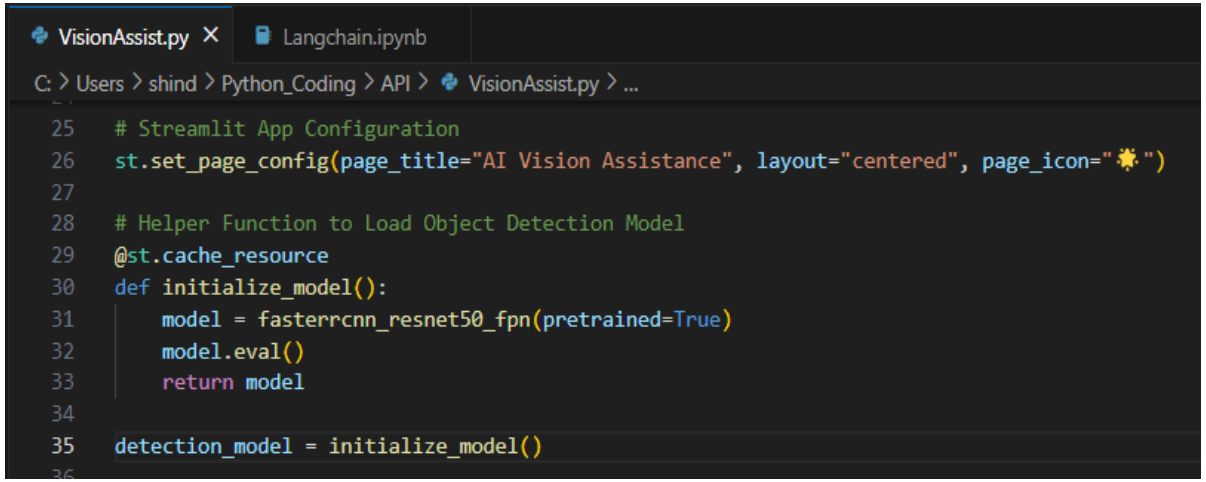
fasterrcnn_resnet50_fpn is a pre-trained object detection model available in PyTorch's torchvision.

Model Details:

Faster R-CNN: A robust object detection algorithm.

ResNet-50 Backbone: Used for feature extraction.

Pretrained=True: Loads a model trained on the COCO dataset.



4. Installation Commands

To install the required libraries, use the following commands in your terminal or command prompt:

```
pip install streamlit
pip install torch torchvision
pip install pytesseract
sudo apt install tesseract-ocr  # For Linux systems
brew install tesseract          # For macOS
pip install opencv-python
pip install Pillow
pip install pyttsx3
pip install python-dotenv
pip install google-generativeai
```

INNOMATICS RESEARCH LABS

Environment Setup

5. Setup Tips

Tesseract Path Configuration (Windows):

If Tesseract is not in your PATH, specify its location in your code:

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

API Key Placement:

Store sensitive keys like API keys in a .env file to avoid hardcoding them into the script.

By setting up these libraries and pre-trained models, the project integrates key functionalities like object detection, text recognition, and AI-driven scene description efficiently.

Core Functionalities

1. Text Extraction

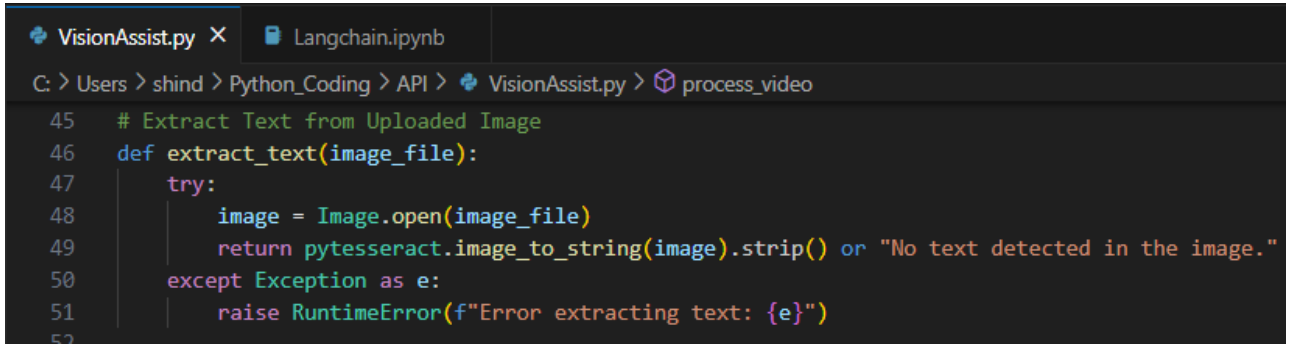
Purpose: Extracts text from an image and narrates it.

Functionality:

Opens an uploaded image and uses pytesseract to extract text.
Handles errors gracefully, providing user feedback when text extraction fails.

How it Works:

Uses pytesseract to process the uploaded image and extract text.
Text is displayed on the interface and narrated using pyttsx3.



2. Object Detection

Purpose: Identifies objects in the image, highlights them, and narrates key details.

Functionality:

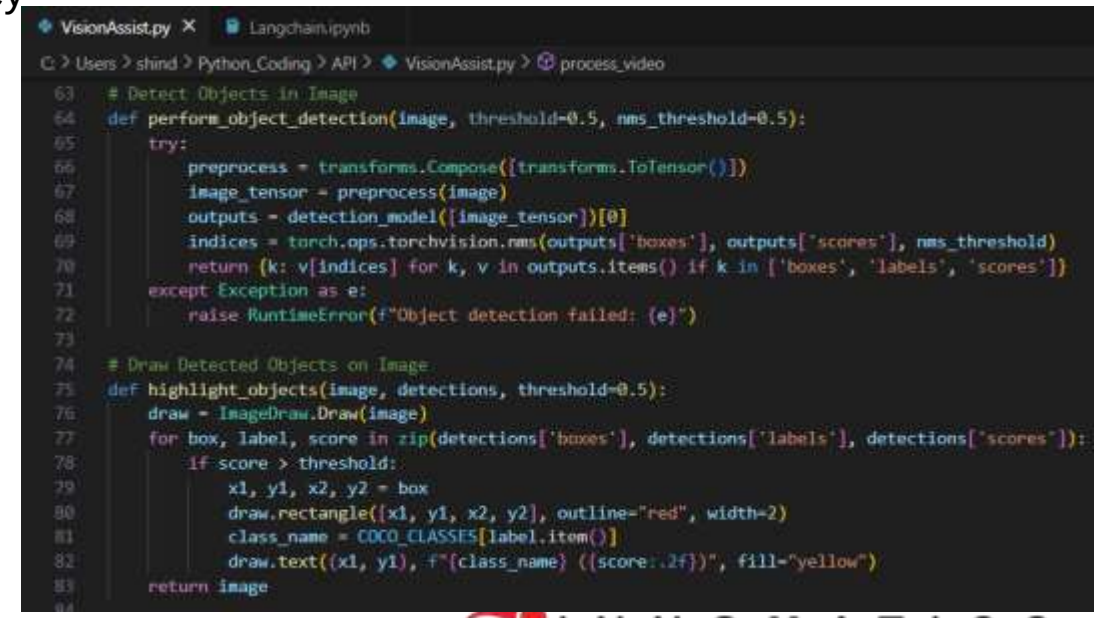
Converts an image to a tensor for processing.
Performs object detection using a Faster R-CNN model.
Applies non-maximum suppression (NMS) to eliminate overlapping detections.

