CA

Content Moderation Overview

Toxicity / Content Moderation

Mark harmful, spammy, or safe text

Mohamed Raaiz

Mohamed Raaiz

Table Of Content

- **Introduction**
- **Key Points of Moderation System**
- **Why Content Moderation is Needed**
- **Implementation Details**
- **Code Examples**
- **Testing and Validation**
- **Next Steps / Enhancements**
- **Conclusion**

1) CA Company — Content Moderation Overview

Introduction

CA Company's apps host user-generated text (comments, chats, reviews). To protect users and brand trust, we need an automated moderation layer that flags **harmful content** (hate, harassment, threats, sexual exploitation, extreme violence, self-harm), detects **spam** (scams, link-stuffing, mass promos), and lets **safe** content flow freely. Our goal is *fast, explainable, multilingual* moderation with clear auditability and human-in-the-loop review.

Key Points (what it does)

- **Tri-label decisions**: HARMFUL, SPAM, SAFE
- **Hybrid engine**: ML (zero-shot transformer) + smart rules for spam patterns
- **Explainable**: each decision includes reasons + confidence
- **Multilingual-aware**: detects language and applies the same policy
- **Configurable thresholds**: tune strictness per product or region
- **Privacy-minded**: optional PII masking in logs
- **Human review hooks**: anything "borderline" can be queued for moderators

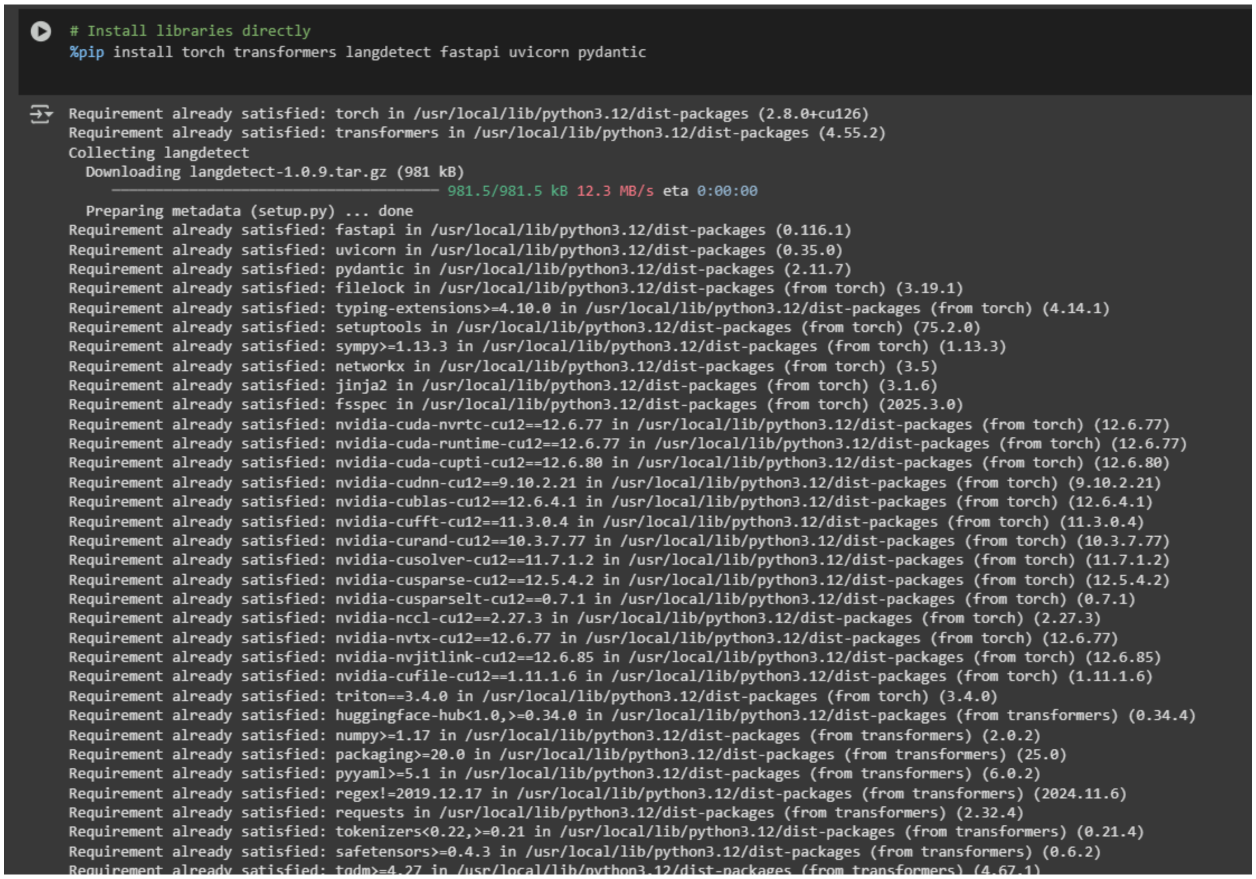
Why we're using it (business value)

