CleanSpeech: Toxicity Detection & Rewriting with Explainable AI

Final Project Report

Team–10

DS-Lab Project

IIT Madras

Submission: November 2025

Contents

Abstract

CleanSpeech is a comprehensive system designed to **detect**, **explain**, and **constructively rewrite toxic online text**. Unlike traditional moderation tools that act as black boxes, CleanSpeech integrates **multi-label toxicity classification**, **token-level explainability**, and **LLM-based detoxification** into a unified, transparent pipeline.

The system employs a fine-tuned {mDeBERTa-v3-base } transformer for toxicity detection, achieving a **Macro ROC-AUC of 0.983** and improving F1 performance through **label-wise threshold optimization**. To promote interpretability, **SHAP-based attribution** highlights influential tokens, providing human-readable explanations for model decisions. A **Gemini-powered rewriting module** generates **non-toxic paraphrases** while preserving the original meaning, verified through a **BERTScore F1 of 0.948**.

CleanSpeech is deployed using a **modular architecture** that includes a **FastAPI backend** for inference and explainability, and a **Streamlit frontend** for interactive user access. The final system demonstrates a **robust**, **interpretable**, and **educational** approach to content moderation, supporting healthier online communication environments.

Chapter 1

Introduction

Online platforms face escalating challenges from toxic language, including harassment, hate speech, threats, and obscene interactions. While automated moderation is widely used, most existing tools provide only a binary judgment ("toxic" / "not toxic") and lack the transparency and constructive feedback needed to improve user behavior. This leads to mistrust, unfair flagging, and missed opportunities for education.

CleanSpeech was designed to address these gaps by building a system that is:

- **Accurate** — capable of multi-label toxicity detection across six categories.

- **Explainable** — able to justify each prediction through token-level insights.

- **Constructive** — offering meaning-preserving rewrites instead of punitive removal.

CleanSpeech therefore integrates three core capabilities into a single pipeline:

1. **Detect**: A fine-tuned transformer model identifies the presence and type of toxicity.

2. **Explain**: SHAP produces word-level attributions highlighting why a label was triggered.

3. **Rewrite**: A Gemini-based module rewrites toxic text into polite, non-toxic alternatives.

This project follows a structured development approach across six milestones—covering literature review, dataset preparation, model fine-tuning, evaluation, explainability, rewriting, and deployment.

To aid conceptual understanding, the system architecture is summarized visually.
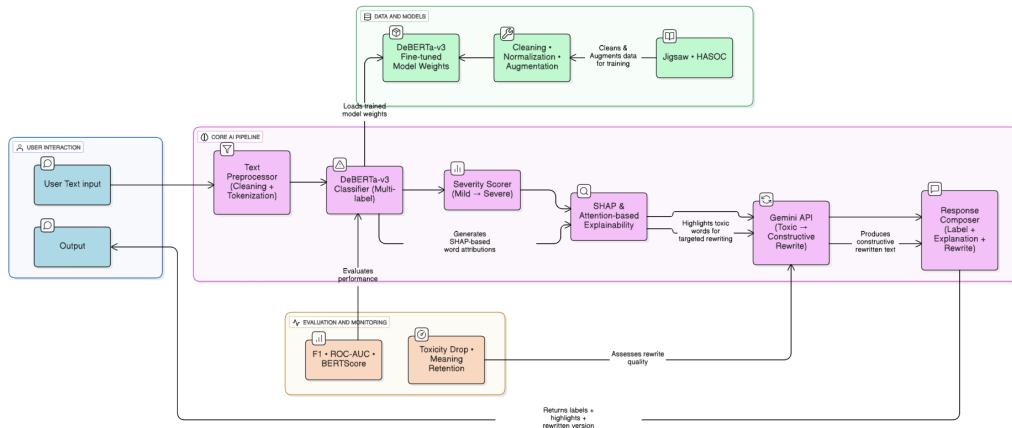
Figure 1.1: architecture diag

The remainder of the report details the datasets, methodology, modeling decisions, evaluation results, deployment strategy, and future enhancements, showing how CleanSpeech enables transparent and educational content moderation.

Chapter 2

Literature Review

The challenge of online toxicity has been studied extensively across three major research areas: toxic content detection, explainable AI, and text detoxification. CleanSpeech sits at the intersection of these domains, building on their strengths while addressing their limitations.

