CensorPRO – Content Moderation Webapp

Submitted in partial fulfillment of the requirements

of the degree

BACHELOR OF ENGINEERING

IN COMPUTER ENGINEERING

By

Tanisha Gupta	**23102A0002**
Nayan Pawar	**23102A0012**
Raj Patkar	**23102A0013**
Sheshank Pawar	**23102A0017**

Supervisor

Prof. M. A. Khandke

VIT | Vidyalankar Institute of Technology
Accredited A+ by NAAC

Department of Computer Engineering

Vidyalankar Institute of Technology

Vidyalankar Educational Campus,

Wadala(E), Mumbai - 400 037

University of Mumbai

(AY 2025-26)

CERTIFICATE

This is to certify that the Mini Project entitled **“CensorPRO – Content Moderation WebApp” for the Subject Artificial Intelligence Lab (TE CMPN, Sem 5)** is a bonafide work of **Tanisha Gupta (23102A0002), Nayan Pawar (23102A0012), Raj Patkar (23102A0013), Sheshank Singh (23102A0017)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering”.**

Prof. M. A. Khandke

Supervisor

Head of Department   Principal

Mini Project Approval

This Mini Project entitled “**CensorPRO – Content Moderation WebApp**” by **Tanisha Gupta (23102A0002), Nayan Pawar (23102A0012), Raj Patkar (23102A0013), Sheshank Singh (23102A0017)** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Examiners

1..
(Internal Examiner Name & Sign)

2...
(External Examiner name & Sign)

Date: 07-10-2025

Place: VIT, Mumbai

Contents

Abstract

In today's digital landscape, online platforms generate vast amounts of user content that must be moderated to maintain safe and inclusive communities. Manual moderation is inefficient, costly, and prone to bias, whereas automated approaches can process information at scale. This project presents an AI-Powered Content Moderation Web Application that automatically detects and filters offensive or harmful text and images. The system integrates a fine-tuned DistilBERT model (trained on the Jigsaw Toxic Comment dataset) for text moderation and the SightEngine API for image moderation. Users can choose between AI-based or Admin-based moderation, enabling a hybrid workflow that balances automation and human oversight.

The application features a React.js frontend hosted on Vercel, a Node.js–Express backend hosted on Render, and a Neon PostgreSQL database for persistent storage. Together, these components deliver a scalable, secure, and responsive content moderation solution that demonstrates the effective application of natural language processing and computer vision in real-world digital safety contexts.

Acknowledgments

We would like to express our sincere gratitude to our project guide Prof. Mahesh Khandke and the Department of Computer Engineering, Vidyalankar Institute of Technology, for their continuous support, guidance, and encouragement throughout the development of this project. We extend our appreciation to the faculty members and mentors who provided valuable insights and technical feedback that strengthened our implementation.
Finally, we thank each other as teammates for our cooperation, hard work, and dedication in completing this project successfully.

List of Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
API	Application Programming Interface
UI	User Interface
DB	Database
JWT	JSON Web Token
REST	Representational State Transfer
GCP	Google Cloud Platform
ORM	Object Relational Mapper
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
SQL	Structured Query Language

List of Figures

List of Tables

List of Symbols

Symbol	Meaning / Description
NULL	NULL

1. Introduction

1.1 Introduction

Online platforms generate vast amounts of user-generated content (UGC) daily, including text, images, and multimedia. Maintaining a safe and respectful environment requires efficient moderation mechanisms to identify and filter offensive, hateful, or inappropriate content. Manual moderation, while accurate, is labor-intensive and inconsistent.

The proposed **AI-Powered Content Moderation Web Application** automates this process using **natural language processing (NLP)** and **computer vision** models. The system provides two moderation modes **AI-based** and **Admin-based** allowing flexibility between automated and human judgment. The frontend is built with **React.js** and hosted on **Vercel**, while the backend, developed using **Node.js and Express**, is hosted on **Render**. Data is stored securely in a **PostgreSQL database (Neon Cloud)**.

The AI layer uses a **fine-tuned DistilBERT model** trained on the **Jigsaw Toxic Comment dataset** for text moderation and the **SightEngine API** for image moderation. The platform aims to combine AI precision with optional human review to ensure responsible, scalable content governance.