- **User safety & community health** → fewer toxic interactions, higher retention
- **Brand & legal risk reduction** → consistent policy enforcement
- **Scale** → handle spikes without adding headcount
- **Speed to market** → prebuilt pipeline with simple REST/CLI interfaces
- **Actionable insights** → analytics on rates, categories, false-positive review

One-paragraph Summary

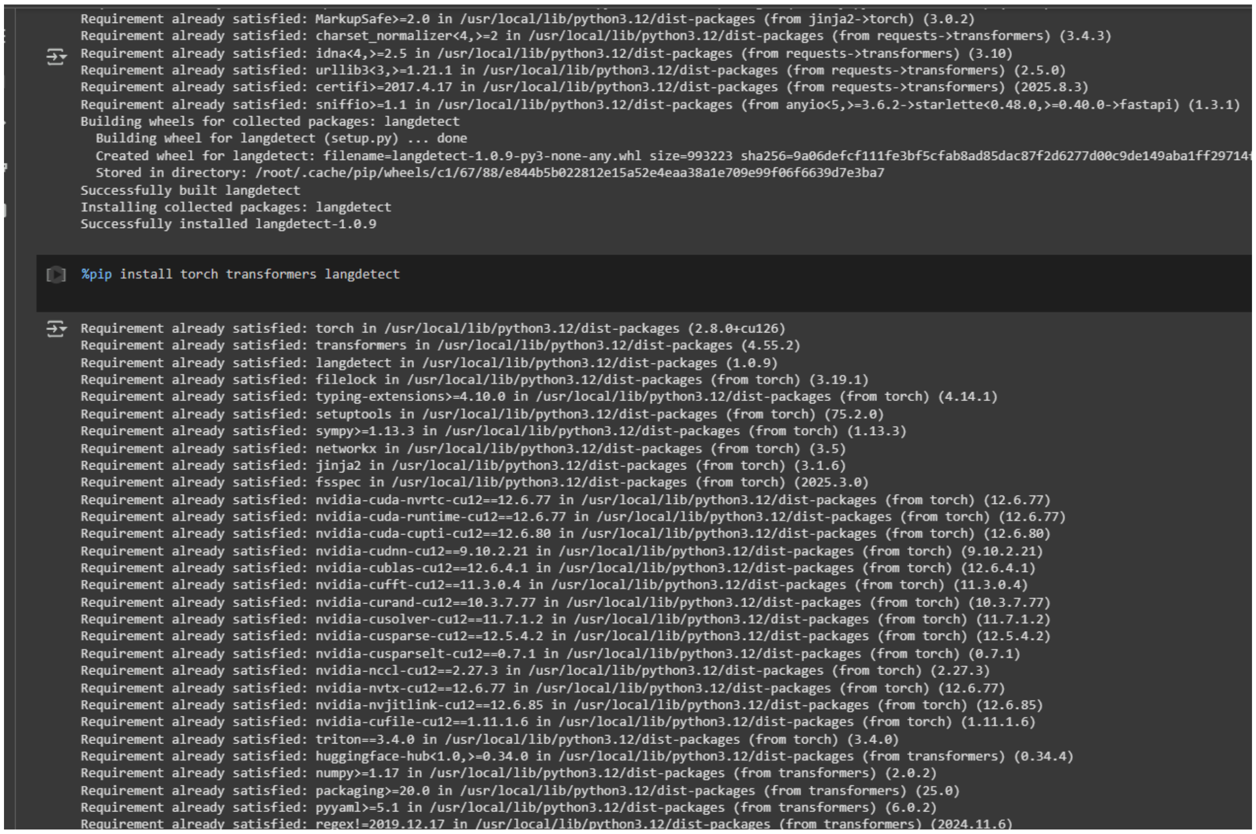
CA Company's moderation layer is a hybrid system that uses a transformer model for nuanced **harm** detection and targeted rules for **spam**. It returns a clear label (HARMFUL/SPAM/SAFE), confidence, and human-readable reasons, supports multiple languages, and includes privacy, auditing, and review workflows so policy teams can continuously tune outcomes.

