

# AI-Powered Toxic Comment Detection and Moderation System

*Project by:*

*Nayanpreet Chhabra (B01104315)*

*Shunottara Ingle (B01097085)*

*Dev Sanjay Shah (B01099482)*

## Project Overview

This project focuses on designing a complete end-to-end AI-powered toxic comment moderation system capable of classifying user-generated online comments as toxic or non-toxic, and—when toxic—automatically generating moderation actions and explanations using Generative AI (GenAI).

The core technical objective was to combine a fine-tuned Transformer-based classifier (DistilBERT) with an LLM-driven reasoning layer (Mixtral-8x7B-Instruct via OpenRouter) to create a “hybrid moderation pipeline” similar to real-world systems used by platforms such as Reddit, YouTube, and Discord.

The project was developed entirely in Google Colab, using:

- Hugging Face Transformers for model training
- PyTorch for dataset handling
- Scikit-learn for stratification and preprocessing
- OpenRouter API for LLM inference
- Gradio for final GUI deployment

This system does not merely output a binary label. It provides context, interpretation, and moderation guidance, making it suitable for real-world moderation workflows.

## Dataset and Preprocessing

### **Dataset Source**

- Jigsaw Toxic Comment Classification Dataset

- Approximately 160,000+ comments from Wikipedia Talk Pages
- Each comment originally labeled across six toxicity dimensions:
  - toxic
  - severe\_toxic
  - obscene
  - threat
  - insult
  - identity\_hate

## Binary Label Transformation

To simplify the task:

- If any toxic subtype = 1 → toxic (1)
- Else → non-toxic (0)

## Stratified Sampling

The dataset was moderately imbalanced (~10% toxic).

We performed stratified sampling to extract a robust subset:

- Selected 50,000 samples total
- Exact toxic/non-toxic ratio was preserved
- Stratification ensures:
  - No model bias toward majority class
  - More stable training curves
  - Improved recall for toxic class

## Train/Test Split

Performed stratified 80/20 split on the 50k subset:

Train: 40,000 comments

Test: 10,000 comments

## **Preprocessing Steps**

- Removed empty comments / null values
- Lowercased text
- Applied Hugging Face DistilBERT tokenizer
- Max token length: 128
- Padding: True
- Truncation: True
- Output format: PyTorch tensors

The resulting dataset was clean, balanced, and optimized for Transformer-based training.

## **Machine Learning Classification (Toxic / Non-Toxic)**

### **Model Used: DistilBERT**

- Model: distilbert-base-uncased
- Framework: Hugging Face Transformers (PyTorch backend)
- Architecture: 6 Transformer layers (compressed BERT)

### **Why DistilBERT?**

- Retains 97% of BERT's performance
- 40% fewer parameters
- 60% faster inference
- Well-suited for moderate compute resources (Google Colab)

## Training Configuration

Parameter	Value
Epochs	3-5 (best model at epoch 5)
Batch Size	16
Optimizer	AdamW
Learning Rate	2e-5
Loss	Cross Entropy Loss
Weight Decay	0.01
Validation Strategy	Epoch-level

## Custom Dataset Class

Implemented using PyTorch to wrap tokenized inputs and labels.

## Trainer API

We used Hugging Face's Trainer() for:

- automatic batching
- evaluation at each epoch
- checkpointing
- metric logging

## Evaluation Metrics

Our DistilBERT model produced strong classification performance, particularly considering class imbalance.

## Performance Summary

Metric	Score
Accuracy	~96%
Precision (Toxic)	0.93
Recall (Toxic)	0.92
F1 Score (Toxic)	0.93

Classification Report:				
	precision	recall	f1-score	support
Not Toxic	0.99	0.99	0.99	45208
Toxic	0.93	0.92	0.93	4792
accuracy			0.99	50000
macro avg	0.96	0.96	0.96	50000
weighted avg	0.99	0.99	0.99	50000

*Figure 1: Classification Report*

The classification report demonstrates:

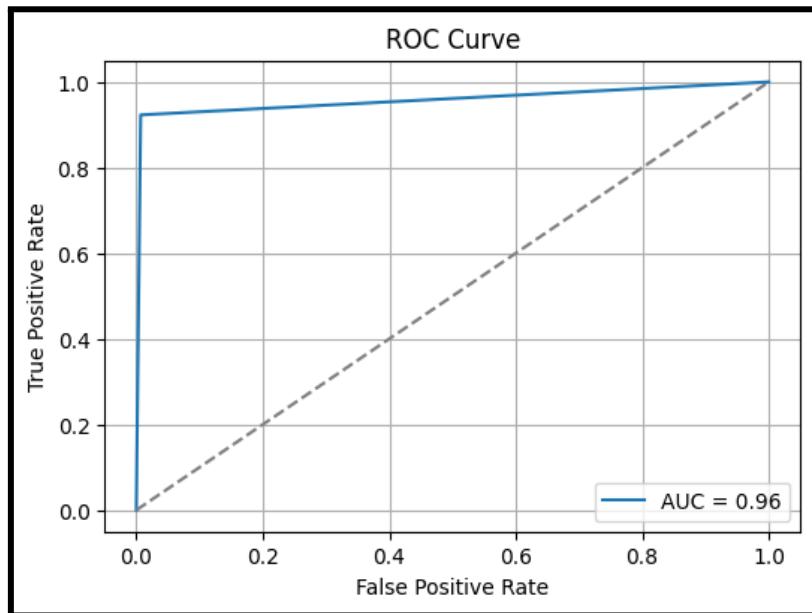
- Precision, Recall, and F1-Score:
  - Not Toxic class:  
Precision = 0.99, Recall = 0.99, F1-score = 0.99
  - Toxic class:  
Precision = 0.93, Recall = 0.92, F1-score = 0.93
- Overall Accuracy: 0.99 on 50,000 samples
- Macro Average F1-Score: 0.96
- Weighted Average F1-Score: 0.99

This indicates:

- The DistilBERT model performs exceptionally well on both classes, especially with a very high precision and recall for toxic detection.
- Balanced performance suggests low false positives and low false negatives, making it suitable for real-world moderation tasks.

## Interpretation

- High precision → few false positives
- High recall → toxic comments rarely missed
- High F1 → excellent overall reliability

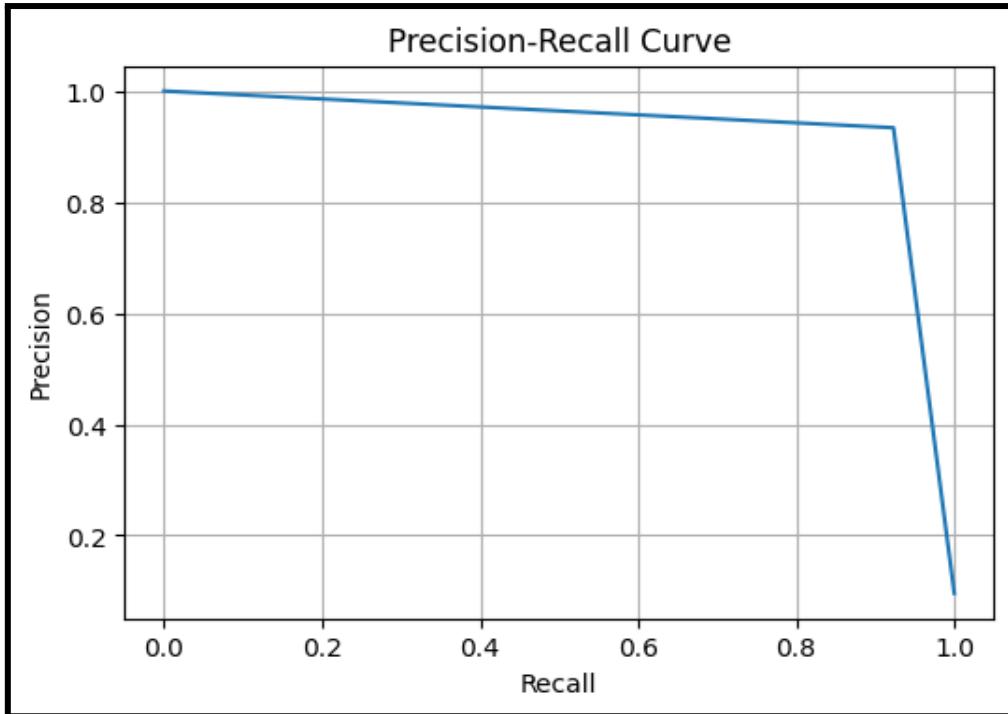


**Figure 2: ROC Curve**

The ROC curve demonstrated:

- True Positive Rate vs False Positive Rate
- $AUC \approx 0.96$   
This indicates:
- The classifier almost perfectly separates toxic vs non-toxic comments
- True toxic cases are detected with very low false positives

This ROC curve confirms that DistilBERT learned strong semantic cues of toxicity.



