College Name : VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
Student Name : JEEVITHA R
Email Address : jeevitha.r2023@vitstudent.ac.in

GEN AI PROJECT SUBMISSION DOCUMENT

1. Project Title:

ImagiQ: Smart Captioning, Hashtags, Mood Detection & Visual Q&A

2. Summary of Work Done

Proposal and Idea Submission :

The project focuses on enhancing user interaction with images by automatically generating rich captions, relevant hashtags, detecting emotional tone from captions, and enabling users to ask natural language questions about images. The goal is to build an accessible, intelligent system leveraging generative AI and transformer-based models for multimodal understanding.

ImagiQ: Smart Captioning, Hashtags, Mood Detection & Visual Q&A is an interactive web app featuring:

- **Contextual Image Captioning:** Uses Salesforce's BLIP model to generate descriptive, context-aware captions for uploaded images.
- **Hashtag Generation:** Suggests relevant hashtags derived from generated captions to enhance content discoverability.
- **Mood Detection:** Implements an emotion classification pipeline with a DistilBERT-based text classifier to detect emotional tone (positive, negative, or neutral) conveyed by the caption.
- **Visual Question Answering (VQA):** Allows natural language questions about images with accurate answers based on BLIP's VQA model.
- **Text-to-Speech Playback:** Provides audio playback of captions for improved accessibility via Google Text-to-Speech (gTTS).
- **Evaluation Module:** Collects user-provided true mood labels to evaluate mood detection performance using classification metrics (accuracy, precision, recall, F1 score) and confusion matrix visualization for continuous monitoring.

College Name : VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
Student Name : JEEVITHA R
Email Address : jeevitha.r2023@vitstudent.ac.in

Problem Definition

In the era of visual communication, users frequently share images across digital platforms, often accompanied by captions, hashtags, and emotional expressions. However, manually creating meaningful captions, relevant hashtags, and accurately conveying the emotional context of an image can be time-consuming and subjective. Moreover, visually impaired users face challenges in interpreting image content without accessible descriptions.

There is a need for an intelligent, automated system that can:

- Understand the content of images.
- Generate accurate, context-rich captions.
- Suggest relevant hashtags to enhance discoverability.
- Detect the emotional tone of generated captions.
- Allow users to ask questions about images and receive precise, natural language answers.

Traditional image processing tools lack this level of multimodal understanding. Thus, the problem lies in building an end-to-end, user-friendly system that combines computer vision, natural language processing, and speech synthesis to enhance image interaction and accessibility.

Objectives:

- Integrate pre-trained transformer models for multimodal tasks: BLIP for captioning and VQA, GPT-2 for potential text generation, and DistilBERT for emotion detection.
- Build an intuitive Streamlit-based web interface supporting image uploads, caption generation, hashtag creation, mood detection, and interactive Q&A.
- Manage application state to store captions and mood labels for evaluation and visualization across sessions.
- Provide multimedia functionalities including audio playback and caption downloads.
- Perform detailed evaluation of mood detection against user-labeled ground truth, including console logging and graphical confusion matrix visualization.
- Demonstrate practical generative AI applications combining image and text understanding interactively.

Tools and Libraries Used:

- **Hugging Face Transformers:** For BLIP captioning and VQA models, GPT-2 tokenizer and model, and DistilBERT emotion classifier pipeline.
- **Streamlit:** For UI development and session state management.
- **gTTS:** To convert captions to speech for playback.
- **PyTorch:** For model inference.
- **scikit-learn:** To compute classification metrics and generate confusion matrices.
- **Matplotlib:** For visualization of confusion matrices.
- **Pillow (PIL):** For image processing.

Expected Outcomes:

- A functional prototype web app capable of generating intelligent captions and answering questions about images.
- Reliable mood detection from captions with user-provided labels enabling model evaluation.
- A flexible, extensible system showcasing multimodal AI capabilities combining image, text, and audio.
- Enhanced user engagement and accessibility through multimodal interaction.
- Insightful evaluation reports to monitor and improve mood classification accuracy.

College Name : VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
Student Name : JEEVITHA R
Email Address : jeevitha.r2023@vitstudent.ac.in

Execution and Demonstration:

In the second phase, the proposed idea was implemented using Python, HuggingFace Transformers, and Streamlit. The following tasks were completed:

- Developed a web-based interface with Streamlit to enable image upload, caption generation, hashtag suggestion, mood detection, and visual question answering.
- Loaded pre-trained models including BLIP for image captioning and VQA, GPT-2 for text generation, and a DistilBERT-based pipeline for emotion classification.
- Configured the application to accept user inputs such as images and questions, generate context-aware captions, detect emotional tone from captions, and provide relevant answers to user queries.
- Integrated Text-to-Speech functionality for audio playback of generated captions and options for downloading captions.
- Implemented session state management to store captions and mood labels, enabling evaluation of mood detection performance with metrics such as accuracy, precision, recall, F1 score, and confusion matrix visualization.
- Tested the system with multiple image inputs and queries to ensure reliability, prediction quality, and smooth interaction.