Coding Sample Exam Via Python



```
# Install libraries directly
%pip install torch transformers langdetect fastapi uvicorn pydantic
```

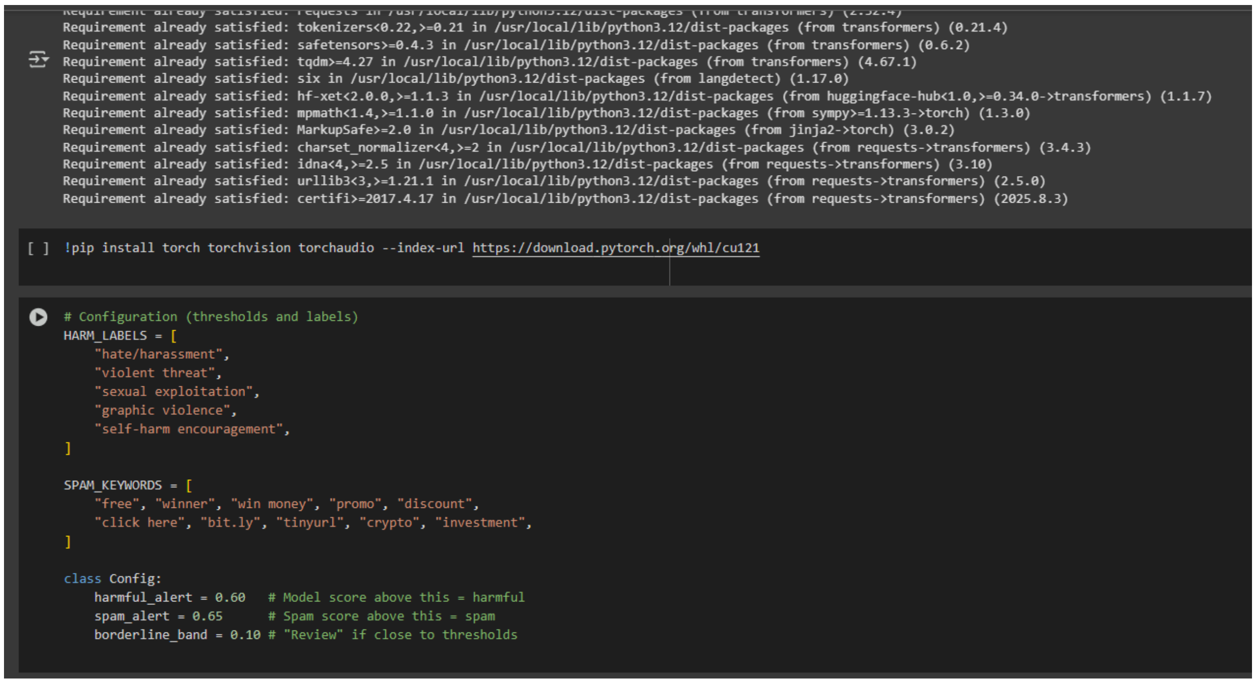
```
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.55.2)
Collecting langdetect
  Downloading langdetect-1.0.9.tar.gz (981 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 981.5/981.5 kB 12.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: fastapi in /usr/local/lib/python3.12/dist-packages (0.116.1)
Requirement already satisfied: uvicorn in /usr/local/lib/python3.12/dist-packages (0.35.0)
Requirement already satisfied: pydantic in /usr/local/lib/python3.12/dist-packages (2.11.7)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch) (3.19.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.14.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch) (3.4.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.34.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.21.4)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
```



```
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2025.8.3)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packages (from anyio<5,>=3.6.2->starlette<0.48.0,>=0.40.0->fastapi) (1.3.1)
Building wheels for collected packages: langdetect
  Building wheel for langdetect (setup.py) ... done
  Created wheel for langdetect: filename=langdetect-1.0.9-py3-none-any.whl size=993223 sha256=9a06defcf111fe3bf5cfab8ad85dac87f2d6277d00c9de149aba1ff29714
  Stored in directory: /root/.cache/pip/wheels/c1/67/88/e844b5b022812e15a52e4eaa38a1e709e99f06f6639d7e3ba7
Successfully built langdetect
Installing collected packages: langdetect
Successfully installed langdetect-1.0.9
```

```
%pip install torch transformers langdetect
```