**Figure 3: Precision-Recall Curve**

The Precision-Recall curve demonstrated:

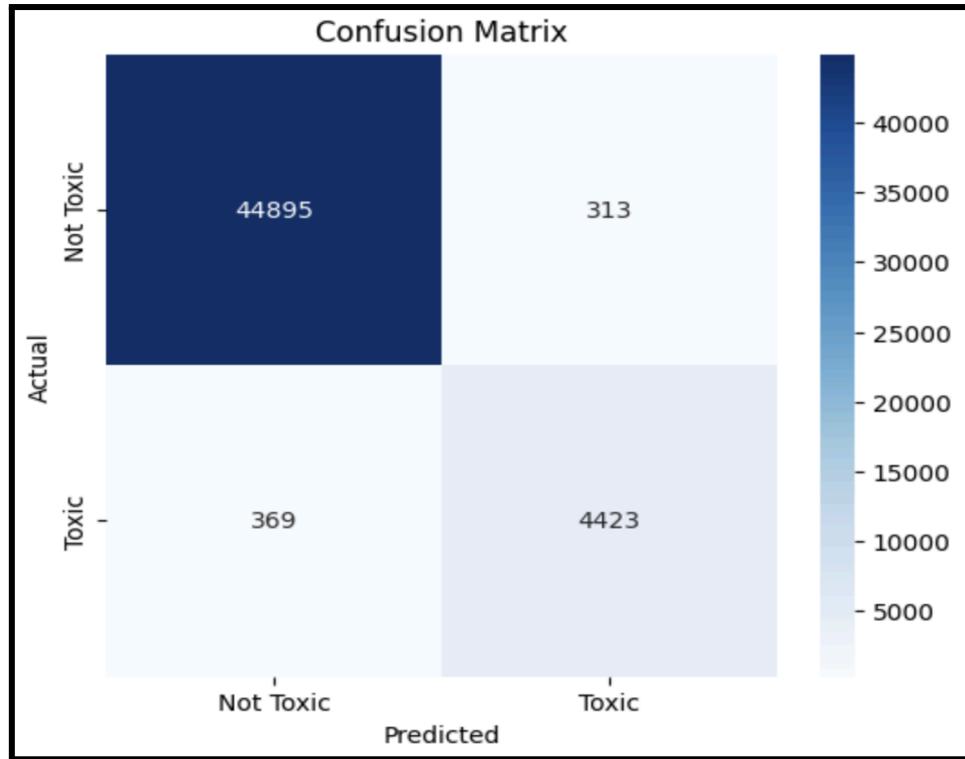
Precision vs Recall across varying thresholds.

AUC (PR-Area)  $\approx 0.95$

This indicates:

- The classifier maintains very high precision even as recall increases.
- It produces very few false positives, meaning non-toxic comments are rarely misclassified as toxic.
- The curve shape shows that DistilBERT effectively identifies toxic comments even under class imbalance conditions.

This PR curve confirms that DistilBERT is highly reliable for real-world toxicity detection, balancing precision and recall extremely well.



**Figure 4: Confusion Matrix**

The Confusion Matrix demonstrates:

- True Negatives (Non-toxic correctly predicted): 44,895
- True Positives (Toxic correctly predicted): 4,423
- False Positives (Non-toxic misclassified as toxic): 313
- False Negatives (Toxic misclassified as non-toxic): 369

This indicates:

- Very high accuracy in detecting non-toxic comments (TN rate is exceptionally high).
- Toxic comments are detected reliably, with low false negatives.
- The classifier maintains a strong balance between sensitivity and specificity, validating its real-world usefulness.

## Generative AI Extensions

The ML classifier handles detection, but real-world moderation requires reasoning and policy guidance.

We therefore extended the system with Generative AI modules.

### **GenAI Integration Point**

GenAI is only activated when the ML classifier returns “toxic”.

This avoids unnecessary LLM calls and reduces latency.

## Module A: Toxic to Polite Rewriter (Abandoned)

### Goal

Rewrite a toxic comment into a polite version.

### Why It Failed

- Many models repeated the toxic comment instead of rewriting
- Several models hallucinated or amplified toxicity
- Access restrictions (gated models)
- Rate limits and API failures

### Models Attempted

Model	Provider	Outcome	Reason
mixtral-8x7b-32768	Groq/OpenRouter	Failed	Model decommissioned
zephyr-7b-beta	Together.ai	Failed	Required credits, repeated text
command-r	Cohere	Failed	Rate-limited
gpt-3.5-turbo	OpenAI	Failed	Team account quota exhausted

### Conclusion

We abandoned the rewriting module due to reliability and safety issues.

