Some Personal Information about Me:

Team Name:	DL Team at ITSOLERA PVT LTD
Project Title:	AI-Powered Content Moderation System
Submitted By:	Muhammad Maaz
City:	Peshawar
Date:	25June, 2025
Education:	BS in Computer Science
Institution:	Government Superior Science College, Peshawar
Affiliated With:	University of Peshawar

Title: AI-Powered Content Moderation System Project Lead: DL Team at ITSOLERA PVT LTD

Project Proposal

1. Introduction

In today's digital age, managing user-generated content is vital to protect platforms and their communities from harmful, abusive, or explicit material. As content volumes increase rapidly, manual moderation becomes inefficient and error-prone. This project proposes the development of an AI-powered content moderation system that uses deep learning to automatically analyze, filter, and report inappropriate content in real-time.

The system will moderate both text and image content, using Natural Language Processing (NLP) and Convolutional Neural Networks (CNN), respectively. It also offers platform-specific customization, real-time performance, and a user-friendly web interface.

---

2. Objectives

- Develop a deep learning system for text and image moderation.
- Implement real-time filtering and reporting features.
- Provide customizable moderation policies tailored to platform-specific needs.
- Ensure high accuracy and efficiency in content detection.

---

3. Scope of Work

Phase 1: Requirement Analysis and Planning

- Define types of inappropriate content.

- Identify platform-specific moderation policies.
- Develop a project roadmap and milestone schedule.

Phase 2: System Design

- Design modular architecture for text and image moderation pipelines.
- Create a framework for easily updating moderation policies.

Phase 3: Development

- Implement NLP-based text moderation using models like BERT.
- Implement CNN-based image moderation using models like ResNet.
- Build a real-time web interface using Flask.

Phase 4: Testing and Validation

- Test model performance using precision, recall, and F1-score.
- Collect feedback from test users and platform administrators.

Phase 5: Deployment and Maintenance

- Deploy the system to target platforms.
- Monitor performance and integrate updates as required.

---

4. Methodology and Tools

Languages and Frameworks: Python, Flask
Deep Learning Libraries: TensorFlow, PyTorch
NLP Tools: SpaCy, NLTK, Hugging Face Transformers
Image Processing Tools: OpenCV, Pillow
Frontend Technologies: HTML, CSS, JavaScript
Deployment Tools (optional): Docker, Heroku, or AWS for cloud deployment

---

5. Proposed Features

- Real-time text moderation using a BERT-based classifier.
- Real-time image moderation using ResNet-based classifier.
- Admin dashboard for customizing moderation thresholds.
- Automatic flagging and reporting of harmful content.
- Clean, intuitive web interface with input validation.

---

6. Expected Outcomes

- An end-to-end AI system capable of detecting and moderating harmful content.
- Reduction in manual moderation workload.
- Increased platform safety and compliance with content policies.

---

7. Timeline Overview

Phase	Description	Duration
1	Requirements & Planning	1 week
2	Architecture & Policy Design	1 week
3	Core Development (Text + Image Modules)	2–3 weeks
4	Testing & User Feedback	1 week
5	Deployment & Maintenance Setup	1 week

Source Code Implementation

1. Folder Structure

```
content_moderation/
│
├── app.py
├── text_moderation.py
├── image_moderation.py
├── requirements.txt
├── templates/
│   └── index.html
├── static/
│   ├── style.css
│   └── script.js
├── uploads/  # Directory for uploaded images
```

---

2. Python Code

text_moderation.py

```
from transformers import pipeline

from PIL import Image  # For image handling

# === Text Moderation Pipeline ===

text_moderator = pipeline("text-classification", model="unitary/toxic-bert", return_all_scores=True)


def moderate_text(text):

    """
```

```
    Analyze text for multiple types of toxicity or inappropriate
content.
    Returns a list of dictionaries like:
    [
        {'label': 'toxic', 'score': 0.98},
        {'label': 'insult', 'score': 0.67},
        {'label': 'obscene', 'score': 0.45},
        ...
    ]
    """
    results = text_moderator(text)[0]  # Only one text input
    sorted_results = sorted(results, key=lambda x: x['score'],
reverse=True)
    return sorted_results
```