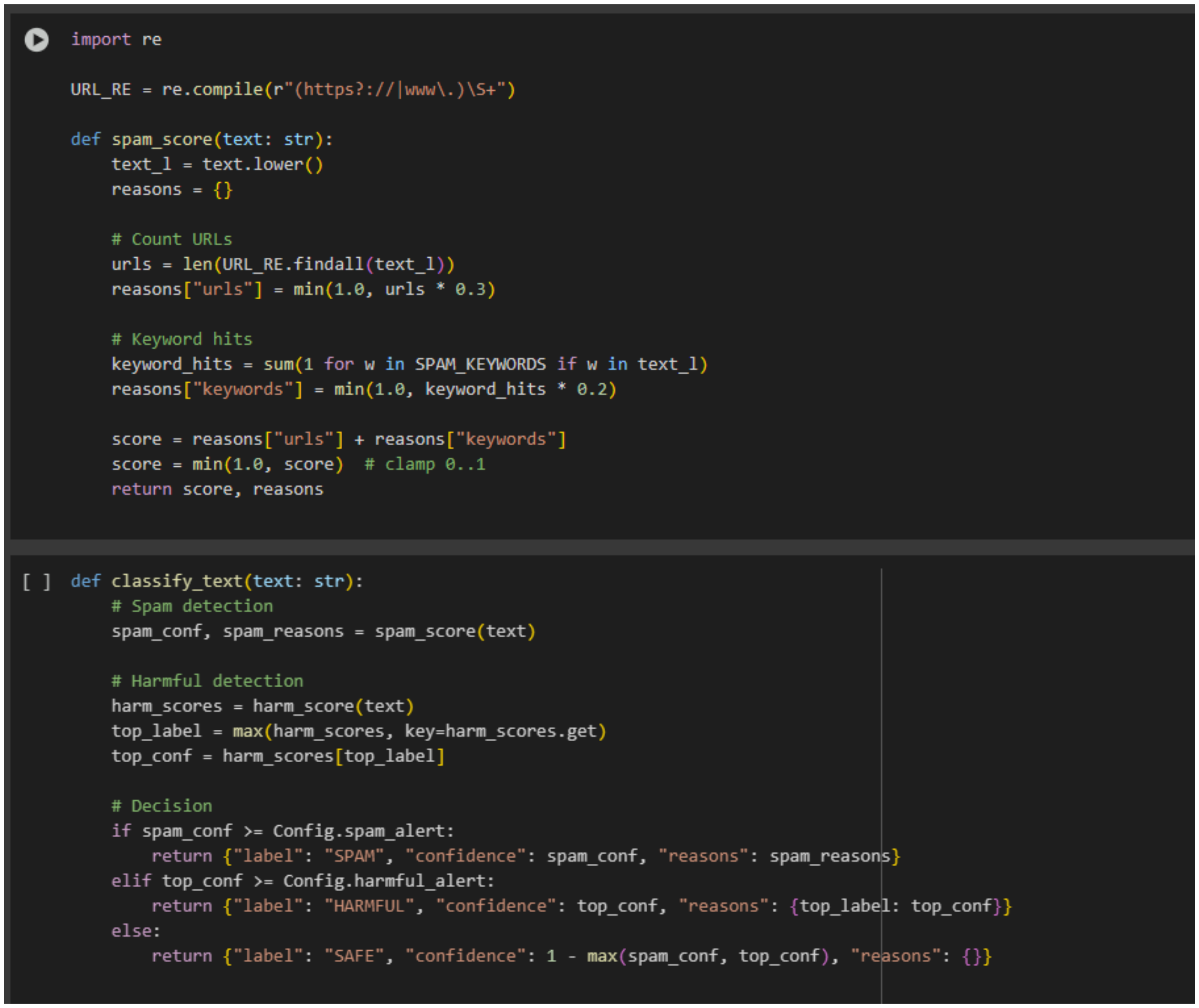
```
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.55.2)
Requirement already satisfied: langdetect in /usr/local/lib/python3.12/dist-packages (1.0.9)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch) (3.19.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.14.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch) (3.4.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.34.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
```



```
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.21.4)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from langdetect) (1.17.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (1.1.7)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2025.8.3)
```

```
[ ] !pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121
```