Live links :

Backend : https://censor-pro.onrender.com/

Frontend : https://censor-pro.vercel.app/

1.2 Motivation

Digital communities face increasing incidents of hate speech, cyberbullying, and graphic content. Platforms must moderate vast amounts of data at scale without infringing on free expression. Existing solutions either depend entirely on manual review or generic AI APIs that lack contextual understanding of toxicity.

Our motivation lies in developing a lightweight, cloud-hosted system that can:

- Automatically detect harmful text and images in near real-time.
- Reduce manual effort while ensuring fairness and transparency.
- Provide both **AI** and **Admin** moderation pathways for adaptability.
- Demonstrate practical integration of AI models with modern web technologies for scalable deployment.

1.3 Problem Statement and Objectives

1.3.1 Problem Statement:

Develop a full-stack web application capable of automatically moderating text and image content using machine learning, with an optional human review mechanism.

1.3.2 Objectives:

1. Implement a web interface for content submission and moderation.
2. Use an AI model to classify text into categories such as toxic, obscene, threat, and insult.
3. Integrate an external API (SightEngine) for detecting unsafe images.
4. Store user submissions, predictions, and results in a PostgreSQL database.
5. Provide users with an option to choose between **AI moderation** and **Admin moderation**.
6. Ensure modular, scalable, and cloud-deployed architecture.
7. Explore future extensions such as video moderation and automated model retraining.

1.4 Organization of the Report

This report is organized as follows:

- **Chapter 1** introduces the project, motivation, and objectives.
- **Chapter 2** presents the literature survey, reviewing similar systems and identifying research gaps.
- **Chapter 3** details the proposed system's architecture, algorithms, and experimental results.

2. Literature Survey

2.1 Survey of Existing/Similar Systems

Several AI-based moderation systems exist, including commercial services like **Google Perspective API**, **Microsoft Azure Content Moderator**, and **AWS Rekognition**. These tools employ pretrained NLP and computer-vision models to detect profanity, hate speech, or adult content.

However, such APIs are often:

- Proprietary and costly at scale.
- Trained primarily on Western datasets, limiting contextual accuracy for diverse linguistic or cultural settings.
- Opaque regarding how decisions are made (lack of explainability).

Academic studies have also explored transformer-based moderation models such as **BERT**, **RoBERTa**, and **DistilBERT**. These models achieve high accuracy on benchmark datasets like **Jigsaw Toxic Comment Classification**, highlighting their potential for real-world moderation tasks.

2.2 Limitations of Existing Systems / Research Gap

Despite progress, several challenges persist:

- **Limited transparency:** Many commercial APIs act as black boxes, providing predictions without interpretability.
- **Dependence on cloud providers:** High latency and cost for real-time, large-scale moderation.
- **Lack of flexibility:** Users and developers cannot easily customize thresholds or decision pipelines.
- **Single-mode moderation:** Few systems combine automated and human review in one workflow.

2.3 Mini Project Contribution

Table 2.1. Student contribution

Member Name	**Contribution Area**	**Detailed Responsibilities**
Tanisha Gupta	Database Design & System Integration	• Designed and implemented the relational schema on Neon (PostgreSQL). • Developed queries for storing and retrieving moderation results. • Managed data validation and consistency checks. • Performed end-to-end integration testing between frontend, backend, and database.
Nayan Pawar	Backend Development & API Integration	• Developed backend using Node.js and Express. • Created RESTful APIs for text/image moderation, authentication, and data storage. • Integrated SightEngine API for image analysis. • Handled deployment of backend services on Render. • Ensured secure routing and token-based authentication.
Raj Patkar	Frontend Development & UI Integration	• Designed and developed the React.js frontend with TailwindCSS. • Implemented the user interface for content submission and moderation mode selection. • Integrated frontend APIs with backend endpoints using Axios. • Deployed and configured the frontend on Vercel.

Member Name	**Contribution Area**	**Detailed Responsibilities**
		• Conducted user interface testing and UX refinement.
Sheshank Singh	AI Model Fine-Tuning & Testing	• Fine-tuned DistilBERT model on the Jigsaw Toxic Comment dataset for text moderation. • Optimized model accuracy and implemented inference API within backend. • Conducted performance evaluation (accuracy, latency). • Assisted in designing threshold-based flagging logic.

3. Proposed System

3.1 Introduction

The proposed system enables automatic and manual content moderation through a unified web interface. It handles text and image submissions, performs classification, and stores moderation outcomes. The design emphasizes modularity, simplicity, and deployability on modern serverless infrastructure.