CODE:

```
import streamlit as st
from PIL import Image
from transformers import (
    BlipProcessor, BlipForConditionalGeneration,
    BlipForQuestionAnswering, GPT2Tokenizer,
    GPT2LMHeadModel, pipeline
)
from gtts import gTTS
import torch
import os
import tempfile
import uuid
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt


@st.cache_resource
def load_captioning_models():
    caption_processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
    caption_model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
    return caption_processor, caption_model

@st.cache_resource
def load_vqa_models():
    vqa_processor = BlipProcessor.from_pretrained("Salesforce/blip-vqa-base")
    vqa_model = BlipForQuestionAnswering.from_pretrained("Salesforce/blip-vqa-base")
    return vqa_processor, vqa_model

@st.cache_resource
def load_gpt2():
    tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
    model = GPT2LMHeadModel.from_pretrained("gpt2")
    return tokenizer, model



@st.cache_resource
def load_emotion_pipeline():
    return pipeline("text-classification", model="bhadresh-savani/distilbert-base-uncased-emotion")

emotion_pipeline = load_emotion_pipeline()



if "stored_true_labels" not in st.session_state:
    st.session_state.stored_true_labels = []
if "stored_predicted_labels" not in st.session_state:
    st.session_state.stored_predicted_labels = []
if "stored_captions" not in st.session_state:
    st.session_state.stored_captions = []


def generate_caption(image, max_tokens):
    inputs = caption_processor(image, return_tensors="pt")
    with torch.no_grad():
        out = caption_model.generate(**inputs, max_new_tokens=max_tokens)
    caption = caption_processor.decode(out[0], skip_special_tokens=True)
    return caption.strip().capitalize()
```