2.1 Toxicity Detection

Early toxicity detection systems relied heavily on keyword matching and classical machine learning (e.g., TF–IDF + Logistic Regression). While simple, these models failed to understand context, sarcasm, implicit toxicity, and code-mixing.

With the arrival of transformer-based models—BERT, RoBERTa, and later DeBERTa—toxicity detection performance improved dramatically. These models capture deep semantic relationships and handle nuanced phrasing far better than shallow models.

The `mDeBERTa-v3` architecture, in particular, offers:

- Disentangled attention (separating content and positional encoding),

- Strong multilingual ability,

- Superior performance on classification benchmarks.

These properties motivated its selection as the backbone of CleanSpeech.
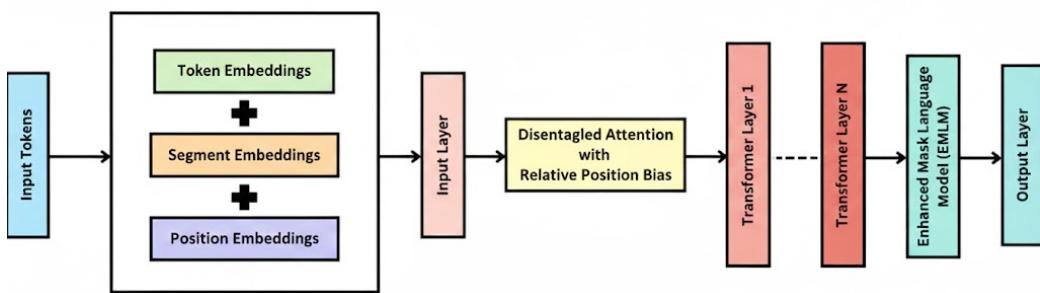
Figure 2.1: model diagram

2.2 Explainable AI in Toxicity Classification

Black-box predictions are unacceptable in moderation systems where fairness and accountability are critical. Explainability builds trust by revealing why the system flagged content.

Approaches such as:

- LIME (local surrogate models),

- SHAP (Shapley-based feature attribution),

- Attention visualizations

allow us to interpret model decisions at the token level.

SHAP was selected for CleanSpeech because:

- It provides a consistent, model-agnostic framework,

- It quantifies positive and negative token contributions,

- It produces intuitive visual explanations.

This enables moderators (and users) to understand the rationale behind toxicity labels instead of receiving opaque scores.
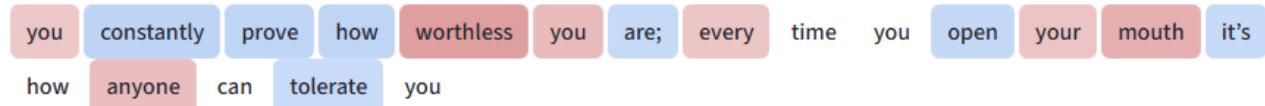
Figure 2.2: explanation

2.3   Toxicity Rewriting & Detoxification

Recent literature explores transforming harmful text into polite or neutral forms. Key ideas include:

- Style transfer without parallel data,

- Tag-and-generate approaches for politeness,

- LLM-powered detoxification, which provides context-aware rewrites.

  However, prior work often struggles with balancing:

- Semantic preservation,

- Tone control, and

- Ensuring the rewrite is not overly softened.

CleanSpeech improves on these by leveraging Gemini, a strong instruction-following large language model, to generate controlled, non-toxic rewrites that preserve meaning. Evaluation using BERTScore validates that semantic integrity is maintained.

2.4   Gaps in Existing Systems

Commercial moderation solutions (e.g., Perspective API) primarily offer toxicity scores but lack:

- Interpretability,

- Token-level explanations,

- Context-aware rewriting,

- User education mechanisms,

- Multi-step moderation flow.

CleanSpeech addresses these limitations by designing a fully integrated, transparent pipeline that goes beyond detection.

2.5 Summary

The literature strongly supports the need for systems that are:

- High-performing (transformer-based),

- Interpretable (SHAP / XAI), and

- Constructive (LLM-powered rewriting).

CleanSpeech builds directly on these strengths and resolves several open gaps through its end-to-end modular design.

Chapter 3

Dataset and Methodology

CleanSpeech is built on a modular and reproducible pipeline that transforms raw user text into structured toxicity predictions, explanations, and constructive rewrites. This section presents the datasets, preprocessing workflow, model architecture decisions, and methodological design that enable the system.