How it Works:

Converts the image into a tensor using torchvision.transforms.
Runs it through a pre-trained Faster R-CNN model (fasterrcnn_resnet50_fpn) for object detection.
Uses non-maximum suppression to filter overlapping bounding boxes.

Output Highlighting:

Highlights objects on the image with bounding boxes and labels.



INNOMATICS RESEARCH LABS

Core Functionalities

3. Scene Description

Purpose: Provides an AI-generated description of the environment and objects in an image.

Functionality:

Sends an image to a generative AI model for scene analysis.

Provides descriptive text highlighting objects, people, and activities.

How it Works:

Uses the Gemini model from Google Generative AI to generate descriptions.

Designed to be concise, focusing on essential elements like objects, people, and actions.

VisionAssist.py ● Langchain.ipynb

C: > Users > shind > Python_Coding > API > VisionAssist.py > describe_scene

```python
def describe_scene(image_path):
    """Describe the scene in the given image using an AI model."""
    try:
        # Alternatively, using PIL to open the image and convert it to bytes
        img = Image.open(image_path)

        prompt = (
                "Analyze the uploaded image and describe the scene and object in clear and"
                "simple language in very short to assist visually impaired users like"
                "crossing roads and detecting objects. Answer must include key"
                "details with highlights about the environment, objects, people, and"
                "actions present in the image."
            )
        # Assuming genai is pre-configured with your API key
        response = genai.GenerativeModel("gemini-1.5-pro").generate_content([prompt, img]).text
        return response.strip()
    except Exception as e:
        return f"Error describing scene: {e}"
```

4. Text-to-Speech Conversion

Purpose: Converts generated text into speech for auditory feedback.

Functionality:

Uses pyttsx3 to convert text into speech.

Maintains session state to ensure efficient audio playback.

How it Works:

Initializes a text-to-speech engine using pyttsx3.

Plays audio feedback of the generated descriptions or detected text.

VisionAssist.py ● Langchain.ipynb

C: > Users > shind > Python_Coding > API > VisionAssist.py > describe_scene

```python
# Convert Text to Speech
def speak_text(text):
    try:
        if "audio_engine" not in st.session_state:
            st.session_state.audio_engine = pyttsx3.init()
        st.session_state.audio_engine.say(text)
        st.session_state.audio_engine.runAndWait()
    except Exception as e:
        raise RuntimeError(f"Text-to-speech conversion failed: {e}")
```

INNOMATICS RESEARCH LABS

Core Functionalities

5. Real-Time Video Processing

Purpose: Captures live video, extracts frames, and processes them for object detection and scene description.

Functionality:

Captures video from the webcam, processes frames, and describes each frame's content.

Stops video capture after a set time to ensure timely output.

How it Works:

Captures frames using OpenCV.

Saves each frame and processes it for scene descriptions and object detection.

Narrates the scene description for user assistance.

VisionAssist.py ● Langchain.ipynb

C: > Users > shind > Python_Coding > API > VisionAssist.py > describe_scene

```python
def process_video():
    """Capture video, save frames, and describe scenes."""
    cap = cv2.VideoCapture(0)  # 0 for default webcam
    frame_count = 0
    start_time = time.time()  # Start timer when capture begins
    pause_time = 15  # Time (in seconds) to capture video before pausing

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Calculate elapsed time
        elapsed_time = time.time() - start_time

        # Save the current frame
        frame_path = save_frame(frame, frame_count)

        # Describe the scene (always process each frame)
        scene_description = describe_scene(frame_path)

        # Display the processed frame and description in Streamlit
        st.image(frame, channels="BGR", caption=f"Frame {frame_count}")
        st.write(f"Frame {frame_count} Description: {scene_description}")
        speak_text(scene_description)

        frame_count += 1

        # Break after pausing for the specified time
        if elapsed_time >= pause_time:  # Pause after 15 seconds
            st.write("Video capture paused after 15 seconds.")
            break  # Exit the loop after 15 seconds of capture

    cap.release()
    st.success("Video processing completed. All frames saved.")
```

INNOMATICS RESEARCH LABS

User Interaction

User Interaction

This section outlines how users interact with the application. Users can upload images or capture videos, and the app processes the input to return results in various formats, such as annotated images, extracted text, or synthesized speech.

User Interaction Flow

Image Upload: Users upload an image through the app interface.
Video Capture: Users can initiate video capture using their webcam.
Processing: The app performs tasks such as object detection, text recognition, or scene description.

VisionAssist.py ● Langchain.ipynb

C: > Users > shind > Python_Coding > API > VisionAssist.py > ...

```
166  # Streamlit interface
167  st.title("Real Time Video Assist for Visually Impaired")
168  if st.button("Start Video Capture"):
169      process_video()
170
171
172  # Sidebar with Buttons
173  st.sidebar.title("Actions to Perform")
174  text_button = st.sidebar.button("Describe Image 📜")
175  scene_button = st.sidebar.button("Read Text from Image 🖉")
176  detect_button = st.sidebar.button("Detect Objects ⋮")
177  assist_button = st.sidebar.button("Personal Assist")
178  audio_button = st.sidebar.button("Stop Audio 🔕")
179
180  # Center Layout for Upload Section
181  st.title("Image Assistance for Visually Impaired")
182  uploaded_images = st.file_uploader("Upload multiple images:", type=["jpg", "jpeg", "png"], accept_multiple_files=True)
```