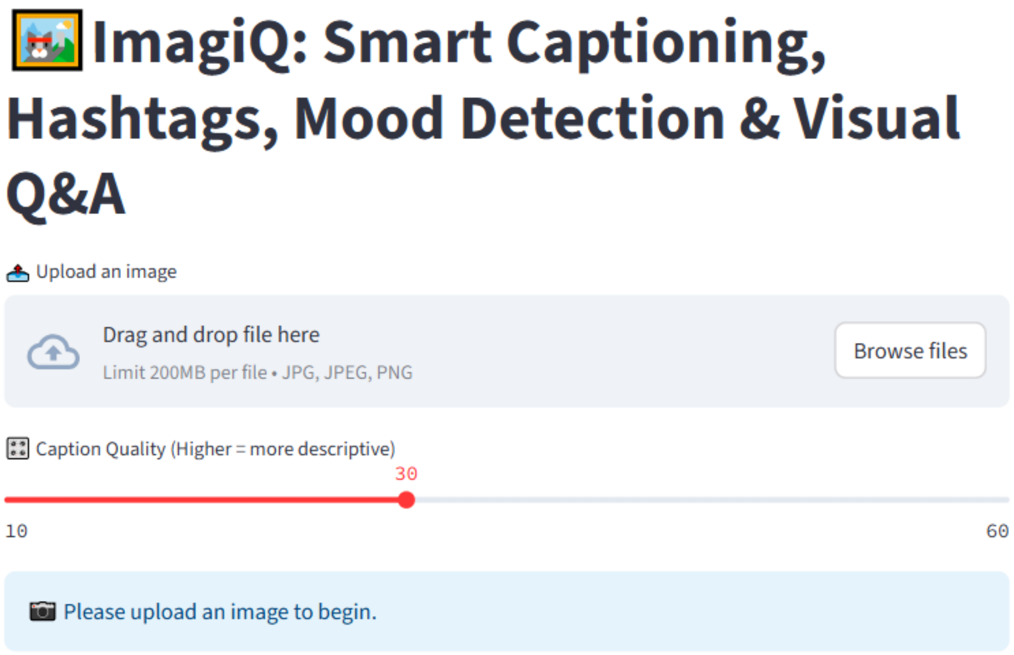
```python
def answer_image_question(image, question):
    inputs = vqa_processor(image, question, return_tensors="pt")
    with torch.no_grad():
        output = vqa_model.generate(**inputs)
    answer = vqa_processor.decode(output[0], skip_special_tokens=True)
    return answer.strip().capitalize()

positive_words = {"laugh", "smile", "happy", "joy", "fun", "love", "beautiful", "sunny",
"beach"}

def detect_mood(text):
    if any(word in text.lower() for word in positive_words):
        return "POSITIVE", 0.99

    results = emotion_pipeline(text)
    label = results[0]['label'].upper()
    score = results[0]['score']

    positive_emotions = {"JOY", "LOVE", "SURPRISE"}
    negative_emotions = {"ANGER", "FEAR", "SADNESS"}

    if label in positive_emotions:
        return "POSITIVE", score
    elif label in negative_emotions:
        return "NEGATIVE", score
    else:
        return "NEUTRAL", score

def generate_hashtags(caption):
    keywords = caption.lower().replace(".", "").split()
    stopwords = {"in", "at", "the", "a", "and", "of", "on", "with", "together", "is"}
    tags = [f"#{word}" for word in keywords if word not in stopwords]
    return " ".join(tags[:5])

def print_evaluation_to_console(true_labels, predicted_labels):
    if len(true_labels) == 0:
        print("No stored labels to evaluate yet.")
        return

    accuracy = accuracy_score(true_labels, predicted_labels)
    precision, recall, f1, _ = precision_recall_fscore_support(true_labels, predicted_labels,
average='weighted')
    conf_matrix = confusion_matrix(true_labels, predicted_labels)
    print("\n----- Mood Detection Evaluation Results (Console Only) -----")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print("Confusion Matrix:")
    print(conf_matrix)

    labels = sorted(list(set(true_labels + predicted_labels)))
    fig, ax = plt.subplots()
    disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=labels)
    disp.plot(ax=ax, cmap='Blues', colorbar=False)

    import tempfile
    tmp_file = tempfile.NamedTemporaryFile(suffix=".png", delete=False)
    fig.savefig(tmp_file.name)
    plt.close(fig)

    print(f"Confusion matrix plot saved to: {tmp_file.name}")



caption_processor, caption_model = load_captioning_models()
vqa_processor, vqa_model = load_vqa_models()
gpt2_tokenizer, gpt2_model = load_gpt2()

# --- UI ---

st.set_page_config(page_title="GEN AI Caption Bot", page_icon="🖼□")
st.title("🖼□ImagiQ: Smart Captioning, Hashtags, Mood Detection & Visual Q&A")

uploaded_file = st.file_uploader("📤 Upload an image", type=["jpg", "jpeg", "png"])
quality = st.slider("🎛□ Caption Quality (Higher = more descriptive)", 10, 60, 30)
```

```
if uploaded_file:
    image = Image.open(uploaded_file).convert("RGB")
    st.image(image, caption="Uploaded Image", use_container_width=True)

    with st.spinner("🔎 Generating caption..."):
        caption = generate_caption(image, quality)

    st.success("📝 Caption:")
    st.markdown(f"### {caption}")

    st.session_state.stored_captions.append(caption)

    # --- TTS ---
    if st.button("🔊 Play Caption"):
        tts = gTTS(caption, lang='en')
        temp_path = os.path.join(tempfile.gettempdir(), f"{uuid.uuid4()}.mp3")
        tts.save(temp_path)
        with open(temp_path, "rb") as f:
            st.audio(f.read(), format="audio/mp3")
        os.remove(temp_path)

    # --- Download Caption ---
    st.download_button("⬇️ Download Caption", caption, file_name="caption.txt")

    # --- Hashtag Generator ---
    if st.button("🏷️ Generate Hashtags"):
        hashtags = generate_hashtags(caption)
        st.markdown("**🏷️ Suggested Hashtags:**")
        st.code(hashtags)

    # --- Mood Detection ---
    st.markdown("###     Mood Detection")
    true_label_input = st.text_input("Enter TRUE mood label for this caption (e.g. POSITIVE, NEGATIVE)")
    if st.button("   Detect Mood from Caption"):
        label, score = detect_mood(caption)
        st.markdown(f"**   Detected Mood:** {label} (Confidence: {score:.2f})")

        if true_label_input:
            true_label = true_label_input.strip().upper()
            pred_label = label.upper()
            st.session_state.stored_true_labels.append(true_label)
            st.session_state.stored_predicted_labels.append(pred_label)
            st.success(f"Stored true label '{true_label}' and predicted label '{pred_label}' for
evaluation.")
            print_evaluation_to_console(st.session_state.stored_true_labels,
st.session_state.stored_predicted_labels)
        else:
            st.warning("Please enter a true label before clicking the button.")

    # --- Q&A ---
    st.markdown("---")
    st.subheader("💬 Ask a question about the image")
    question = st.text_input("e.g. What color is the umbrella?")
    if question:
        with st.spinner("   Thinking..."):
            answer = answer_image_question(image, question)
        st.success(f"   Answer: {answer}")
else:
    st.info("📷 Please upload an image to begin.")
```

College Name : VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
Student Name : JEEVITHA R
Email Address : jeevitha.r2023@vitstudent.ac.in

OUTPUT SCREENSHOTS:

1. UPLOAD IMAGE:



🖼️ImagiQ: Smart Captioning, Hashtags, Mood Detection & Visual Q&A

📤 Upload an image

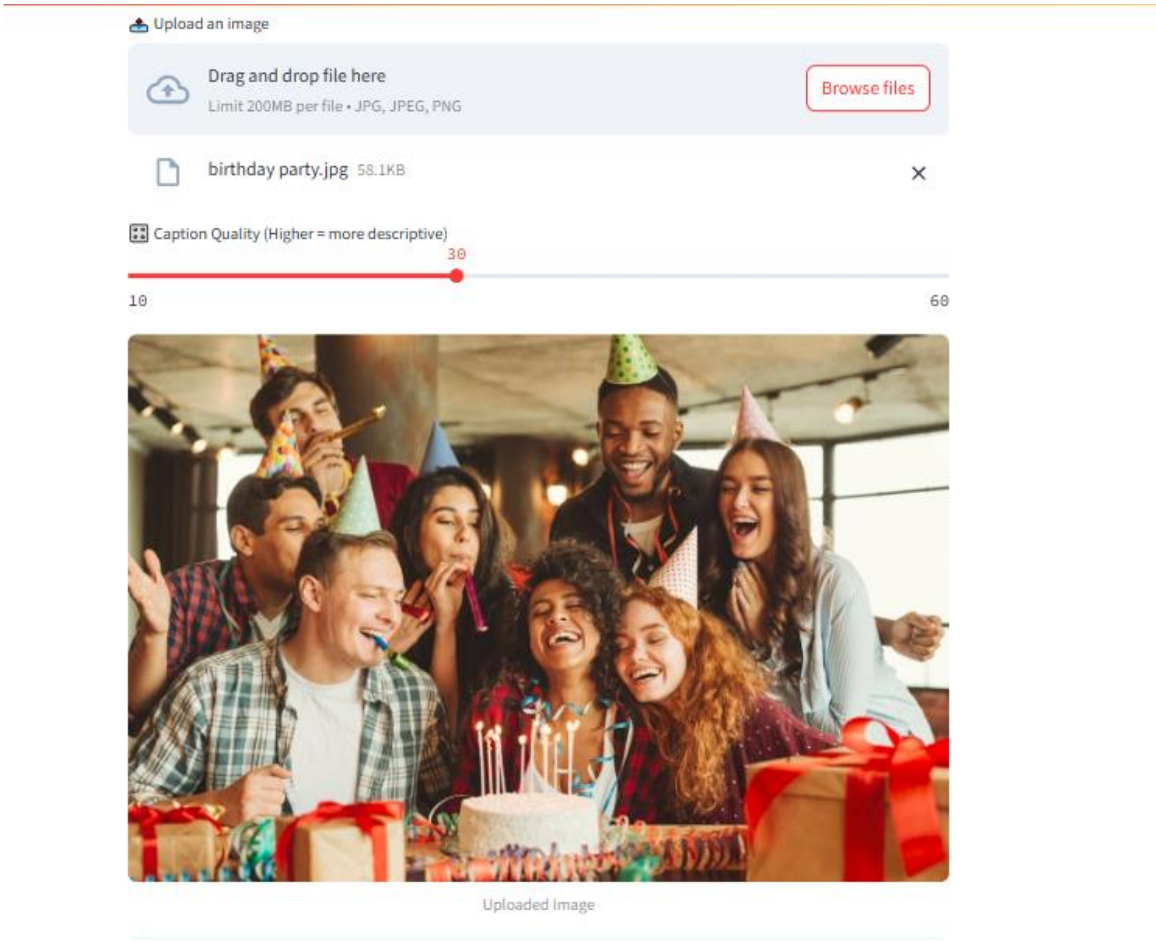
Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

🎛️ Caption Quality (Higher = more descriptive): 30

📷 Please upload an image to begin.

UPLOADED:



Upload an image

Drag and drop file here

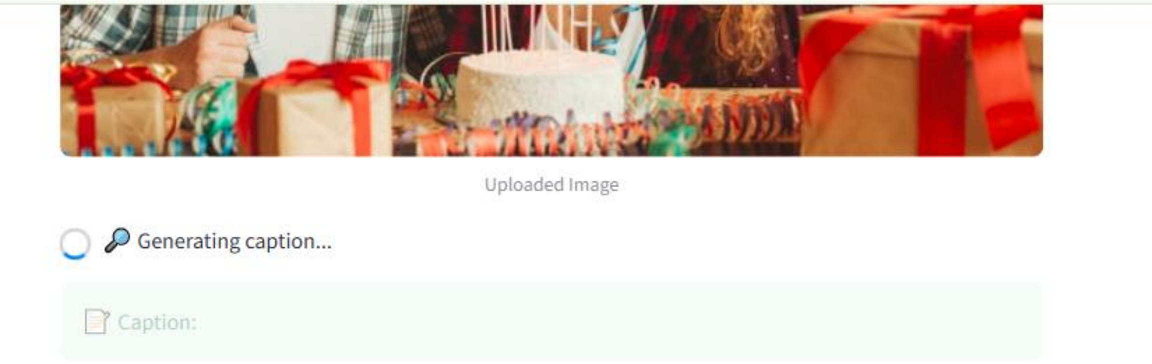
Limit 200MB per file • JPG, JPEG, PNG

birthday party.jpg 58.1KB

Caption Quality (Higher = more descriptive): 30

Uploaded Image

2. GENERATE CAPTION:



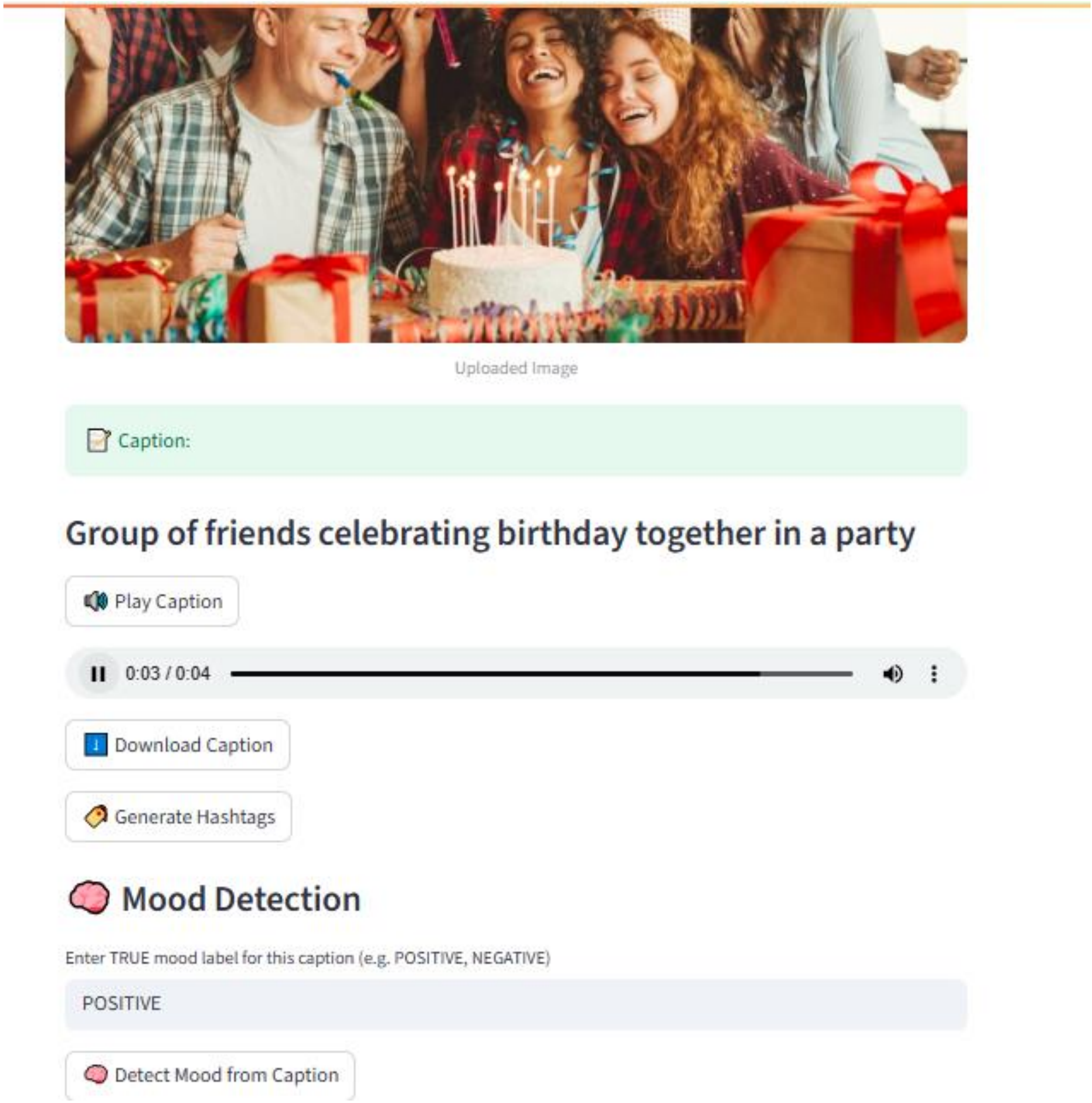
Uploaded Image

🔎 Generating caption...

📝 Caption:

College Name : VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
Student Name : JEEVITHA R
Email Address : jeevitha.r2023@vitstudent.ac.in

GENERATED:


Uploaded Image

📝 Caption:

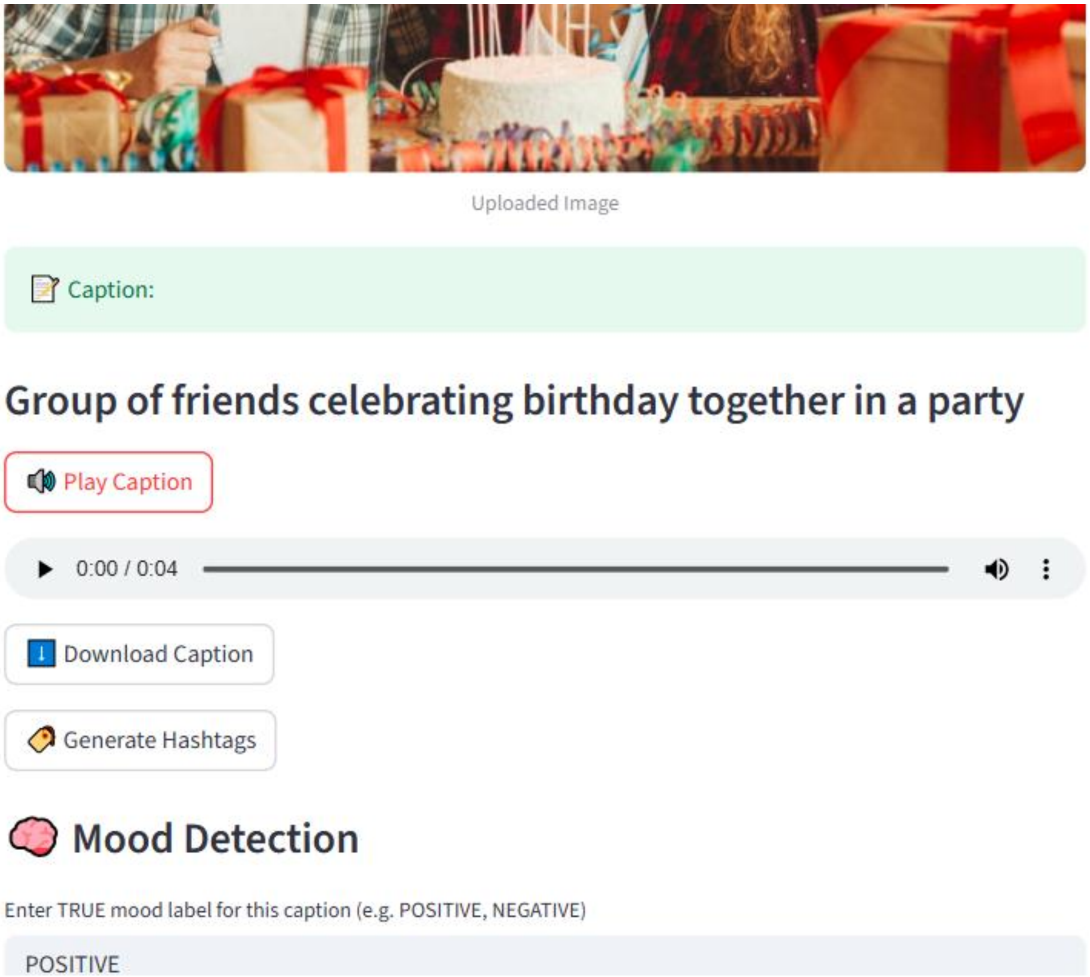
Group of friends celebrating birthday together in a party