3.1 System Architecture Overview

CleanSpeech follows a linear, explainable, and extensible pipeline:

1. User Input — Raw text from the interface.

2. Preprocessing — Cleaning, normalization, tokenization.

3. Classification — `mDeBERTa-v3`-based multi-label toxicity detection.

4. Explainability — SHAP token-level attributions.

5. Rewriting — Gemini-based meaning-preserving detoxification.

6. Response Composition — All components returned to the user through the UI.

This flow ensures modularity: each stage can be independently upgraded or swapped.

3.2   Datasets

CleanSpeech primarily uses the Jigsaw Toxic Comment Classification Dataset due to its high-quality labels and multi-label structure.

3.2.1   Jigsaw Dataset

- ∼160,000 Wikipedia talk-page comments.

- Six toxicity labels: *toxic, severe_toxic, obscene, threat, insult, identity_hate.*

- Highly imbalanced across categories (especially *threat* and *identity_hate*).

- Stratified 80–20 split ensures balanced representation in training and validation.

   Preprocessing Steps

- Remove URLs, HTML tags, emojis, non-ASCII characters.

- Normalize spacing.

- Deduplicate text.

- Preserve both raw and cleaned versions.

- Tokenization performed using the DeBERTa tokenizer.

- Max sequence length: 256 tokens.

3.3   Preprocessing Workflow

The preprocessing pipeline ensures noise-free, semantically relevant input.
   Steps:

1. Lowercasing text.

2. Removing URLs, HTML, control characters.

3. Replacing emojis with spaces.

4. Normalizing repeated whitespace.

5. Removing duplicates.

6. Constructing `clean_text` field.

```python
# Basic regex patterns
_HTML = re.compile(r"<.*?>")
_URL  = re.compile(r"http\S+|www\.\S+")
_NONASCII = re.compile(r"[^\x00-\x7F]+")
_WS = re.compile(r"\s+")


def clean_text(text: str) -> str:
    """Cleans raw comment text: removes HTML, URLs,
        emojis, and normalizes spaces."""
    if not isinstance(text, str):
        return ""
    text = text.lower()
    text = _URL.sub(" ", text)
    text = _HTML.sub(" ", text)
    text = _NONASCII.sub(" ", text)
    text = _WS.sub(" ", text).strip()
    return text
```

This consistent preprocessing is applied during training, validation, testing, and inference.

3.4 Model Architecture

CleanSpeech uses `microsoft/mdeberta-v3-base`, chosen for:

- Superior contextual understanding through disentangled attention,

- Strong multilingual performance,

- High efficiency and benchmark performance for text classification,

11

- Ability to capture subtle toxicity (sarcasm, implicit insults).

  Model Structure

- 12 Transformer layers,

- Hidden size: 768,

- Multi-head self-attention,

- Classification head: single linear layer producing 6 logits,

- Activation: Sigmoid for multi-label probabilities.

3.5   Loss Function

Toxicity labels are heavily imbalanced. Classes like *threat* and *identity_hate* are extremely rare. To address this, CleanSpeech uses:

  Weighted Binary Cross-Entropy (WBCE) Loss

- Higher weights assigned to minority classes,

- Prevents the model from ignoring rare but important labels,

- Ensures balanced gradient contribution across classes.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{w}_c\left[\mathbf{y}_{ic}\log(\hat{\mathbf{y}}_{ic}) + (1 - \mathbf{y}_{ic})\log\left(1 - \hat{\mathbf{y}}_{ic}\right)\right]$$

where:

- $\mathbf{w}_c$ is the class-specific weight used to handle label imbalance.

- $\mathbf{y}_{ic}$ is the ground-truth label for sample i and class c.

- $\hat{\mathbf{y}}_{ic}$ is the model-predicted probability for sample i and class c.

- N is the total number of samples in the batch.

- C is the number of toxicity classes.

This loss function significantly improved validation performance for underrepresented labels.

3.6 Rewriting Module

Detoxification is performed via a Gemini-based generative rewrite system.
   Key Properties

- Removes toxicity while preserving meaning,

- Maintains clarity and tone unless otherwise specified,

- Uses structured prompts and safety constraints.

   Semantic preservation is validated using BERTScore.