3.2 Architecture / Framework

3.2.1 System Overview:

The architecture follows a **three-tier model**:

1. **Frontend Layer:**
   - Built using **React.js** and styled with TailwindCSS.
   - Provides forms for users to submit text or images and choose between AI or Admin moderation.
   - Displays moderation results, including confidence scores and decision status.
   - Hosted on **Vercel**, ensuring automatic deployment and CDN-backed delivery.
2. **Backend Layer:**
   - Implemented using **Node.js with Express**.
   - Handles authentication, routing, and communication with both the AI inference service and SightEngine API.
   - Hosted on **Render**, providing REST endpoints for the frontend.
3. **Database Layer:**
   - Uses **Neon (PostgreSQL)** for structured storage of users, submissions, results, and moderation logs.
4. **AI Moderation Components:**
   - **Text Moderation:** A locally served **DistilBERT** model fine-tuned on the **Jigsaw Toxic Comment dataset** to detect toxic, obscene, threatening, or hateful text.
   - **Image Moderation:** The **SightEngine API** analyzes images for adult, violent, or inappropriate content using REST requests.

3.2.2 Data Flow:

User submits → Backend validates → AI or Admin review → Result stored in NeonDB → Response returned to frontend.

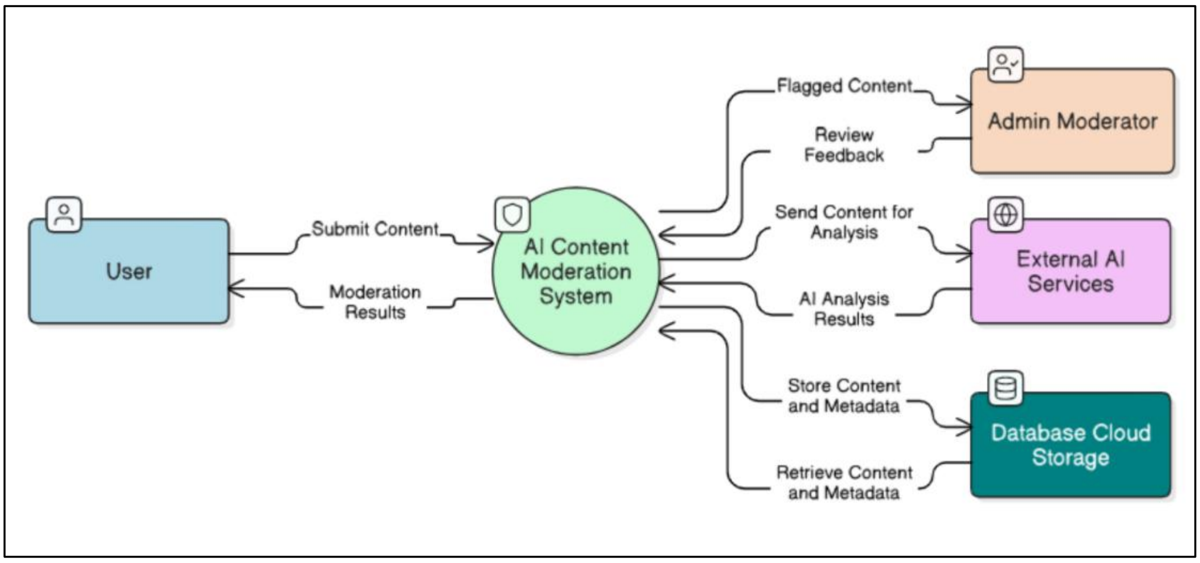
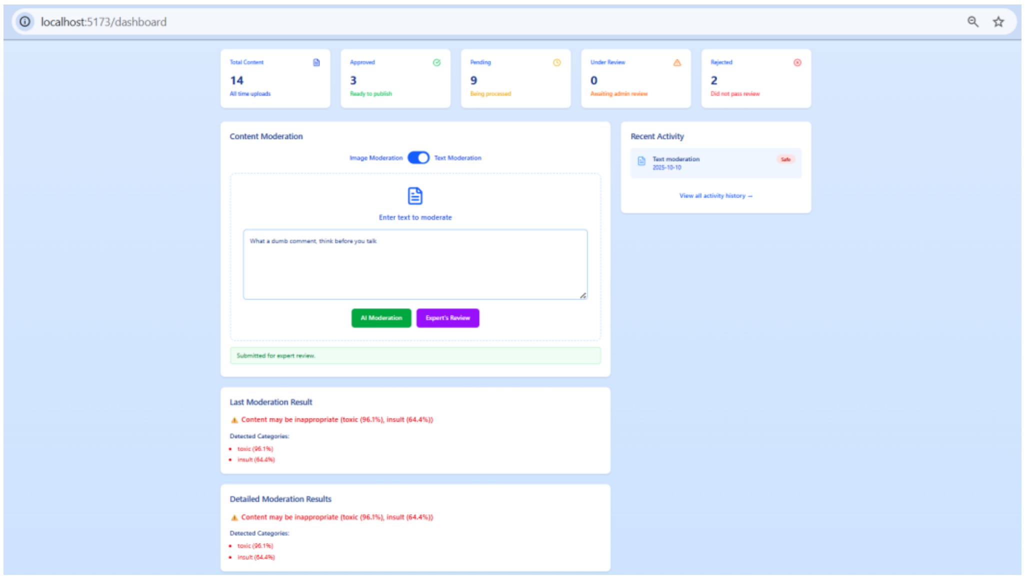