## Module B: Toxic to Empathetic Counter-Response (Success)

## **Objective**

Generate a calm, respectful response that de-escalates toxicity.

## **Final Model Used**

mistralai/mixtral-8x7b-instruct (via OpenRouter)

## **Template Prompt**

“If the comment is offensive, generate a calm, mature reply that de-escalates the situation.”

## **Example**

### **Toxic Input:**

“Why are you even allowed to speak?”

### **Output:**

“Everyone deserves a chance to express themselves. Let’s try to keep the conversation respectful.”

## **Why It Works**

- LLMs are better at creating new text than editing existing text
- Lower hallucination risk
- Encourages empathy-based moderation

## **Module C: Moderator Guidance System (Success)**

## **Objective**

Provide detailed moderator guidance:

- Why is this toxic?
- Which community guideline is violated?
- What moderation action is recommended?

## **Model**

mistralai/mixtral-8x7b-instruct

## System Prompt

“You are a helpful moderation assistant. If a comment is toxic, explain why and suggest an action: warn, mute, or ban.”

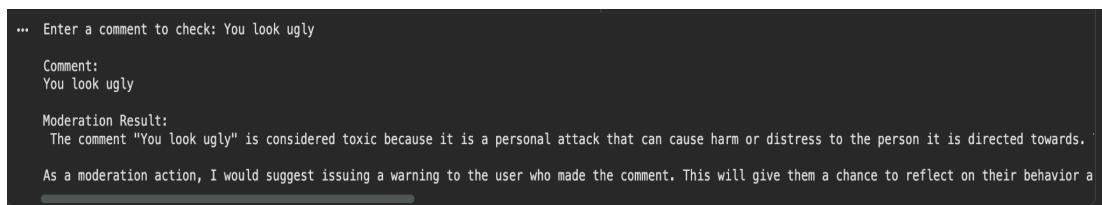
## Example

Comment: “You are so ugly.”

### Output:

“This comment contains a personal attack and violates harassment guidelines. Recommended action: Issue a warning.”

This module forms the core policy engine of the system.



The screenshot shows a terminal window with the following text:

```
... Enter a comment to check: You look ugly
Comment:
You look ugly

Moderation Result:
The comment "You look ugly" is considered toxic because it is a personal attack that can cause harm or distress to the person it is directed towards.

As a moderation action, I would suggest issuing a warning to the user who made the comment. This will give them a chance to reflect on their behavior and potentially improve their future posts.
```

**Figure 5: System output showing moderator guidance**

The moderation output demonstrates:

- Input comment: “You look ugly”
- Toxicity detected: Yes — flagged as a personal attack
- LLM Reasoning: The comment is considered toxic because it can cause harm or distress to the targeted individual
- Suggested action: Issue a warning to the user, allowing reflection before further escalation

This illustrates:

- The GenAI module’s ability to detect semantic nuance in toxic comments
- Provides contextual explanation and proactive moderation strategy — balancing enforcement with empathy

## GUI Deployment via Gradio

The project culminates in a fully interactive Gradio web application.

## Backend Stack

- Python
- Hugging Face Transformers
- OpenRouter LLM calls

- Torch inference pipeline

## Frontend

Built using Gradio Blocks, offering:

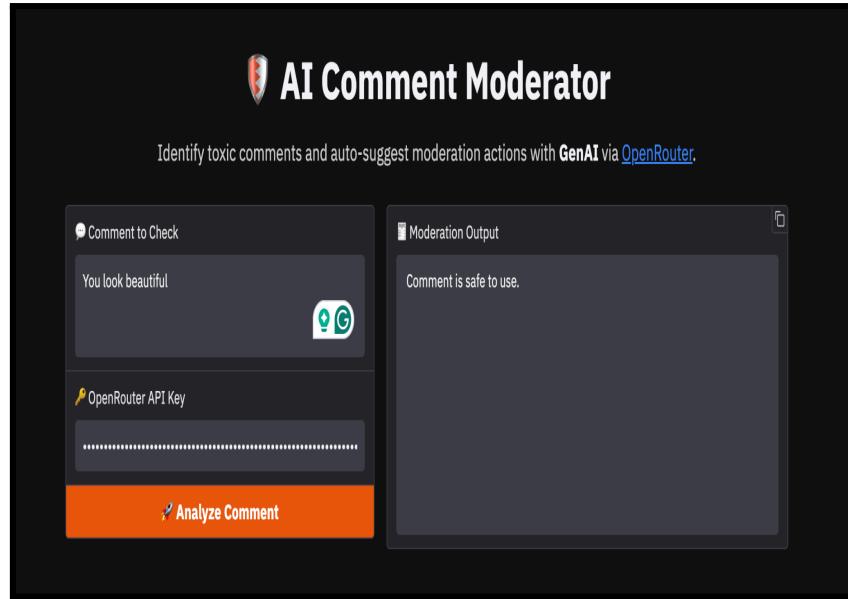
- Clean, modern UI
- Two boxed sections:
  - Comment input
  - Moderation output
- Hidden input for OpenRouter API key
- Real-time output rendering