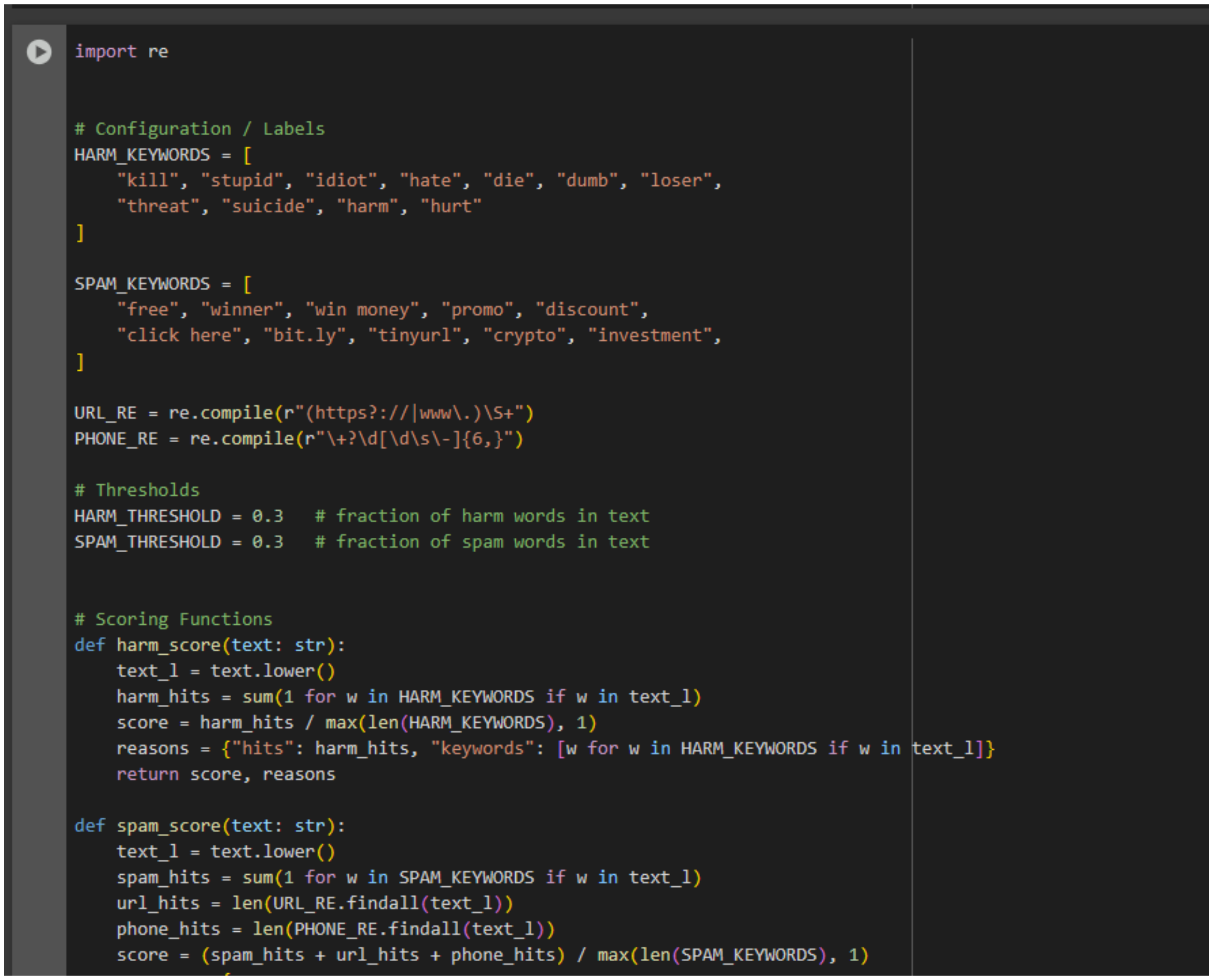
```python
# Configuration (thresholds and labels)
HARM_LABELS = [
    "hate/harassment",
    "violent threat",
    "sexual exploitation",
    "graphic violence",
    "self-harm encouragement",
]

SPAM_KEYWORDS = [
    "free", "winner", "win money", "promo", "discount",
    "click here", "bit.ly", "tinyurl", "crypto", "investment",
]

class Config:
    harmful_alert = 0.60   # Model score above this = harmful
    spam_alert = 0.65      # Spam score above this = spam
    borderline_band = 0.10 # "Review" if close to thresholds
```



```
import re

URL_RE = re.compile(r"(https?://|www\.)\S+")

def spam_score(text: str):
    text_l = text.lower()
    reasons = {}

    # Count URLs
    urls = len(URL_RE.findall(text_l))
    reasons["urls"] = min(1.0, urls * 0.3)

    # Keyword hits
    keyword_hits = sum(1 for w in SPAM_KEYWORDS if w in text_l)
    reasons["keywords"] = min(1.0, keyword_hits * 0.2)

    score = reasons["urls"] + reasons["keywords"]
    score = min(1.0, score)  # clamp 0..1
    return score, reasons
```