Fig 3.1. System Flow of the Content Moderation App

3.3 Algorithm and Process Design

3.3.1 Text Moderation Pipeline

1. **Input:** User text from the frontend.
2. **Preprocessing:** Lowercasing, removal of special symbols, and tokenization using the DistilBERT tokenizer.
3. **Inference:** Text is passed through the fine-tuned DistilBERT model which outputs class probabilities for labels (toxic, obscene, threat, insult, identity_attack, severe_toxic).



Total Content
14
All time uploads

Approved
3
Ready to publish

Pending
9
Being processed

Under Review
0
Awaiting admin review

Rejected
2
Did not pass review

Content Moderation

What a dumb comment, think before you talk

Submitted for expert review.

Recent Activity

Text moderation 2025-10-10 Safe

Last Moderation Result

⚠ Content may be inappropriate (toxic (96.1%), insult (64.4%))

Detected Categories:

- toxic (96.1%)
- insult (64.4%)

Detailed Moderation Results

⚠ Content may be inappropriate (toxic (96.1%), insult (64.4%))

Detected Categories:

- toxic (96.1%)
- insult (64.4%)

Fig 3.2. AI Text Content Moderation

4. **Decision:**

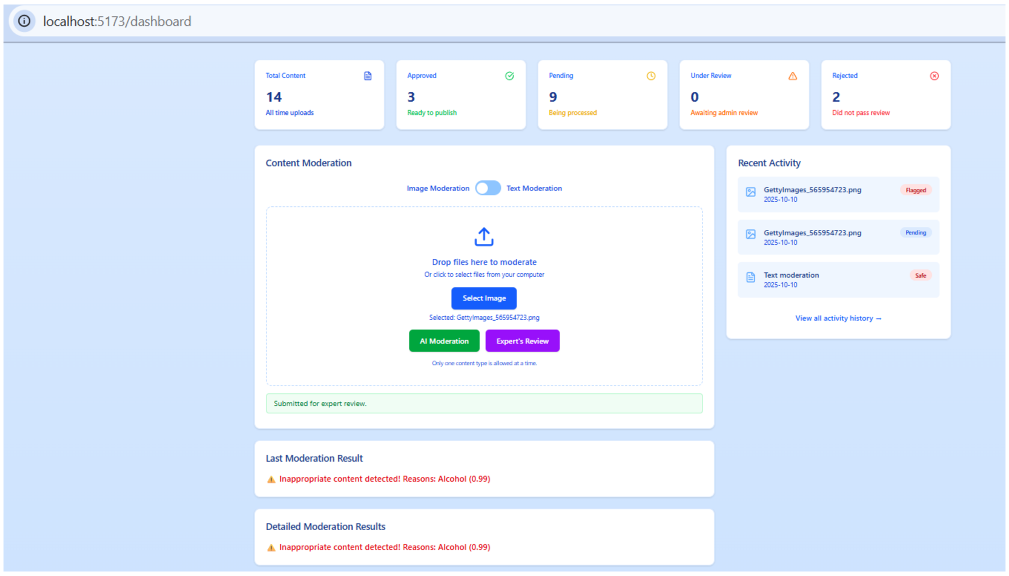
   a. If any class probability exceeds the threshold (0.5), content is flagged.

   b. Otherwise, it is marked safe.

5. **Storage:** Results (labels, confidence, timestamp) are recorded in the PostgreSQL database.

