Comment Toxicity Detection

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted by

Varikutla Sai Manoj	**:**	**2211CS020527**
Sravya Vasa	**:**	**2211CS020528**
Veeramalla Sai Charan	**:**	**2211CS020530**
Vemula Sravani	**:**	**2211CS020532**
Vemuri Pranay	**:**	**2211CS020533**

Under the Guidance of

Prof.K.ManojSagar

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, "**Comment Toxicity Detection"** submitted by Varikutla Sai Manoj (2211CS020527), Vasa Sravya (2211CS020528), Veeramalla Sai Charan (2211CS020530), Vemula Sravani(2211CS020532), Vemuri Pranay Chowdary (2211CS020533) B. Tech III year I semester, Department of CSE (AI &ML) during the year 2024-2025. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Prof.K.Manoj Sagar **Dr.A.Sivaranjani**

DEAN CSE(AI&ML) **EXTERNAL EXAMINER**

Dr.Thayyaba Khatoon

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all the efforts with success. We are very grateful to our project mentor Prof.K.Manoj Sagar, for the guidance, inspiration and constructive suggestions that helped us in the development of this application.

We are much obliged to Prof.Shiva Kumar(Application Development Incharge) for encouraging and supporting us immensely by giving many useful inputs with respect to the topic chosen by us, throughout the development of the application ensuring that our project is a success.

We also express our heartfelt gratitude to Dr. Thayyaba Khatoon(Dean of the Department), for giving all of us such a wonderful opportunity to explore ourselves and the outside world to work on the real-life scenarios where the machine learning is being used nowadays.

We also thank our parents and family at large for their moral and financial support in funding the project to ensure successful completion of the project.

ABSTRACT

This project tackles the challenge of detecting toxic comments on online platforms through a deep learning-based approach. Toxic comments—including hate speech, threats, and profanities—pose a major risk to user experience and community well-being. Traditional moderation approaches often struggle with the complex, context-dependent nature of abusive language, which can vary widely across platforms and communities. This documentation describes a solution leveraging transformer-based natural language processing (NLP) models, specifically a fine-tuned DistilBERT model, to enhance toxicity detection capabilities. The model, trained on a large and diverse dataset covering multiple forms of toxicity, can detect both explicit and subtle toxic language. Supplementary techniques include text preprocessing, sentiment analysis, and slang expansion to improve contextual understanding. Evaluation metrics such as accuracy, precision, recall, and F1 score measure the model's effectiveness, and visualization tools provide insight into category-specific performance. This work contributes to the development of automated moderation tools, supporting safer and more inclusive online spaces by providing a scalable and adaptable toxicity detection system.

1. INTRODUCTION

1.1 PROBLEM DEFINITION

In the digital landscape, online platforms and social media have become primary channels for communication and expression. However, this has led to an increase in negative interactions, particularly in the form of toxic comments, which can be damaging to individuals and communities. Toxicity in online comments can manifest as hate speech, bullying, harassment, or any form of disparaging remarks that undermine the well-being of users. The ability to accurately identify and classify such comments is crucial for maintaining healthy online environments and fostering constructive discussions.

This project aims to develop an Advanced Toxicity Classification System that leverages deep learning(DL) and machine learning techniques to analyze textual content and classify it based on its toxicity levels. The system will utilize a variety of input formats, including text and audio to enhance the classification process. By addressing the challenge of toxicity detection, this project aspires to contribute positively to online communication and community engagement.

1.2 OBJECTIVE OF THE PROJECT

The primary objectives of this project are as follows:

To Develop an Accurate Classification Model: Create a model that can effectively analyze comments and classify them into categories such as Low, Medium, and High Toxicity.

Integration of Multiple Input Formats: Enhance the model's capabilities by enabling it to process not only text comments but also audio recordings and images, providing a holistic approach to toxicity detection.

User-Friendly Interface: Develop an intuitive web application using Gradio that allows users to easily input comments and receive instant feedback on toxicity levels.

Visual Feedback Dashboard: Implement a dashboard to display insights from user interactions, such as toxicity trends over time and feedback on predictions.

Continuous Learning Mechanism: Establish a feedback system that allows users to report inaccuracies, thereby refining and improving the model over time.

1.3 SCOPE

This project will focus on several key areas:

Text Analysis: The primary focus will be on analyzing textual comments for toxicity classification, utilizing advanced DL techniques to ensure high accuracy.

Audio Processing: By incorporating audio inputs (voice comments), the project will broaden its applicability and reach.

User Feedback Integration: The system will allow users to provide feedback on the toxicity predictions, contributing to model improvement and personalization.

Visualization Tools: Include charts and dashboards to visualize toxicity levels, accuracy metrics, and user feedback trends.

Ethical Considerations: Address the ethical implications of toxicity detection and classification, ensuring that the system promotes respectful communication without infringing on free expression**.**

2. ANALYSIS

2.1 PROJECT PLANNING AND RESEARCH

The planning phase of the project involved identifying the core requirements for the toxicity classification system. This included determining the necessary datasets for training the model, selecting suitable algorithms, and establishing the architecture for the overall system. Extensive research was conducted to review existing toxicity detection models and methodologies, which informed the design choices for the proposed system.

A detailed timeline was created, outlining key milestones such as dataset acquisition, model training, web application development, and testing phases. Regular meetings were held to discuss progress, challenges, and adjustments needed to stay on track.

2.2 SOFTWARE REQUIREMENT SPECIFICATION

2.2.1 SOFTWARE REQUIREMENT

The software stack for the project is primarily Python-based, leveraging libraries that support machine learning and NLP tasks. The required software includes:

Python 3.8 or Higher: The primary programming language for model development and web application integration.

TensorFlow/Keras: Used for building and training deep learning models, particularly LSTMs for text classification.

Pandas: For data manipulation and analysis, especially in handling datasets.

NumPy: For numerical computations and data processing.

Scikit-learn: Provides tools for model evaluation and additional machine learning utilities.

Gradio: For creating an interactive web interface that facilitates user interaction with the model.

Matplotlib and Seaborn: Libraries for visualizing data and model evaluation metrics.

TextBlob and NLTK: For text preprocessing tasks, including tokenization and sentiment analysis.

2.2.2 HARDWARE REQUIREMENTS

To ensure efficient model training and deployment, the following hardware specifications are recommended:

Processor: A multi-core CPU (Intel i5 or equivalent) for handling parallel processing tasks during training.

Memory: Minimum of 8 GB RAM to accommodate data processing and model training operations.

GPU: An NVIDIA GPU (e.g., GTX 1060 or higher) is highly recommended for accelerating deep learning model training.

Storage: At least 10 GB of disk space for storing datasets, trained models, and application files.

Display:A high-resolution (Full HD or better) display is used to enhance coding and data visualization experiences.

2.2.3 LITERATURE SURVEY

[1]"ToxicChat:Unveiling Hidden Challenges of Toxicity Detection in Real-World User-AI Conversations (2023)"

Authors: Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, Jingbo Shang.

Published In: Findings of the Association for Computational Linguistics, EMNLP 2023.