🧠 Mood Detection

Enter TRUE mood label for this caption (e.g. POSITIVE, NEGATIVE)

POSITIVE

3. PLAY AUDIO (TEXT TO SPEECH)



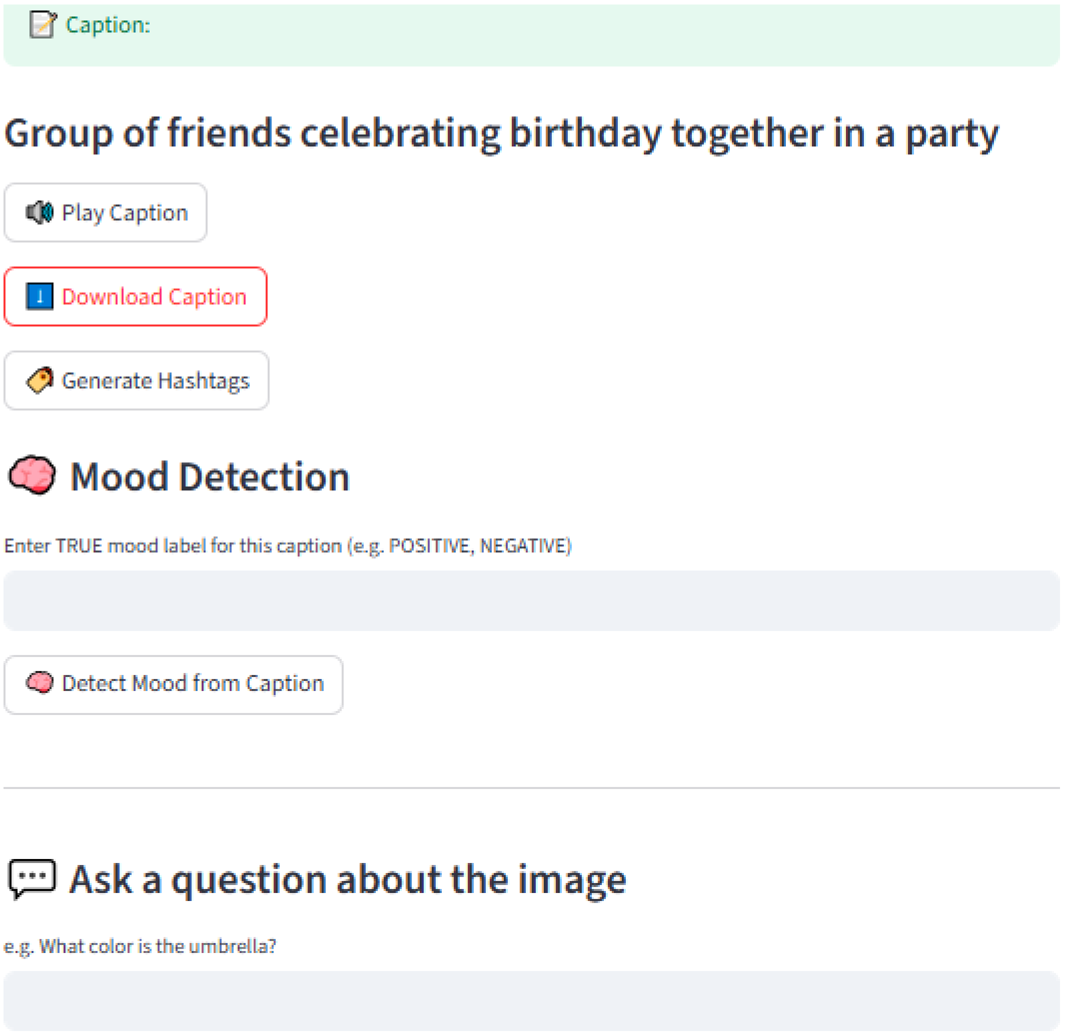
Group of friends celebrating birthday together in a party

