//Python code

```
pip install torch transformers pandas gradio textblob pytesseract pillow speechrecognition matplotlib seaborn
import os
import re
import pandas as pd
import numpy as np
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from sklearn.metrics import accuracy_score
from textblob import TextBlob
import seaborn as sns
import matplotlib.pyplot as plt
import gradio as gr
import pytesseract
from PIL import Image
import speech_recognition as sr
from datetime import datetime
# Load a pre-trained transformer model and tokenizer (DistilBERT fine-tuned for multi-label classification)
MODEL_NAME = "unitary/toxic-bert"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME)
# Define toxicity categories
categories = ["Toxic", "Severe Toxic", "Obscene", "Threat", "Insult", "Identity Hate"]
```

```
# Slang and abbreviation dictionary
slang_dict = {
    "lol": "laughing out loud",
    "brb": "be right back",
    "gtg": "got to go",
    "ttyl": "talk to you later",
    "idk": "I don't know",
    "omg": "oh my god",
    "btw": "by the way"
}
# Feedback storage
feedback_data = []
# Function to preprocess text
def preprocess_text(text):
    # Expand slang and abbreviations
    for slang, expansion in slang_dict.items():
        text = text.replace(slang, expansion)

    # Spelling correction
    corrected_text = str(TextBlob(text).correct())

    # Remove URLs, mentions, and hashtags
    corrected_text = re.sub(r'http\S+', '<URL>', corrected_text)
    corrected_text = re.sub(r'@\w+', '<MENTION>', corrected_text)
    corrected_text = re.sub(r'#\w+', '<HASHTAG>', corrected_text)

    # Sentiment analysis with TextBlob
    sentiment = TextBlob(corrected_text).sentiment.polarity
    if sentiment > 0.1:
        sentiment_label = "Positive"
    elif sentiment < -0.1:
        sentiment_label = "Negative"
```

```
    else:
        sentiment_label = "Neutral"

    return corrected_text, sentiment_label
    # Function to transcribe audio input to text
    def transcribe_audio(audio_file):
        recognizer = sr.Recognizer()
        try:
            with sr.AudioFile(audio_file) as source:
                audio = recognizer.record(source)
            return recognizer.recognize_google(audio)
        except sr.UnknownValueError:
            return "Sorry, I could not understand the audio."
        except sr.RequestError:
            return "Could not request results from Google Speech Recognition service."
        except Exception as e:
            return f"Error processing audio: {str(e)}"
# Function to extract text from an image
def extract_text_from_image(image_file):
    try:
        img = Image.open(image_file)
        text = pytesseract.image_to_string(img)
        return text
    except Exception as e:
        return f"Error processing image: {str(e)}"
# Function to make predictions with explanations
def advanced_toxicity_predictor(text, audio_file, image_file, feedback, toxicity_threshold, sentiment_threshold):
    global feedback_data
    if not text.strip() and audio_file is None and image_file is None:
        return "Error: Please provide input text, upload an audio file, or an image.", "<div style='color:red;font-weight:bold;'>No input provided</div>"
```

```
#    Process    input
processed_text = ""
if text.strip():
   processed_text   =   text
elif audio_file is not None:
   processed_text = transcribe_audio(audio_file)
elif image_file is not None:
   processed_text = extract_text_from_image(image_file)

# Preprocess the text
processed_text, sentiment = preprocess_text(processed_text)

# Tokenize and prepare inputs for model
inputs = tokenizer(processed_text, return_tensors="pt", truncation=True, padding=True)
outputs = model(**inputs)
predictions = torch.sigmoid(outputs.logits).detach().numpy()[0]

# Interpret predictions with thresholds
toxic_levels = []
for idx, score in enumerate(predictions):
   if score >= toxicity_threshold:
      toxic_levels.append((categories[idx], score))

# Determine overall toxicity level and color
if toxic_levels:
   overall_level = " & ".join([f"{category} ({score:.2f})" for category, score in toxic_levels])
   color = "red"  # Show red if any category is flagged as toxic
else:
   overall_level = "Low Toxicity"
   color = "green"
```

```
# Final assessment based on sentiment
if sentiment == "Neutral" and overall_level == "Low
    Toxicity":overall_level = "Neutral"
    color = "blue"

# Log the timestamp of the comment
timestamp = datetime.now().strftime("%Y-%m-
%d %H:%M:%S")
feedback_data.append((processed_text,                feedback,
timestamp))

#                Create
probability        bar
chart
plt.figure(figsize=(
10, 5))
plt.bar(categories, predictions, color=['red', 'orange', 'yellow', 'blue', 'green', 'purple'])
plt.axhline(y=toxicity_threshold, color='gray', linestyle='--', label='Toxicity Threshold')
plt.xlabel("Toxicity Categories")
plt.ylabel("Probabilities")
plt.title("Probability Distribution of Toxicity
Categories")plt.xticks(rotation=45)
plt.ylim(0, 1) # Set y-axis limit
from        0        to        1plt.legend()
plt.tight_layout()
plt.savefig('toxicity_probabilitie
s.png')plt.close()

return (f"Toxicity Level: {overall_level}\nConfidence:
        {predictions}\n", f"<div style='color:{color};font-
        weight:bold;'>{overall_level}</div>",
```

```
                'toxicity_probabilities.png')

        # Function to display feedback dashboard
        def display_feedback_dashboard():
            if not feedback_data:
                return "No feedback received yet."

    feedback_df = pd.DataFrame(feedback_data, columns=["Comment", "Feedback", "Timestamp"])
    return feedback_df.to_html(index=False)

    # Confusion Matrix plot function
    def plot_confusion_matrix(y_true, y_pred, labels=categories):
        conf_matrix = multilabel_confusion_matrix(y_true, y_pred)
        fig, axes = plt.subplots(2, 3, figsize=(12, 8))
        axes = axes.ravel()
        for i, cm in enumerate(conf_matrix):
            sns.heatmap(cm, annot=True, fmt="d", ax=axes[i], cmap="Blues", cbar=False,
                        xticklabels=["Non-Toxic", "Toxic"], yticklabels=["Non-Toxic", "Toxic"])
            axes[i].set_title(f'{labels[i]} Confusion Matrix')
            axes[i].set_xlabel('Predicted Label')
            axes[i].set_ylabel('True Label')
        plt.tight_layout()
        plt.savefig("multi_conf_matrix.png")
        plt.close()
        return "multi_conf_matrix.png"

    # Gradio Interface
    def gradio_interface():
        with gr.Blocks() as interface:
            gr.Markdown("## ToxiComment")
            gr.Markdown("This tool classifies comment toxicity into Low, Medium, or High, with support for text,
audio, and image inputs. It also provides a feedback dashboard.")

            with gr.Row():
```

```
            with gr.Column():
                input_textbox = gr.Textbox(label="Comment (Text Input)", placeholder="Enter your comment
here...")
                audio_input = gr.Audio(label="Record Audio (optional)", type="filepath")
                image_input = gr.Image(label="Upload Image (optional)", type="filepath")
                feedback_textbox = gr.Textbox(label="Feedback (optional)", placeholder="Provide feedback here...")
                toxicity_threshold = gr.Slider(0.3, 0.7, value=0.5, step=0.05, label="Toxicity Sensitivity Threshold")
                sentiment_threshold = gr.Slider(0.1, 0.5, value=0.3, step=0.05, label="Sentiment Sensitivity
Threshold")
                submit_button = gr.Button("Submit")

            with gr.Column():
                prediction_output = gr.Textbox(label="Prediction", interactive=False)
                toxicity_level_output = gr.HTML(label="Visual Toxicity Level")
                probability_chart_output = gr.Image(type="filepath", label="Probability Chart", interactive=False)

        submit_button.click( advanced_
            toxicity_predictor,
            inputs=[input_textbox, audio_input, image_input, feedback_textbox, toxicity_threshold,
sentiment_threshold],
            outputs=[prediction_output, toxicity_level_output, probability_chart_output]
        )
        gr.Markdown("### Feedback Dashboard")
        feedback_dashboard_output = gr.HTML(label="Feedback Dashboard")
        feedback_dashboard = gr.Button("Show Feedback Dashboard")
        feedback_dashboard.click(
            display_feedback_dashboard,
            inputs=[],
            outputs=feedback_dashboard_output
        )
    # Launch the Gradio interface
    interface.launch()
# Run the interface
gradio_interface()
```