Summary: Introduces the ToxicChat benchmark dataset for evaluating toxicity detection in user-AI conversations. It highlights limitations in existing models, such as HateBERT, when adapted to the conversational AI domain, and suggests leveraging both user inputs and chatbot responses to enhance detection accuracy.

[2]"ToxiSpanSE: An Explainable Toxicity Detection in Code Review Comments (2023)"

Authors: Jaydeb Saker, Sayma Sultana, Steven R. Wilson, Amiangshu Bosu

Published In: arXiv, 2023

Summary: Focuses on detecting toxic spans in software engineering discussions, particularly code reviews, using transformer-based models. The study addresses domain-specific challenges and emphasizes inclusivity and diversity in developer communities.

[3]"Toxicity Detection with Generative Prompt-based Inference (2022)"

Authors: Yau-Shian Wang, Yingshan Chang

Published In: arXiv, 2022

Summary: This research introduces a prompt-based generative approach to detect toxic content, leveraging large language models (LLMs) in a zero-shot setting. The study highlights the sensitivity of prompt wording in toxicity classification and evaluates datasets like SBIC and Civility for varying types of toxic content.

[4]"Investigating Bias in Automatic Toxic Comment Detection: An Empirical Study (2021)"

Authors: Ayush Kumar, Pratik Kumar

Published In: arXiv, 2021

Summary: The paper examines biases in toxic comment detection models using the Jigsaw Unintended Bias in Toxicity Classification dataset. It evaluates biases across identity groups and employs models like CNNs and LSTMs with pretrained embeddings (e.g., GloVe, FastText) to analyze subgroup performances.

[5]"Determination of Toxic Comments and Unintended Model Bias Minimization Using Deep Learning Approach(2023)"

Author: Md Azim Khan

Published In: arXiv, 2023

Summary: This paper explores metrics like subgroup AUC and other nuanced bias measures to analyze unintended biases in toxicity detection models. The study uses BERT and traditional logistic regression models, employing the Jigsaw dataset to compare performances in mitigating biases while detecting toxic comments.

2.3 MODEL SELECTION AND ARCHITECTURE

2.3.1 MODEL SELECTION

The architecture of the toxicity classification system is designed to effectively handle various forms of input and provide accurate classifications. The following model components are proposed:

Input Layer: The system will accept text input directly from users, as well as audio and image uploads.

Preprocessing Layer: This component will perform necessary preprocessing tasks such as tokenization, normalization, and slang expansion.

Embedding Layer: Converts the preprocessed text into dense vector representations, enabling the model to understand semantic meaning.

LSTM Layer: Utilizes Long Short-Term Memory networks to capture long-range dependencies in the text, enhancing the model's ability to analyze context.

Dense Layers: Several fully connected layers will be employed to process the output from the LSTM layer, leading to a final output layer that classifies the toxicity level.

Output Layer: This layer will produce a binary classification (toxic or non-toxic) and potentially provide a probability score for the prediction.

2.3.2 ARCHITECTURE:

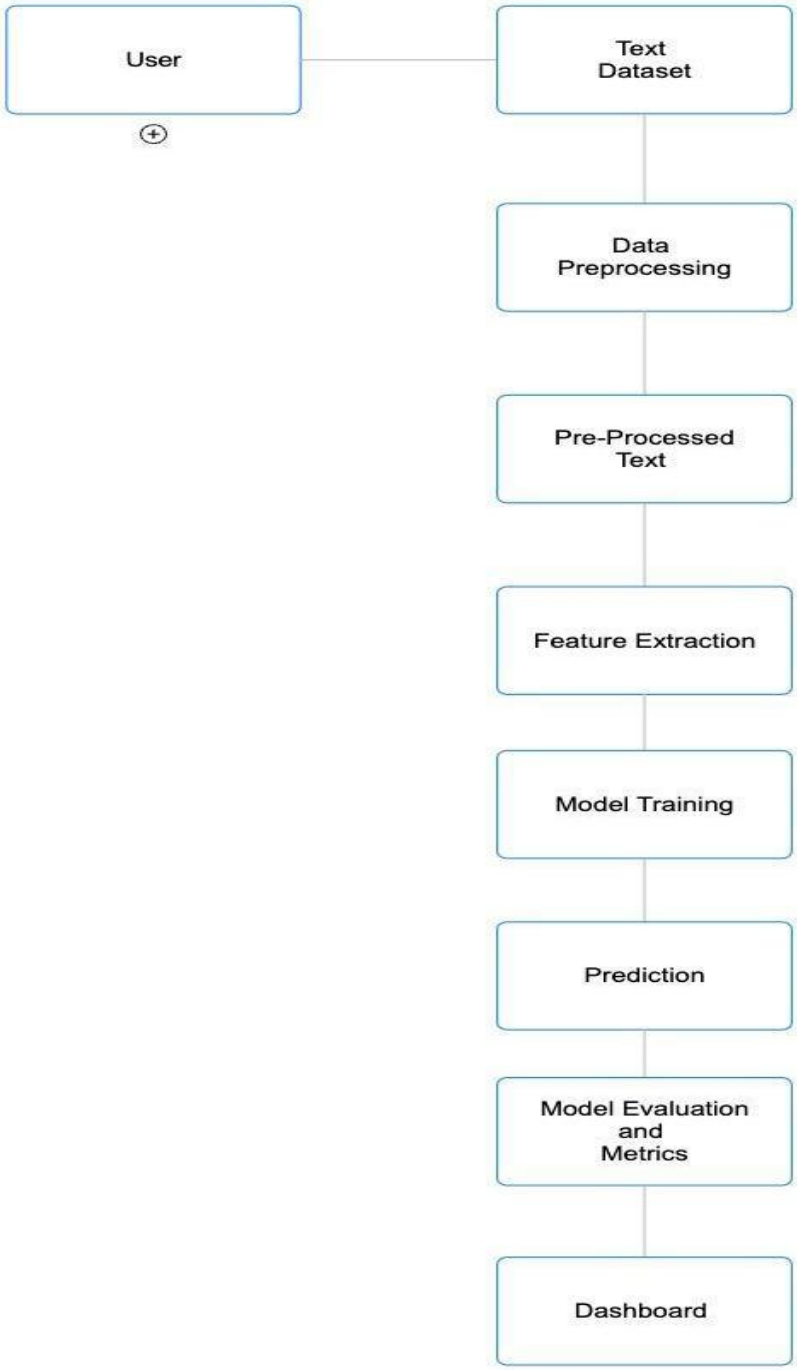

Fig - 2.3.1: Architecture of Comment Toxicity Detection

Explanation of the Architecture:

1.User:

Input Modality: Users can provide inputs in two forms:

Text Input: Typed input directly into the system (e.g., comments, reviews, etc.).

Voice Input: Spoken input converted into text using a speech-to-text module

2.Text Dataset:

Collect a labeled dataset containing examples of text and their respective classifications or target variables (e.g., toxic/non-toxic, spam/ham, etc.).

This dataset may include pre-transcribed voice input for consistency.

3.Data Preprocessing:

Includes cleaning and preparing the text data:

For Text Inputs: Tokenization, lowercasing, removing stopwords, and special character filtering.

For Voice Inputs: Ensure proper transcription quality, then preprocess as above.