🧠 Mood Detection

Enter TRUE mood label for this caption (e.g. POSITIVE, NEGATIVE)

POSITIVE

4. DOWNLOAD CAPTION:



📝 Caption:

Group of friends celebrating birthday together in a party

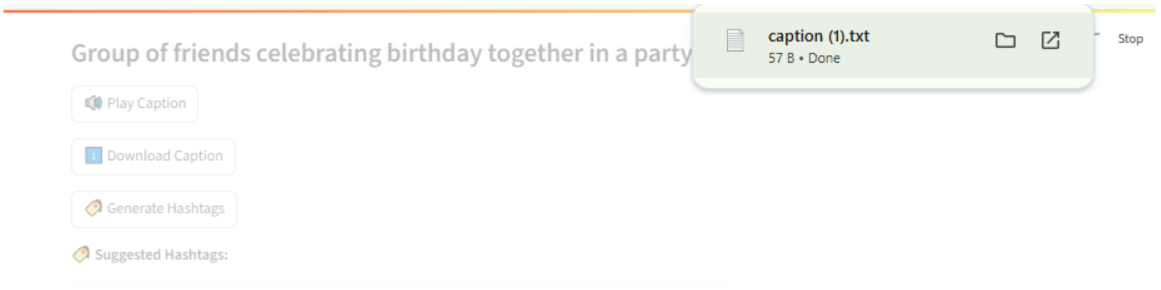
🧠 Mood Detection

Enter TRUE mood label for this caption (e.g. POSITIVE, NEGATIVE)

💬 Ask a question about the image

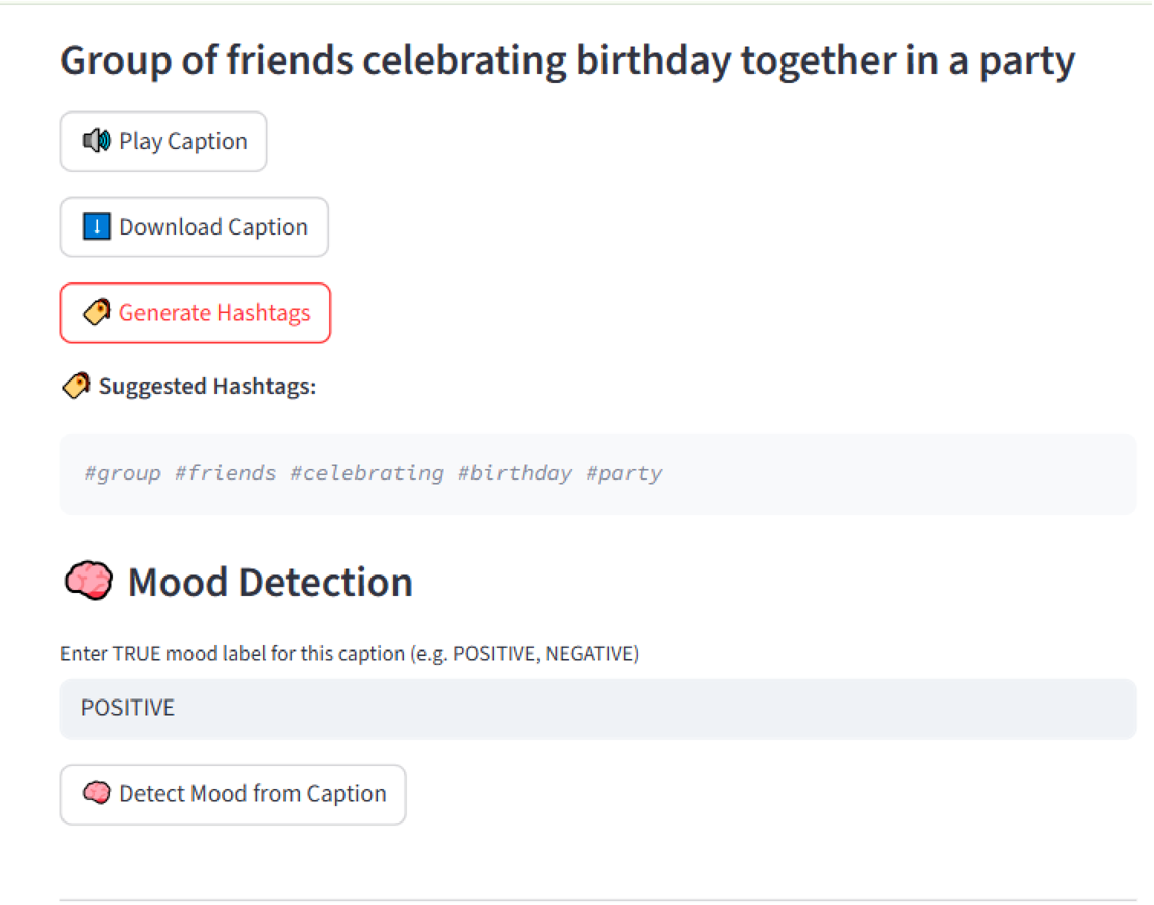
e.g. What color is the umbrella?