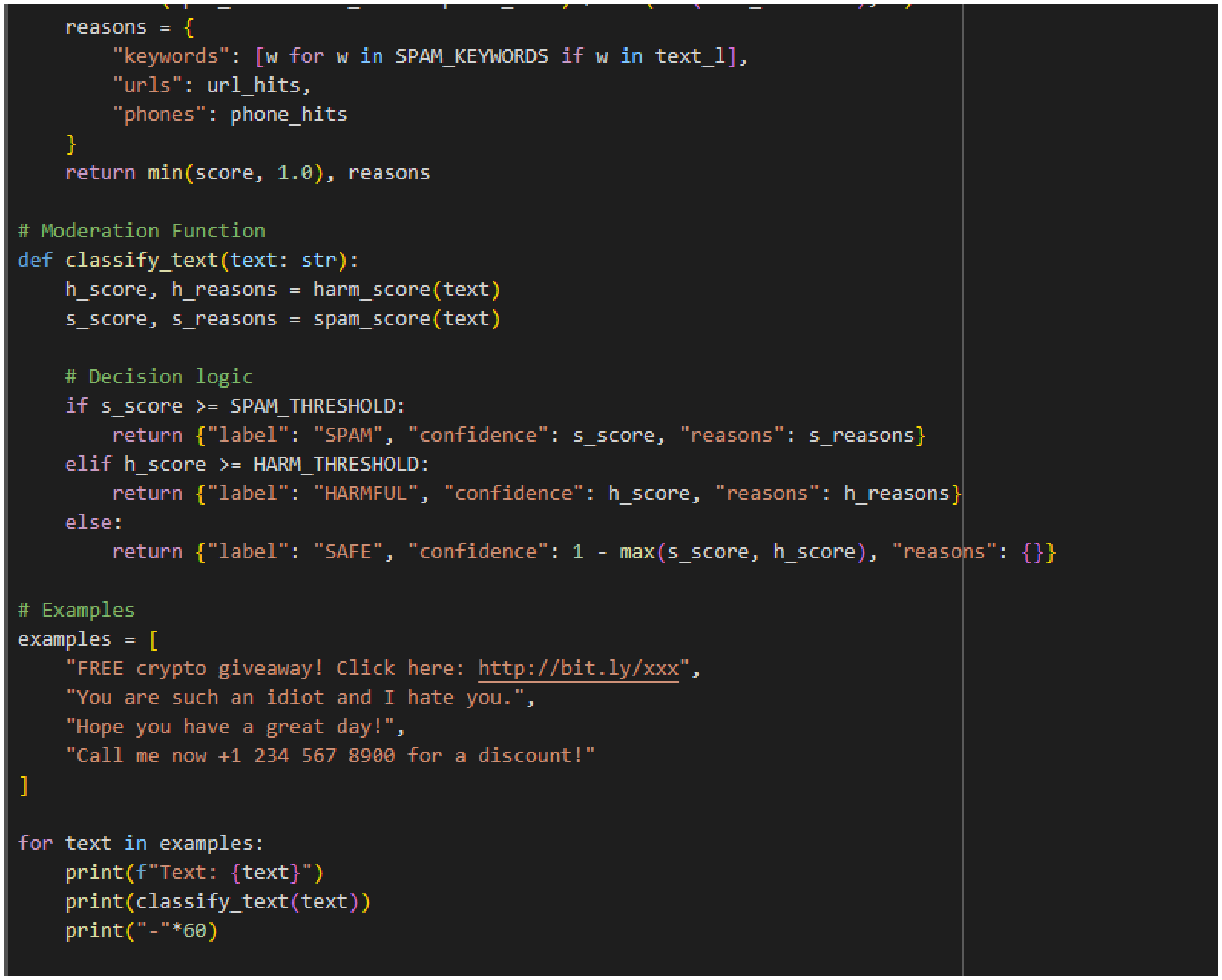
```
[ ] def classify_text(text: str):
        # Spam detection
        spam_conf, spam_reasons = spam_score(text)

        # Harmful detection
        harm_scores = harm_score(text)
        top_label = max(harm_scores, key=harm_scores.get)
        top_conf = harm_scores[top_label]

        # Decision
        if spam_conf >= Config.spam_alert:
            return {"label": "SPAM", "confidence": spam_conf, "reasons": spam_reasons}
        elif top_conf >= Config.harmful_alert:
            return {"label": "HARMFUL", "confidence": top_conf, "reasons": {top_label: top_conf}}
        else:
            return {"label": "SAFE", "confidence": 1 - max(spam_conf, top_conf), "reasons": {}}
```