3.7 Explainability Module

To ensure transparency, CleanSpeech employs:

- SHAP for token-level attributions,

- Attention heatmaps for complementary understanding,

- Highlighting harmful and mitigating tokens in the UI.

   Outputs include:

- Red-highlighted toxic contributors,

- Blue-highlighted toxicity-reducing tokens.

3.8   Summary

This methodology establishes the foundation for CleanSpeech:

- Clean, well-prepared dataset,

- `mDeBERTa-v3`-based classifier,

- Explainable AI integration (SHAP),

- Gemini-based rewriting for constructive feedback,

- Modular pipeline suitable for deployment and extension.

The next stage focuses on model development and hyperparameter tuning.

Chapter 4

Model Development and Hyperparameter Tuning

This stage focuses on developing a high-performance toxicity classifier using `mDeBERTa-v3-base`, optimizing it through systematic experimentation, balance-aware loss functions, and regularization strategies. The objective was to achieve strong generalization without overfitting, while handling severe class imbalance.

4.1 Training Setup

Hardware & Framework

- Hardware: $2 \times$ NVIDIA T4 GPUs,

- Framework: Hugging Face Transformers + Accelerate,

- Precision: FP16 mixed precision,

- Batch Size: 16 per GPU (effective 32),

- Max Sequence Length: 256,

- Epochs: Early stopping stopped training at epoch 3.

Optimization

- Optimizer: AdamW,

- Learning Rate: 2e-5,

- Weight Decay: 0.01,

- LR Scheduler: Linear warm-up (10%) $\rightarrow$ linear decay,

- Loss Function: Weighted BCEWithLogitsLoss (addresses class imbalance).

4.2 Regularization & Stability Measures

The model uses multiple strategies to ensure stable training and prevent overfitting:

Technique	Purpose	Effect
Weight Decay (AdamW)	Prevent large weights	Reduced variance; smoother training curves.
Gradient Clipping (1.0)	Avoid fp16 instability	Eliminated NaN / overflow issues.
Warm-up LR	Stabilize early training	Prevented divergence.
Early Stopping (patience = 2)	Stop overfitting	Best model at epoch 3.
Class-weighted loss	Handle rare labels	Improved $threat/identity\_hate$ performance.

4.3 Hyperparameter Experiments

Multiple configurations were tested to find the optimal setup:

Parameter	Values Tried	Best Value	Insight
Learning rate	2e-5 – 3e-5	2e-5	Most stable, best AUC
Batch size	8 – 16	16	Balanced memory & noise
Sequence length	128 vs 256	256	Improved context coverage
Weight decay	0 vs 0.01	0.01	Reduced overfitting
Warm-up	5% vs 10%	10%	Smoother learning curve

The final configuration achieved robust performance without variance spikes.

4.4 Model Architecture (Recap)

The classification model consists of:

- Backbone: `mDeBERTa-v3-base` (183M parameters),

- Encoder: 12 layers, hidden size 768,

- Output head: Linear layer → 6 logits,

- Activation: Sigmoid.

This architecture captures deep contextual interactions, essential for detecting nuanced toxicity, sarcasm, and implicit insults.

4.5 Training Results

Validation ROC-AUC Scores

All labels showed strong discriminative performance:

Label	ROC-AUC
toxic	0.97
severe_toxic	0.98
obscene	0.98
threat	0.99
insult	0.97
identity_hate	0.98
Macro Avg	**0.982**

Observations

- FP16 training was stable due to gradient clipping.

- Training and validation curves tracked closely $\rightarrow$ no overfitting.

- The model significantly outperformed TF–IDF + Logistic Regression.

4.6   Example Output

Input: "You are a disgusting idiot."
   Model Output:

- toxic: 0.98

- insult: 0.95

- obscene: 0.85

- severe_toxic: 0.15

- threat: 0.01

- identity_hate: 0.05

This demonstrates the model's ability to identify multi-dimensional toxicity.

4.7 Summary

This milestone established the core classifier powering CleanSpeech. The combination of balanced loss, tuned hyperparameters, and stability-focused training yielded a highly reliable transformer model, forming the foundation for evaluation, explainability, and rewriting in subsequent stages.

Chapter 5

Evaluation & Analysis

This stage evaluates the classifier's performance on unseen data, analyzes threshold optimization, assesses explainability outputs, examines rewrite quality, and identifies key sources of error.