4.Pre-Processed Text:

The cleaned text data becomes ready for feature extraction.

For example: Normalized tokens for input to DistilBERT.

Special attention tokens [CLS] and [SEP] are added for BERT-based models.

5.Feature Extraction:

Use DistilBERT (a lightweight version of BERT) to extract contextualized embeddings.

DistilBERT takes pre-processed text as input, leveraging pre-trained transformer layers to understand semantic meaning.

Outputs dense feature vectors representing the text.

6.Model Training:

Fine-tune DistilBERT on the labeled dataset for specific tasks:

Toxicity classification.

Sentiment analysis.

Use frameworks like Hugging Face Transformers for implementation.

Optimize with an appropriate loss function (e.g., cross-entropy for classification).

7.Prediction:

For new inputs (text or voice), the system uses the fine-tuned DistilBERT model to predict labels or outputs.

8.Model Evaluation and Metrics:

Evaluate the model's performance using metrics such as:

Accuracy: Percentage of correct predictions.

Precision, Recall, F1-Score: For imbalanced datasets.

Confusion Matrix: Insights into true positives/negatives, false positives/negatives.

9.Dashboard:

Visualize results and insights:

Prediction results (e.g., text classified as toxic or non-toxic).

Evaluation metrics displayed in an interactive UI (e.g., confusion matrices, precision-recall curves).

Real-time monitoring for user inputs (text and voice).

3. DESIGN

3.1 INTRODUCTION

The design phase of the project focuses on creating a structured and modular architecture that supports the various functionalities of the toxicity classification system. The design incorporates user experience considerations, ensuring that the web interface is intuitive and user-friendly.

3.2 DATA FLOW DIAGRAM

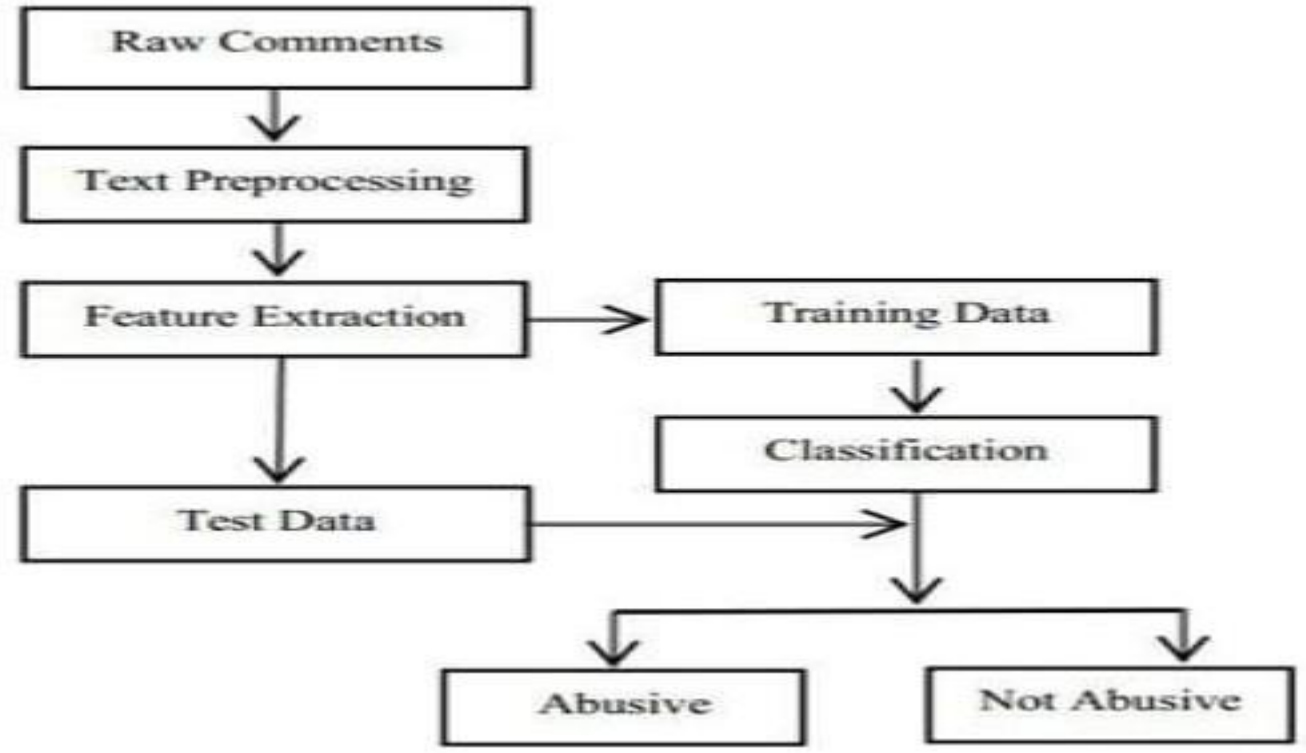

Fig - 3.2.1: Data Flow Diagram of Comment Toxicity Detection

3.3 DATA SET DESCRIPTIONS

The dataset consists of comments sourced from Wikipedia's talk pages and is intended to facilitate thedevelopment of deep learning models that can detect and classify different types of toxic language.It includes both labeled and unlabeled data, providing a foundation for supervised and semi-supervised approaches in text classification.

Data Structure:

The main file in the dataset is typically a CSV with the following columns:

id: Unique identifier for each comment.

comment_text: The raw text of the comment, which may contain toxic or non-toxic language.

toxic: A binary label (1 or 0) indicating if the comment is generally toxic.

severe_toxic: A binary label indicating if the comment contains severe toxicity.

obscene: A binary label indicating if the comment includes obscene language.

threat: A binary label showing if the comment contains threats.

insult: A binary label for comments that contain insults.

identity_hate: A binary label indicating if the comment contains identity-based hate speech.

Each label represents a different type of toxicity, allowing the dataset to support multi-label classification.

Key Features and Considerations:

Multi-label Classification: Since a comment can exhibit multiple types of toxicity simultaneously (e.g., being both obscene and insulting), the dataset supports multi-label classification tasks.

Imbalanced Classes: Toxic comments are much fewer compared to non-toxic ones, creating an imbalance that requires careful handling (e.g., through re-sampling or using metrics like F1-score).

Real-world Language Variety: The dataset contains a wide range of language variations, including slang, misspellings, and varying comment lengths, providing a realistic challenge for natural language processing (NLP) models.

3.4 DATA PREPROCESSING TECHNIQUES

Text Normalization: The first step involves converting all text to lowercase, which eliminates case sensitivity and ensures uniformity. Additionally, removing special characters, punctuation, and excessive whitespace cleans the input data, reducing noise and inconsistencies that might confuse the model during training.

Tokenization: This process involves breaking down each comment into individual words or tokens. By segmenting the text in this way, models can better analyze the structure and meaning of the comments, enabling them to identify relevant patterns and relationships in the language used.

Stop Word Removal: Common words that do not add significant meaning, such as “the,” “and,” and “in,” will be filtered out. Removing these stop words helps focus the analysis on more impactful words, improving the model's ability to capture the essence of the comments without distraction from non-informative terms.