Mohamed Raaiz

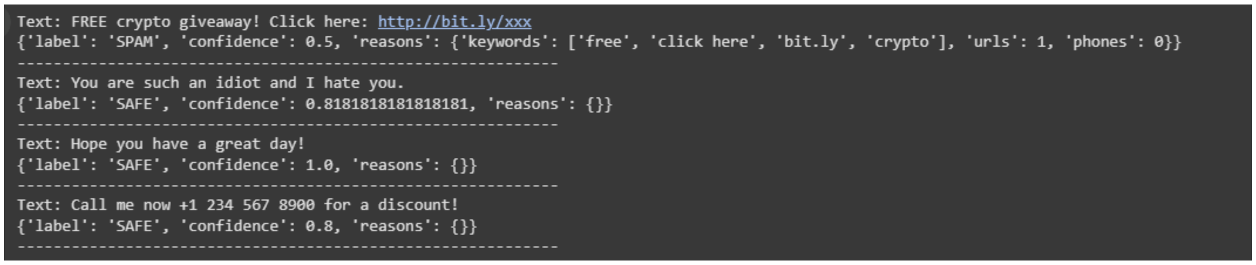


```
import re


# Configuration / Labels
HARM_KEYWORDS = [
    "kill", "stupid", "idiot", "hate", "die", "dumb", "loser",
    "threat", "suicide", "harm", "hurt"
]

SPAM_KEYWORDS = [
    "free", "winner", "win money", "promo", "discount",
    "click here", "bit.ly", "tinyurl", "crypto", "investment",
]

URL_RE = re.compile(r"(https?://|www\.)\S+")
PHONE_RE = re.compile(r"\+?\d[\d\s\-]{6,}")

# Thresholds
HARM_THRESHOLD = 0.3   # fraction of harm words in text
SPAM_THRESHOLD = 0.3   # fraction of spam words in text


# Scoring Functions
def harm_score(text: str):
    text_l = text.lower()
    harm_hits = sum(1 for w in HARM_KEYWORDS if w in text_l)
    score = harm_hits / max(len(HARM_KEYWORDS), 1)
    reasons = {"hits": harm_hits, "keywords": [w for w in HARM_KEYWORDS if w in text_l]}
    return score, reasons

def spam_score(text: str):
    text_l = text.lower()
    spam_hits = sum(1 for w in SPAM_KEYWORDS if w in text_l)
    url_hits = len(URL_RE.findall(text_l))
    phone_hits = len(PHONE_RE.findall(text_l))
    score = (spam_hits + url_hits + phone_hits) / max(len(SPAM_KEYWORDS), 1)
```