INNOMATICS RESEARCH LABS

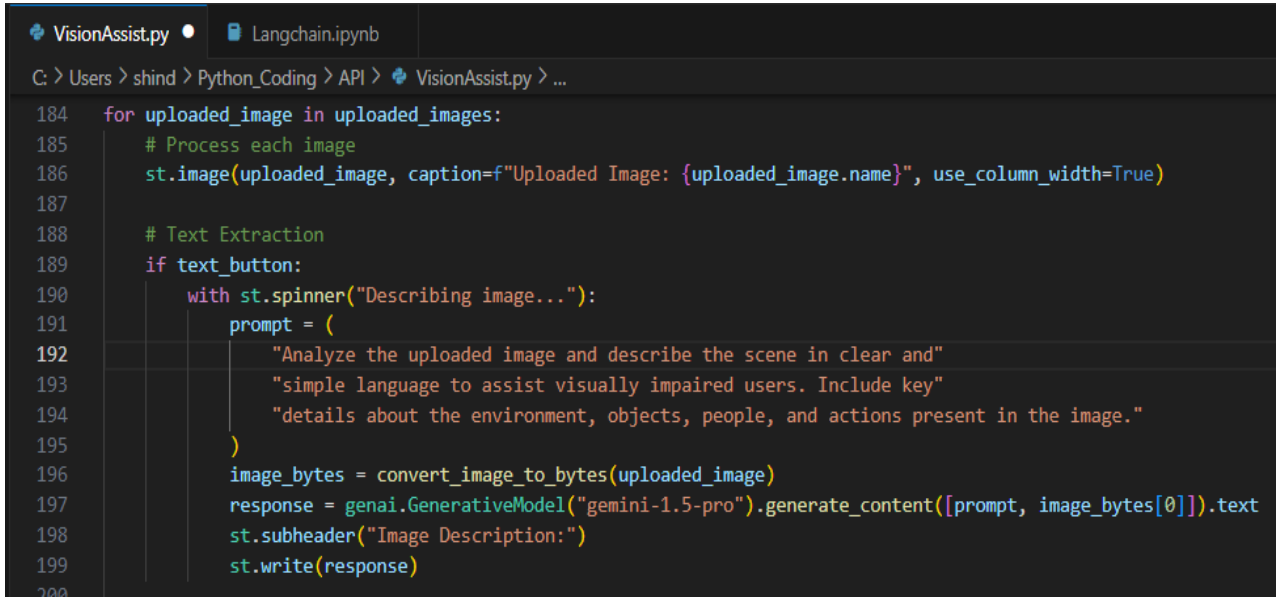
Output Delivery

1. Text Extraction (if text_button)
This section handles the generation of a textual description of the uploaded image.

Functionality:
Prompts the Generative AI model to analyze the uploaded image and provide a clear description of the scene for visually impaired users.
Focuses on generating text that highlights the environment, objects, people, and actions within the image.

Process:
Converts the uploaded image into bytes for compatibility with the AI model.
Passes a descriptive prompt along with the image to the generative model.
Displays the AI-generated description to the user.

Output:
A written description of the image scene.



INNOMATICS RESEARCH LABS

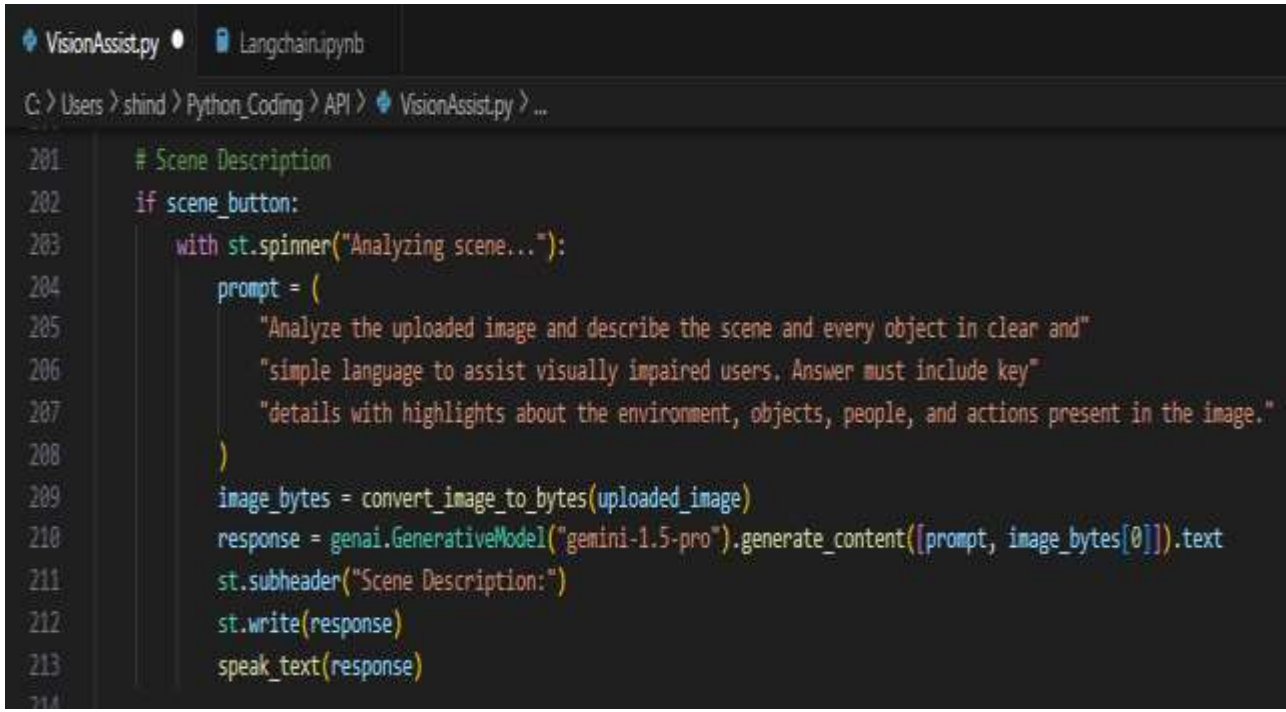
Output Delivery

2. Scene Description (if scene_button)
This section is similar to text extraction but focuses more on detailed scene analysis.

Functionality:
Provides an in-depth explanation of the scene, identifying specific objects, people, and their interactions.
Highlights key environmental details in clear language.

Process:
Converts the uploaded image into bytes.
Sends a scene-specific prompt to the AI model.
Displays the description and optionally speaks the text for the user.

Output:
A detailed scene description emphasizing all objects and environmental features.



INNOMATICS RESEARCH LABS

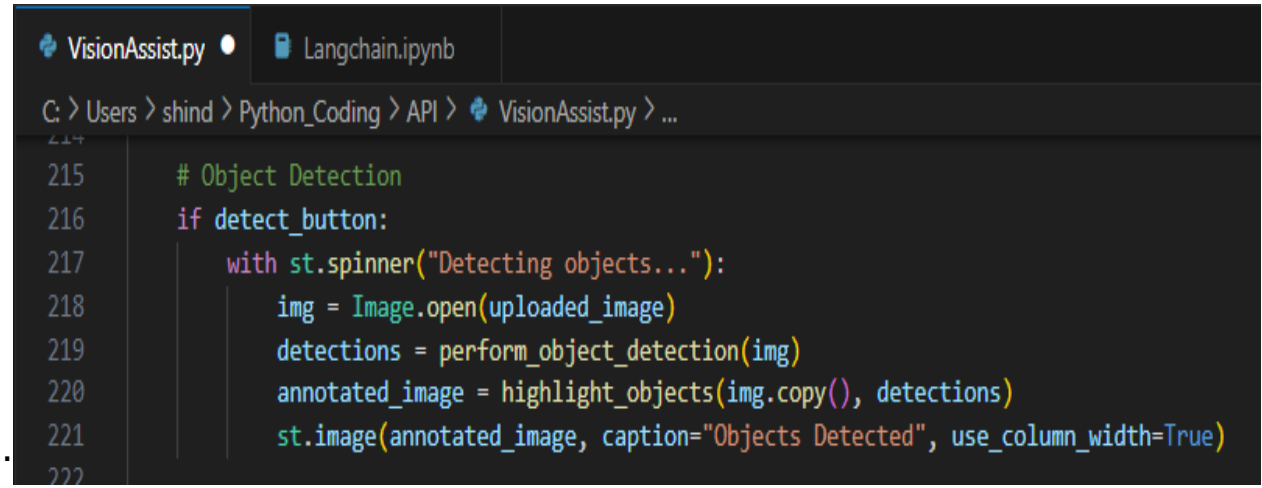
Output Delivery

3. Object Detection (if detect_button)
This section performs object detection on the uploaded image.

Functionality:
Detects objects in the image and annotates it with bounding boxes and labels.
Enhances the image to provide visual feedback on identified objects.