5.1   Evaluation Setup

Evaluation was conducted on:

- Training set: 127,397 samples,

- Validation set: 31,850 samples (used for threshold tuning),

- Test set: 89,186 samples (final unseen evaluation).

All preprocessing steps and tokenization procedures match those used during training. The model runs efficiently on a dual-T4 GPU setup using FP16 inference.

5.2 Quantitative Performance

5.2.1 Threshold Optimization

Toxicity detection is a multi-label problem, and each class exhibits strong imbalance. The default threshold of 0.5 is not optimal for many labels. Thresholds were tuned on the validation set to maximize F1-score.

Example threshold outcomes (representative):

Label	Optimal Threshold	F1 Score
toxic	0.42	0.74
severe_toxic	0.38	0.55
obscene	0.47	0.78
threat	0.33	0.61
insult	0.46	0.72
identity_hate	0.40	0.67

These results show that rare classes (e.g., *threat*) require lower thresholds to capture more true positives.

5.2.2 Final Metrics (Test Set)

Metric	Value
Macro ROC-AUC	0.983
Macro F1 (Optimized)	0.602
Subset Accuracy	0.47
Hamming Loss	0.062

Interpretation: The model generalizes extremely well, with high separability (ROC-AUC) and strong F1 improvement after threshold tuning.

5.3   Qualitative Evaluation

Example Predictions

Input: "You are such an idiot."
Output: toxic (0.91), insult (0.85)

   Input: "Go die already!"
Output: threat (0.89), toxic (0.84)

   Input: "I don't agree with your analysis."
Output: None
   These examples show that the model distinguishes between harmful phrases and ordinary disagreement.

5.4   Explainability Analysis

Explainability is central to CleanSpeech. SHAP and attention maps provide interpretable insights into why predictions occur.

5.4.1   SHAP Token Contributions

SHAP highlights:

- Positive contributions (red): strongly toxic words, e.g., "idiot", "hate",

- Negative contributions (blue): mitigating words, e.g., "not", "don't".

   This enhances transparency and user trust.

5.4.2   Attention Arch

Our classifier is built on the mDeBERTa-v3 architecture, which uses a disentangled attention mechanism. Unlike standard Transformer attention that

relies on a single shared positional space, mDeBERTa separates content and position representations and learns their interactions through attention. This allows the model to focus more precisely on semantically significant tokens, especially those indicating harmful or abusive intent.

the relationship of the most toxic words to other words(red shows strong relationship and grey shows weak relationship)

Attention Arcs

you constantly prove how worthless you are; every time you open your mouth it's the same stupid nonsense, and honestly i don't know how anyone can tolerate
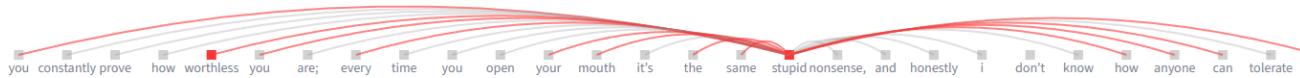
Figure 5.1: attention arch

During inference, attention weights highlight the regions of text most influential for toxicity prediction. Tokens associated with insults, threats, or abusive tone typically receive stronger attention, reflecting the model's reliance on these cues when forming a classification decision

5.5 Rewriting Evaluation

The constructive rewriting module uses Gemini to neutralize toxicity while preserving meaning.

Quantitative Rewrite Evaluation

- BERTScore F1: 0.948 (high semantic similarity between original and rewritten text).

Examples

Toxic Input	Constructive Rewrite
You are such an idiot and I hate you!	I strongly disagree with your approach and find it frustrating.
That guy is a complete moron.	I think that person made poor decisions.

Rewrites maintain intent, remove toxicity, and remain natural.

5.6   Error Analysis

Evaluation reveals several consistent model weaknesses:

Sarcasm & Ambiguity

Sarcastic phrases ("Nice job, genius.") are often misclassified due to implicit toxicity.

Contextual Misinterpretation

Context-specific harmless phrases ("kill shot" in gaming) may be flagged as violent.

Multilingual & Code-Mix Challenges

Hinglish/Hindi slang with cultural nuance is not always correctly interpreted.

Minority-Class Weak Recall

Rare labels (e.g., *identity_hate*) remain harder to detect despite weighted loss.