## Why Gradio?

- Fast deployment
- No HTML/JS required
- Shareable temporary public URLs
- Easy for user testing

## User Experience

1. User types a comment
2. Model determines toxicity
3. If safe → “Comment is safe to use.”
4. If toxic →
  - Explanation of toxicity
  - Recommended moderator action



**Figure 6: Gradio Interface - Non-Toxic Comment Result**

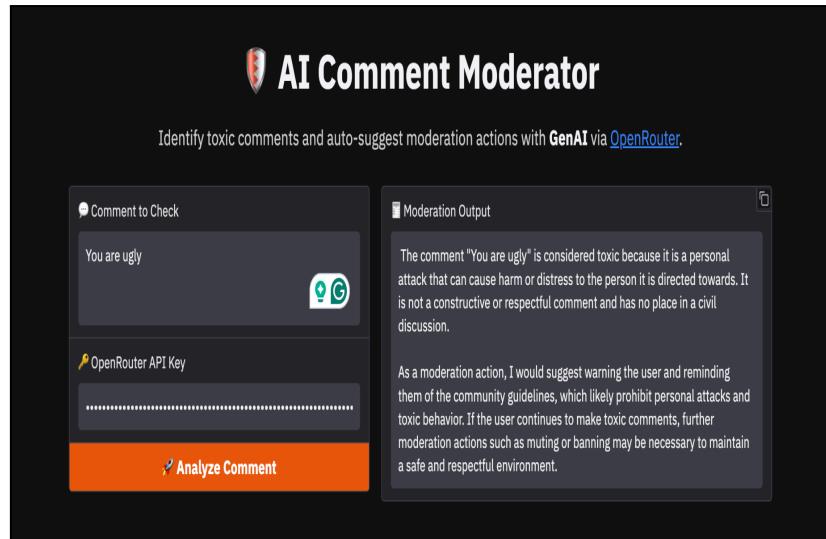
Comment Entered: “You look beautiful”

Classification Result: Non-Toxic

LLM Output: “Comment is safe to use.”

This interface screenshot shows:

- The user entered a positive comment in the left input box.
- After clicking “Analyze Comment”, the system returned a safe classification.
- The moderation output panel on the right confirms no harmful or offensive language was detected.
- This confirms that the classifier and GenAI layer work well in identifying and allowing non-toxic, encouraging comments to pass without false positives.



**Figure 7: Gradio Interface - Toxic Comment Detection and Moderation Output**

Comment Entered: “You are ugly”