Process:
Loads the uploaded image.
Uses a function (perform_object_detection) to detect objects.
Annotates the image using a helper function (highlight_objects).
Displays the annotated image in the app.

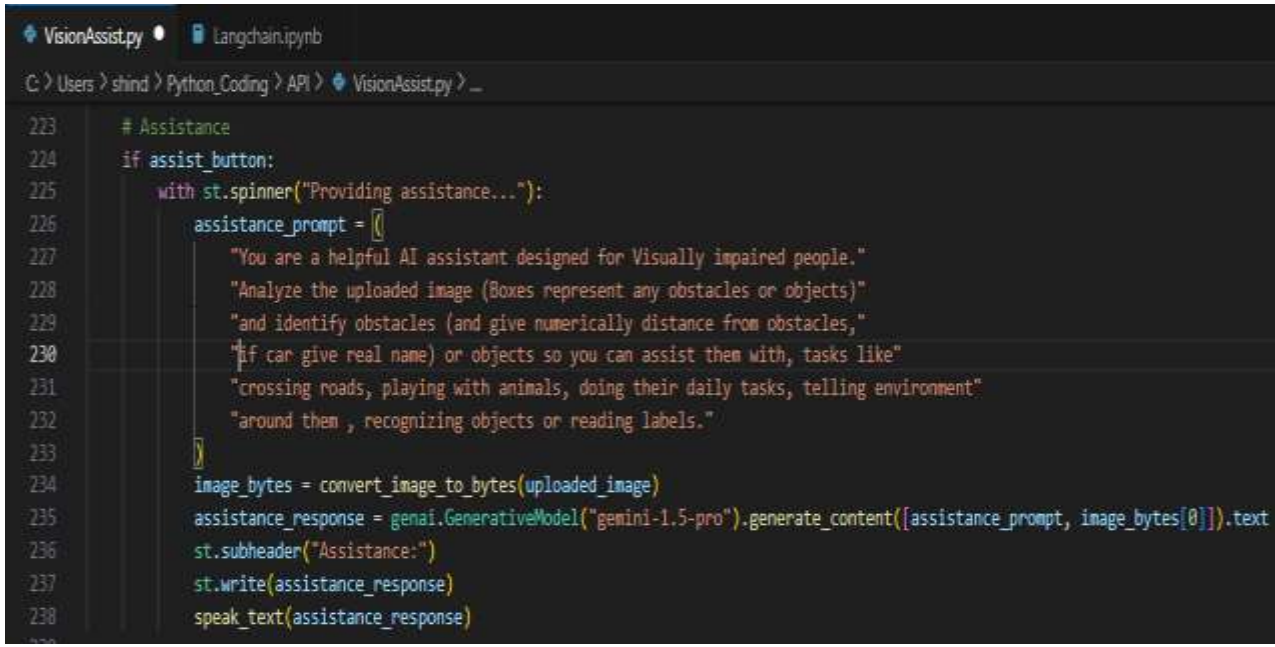
Output:
An annotated image showing detected objects and their locations.



INNOMATICS RESEARCH LABS

Output Delivery

4. Assistance (if assist_button)
This section provides tailored assistance for visually impaired users based on the uploaded image.



Functionality:
Acts as a virtual assistant to analyze the image and provide useful insights.
Focuses on helping users with real-world tasks, such as identifying obstacles, recognizing objects, or describing the surrounding environment.

Process:
Converts the uploaded image into bytes.
Sends a detailed assistance prompt to the generative model.
Displays the AI-generated response and optionally speaks it for the user.

Output:
A helpful description or analysis aimed at assisting visually impaired users.