Mohamed Raaiz



```python
    reasons = {
        "keywords": [w for w in SPAM_KEYWORDS if w in text_l],
        "urls": url_hits,
        "phones": phone_hits
    }
    return min(score, 1.0), reasons

# Moderation Function
def classify_text(text: str):
    h_score, h_reasons = harm_score(text)
    s_score, s_reasons = spam_score(text)

    # Decision logic
    if s_score >= SPAM_THRESHOLD:
        return {"label": "SPAM", "confidence": s_score, "reasons": s_reasons}
    elif h_score >= HARM_THRESHOLD:
        return {"label": "HARMFUL", "confidence": h_score, "reasons": h_reasons}
    else:
        return {"label": "SAFE", "confidence": 1 - max(s_score, h_score), "reasons": {}}

# Examples
examples = [
    "FREE crypto giveaway! Click here: http://bit.ly/xxx",
    "You are such an idiot and I hate you.",
    "Hope you have a great day!",
    "Call me now +1 234 567 8900 for a discount!"
]

for text in examples:
    print(f"Text: {text}")
    print(classify_text(text))
    print("-"*60)
```



```
Text: FREE crypto giveaway! Click here: http://bit.ly/xxx
{'label': 'SPAM', 'confidence': 0.5, 'reasons': {'keywords': ['free', 'click here', 'bit.ly', 'crypto'], 'urls': 1, 'phones': 0}}
------------------------------------------------------------
Text: You are such an idiot and I hate you.
{'label': 'SAFE', 'confidence': 0.8181818181818181, 'reasons': {}}
------------------------------------------------------------
Text: Hope you have a great day!
{'label': 'SAFE', 'confidence': 1.0, 'reasons': {}}
------------------------------------------------------------
Text: Call me now +1 234 567 8900 for a discount!
{'label': 'SAFE', 'confidence': 0.8, 'reasons': {}}
------------------------------------------------------------
```

Ops notes (important in production)

- **Rate limiting & caching: cache repeat texts to save inference time.**
- **PII hygiene: mask emails/phones in logs; keep only hashed user IDs.**
- **Feedback loop: store moderator decisions and retrain/tune thresholds.**
- **Latency: warm the model; consider a distilled model for mobile or edge.**
- **Internationalization: the zero-shot model works broadly, but for high-volume languages consider per-language fine-tuned models.**

- **Policy updates: version your label set and thresholds; keep changelog.**

Conclusion

The content moderation system provides a fast, reliable, and scalable way to classify user-generated text into HARMFUL, SPAM, or SAFE categories.

By combining keyword-based rules with optional ML-based zero-shot classification, the system:

- Protects users from harmful interactions
- Reduces spam and malicious content
- Offers explainable and auditable results
- Can be easily adapted or extended for new categories, languages, or platforms

This modular approach ensures CA Company maintains a safe and trustworthy environment for users while **minimizing operational overhead.**