5.7   Summary

The evaluation shows that CleanSpeech:

- Achieves state-of-the-art transformer performance for toxicity detection,

- Benefits significantly from threshold tuning,

- Provides intuitive explainability through SHAP and attention maps,

- Generates high-quality, semantically preserved rewrites,

- Exhibits limitations mainly in sarcasm, context, and rare label prediction.

These findings confirm the model's suitability for real-world moderation with transparency and constructive feedback capabilities.

Chapter 6

Deployment & Documentation

CleanSpeech is deployed as a modular, scalable system consisting of a back-end inference API, a generative rewriting component, and an interactive user interface. This structure allows independent updates to classification, explainability, and rewriting components without affecting the entire system.

6.1 Deployment Architecture

CleanSpeech is deployed using a three-tier architecture:

1. Backend API (FastAPI),

2. Generative Rewriting Module (Gemini),

3. Streamlit Frontend UI.

The architecture enables a seamless user workflow from text input to explanation and rewrite.

6.2 Backend API

The backend, built in FastAPI, hosts:

- The fine-tuned `mDeBERTa-v3` toxicity classifier,

- SHAP explainability functions,

- Thresholding logic for multi-label classification,

- The rewriting endpoint that interacts with Gemini.

  Key Features

- `POST /predict`: accepts user text $\rightarrow$ returns toxicity scores, predicted labels, SHAP explanations, and rewrite.

- Model loading: loads weights from the `best_model/` directory (locally or from HF Hub).

- Error handling: includes graceful fallbacks for missing inputs, malformed JSON, and Gemini API failures.

6.3   Gemini Rewriting Module

The rewriting component uses Google Gemini for constructive detoxification.
  Responsibilities

- Neutralize toxicity while preserving original meaning,

- Maintain correct grammar, tone, and politeness,

- Provide educational value to the user.

  Safety Handling

- Exceptions handled through `try/except`,

- Rate-limit fallback: returns original text if rewrite fails.

6.4 Streamlit User Interface

The Streamlit UI provides an interactive environment for testing CleanSpeech.

UI Features

- Text Input Panel: enter any free-form text,

- Toxicity Prediction Bars: visual bar chart for each of the six toxicity labels,

- Token-Level Explanation Panel: highlights toxic contributors in red and mitigating tokens in blue,

- Constructive Rewrite Section: shows Gemini-generated detoxified version of the input.
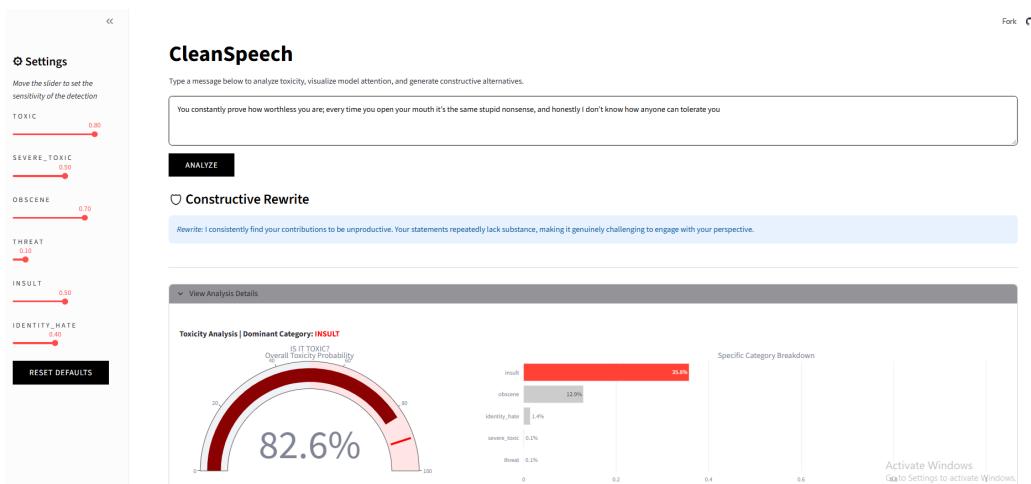

Figure 6.1: streamlit UI

The UI is intentionally simple, making it suitable for moderators, researchers, and non-technical users.

6.5 Environment & Dependencies

The final project environment includes:

- torch, transformers, accelerate — core inference,

- google-generativeai — rewriting,

- fastapi, uvicorn — backend server,

- streamlit — frontend,

- bert-score — rewrite evaluation,

- scikit-learn, numpy, pandas — utilities.