INNOMATICS RESEARCH LABS

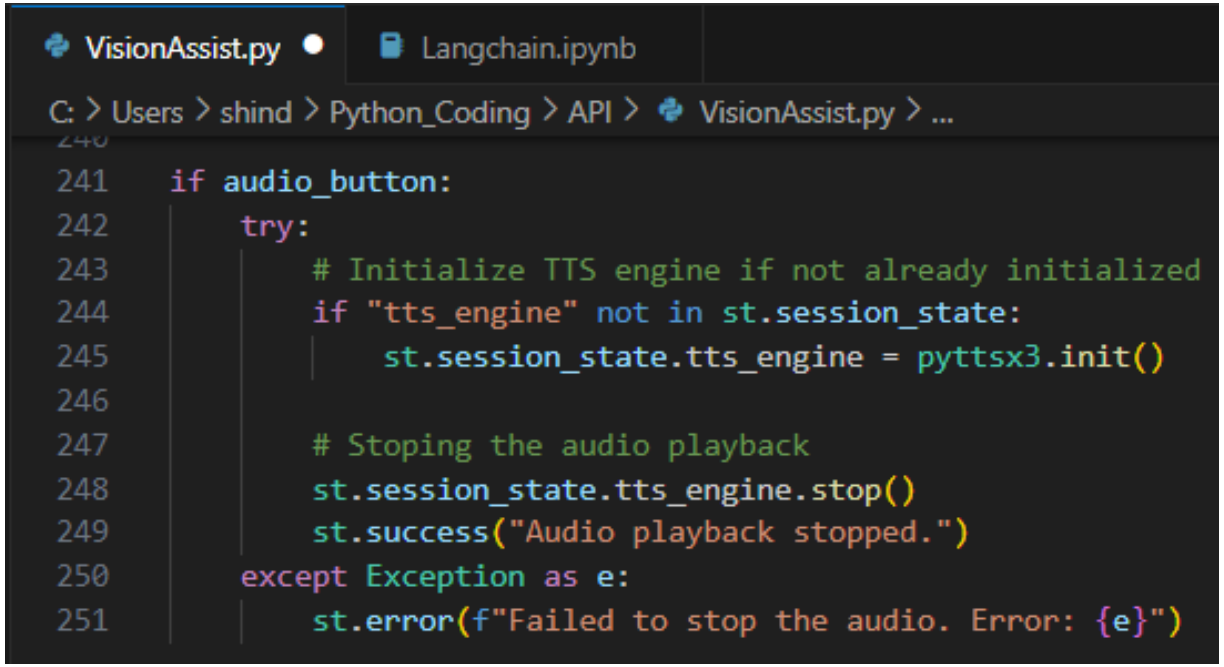
Output Delivery

5. Audio Playback Control (if audio_button)
This section handles stopping text-to-speech (TTS) playback.

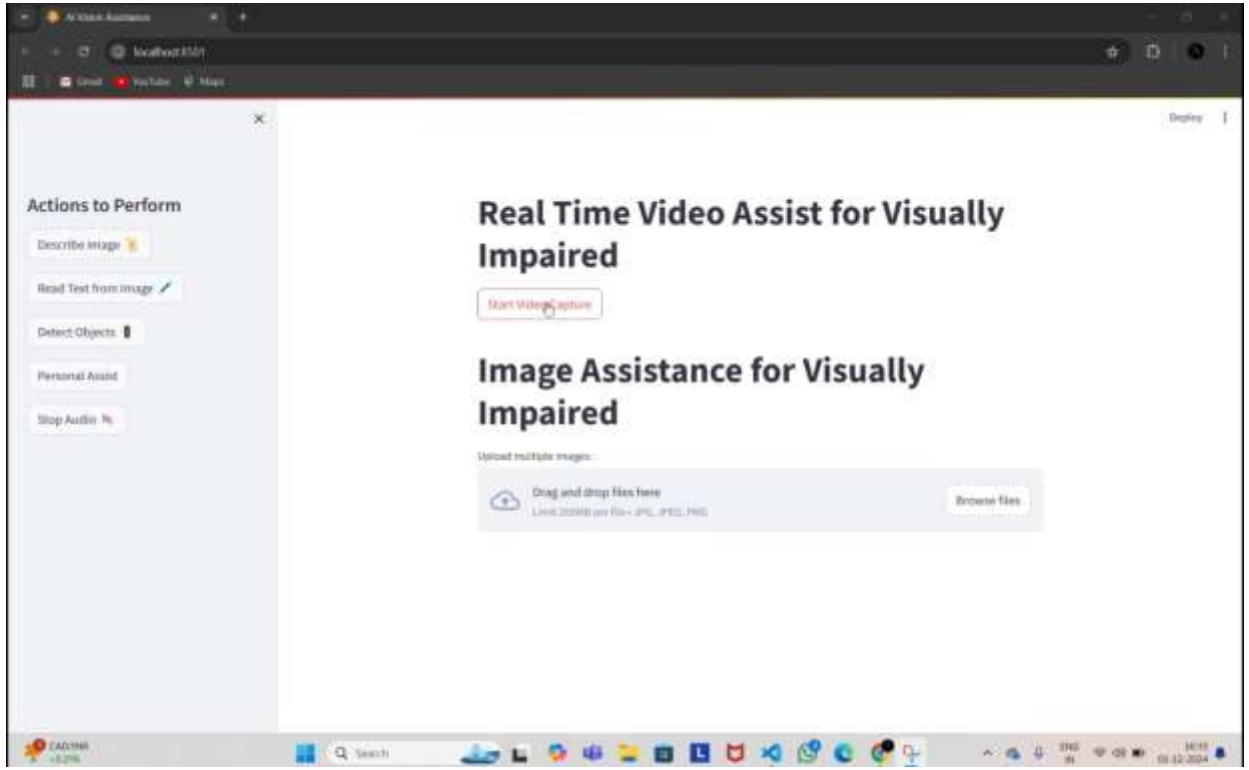
Functionality:
Allows users to stop the playback of any previously generated audio.

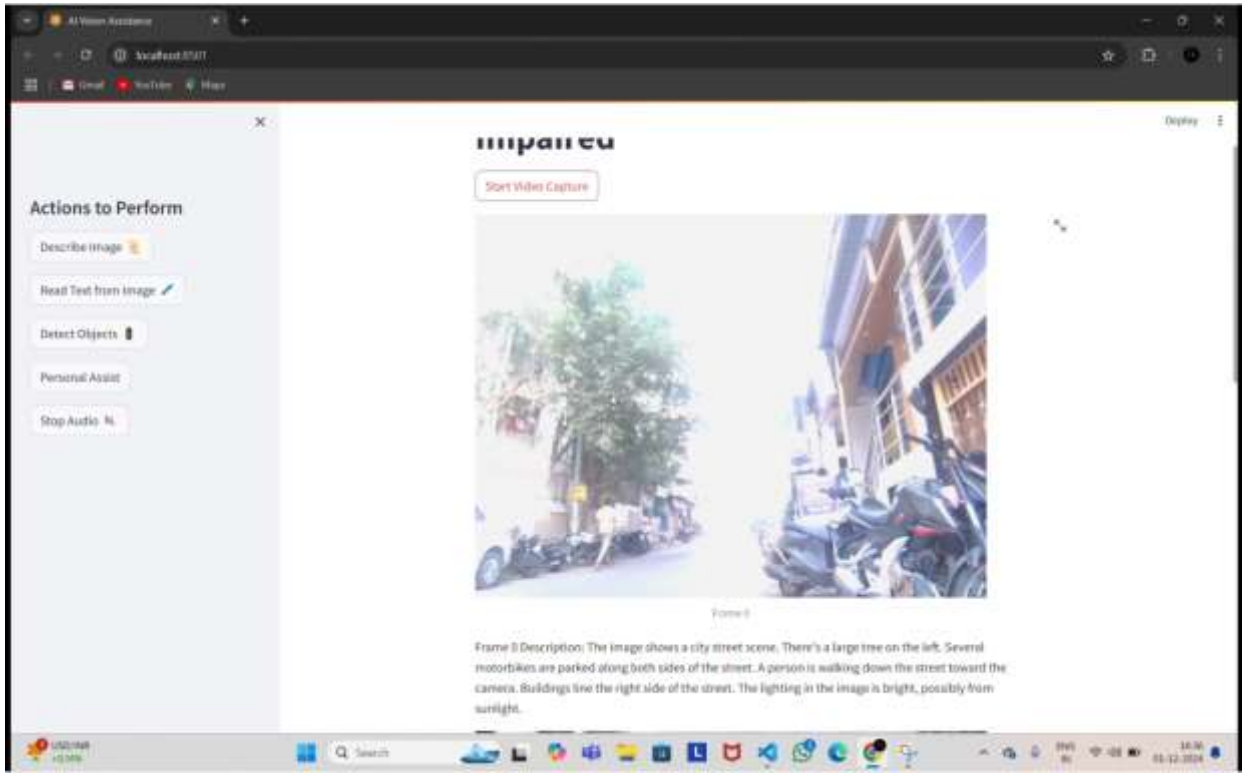
Process:
Initializes a text-to-speech (TTS) engine if it hasn't been initialized yet.
Stops any ongoing audio playback using the TTS engine.
Displays a success or error message depending on the outcome.

Output:
Stops any active audio playback and informs the user.