//WEB GUI

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title> Comment Toxicity Detection</title>
   <!-- font awesome cdn link  -->
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
   <!-- custom css file link -->
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <!-- header section starts -->
   <header class="header">
      <div id="menu-btn" class="fas fa-bars"></div>
      <a href="#" class="logo">ToxiComment</a>
      <nav class="navbar">
         <div id="close-navbar" class="fas fa-times"></div>
         <a href="#home">home</a>
         <a href="#about">about</a>
      </nav>
   </header>
   <!-- header section ends -->
   <!-- home section starts -->
   <section class="home" id="home">
      <div class="content">
```

```
        <h3><span>WELCOME</span></h3>
        <p>To ToxiComment-The comment toxicity predictor.</p>
        <a href=" http://127.0.0.1:7887" class="btn"> Predict</a>
    </div>
    <div class="video-container">
        <video src="images/home.mp4" id="video" loop autoplay muted></video>
    </div>
</section>

<!-- home section ends -->
<!-- about section starts -->
<section class="about" id="about">
    <h1 class="heading">about</h1>
    <div class="container">
        <div class="image-container">
            <img src="images/about-1.jpg" alt="" class="image">
            <div class="controls">
                <span class="control-btn" data-src="images/about-1.jpg"></span>
                <span class="control-btn" data-src="images/about-2.jpg"></span>
                <span class="control-btn" data-src="images/about-3.jpg"></span>
            </div>
        </div>
        <div class="content">
            <span></span>
            <h3></h3>
            <p>Negative online behaviors, like toxic comments, are likely to make people stop expressing themselves and leave a conversation. Platforms struggle to identify and flag potentially harmful or offensive
```

online comments, leading many communities to restrict or shut down user comments altogether. Kaggle issued a challenge to build a multi-label classification model that's able to detect different types of toxicity like threats, obscenity and insults, and thus help make online discussion more productive and respectful.The data for the problem is a dataset of 159,571 comments from Wikipedia's talk page edits. These comments have been flagged for toxic behaviour by human reviewers.

</p>

<ahref="https://medium.com/@alaeddine.grine/toxic-comment-classification-317628632336#:~:text=Negative%20online%20behaviors%2C%20like%20toxic,shut%20down%20user%20comments%20altogether." class="btn">

read more

</section>

<!-- about section ends -->

<!-- footer section starts -->

<section class="footer">

created by GT-13 | all rights reserved!

</section>

<!-- footer section ends -->

<!-- custom js file link -->

<script src="script.js"></script>

</body>

</html>