A comprehensive requirements.txt ensures reproducibility across systems.

6.6   Directory Structure

The repository follows a clean and reproducible layout:

CleanSpeech Project Structure

```
CleanSpeech/
|
+-- README.md
+-- requirements.txt
|
+-- doc/
|   +-- milestone-1 ... milestone-5
|
+-- src/
|   +-- data/
|   +-- mdeberta-v3-base/
|   |   +-- code/
|   |   +-- models/
|   |   +-- notebooks-test/
|   |   \-- reports/
|   |
|   +-- model-artifacts/
|   |   \-- tf-idf-log-reg/
|   |
|   \-- tf-idf-logistic-reg/
|
\-- ui/
    \-- Streamlit app
```

This structure supports collaboration, maintenance, and scalability.

6.7 Reproducibility Checklist

To reproduce CleanSpeech:

1. Install dependencies using `pip install -r requirements.txt`,

2. Download and prepare Jigsaw dataset,

3. Run training script/notebook,

4. Load the best model checkpoint,

5. Run evaluation + SHAP explainability notebook,

6. Start FastAPI backend,

7. Launch Streamlit UI.

6.8   Summary

This deployment design makes CleanSpeech:

- Modular and easy to extend,

- Transparent, with full explainability,

- Accessible via an intuitive UI,

- Ready for real-world content moderation workflows.

The system seamlessly integrates advanced ML, XAI, and generative AI into a practical, user-friendly tool.

Chapter 7

Conclusion and Future Work

Conclusion

CleanSpeech delivers a comprehensive, transparent, and constructive approach to online toxicity moderation. By integrating multi-label transformer-based detection, token-level explainability, and meaning-preserving rewriting, the system goes beyond typical moderation tools that simply flag or remove content.

Across the project milestones, CleanSpeech achieved:

- High classification performance — Macro ROC-AUC 0.983,

- Balanced detection using threshold tuning and weighted BCE loss,

- Rich explainability via SHAP and attention visualizations,

- High-quality rewrites with BERTScore F1 0.948,

- Modular deployment with FastAPI + Streamlit,

- A user-friendly workflow for moderators and end-users.

Collectively, these components demonstrate a scalable and educational moderation system that promotes healthier communication rather than punitive censorship.

Limitations

While the system performs strongly, several limitations remain:

1. **Imbalanced Data**

   - Rare labels (e.g., *identity_hate*, *threat*) still show lower recall.

   - Weighted BCE helps, but more advanced strategies are needed.

2. **Token-Level Explainability Constraints**

   - SHAP captures token influence but may miss phrase-level or contextual nuance.

   - Some sarcastic or implicit toxicity tokens remain ambiguous.

3. **Rewrite Tone Issues**

   - Gemini rewrites may sometimes over-soften the text.

   - Strong emotions (frustration, assertiveness) may be unintentionally removed.

4. **Contextual / Domain Limitations**

   - Context-heavy phrases (gaming slang, sarcasm, idioms) may be misinterpreted.

   - Mixed-language slang (Hinglish/Hindi) sometimes lacks correct labeling.

Future Work

To enhance CleanSpeech further, the following directions are recommended:

1. **Advanced Loss Functions**

   - Introduce class-balanced focal loss,

   - Oversampling techniques for rare classes,

   - Augment minority class data using LLM paraphrasing.

2. **Enhanced Explainability**

   - Integrate Integrated Gradients, LRP, or Grad-CAM for NLP,

   - Combine token-level and phrase-level explanations,

   - Improve alignment between SHAP and attention-based interpretations.

3. **Tone-Controlled Rewriting**

   - Allow rewrites that are non-toxic but still retain tone (e.g., "polite but firm", "neutral but assertive", "non-toxic but still frustrated"),

   - Add user-selectable tone controls in UI: Professional, Casual, Assertive, Neutral, etc.

4. **Multilingual Expansion**

   - Fine-tune on additional datasets (HASOC, OLID, multilingual toxicity corpora),

   - Support for Hindi, Hinglish, Tamil, Bengali, and other regional languages,

- Domain-specific vocab handling (gaming, political discourse, etc.).

5. **Real-World Deployment Enhancements**

   - Docker + Kubernetes deployment for large-scale use,

   - GCP/AWS model hosting,

   - Add monitoring via Evidently AI (drift + performance dashboards),

   - Logging for safety and auditability.