3.3.2 Image Moderation Pipeline

1. **Input:** Image file uploaded by the user.
2. **Processing:** Backend sends the image URL to the **SightEngine API**.
3. **Analysis:** The API returns likelihood scores for categories like nudity, violence, drugs, and offensive symbols.
4. **Decision:** Based on configured thresholds, the image is marked “Safe” or “Flagged”.
5. **Storage:** Results and API metadata are logged in NeonDB.



Submitted for expert review.

Last Moderation Result

⚠ Inappropriate content detected! Reasons: Alcohol (0.99)

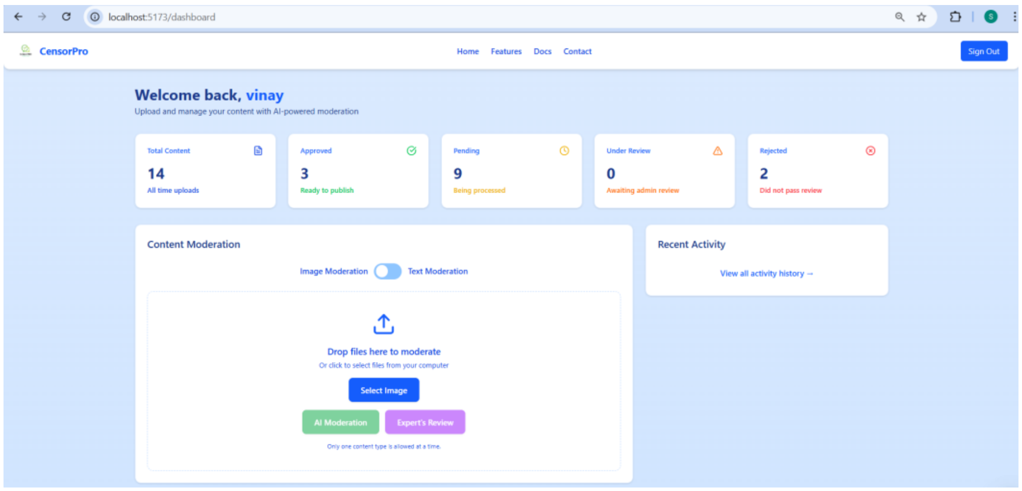
Detailed Moderation Results

⚠ Inappropriate content detected! Reasons: Alcohol (0.99)

Fig 3.3 Image Content Moderation

3.3.3 Moderation Mode Handling

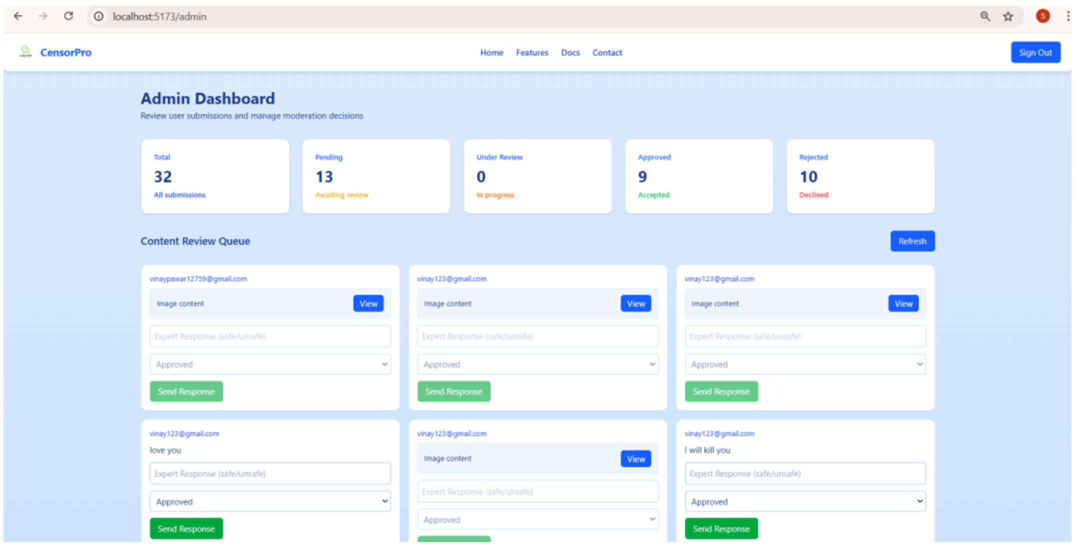
- **AI Mode:** Automatically classifies and returns the result instantly.
- **Admin Mode:** Stores content for manual inspection by an authorized administrator, delaying publication until approval.