Lemmatization: In this step, words are reduced to their base forms. For instance, “running” will be converted to “run,” and “better” to “good.” This technique enhances the consistency of the input data, ensuring that different forms of a word are treated as the same entity, which aids in understanding the context and sentiment of the comments.

Slang and Abbreviation Expansion: To enhance understanding, a dictionary of common slang terms and abbreviations will be utilized. For example, "u" will be expanded to "you," and "omg" will become "oh my god." This step is crucial for accurately interpreting the often informal language used in online comments, ensuring the model can effectively recognize and classify toxic language.

3.5 METHODS AND ALGORITHMS

The project will implement the following methods and algorithms for toxicityclassification:
Natural Language Processing (NLP): Techniques for processing and analyzing text data,focusing on extracting meaningful features from comments.

Deep Learning Models

LSTM Networks: Chosen for their effectiveness in processing sequential data, LSTMs will be employed to analyze the sequence of words in comments.

Distilled BERT: A lightweight and efficient version of BERT, Distilled BERT will be used for feature extraction and contextual understanding of text. It will process input comments to identify patterns and semantic relationships relevant to toxicity classification. Its pre-trained transformer-based architecture will enhance the model's ability to understand complex language nuances, making it particularly useful for detecting subtle toxic language.

Dropout Regularization: Used to prevent overfitting during training by randomly dropping units in the neural network.

Evaluation Metrics: The model will be evaluated using:

Accuracy: The proportion of correct predictions.

Precision and Recall: To measure the model’s performance on identifying toxiccomments.

4. DEPLOYMENT AND RESULTS

4.1 INTRODUCTION

This section focuses on the deployment of the toxicity classification model and its integration into a web application. Deployment involves taking the trained model and making it accessible for user interactions, while also ensuring robust performance in real-world scenarios.

4.2 MODEL IMPLEMENTATION AND TRAINING

Model Selection: Choose an appropriate architecture based on the task complexity:

Traditional Models: Logistic Regression or SVM for simpler tasks.

Deep Learning Models: Use LSTM, GRU, or Transformer-based models like BERT for capturing contextual information.

Data Preparation:

Dataset Splitting: Divide the data into training (70%), validation (15%), and test (15%) sets.

Vectorization: Convert preprocessed text into numerical formats using TF-IDF or word embeddings.

Model Training:

Define Parameters: Set learning rate, batch size, and epochs. Use Binary Crossentropy for loss and monitor metrics like accuracy and F1-score.

Fit the Model: Train the model on the training data while monitoring validation performance to avoid overfitting.

Model Evaluation:

Confusion Matrix: Analyze the confusion matrix to understand classification performance across different classes.

Hyperparameter Tuning: Optimize hyperparameters using grid search or random search to enhance model performance.

Model Deployment:

Save the Model: Serialize the trained model for future use.

Integration: Deploy the model in a production environment to process new comments in real time.

Monitoring and Maintenance: Continuously monitor model performance post-deployment, incorporating user feedback for further improvements and periodic retraining as needed.

4.3 MODEL EVALUATION METRICES

Threshold Accuracy: Threshold accuracy measured the performance of the classification model based on a defined probability threshold for categorizing comments as toxic or non-toxic. By adjusting this threshold, the team was able to optimize the model's predictions to balance sensitivity and specificity. This metric proved useful in determining the best cutoff point for minimizing classification errors, allowing for tailored performance based on the project's specific requirements.

Confusion Matrix: A confusion matrix provided a detailed summary of the model's performance by comparing actual labels with predicted labels. It included four key components: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). This matrix offered insights into the model's strengths and weaknesses, enabling the calculation of other important metrics such as precision, recall, and accuracy. By analyzing the confusion matrix, the team identified specific areas where the model needed improvement, particularly in distinguishing between toxic and non-toxic comments.