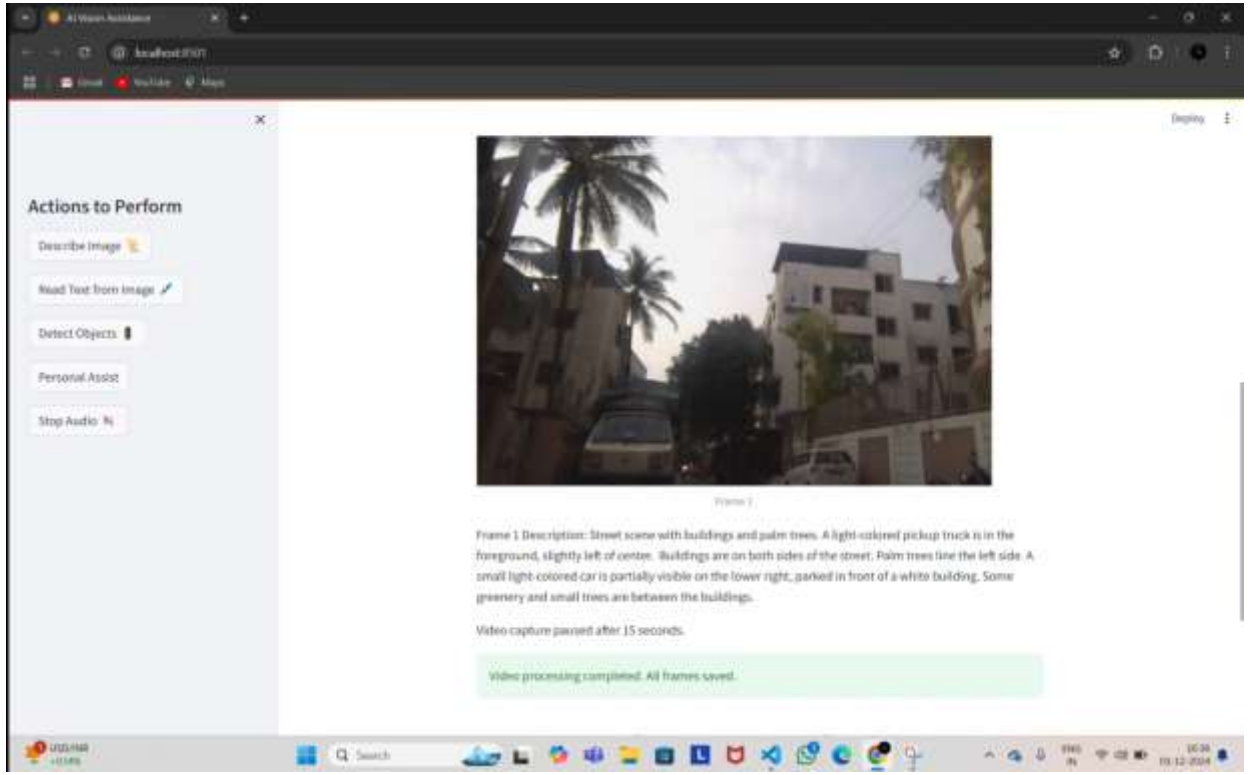
Prototype Testing Results

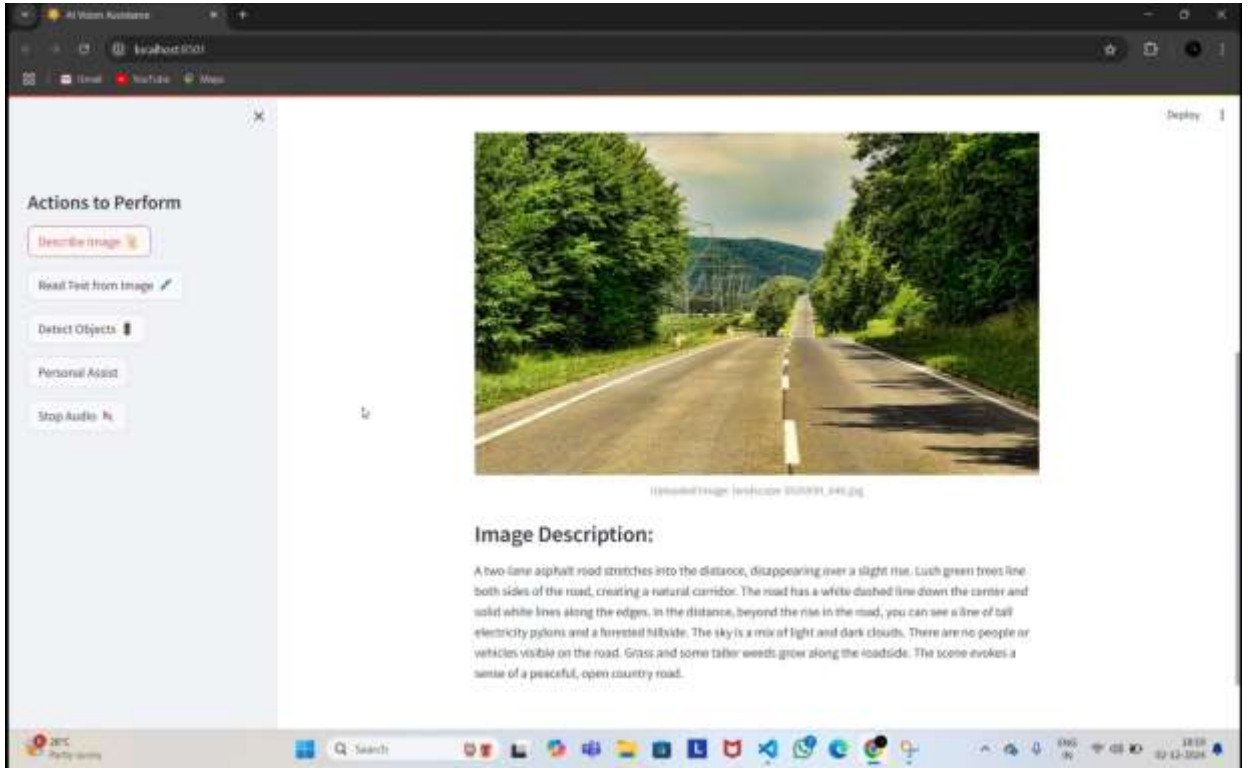




INNOMATICS RESEARCH LABS

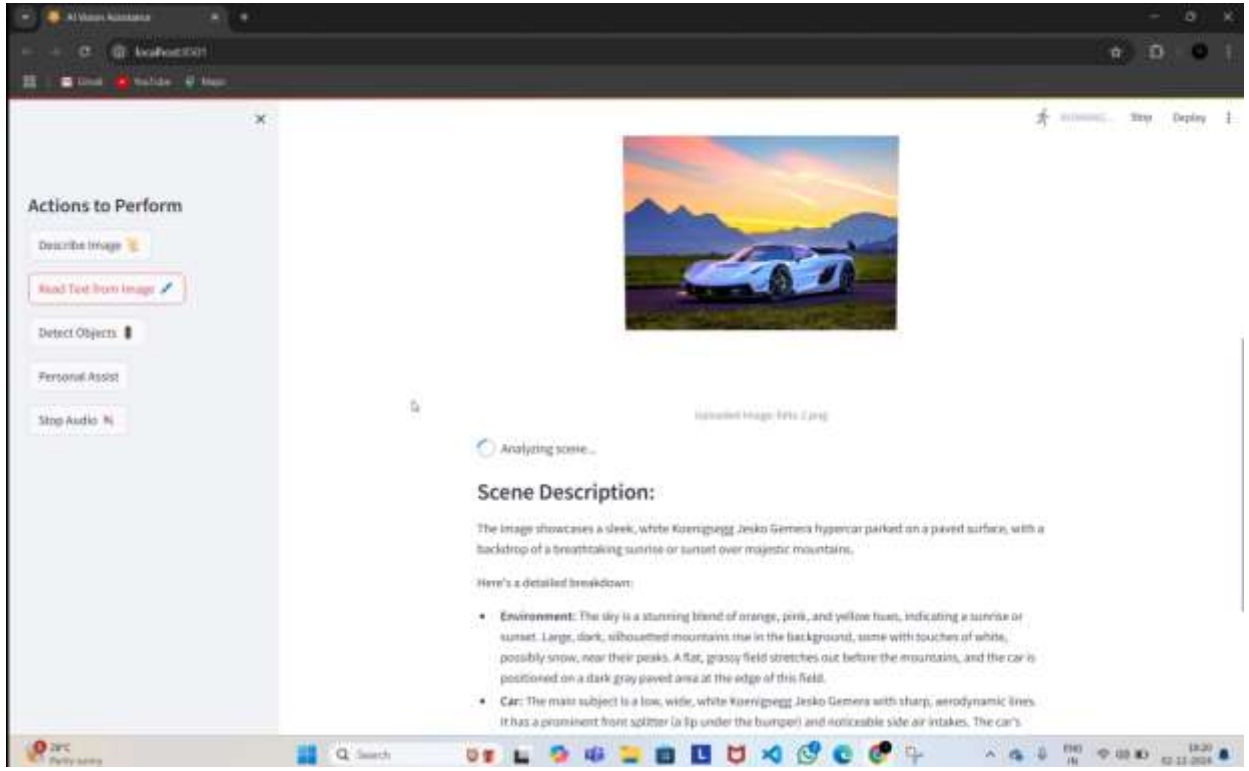
Prototype Testing Results

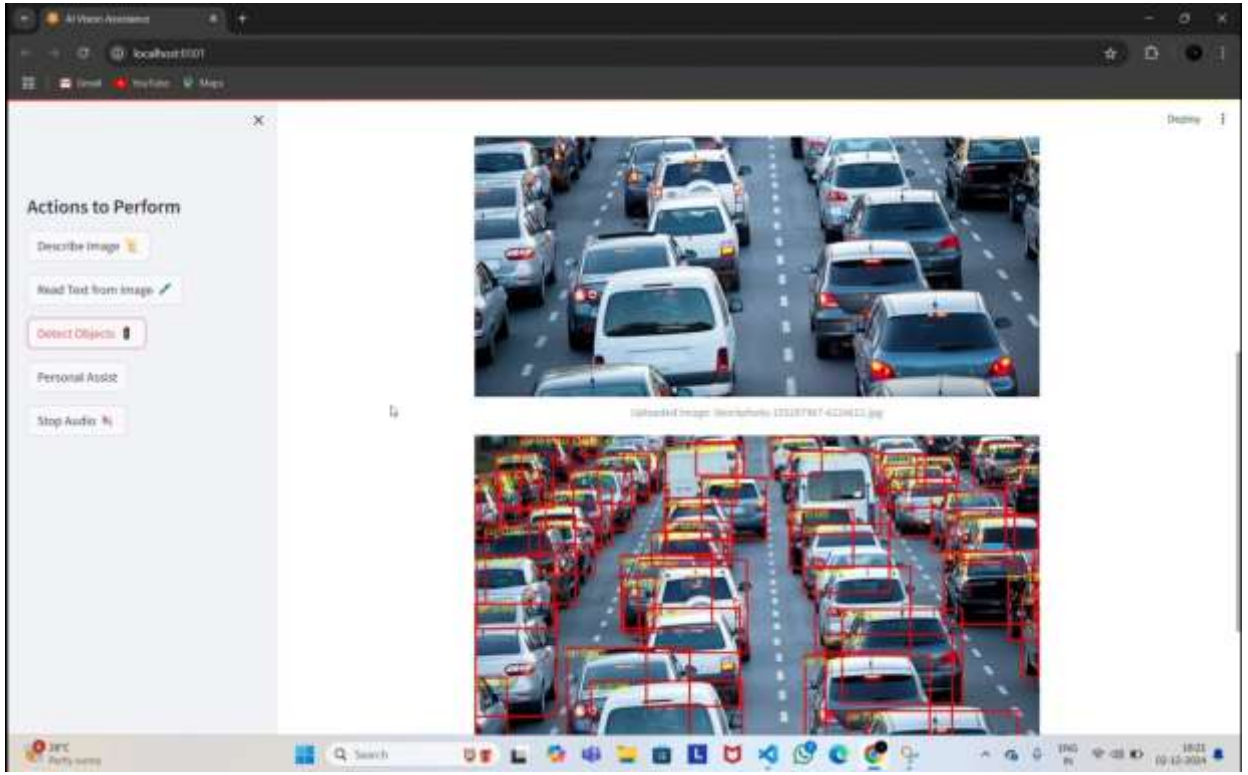




INNOMATICS RESEARCH LABS

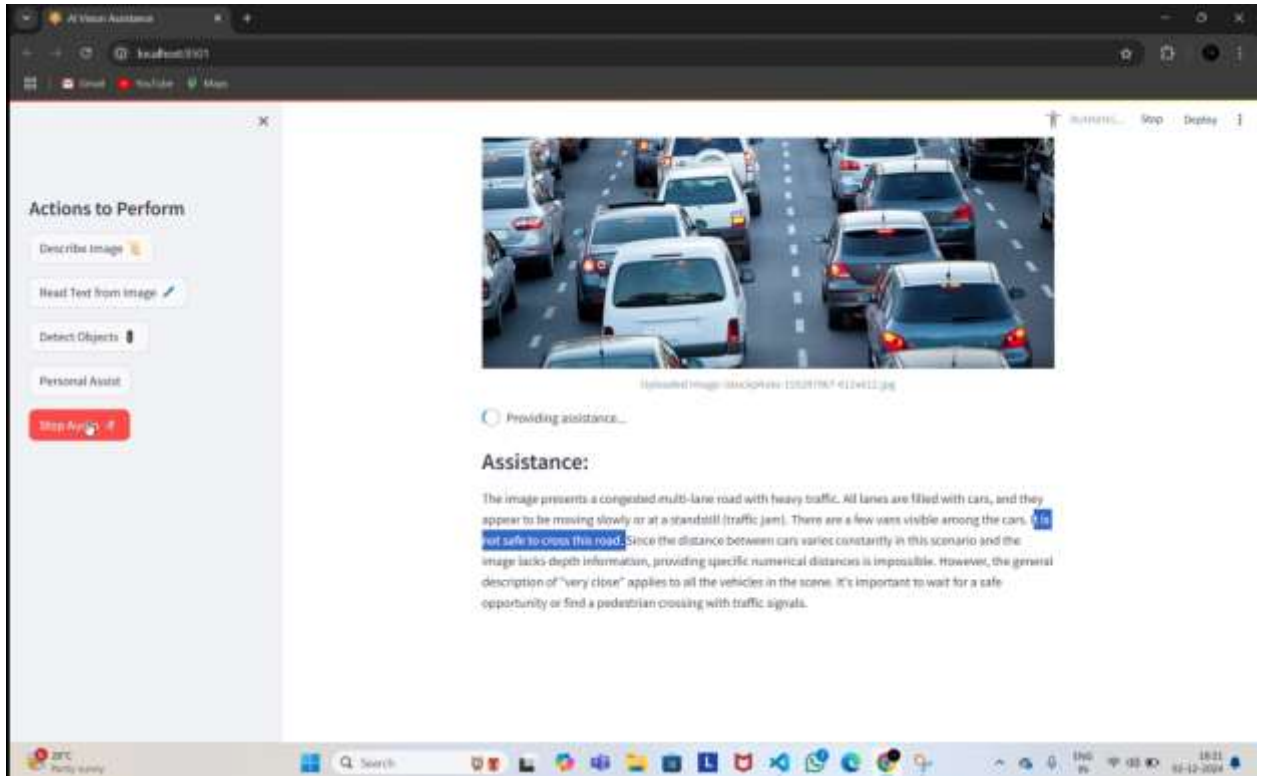
Prototype Testing Results

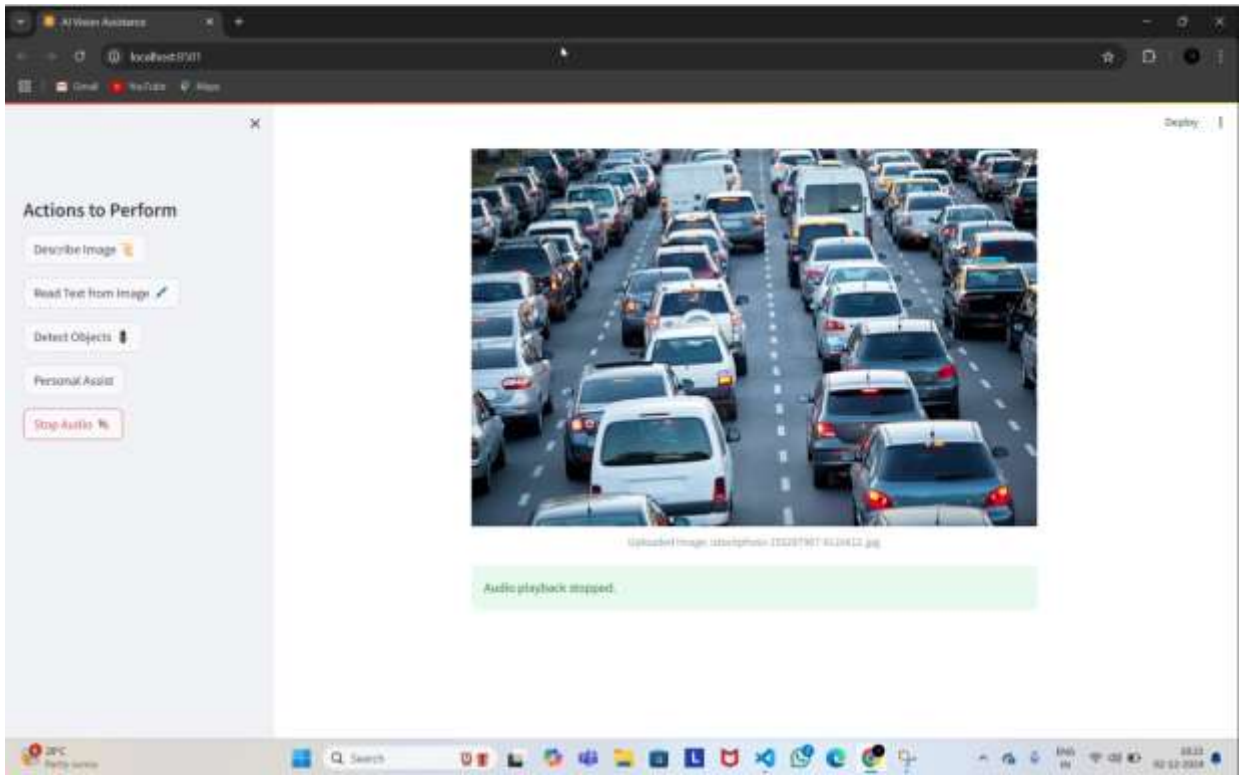




INNOMATICS RESEARCH LABS

Prototype Testing Results





INNOMATICS RESEARCH LABS

Final Conclusion

This application integrates advanced computer vision, AI, and TTS technologies to offer a comprehensive vision assistance tool. Each functionality enhances the independence of visually impaired users by enabling them to:

1. Read text in images.
2. Identify and locate objects in real-time.
3. Understand and interact with their environment effectively.

Future improvements can include expanding object detection capabilities, enhancing scene descriptions with depth information, and integrating multilingual support. Additionally, leveraging edge computing could improve response