---

image_moderation.py

```
from torchvision import models, transforms
from PIL import Image
import torch
from transformers import pipeline
model = models.resnet18(pretrained=True)
model.eval()
labels = ['safe', 'nsfw']  # Example labels
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])
```

```
def moderate_image(image_path):
    image = Image.open(image_path).convert("RGB")
    img_tensor = transform(image).unsqueeze(0)
        with torch.no_grad():
        output = model(img_tensor)
        prediction = torch.argmax(output, 1)
    return labels[prediction.item() % 2]  # Dummy return label
# === Image Moderation Model Initialization ===
try:
    image_moderator = pipeline("image-classification",
model="nateraw/vit-base-patch16-224-in21k-finetuned-nsfw")
except Exception as e:
    print("❌ Error loading image moderation model:", e)
    image_moderator = None
def moderate_image(image_path):
    """
    Analyze image content for NSFW. Returns the top prediction with
explanation.
    """
    if not image_moderator:
        return {"error": "Image moderation model not available."}
    image = Image.open(image_path).convert("RGB")
    results = image_moderator(image)
    # Sort results by score (high to low)
    results = sorted(results, key=lambda x: x['score'], reverse=True)
    top_label = results[0]['label'].lower()
    top_score = results[0]['score']
```

```
    # Debug: print top 3 labels
    print("🔍 Image Classification Results:")
    for res in results[:3]:
        print(f"{res['label']}: {round(res['score'], 2)}")
    # Define safe labels more clearly
    safe_labels = ["neutral", "drawings", "hentai", "neutral", "sexy"]
    nsfw_labels = ["porn", "nsfw"]
    if top_label in nsfw_labels:
        return f"⚠️ NSFW (Confidence: {round(top_score, 2)})"
    elif top_label in safe_labels:
        return f"✅ SAFE (Label: {top_label}, Score: {round(top_score,
2)})"
    else:
        return f"⚠️ Uncertain (Label: {top_label}, Score:
{round(top_score, 2)})"
```

---

app.py

```
from flask import Flask, render_template, request
from text_moderation import moderate_text
from Image_moderation import moderate_image
import os
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
@app.route('/', methods=['GET', 'POST'])
def index():
    text_result = None
```

```
    image_result = None
    if request.method == 'POST':
        user_text = request.form.get('user_text')
        if user_text:
            text_result = moderate_text(user_text)
        file = request.files.get('image')
        if file:
            path = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(path)
            image_result = moderate_image(path)
    return render_template("index.html", text_result=text_result,
image_result=image_result)
if __name__ == '__main__':
    app.run(debug=True)
```

---

3. HTML, CSS, and JavaScript

templates/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>AI Content Moderation</title>
    <link rel="stylesheet" href="/static/style.css">
    <script src="/static/script.js" defer></script>
</head>
<body>
    <div class="container">
        <h1>🛡️ AI Content Moderation</h1>
```

```
        <form method="POST" enctype="multipart/form-data"
class="moderation-form">
            <label for="text-input">✍️ Enter Text:</label>
            <textarea name="user_text" rows="5" cols="50" id="text-
input" placeholder="Write your content here..."></textarea>
            <label for="image-input">🖼️ Upload Image:</label>
            <input type="file" name="image" id="image-input">
            <input type="submit" value="🚀 Moderate Content">
        </form>


        {% if text_result %}
        <div class="result-box">
            <h3>📝 Text Result:</h3>
            <ul>
                {% for item in text_result %}
     <li><strong>{{ item.label }}</strong>: {{ item.score | round(2)
}}</li>
                {% endfor %}
            </ul>
        </div>
        {% endif %}
        {% if image_result %}
        <div class="result-box">
            <h3>🖼️ Image Result:</h3>
            <p>
             {% if image_result is iterable and image_result is not
string %}
                    <ul>
```

```
                        {% for item in image_result %}
     <li><strong>{{ item.label }}</strong>: {{ item.score | round(2)
}}</li>
                        {% endfor %}
                    </ul>
                {% else %}
                    {{ image_result }}
                {% endif %}
            </p>
        </div>
        {% endif %}
    </div>
</body>
</html>
```

---

static/style.css

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(to right, #e0f7fa, #e1bee7);
    padding: 40px;
    margin: 0;}
.container {
    max-width: 600px;
    margin: auto;
    background-color: #ffffffcc;
    padding: 30px;
```

```
    border-radius: 15px;
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
    backdrop-filter: blur(5px);
} h1 {
    text-align: center;
    color: #2c3e50;
    margin-bottom: 20px;
} label {
    display: block;
    margin-top: 15px;
    margin-bottom: 5px;
    color: #333;
    font-weight: bold;}
textarea, input[type="file"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 8px;
    font-size: 15px;
}input[type="submit"] {
    width: 100%;
    background: #007BFF;
    color: white;
    font-size: 16px;
    border: none;
    padding: 12px;
```

```
    border-radius: 8px;
    cursor: pointer;
    transition: background 0.3s ease;
} input[type="submit"]:hover {
    background: #0056b3;
} .result-box {
    margin-top: 20px;
    padding: 15px;
    background: #f5f5f5;
    border-left: 5px solid #007BFF;
    border-radius: 8px;
}
```

---

static/script.js

```
document.addEventListener("DOMContentLoaded", () => {
    const textInput = document.getElementById("text-input");
    const imageInput = document.getElementById("image-input");
    textInput.addEventListener("input", () => {
        if (textInput.value.length > 300) {
            alert("⚠ Text input too long. Consider summarizing.");
        }    });
    imageInput.addEventListener("change", () => {
        const file = imageInput.files[0];
        if (file && !file.type.startsWith("image/")) {
            alert("⃠ Only image files are allowed.");
            imageInput.value = "";
        }
```

```
    });
});
```

---

4. Requirements File

requirements.txt

```
flask
transformers
torch
torchvision
Pillow
```

---

5. Run the System

```
pip install -r requirements.txt
python app.py
```

This full proposal and implementation plan gives a clear, actionable path for building and delivering an AI-Powered Content Moderation System that is customizable, real-time, and accurate.