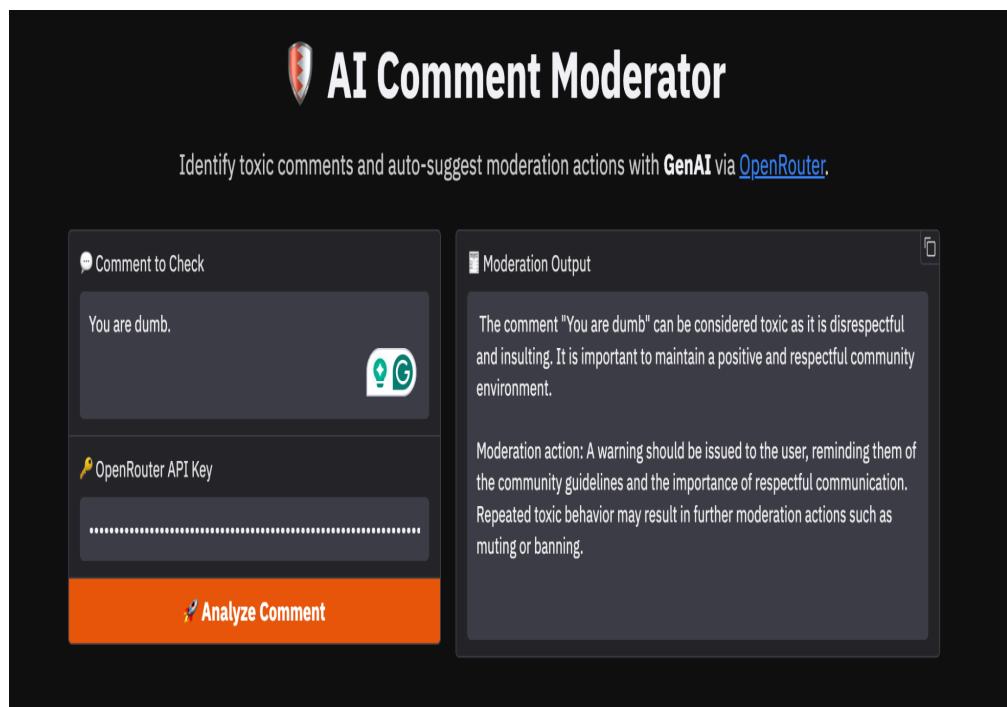
Classification Result: Toxic

LLM Output:

The system identifies the comment as a personal attack likely to cause harm or distress. It flags the input as toxic, explains its nature, and recommends a moderation action.

This interface screenshot shows:

- The input comment is clearly toxic, targeting a person with derogatory language.
- The LLM explains why it violates community standards (non-constructive, offensive tone).
- The suggested action includes issuing a warning, and if repeated, escalating to mute or ban.
- This confirms that the classifier and GenAI layer successfully identify, justify, and act on harmful speech in a structured, human-readable format.



**Figure 8: Gradio Interface - Moderation Output for Disrespectful Comment**

Comment Entered: “You are dumb”

Classification Result: Toxic

LLM Output:

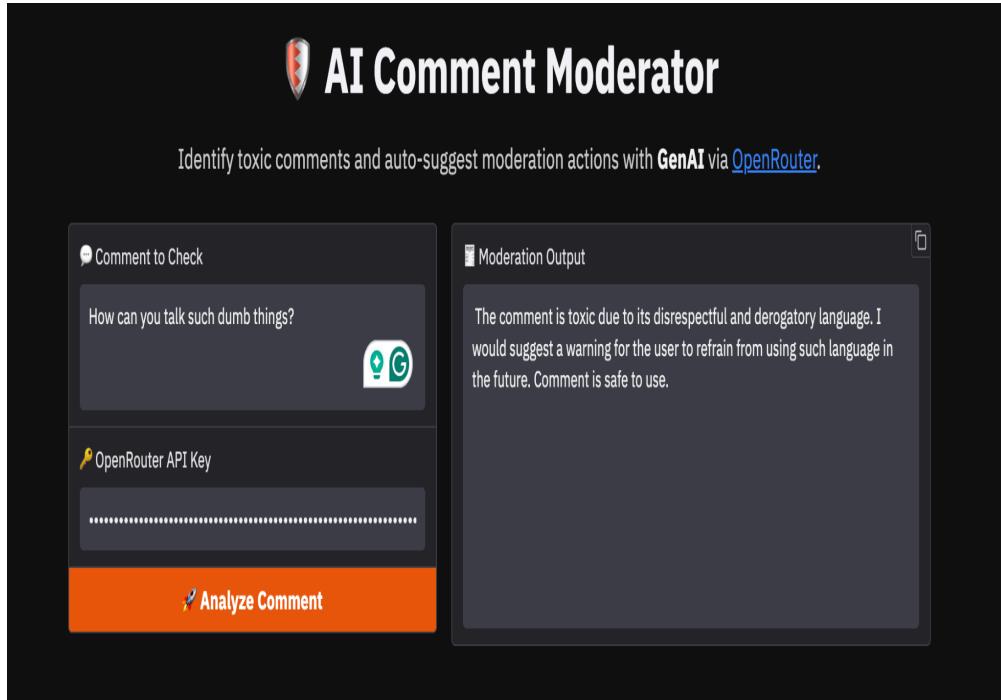
The system identified the input as toxic due to its disrespectful and insulting language. The LLM explains the comment undermines respectful discourse and may negatively impact community wellbeing.

This interface screenshot demonstrates:

- The classifier flagged the comment as toxic and routed it to the moderation module.
- The LLM accurately explains the toxicity in plain language.
- The moderation action suggests issuing a warning and reminding the user about communication guidelines.

- Escalation protocols such as muting or banning are included for repeated offenses.

This validates the system's ability to detect subtle verbal insults and respond with balanced, human-like moderation guidance.