times and allow offline functionality, ensuring uninterrupted usability. Incorporating gesture recognition and voice-command interfaces could further simplify user interaction, providing a seamless experience.

Experience :

1. Knowledge of Core Libraries

I explored various powerful libraries:

Image Processing:

Learned to use OpenCV or Pillow for tasks like image resizing, filtering, and feature extraction.

Gained expertise in image enhancement techniques (e.g., denoising, edge detection).

Machine Learning:

Worked with TensorFlow, PyTorch, or scikit-learn for training or deploying models.

Understood model pipelines, including loading pre-trained models.

OCR (Optical Character Recognition):

Used libraries like Tesseract or EasyOCR to extract text from images.

Improved understanding of handling text recognition in noisy or low-light images.

2. Working with APIs and Pre-trained Models

Integrated APIs like Google Vision API.

Understood the importance of transfer learning for tasks like object detection, text recognition, or scene analysis.

Learned to balance between using lightweight models for speed and heavier models for accuracy.

3. Real-time Processing

Built pipelines for real-time image capture using camera feeds and webcams.

Mastered threading or asynchronous techniques to manage the processing speed of vision tasks.

4. Data Handling and Annotation

Gained insights into preparing datasets for training and testing:

Labeling images for object detection or classification tasks.

Understanding data augmentation techniques to improve model generalization.

5. Debugging and Performance Tuning

Handled issues with model inference speed, memory usage, and accuracy trade-offs.

Debugged cases where the system failed in recognizing or detecting objects accurately.

Overall Takeaway

This project taught you not just technical skills but also problem-solving, adaptability, and innovation. It's a journey that highlights the blend of software engineering, AI, and user-focused design, making you more proficient in cutting-edge technologies and their practical applications.

THANK YOU

INNOMATICS RESEARCH LABS