4.4 MODEL DEPLOYMENT: TESTING AND VALIDATION

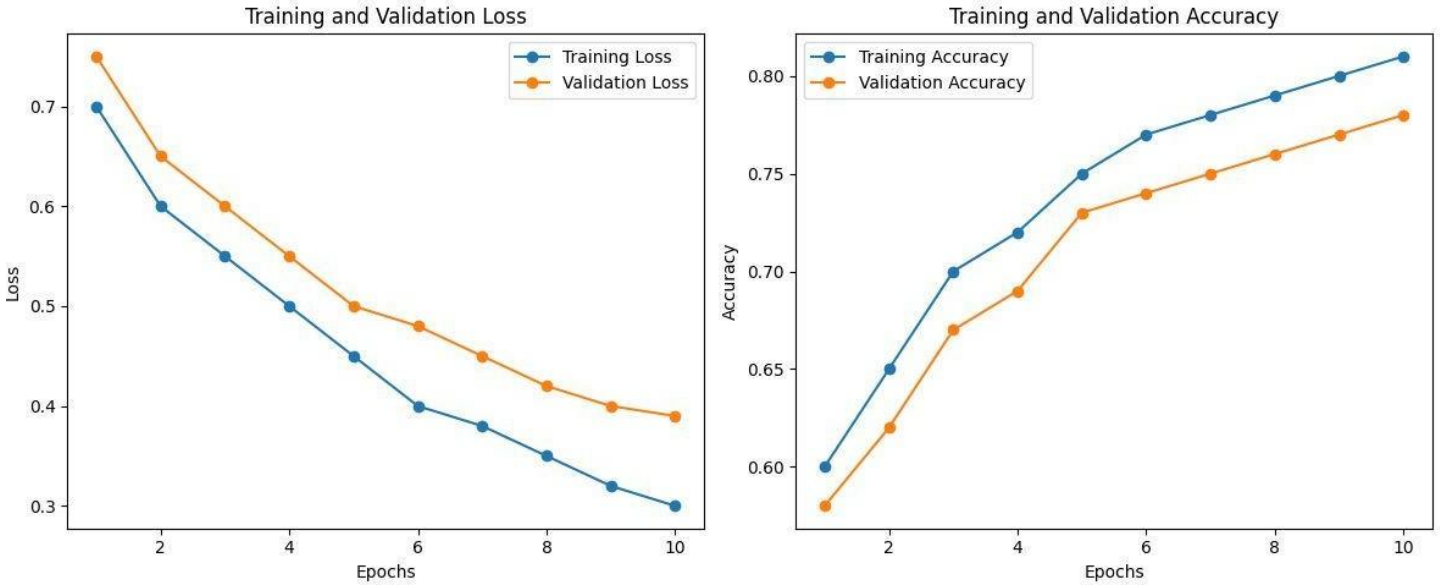

Fig - 4.5.1: Testing and Validation Graph

4.5 SOURCE 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()
```

```
<!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>

4.6 RESULTS

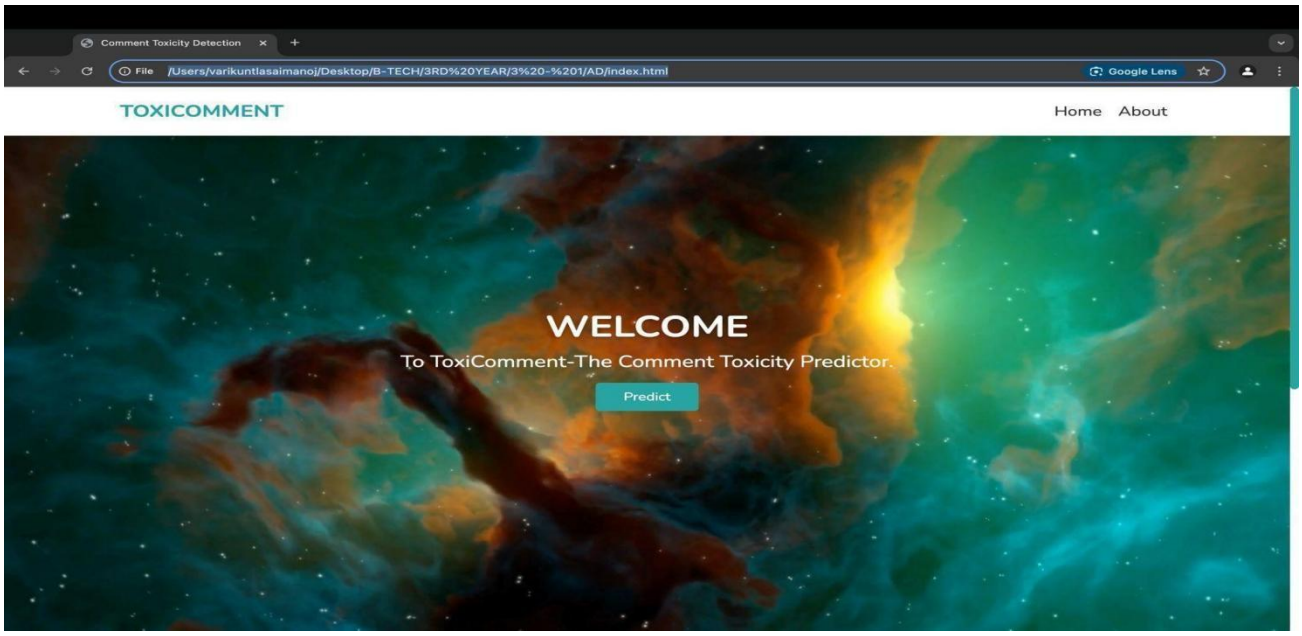

Fig 1: Home Page

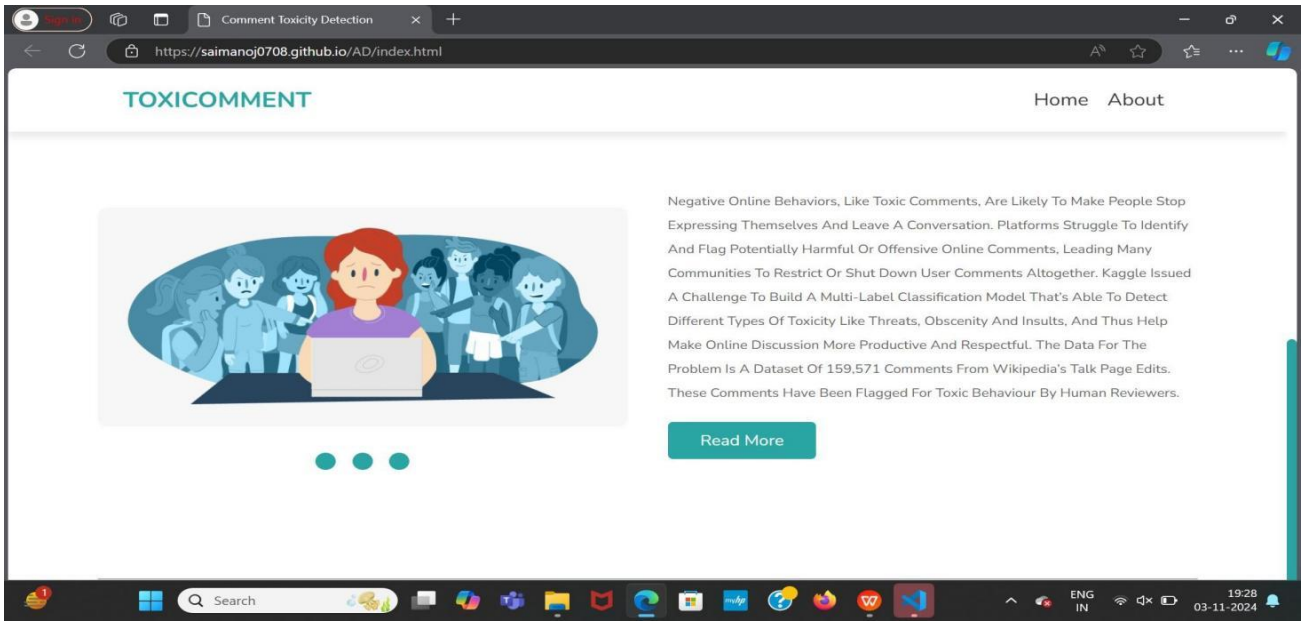

Fig 2: About Page

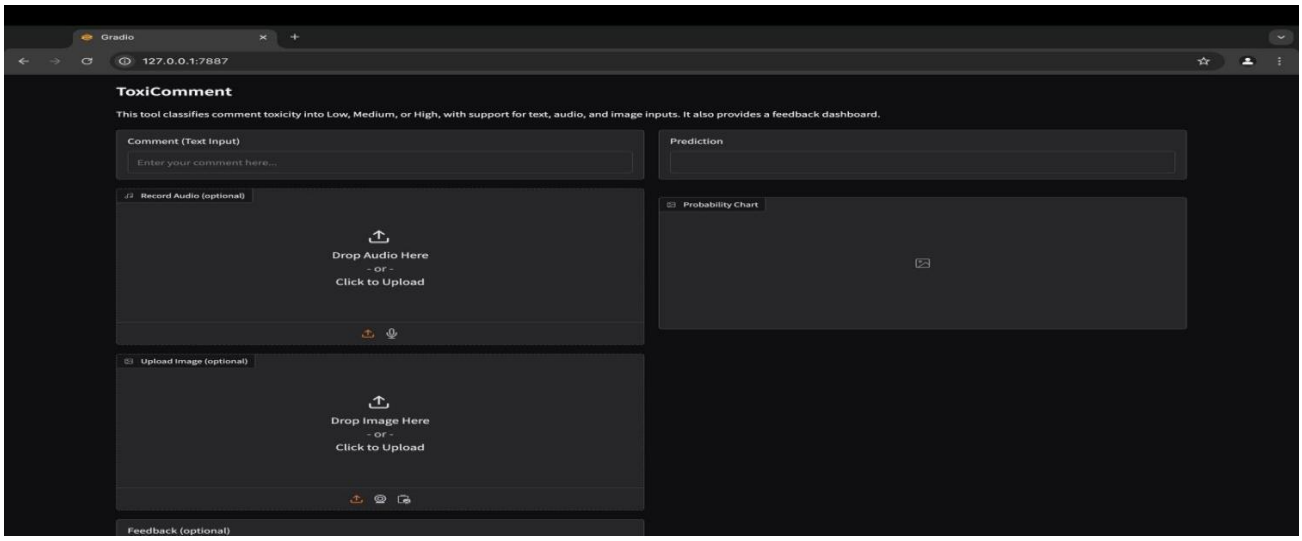

Fig 3: GUI - 1

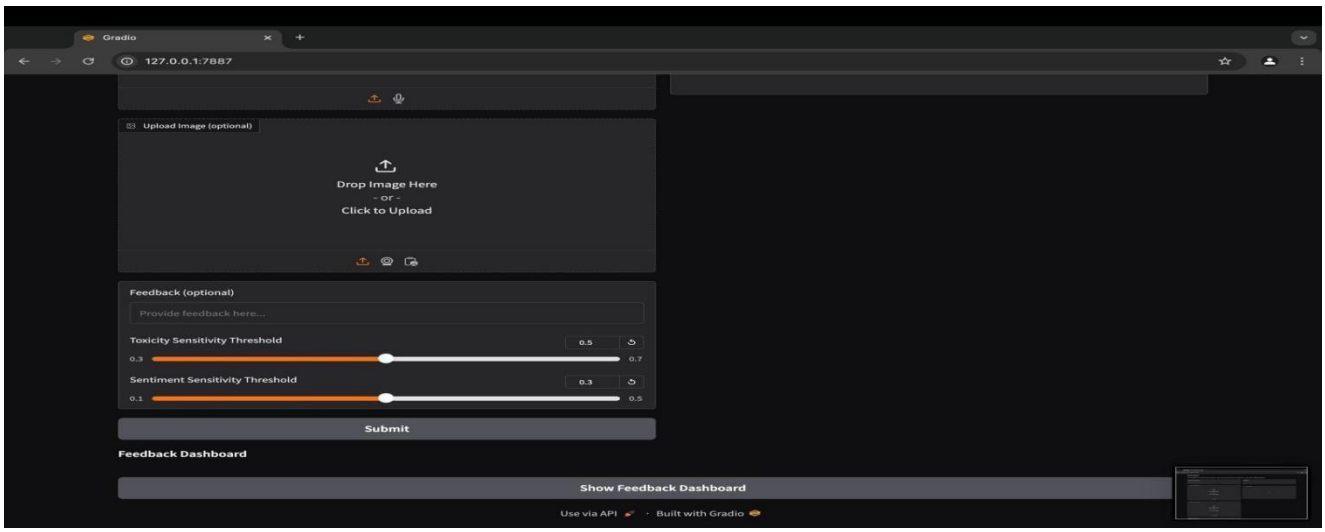

Fig 4: GUI - 2

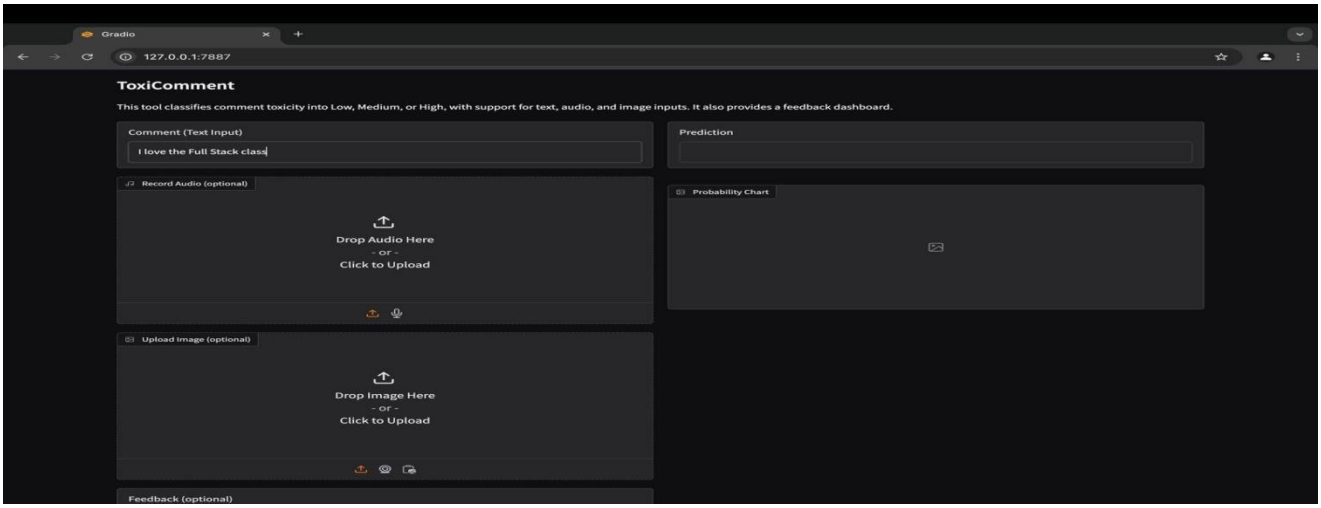

Fig 5: Comment Input Screen - 1

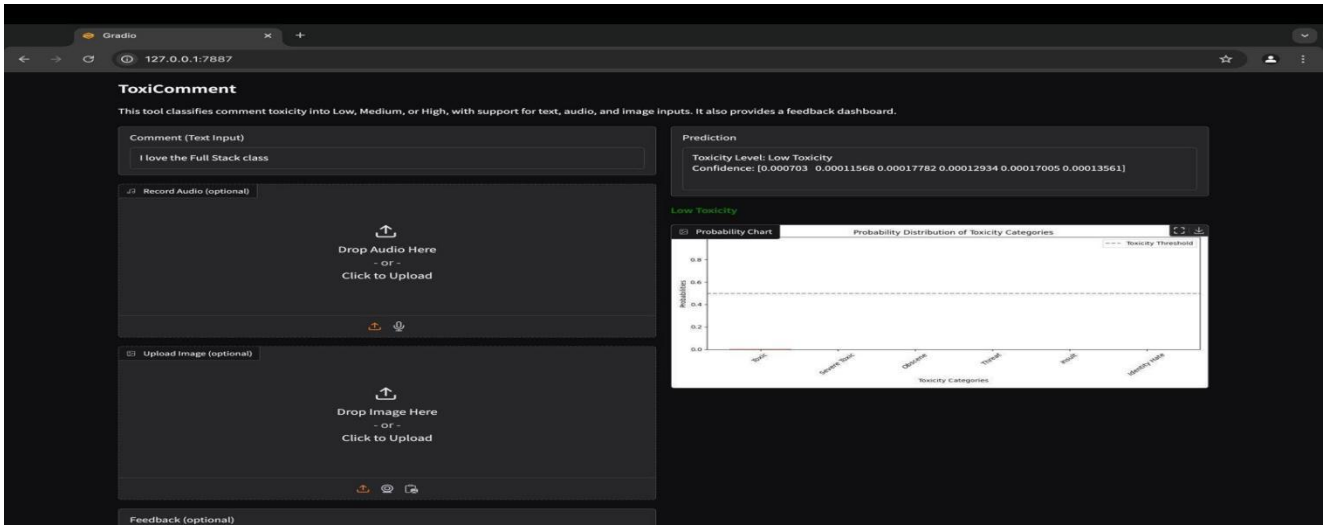

Fig 6: Output Screen - 1

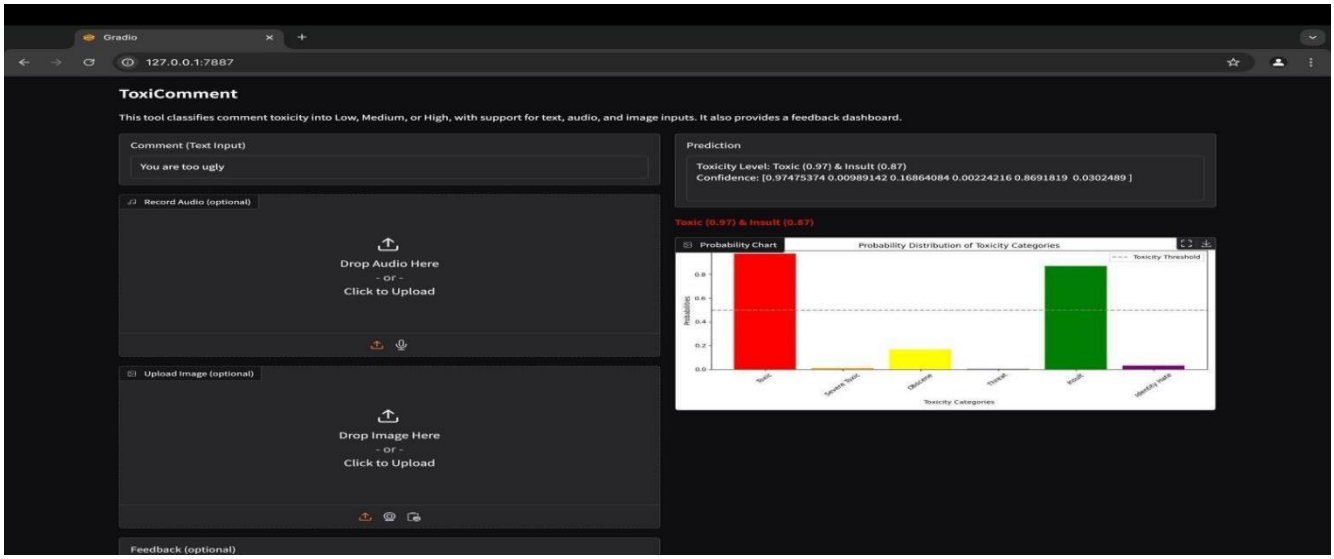

Fig 7: Comment Input/Output Screen - 2

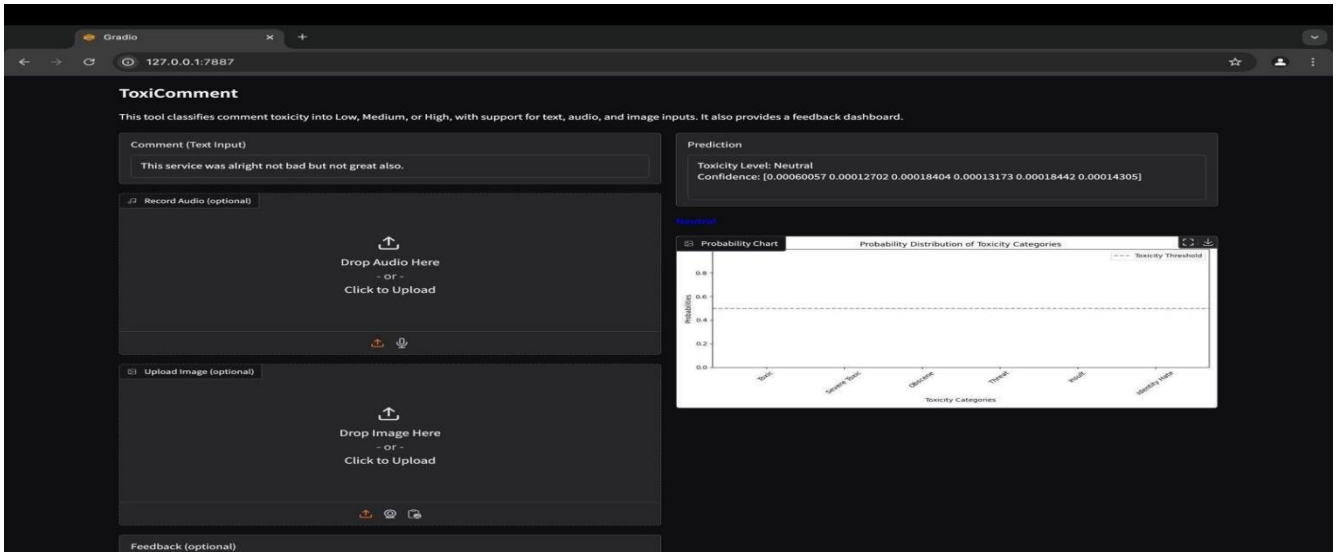