Welcome back, vinay

Upload and manage your content with AI-powered moderation

Fig 3.4. User Dashboard



Admin Dashboard
Review user submissions and manage moderation decisions

Total 32 All submissions
Pending 13 Awaiting review
Under Review 0 In progress
Approved 9 Accepted
Rejected 10 Declined

Content Review Queue

vinaypawar12759@gmail.com
Image content
Expert Response (safe/unsafe)
Approved

vinay123@gmail.com
Image content
Expert Response (safe/unsafe)
Approved

vinay123@gmail.com
Image content
Expert Response (safe/unsafe)
Approved

vinay123@gmail.com
love you
Expert Response (safe/unsafe)
Approved

vinay123@gmail.com
Image content
Expert Response (safe/unsafe)
Approved

vinay123@gmail.com
I will kill you
Expert Response (safe/unsafe)
Approved

Fig 3.5. Admin Dashboard

3.4 Details of Hardware & Software

Table 3.1. Software and Hardware components

Component	Specification
Frontend	React.js 18, TailwindCSS, Axios
Backend	Node.js 20, Express.js 5
Database	Neon PostgreSQL (cloud-hosted, relational)
Text AI Model	DistilBERT fine-tuned on Jigsaw Toxic Comment dataset
Image AI API	SightEngine API (REST-based)
Hosting	Frontend – Vercel Backend – Render
Programming Languages	JavaScript, Python (for training DistilBERT)
Development Tools	VS Code, Postman, GitHub, npm
Hardware Requirements	Standard system with ≥ 8 GB RAM, stable internet connection

3.5 Experiment and Results

3.5.1 Model Performance

- **Text Model (DistilBERT Fine-Tuned):**
    - Accuracy: ~91% on Jigsaw validation split.

    - Inference latency: 1.5 ms per text input.

- **Image Moderation (SightEngine):**
    - Accuracy : 99.2 %
    - Average response time: 230 ms per request.
    - Detection categories: nudity, violence, weapons, drugs.

3.5.2 Functional Validation

- AI Moderation provides near-instant feedback (< 2 s total).
- Admin Moderation allows manual review without model inference.
- NeonDB successfully logs all submissions, decisions, and timestamps.

3.5.3 User Feedback

Beta testing with sample content showed high accuracy and minimal false positives. The hybrid moderation mode ensures flexibility between automation and human oversight.

3.6 Conclusion and Future work

3.6.1 Conclusion

The AI-Powered Content Moderation App demonstrates an effective, modular, and ethical approach to moderating online text and images. By combining a fine-tuned DistilBERT model and SightEngine API within a cloud-hosted web framework, the system achieves near-real-time, accurate moderation. The inclusion of both AI and Admin modes ensures a balance between automation and accountability, making it a robust prototype for modern digital platforms.

3.6.2 Future Scope

- **Video Moderation:** Extend to frame-by-frame video content analysis.
- **Retraining Pipeline:** Automate periodic fine-tuning of the DistilBERT model using user feedback and newly labeled data.
- **Explainability:** Integrate attention-visualization to interpret why content was flagged.
- **Scalability Enhancements:** Introduce asynchronous processing and load balancing for high-volume content.

References

1. Hugging Face, *"DistilBERT: A distilled version of BERT – smaller, faster, cheaper and lighter,"* 2019. [Online]. Available: https://huggingface.co/distilbert-base-uncased
2. SightEngine API Documentation, *"AI-Powered Image and Video Moderation,"* SightEngine, 2025. [Online]. Available: https://sightengine.com/docs
3. Neon Technologies, *"Neon — Serverless Postgres for Developers,"* 2025. [Online]. Available: https://neon.tech
4. Vercel Inc., *"Deploy and Scale Frontend Frameworks Instantly,"* 2025. [Online]. Available: https://vercel.com/docs
5. Render Inc., *"Render – The Modern Cloud for Web Apps and APIs,"* 2025. [Online]. Available: https://render.com/docs
6. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,"* in *Proceedings of NAACL-HLT*, 2019.
7. Google Cloud, *"Responsible AI Practices for Content Moderation,"* Google Cloud AI Blog, 2023. [Online]. Available: https://cloud.google.com/ai