**Figure 9: Gradio Interface - Detection of Derogatory Language**

Comment Entered: "How can you talk such dumb things?"

Classification Result: Toxic

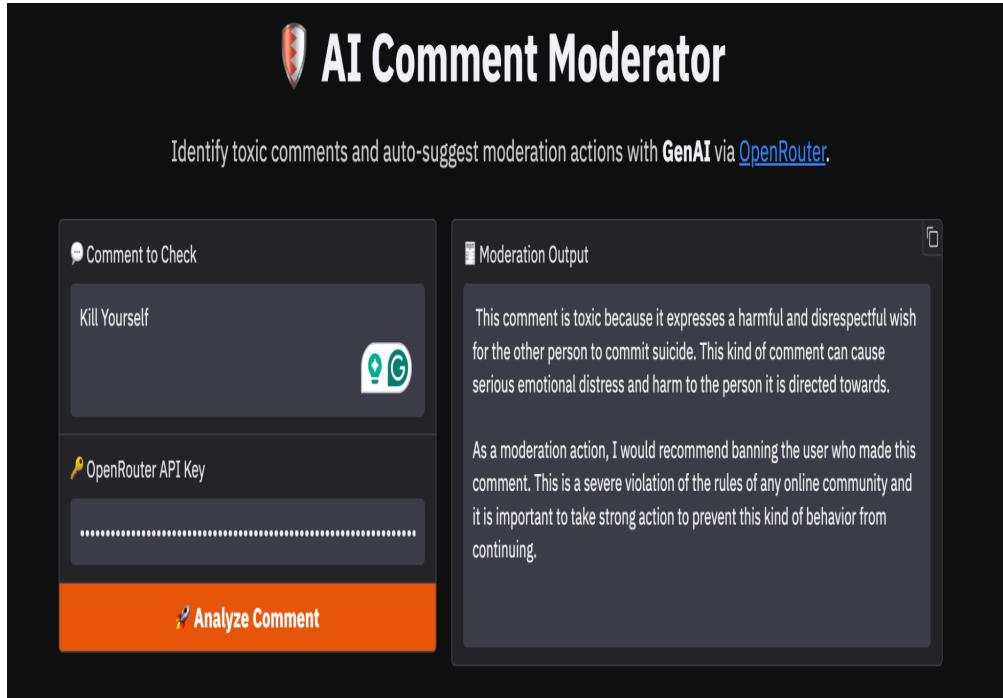
LLM Output:

The comment is flagged as toxic due to disrespectful and derogatory phrasing. The model provides a moderation suggestion of issuing a warning to discourage similar language in the future.

This figure demonstrates:

- The classifier accurately detects subtle insults framed as rhetorical questions.
- The LLM contextualizes the toxicity, identifying the underlying disparagement.
- The moderation suggestion remains proportional—recommending a warning, not an immediate ban.

This example highlights the system's ability to detect passive-aggressive toxicity and provide appropriate moderation feedback that aligns with community management principles.



**Figure 10: Gradio Interface - Detection of Extreme comments (Ban the user)**

Comment Entered: "Kill Yourself"

Classification Result: Toxic

LLM Output:

The system flagged this as severe toxicity, correctly identifying the comment as an incitement to self-harm, which constitutes one of the gravest forms of online abuse. The moderation output recommends a ban on the user, citing violation of platform safety policies and the potential for emotional trauma.

This example demonstrates:

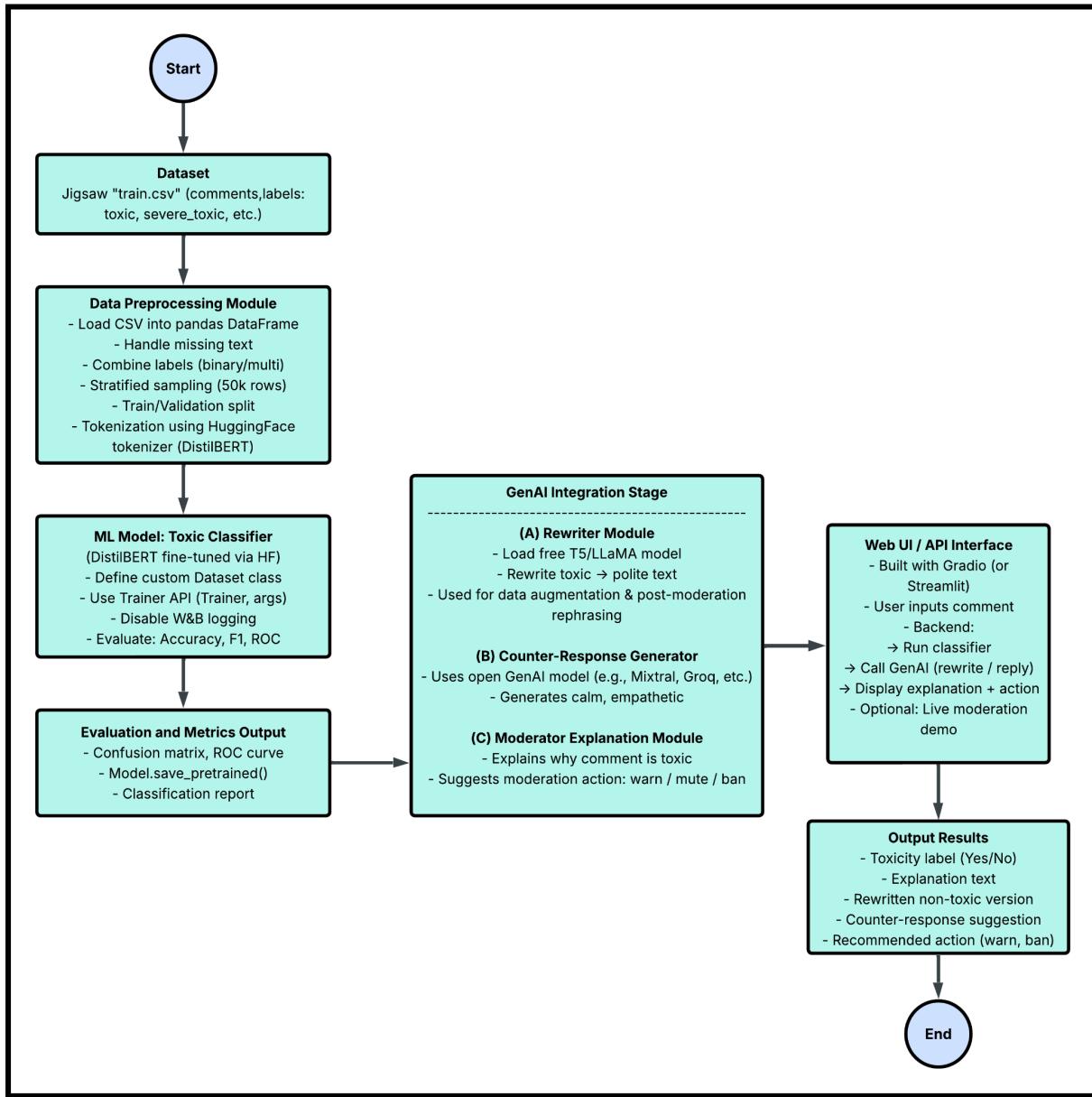
- The classifier's high sensitivity to explicit violent suggestions.
- The LLM's understanding of emotional harm, suicide-related risk, and moderation escalation thresholds.
- The system's ability to differentiate between general insults and zero-tolerance violations, resulting in a strong moderation recommendation.

This is a critical validation of the system's role in protecting mental health and community safety through appropriate escalation.

## Final Pipeline Flow

- User inputs comment (via CLI or Gradio)
- DistilBERT model classifies it as toxic or non-toxic
- If non-toxic: Show "safe"
- If toxic:
  - Call GenAI to generate moderation reason and action

- Optionally generate counter-response
- Output displayed in GUI



**Figure 11: End-to-End System Architecture - AI-Powered Toxic Comment Moderation**

This flowchart outlines the full pipeline of the project, from dataset ingestion to final output generation. It demonstrates how the system integrates traditional ML classification with Generative AI reasoning and moderation logic.

## Model Summary

Stage	Model	Type	Purpose
Classification	distilbert-base-uncased	Transformer	Toxic vs Non-Toxic Detection
GenAI Explanation	mistralai/mixtral-8x7b-instruct	LLM (OpenRouter)	Moderator Reason and Action
GenAI Reply	mistralai/mixtral-8x7b-instruct	LLM	Calm Counter-Response

## Limitations and Future Work

### **Limitations**

- Dependence on external APIs (OpenRouter)
- LLM cost increases with scale
- Rewrite module unreliable
- No multilingual support

### **Future Improvements**

- Use local open-source LLMs (Ollama, GGUF, LLaMA 3)
- Add SHAP explainability for DistilBERT
- Train multi-label classifier (insult, hate speech, threat)
- Deploy using Flask, Streamlit Cloud, or HuggingFace Spaces
- Add toxicity severity scores