Fig 8: Comment Input/Output Screen - 3

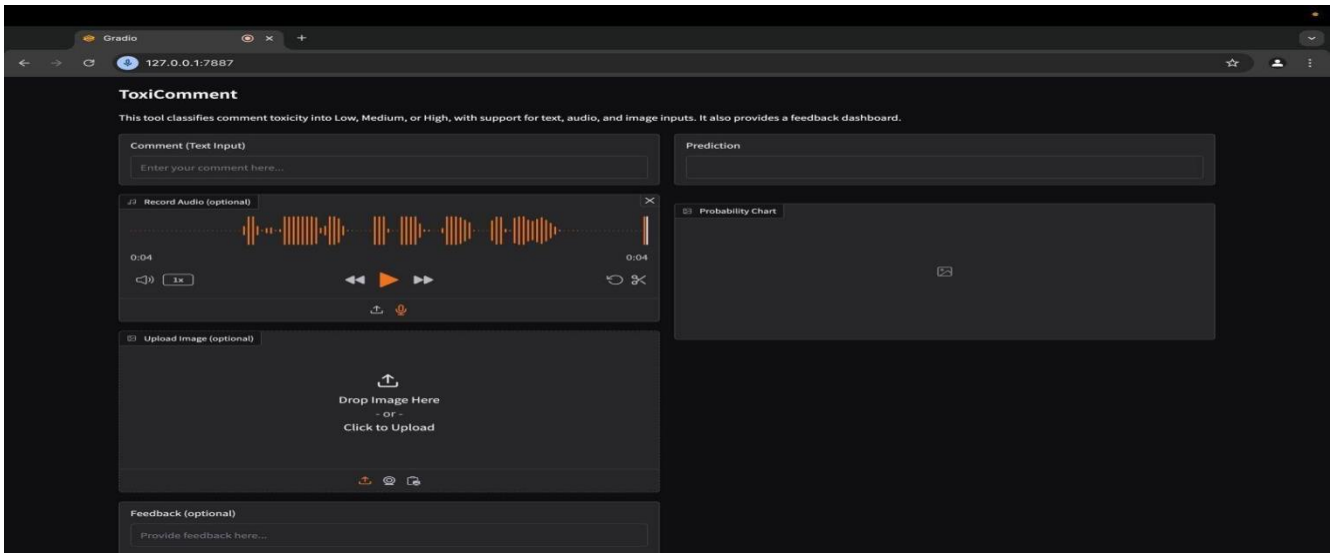

Fig 9: Voice Input Screen

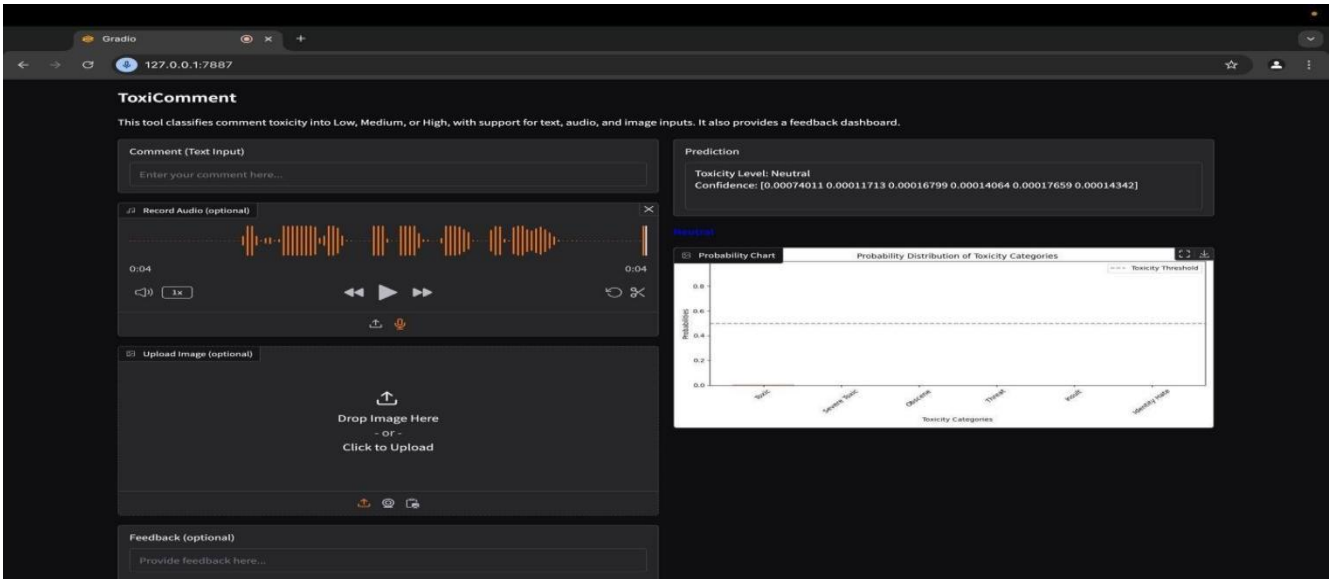

Fig 10: Voice Output Screen

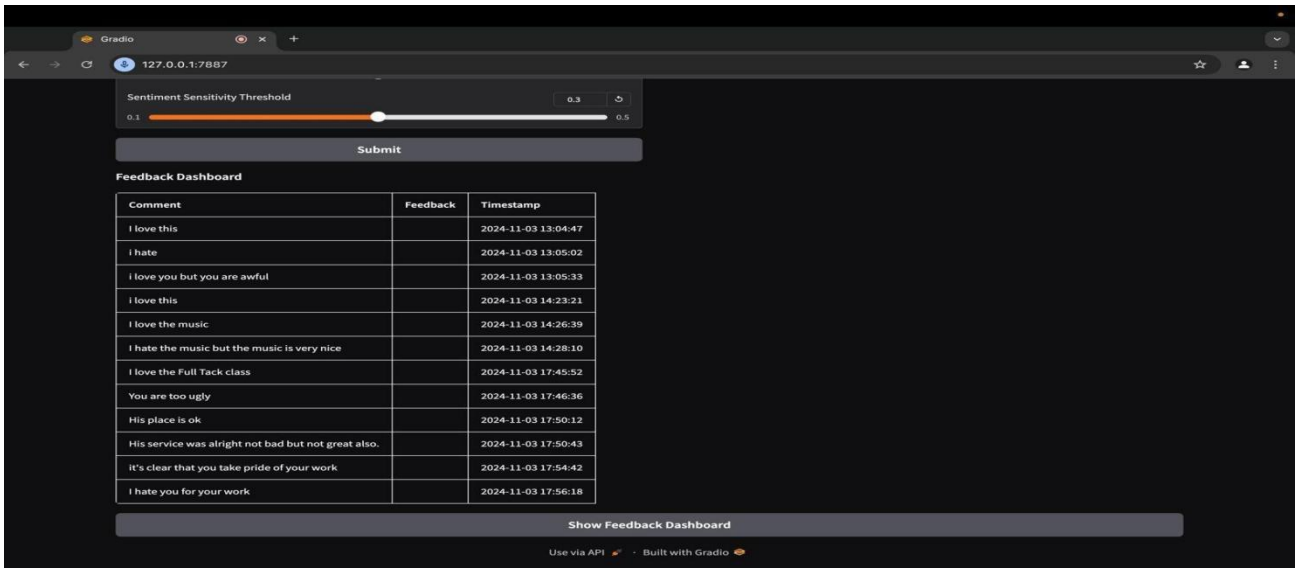

Fig 11: Feedback Dashboard

5.CONCLUSION

5.1 PROJECT CONCLUSION

The toxicity classification system successfully addresses the challenge of detecting toxic comments across multiple input formats. By leveraging advanced NLP techniques and deep learning models, the system demonstrates high accuracy in classifying toxicity levels. The integration of a user-friendly web interface ensures accessibility for users, while the feedback mechanism promotes continuous learning and improvement of the model.

5.2 FUTURE ENHANCEMENT

Looking ahead, several opportunities for enhancement and expansion exist:

