```python
# Importing Required Libraries
import streamlit as st
from transformers import BlipProcessor, BlipForConditionalGeneration
from pytesseract import image_to_string
from PIL import Image
import pyttsx3
import cv2
import numpy as np

# Function for Real-Time Scene Understanding
def describe_image(image_path):
    processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
    model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
    inputs = processor(image_path, return_tensors="pt")
    outputs = model.generate(**inputs)
    return processor.decode(outputs[0], skip_special_tokens=True)

# Function to Extract Text from an Image using OCR
def extract_text(image_path):
    image = Image.open(image_path)
    return image_to_string(image)

# Function to Convert Text to Speech
def text_to_speech(text):
    engine = pyttsx3.init()
    engine.say(text)
    engine.runAndWait()

# Function for Object and Obstacle Detection
def detect_objects(image_path):
    # Load pre-trained model and configuration files (e.g., YOLO)
```

```
net = cv2.dnn.readNet("yolov4.weights", "yolov4.cfg")
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# Load COCO dataset class names
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]

# Read the image
img = cv2.imread(image_path)
height, width, _ = img.shape

# Preprocess the image for YOLO
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outputs = net.forward(output_layers)

# Store detection results
boxes = []
confidences = []
class_ids = []

for output in outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            # Object detected
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
```

```
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    # Apply Non-Maximum Suppression to filter overlapping boxes
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    detected_objects = []
    if len(indexes) > 0:
        for i in indexes.flatten():
            label = str(classes[class_ids[i]])
            detected_objects.append(label)
            x, y, w, h = boxes[i]
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(img, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

    # Save the output image
    output_path = "output.jpg"
    cv2.imwrite(output_path, img)
    return detected_objects, output_path

# Function for Personalized Assistance
def personalized_assistance(image_path):
    text = extract_text(image_path)
    if "milk" in text.lower():
        return "It seems like you uploaded a label for milk. Make sure it's stored in a refrigerator."
    elif "medicine" in text.lower():
        return "This is a medicine label. Please check the expiry date and dosage instructions."
    else:
        return "No specific assistance detected. Please try uploading another image."

# Streamlit App
def main():
    # Title and Description
    st.title("AI-Powered Assistance for Visually Impaired Individuals")
    st.text("Upload an image to get assistance.")

    # Upload an Image
    uploaded_file = st.file_uploader("Choose an image", type=["jpg", "png", "jpeg"])

    if uploaded_file:
        # Display Uploaded Image
        st.image(uploaded_file, caption="Uploaded Image", use_column_width=True)

        # Scene Understanding
        if st.button("Describe Scene"):
            st.write("Analyzing the image...")
            description = describe_image(uploaded_file)
            st.write(f"Scene Description: {description}")

        # Text-to-Speech
        if st.button("Read Text"):
            st.write("Extracting text from the image...")
            extracted_text = extract_text(uploaded_file)
            st.write(f"Extracted Text: {extracted_text}")
            st.write("Converting text to speech...")
            text_to_speech(extracted_text)

        # Object and Obstacle Detection
        if st.button("Detect Objects and Obstacles"):
            st.write("Detecting objects in the image...")
            detected_objects, output_path = detect_objects(uploaded_file.name)
            st.image(output_path, caption="Objects Detected", use_column_width=True)
            st.write(f"Detected Objects: {', '.join(detected_objects)}")

        # Personalized Assistance
        if st.button("Personalized Assistance"):
            st.write("Analyzing for personalized assistance...")
            assistance = personalized_assistance(uploaded_file)
            st.write(f"Assistance: {assistance}")

# Run the App
if __name__ == "__main__":
    main()
```