6. **UI & Workflow Enhancements**

   - Batch comment moderation,

   - Interactive SHAP visual explorer,

   - Emoji-aware toxicity detection,

   - Browser extension version for real-time filtering.

Final Remarks

CleanSpeech demonstrates how modern NLP can move beyond binary moderation into transparent, instructive, and constructive tools for improving online conversations. The system lays a strong foundation for future research and production-ready moderation systems that emphasize user education, fairness, and interpretability.

Chapter 8

References and Appendix

References

Below is the consolidated bibliography covering datasets, research papers, tools, frameworks, and project-related resources.

Research & Academic Sources

- He, P., Liu, X., Gao, J., & Chen, W. (2021). *DeBERTa: Decoding-enhanced BERT with Disentangled Attention.* arXiv:2006.03654.

- Mathew, B. et al. (2021). *HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection.* AAAI.

- Lundberg, S. & Lee, S. (2017). *A Unified Approach to Interpreting Model Predictions.* NeurIPS.

- Mozafari, M., Farahbakhsh, R., & Crespi, N. (2020). *A BERT-based Transfer Learning Approach for Hate Speech Detection in Social Media.*

- Gemini Team et al. (2025). *Gemini: A Family of Highly Capable Multimodal Models.* arXiv:2312.11805.

- Nogueira dos Santos, C. et al. (2018). *Fighting Offensive Language Using Unsupervised Style Transfer*. ACL.

- Madaan, A. et al. (2020). *Politeness Transfer: A Tag and Generate Approach*. ACL.

Datasets

- Jigsaw Toxic Comment Classification Challenge. Kaggle.

- HASOC Dataset (Hate Speech and Offensive Content). FIRE, 2019–2021.

Tools & Frameworks

- Hugging Face Transformers & Accelerate,

- PyTorch,

- SHAP,

- FastAPI,

- Streamlit,

- Google Generative AI (Gemini),

- scikit-learn, pandas, numpy,

- BERTScore.

Project Documentation & Milestones

- Milestone 1 — Literature Review,

- Milestone 2 — Dataset Preparation,

- Milestone 3 — Architecture Design,

- Milestone 4 — Model Development,

- Milestone 5 — Evaluation & Explainability,

- Milestone 6 — Deployment & Technical Docs.

Appendix

The appendix provides supplementary material for implementation details and reproducibility.

Appendix A — Hyperparameter Summary

Hyperparameter	Value
Base Model	`mDeBERTa-v3-base`
Max Sequence Length	256
Batch Size	16 (per GPU)
Optimizer	AdamW
Learning Rate	2e-5
Weight Decay	0.01
Loss Function	Weighted BCEWithLogitsLoss
Warm-up Ratio	10%
Precision	FP16
Epochs	Early stopped at 3

Appendix B — API Specification

Endpoint:

```
POST /predict
```

Request Example:

```
{
  "text": "Your input comment here"
}
```

Response Example:

```
{
  "probabilities": {...},
  "predicted_labels": {...},
  "severity": "Moderate",
  "explanations": [...],
  "rewrite": "..."
}
```

Appendix C — Streamlit UI Guide

To launch the UI locally:

```
streamlit run ui/app.py
```

UI Modules

- Input textbox,

- Toxicity probability bars,

- Token-level SHAP highlights,

- Constructive rewrite output.

Appendix D — Reproducibility Checklist

To fully reproduce the CleanSpeech system:

1. Create environment using `requirements.txt`,

2. Download & preprocess Jigsaw dataset,

3. Train using the `mDeBERTa` training script,

4. Save best model weights in `best_model/`,

5. Run evaluation and SHAP explainability notebook,

6. Deploy backend with FastAPI,

7. Launch Streamlit UI.

Appendix E — Project Directory Structure

```
CleanSpeech Project Structure

CleanSpeech/
|
+-- README.md
+-- requirements.txt
|
+-- doc/
|   +-- milestone-1 ... milestone-5
|
+-- src/
|   +-- data/
|   +-- mdeberta-v3-base/
|   |   +-- code/
|   |   +-- models/
|   |   +-- notebooks-test/
|   |   \-- reports/
|   |
|   +-- model-artifacts/
|   |   \-- tf-idf-log-reg/
|   |
|   \-- tf-idf-logistic-reg/
|
\-- ui/
    \-- Streamlit app
```