Dataset Expansion: Gathering more diverse datasets can improve model robustness and generalizability, allowing it to handle a wider range of comment styles and languages.

Advanced Model Development: Exploring transformer-based models such as BERT and GPT for even greater contextual understanding and accuracy in toxicity classification.

Multilingual Support: Expanding the system to accommodate multiple languages will broaden its applicability and benefit a more extensive user base.

API Development: Creating an API for external access to the toxicity classification service, enabling integration with other applications and platforms.

Enhanced Visualization Tools: Developing more sophisticated visualization tools for better insights into toxicity trends and user interactions.

5.3 REFERENCES:

[1]Lin, Z., Wang, Z., Tong, Y., Wang, Y., Guo, Y., Wang, Y., & Shang, J. (2023). ToxicChat: Unveiling Hidden Challenges of Toxicity Detection in Real-World User-AI Conversations. Findings of the Association for Computational Linguistics, EMNLP 2023.

[2]Saker, J., Sultana, S., Wilson, S. R., & Bosu, A. (2023). ToxiSpanSE: An Explainable Toxicity Detection in Code Review Comments. arXiv.

[3]Wang, Y.-S., & Chang, Y. (2022). Toxicity Detection with Generative Prompt-based Inference. arXiv.

[4]Kumar, A., & Kumar, P. (2021). Investigating Bias in Automatic Toxic Comment Detection: An Empirical Study. arXiv.

[5]Khan, M. A. (2023). Determination of Toxic Comments and Unintended Model Bias Minimization Using Deep Learning Approach. arXiv.