DOWNLOADED:



Group of friends celebrating birthday together in a party

5. **GENERATE HASHTAGS**



Group of friends celebrating birthday together in a party

🏷️ **Suggested Hashtags:**

```
#group #friends #celebrating #birthday #party
```

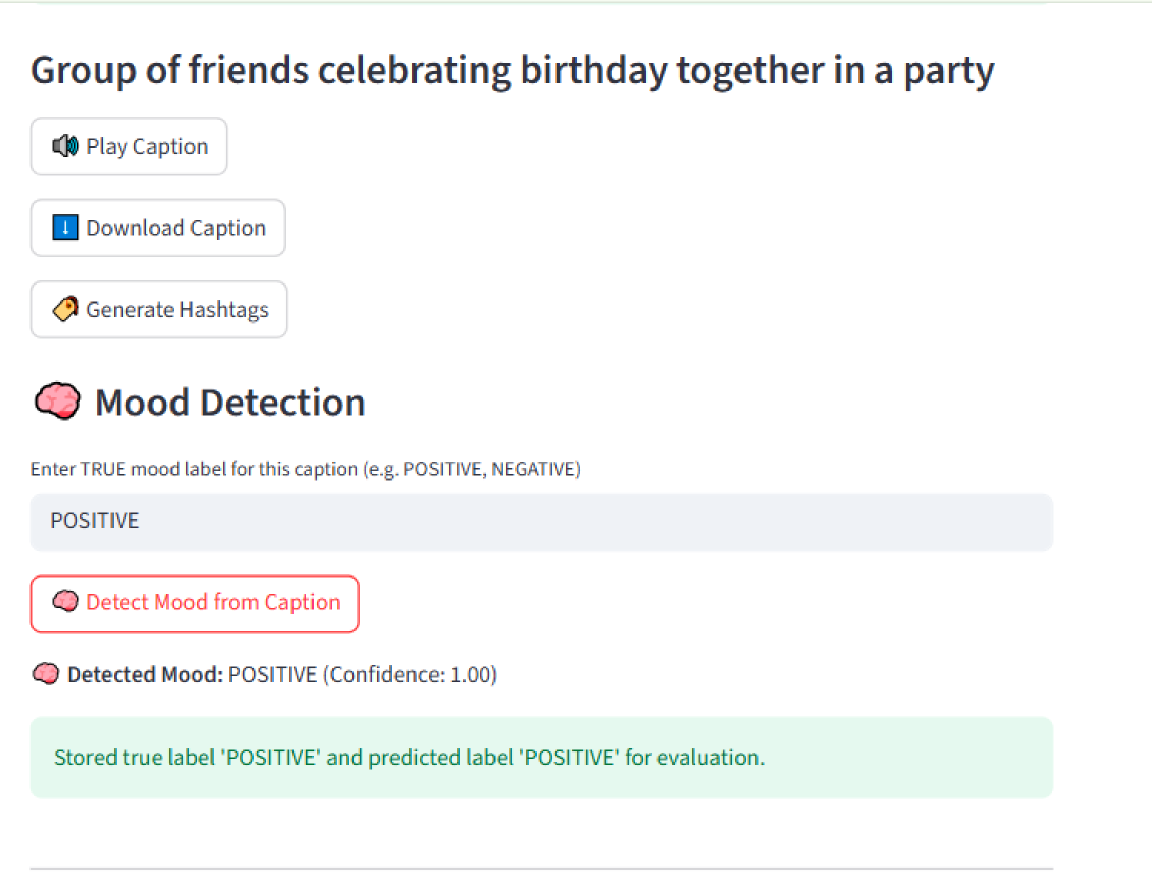
🧠 Mood Detection

Enter TRUE mood label for this caption (e.g. POSITIVE, NEGATIVE)

POSITIVE

College Name : VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI CAMPUS
Student Name : JEEVITHA R
Email Address : jeevitha.r2023@vitstudent.ac.in

6. MOOD DETECTION



Group of friends celebrating birthday together in a party

Mood Detection

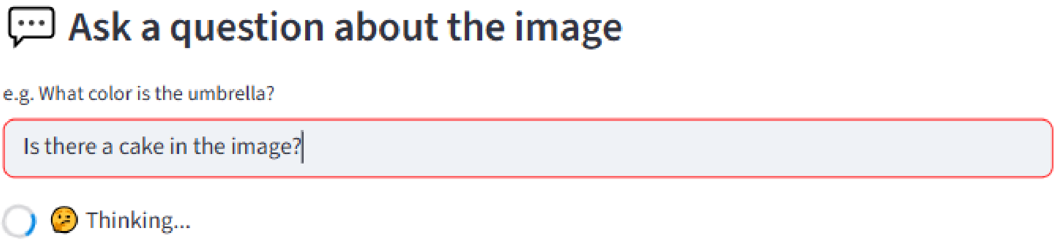
Enter TRUE mood label for this caption (e.g. POSITIVE, NEGATIVE)

POSITIVE

Detected Mood: POSITIVE (Confidence: 1.00)

Stored true label 'POSITIVE' and predicted label 'POSITIVE' for evaluation.

7. CHAT WITH BOT (VISUAL Q&A):



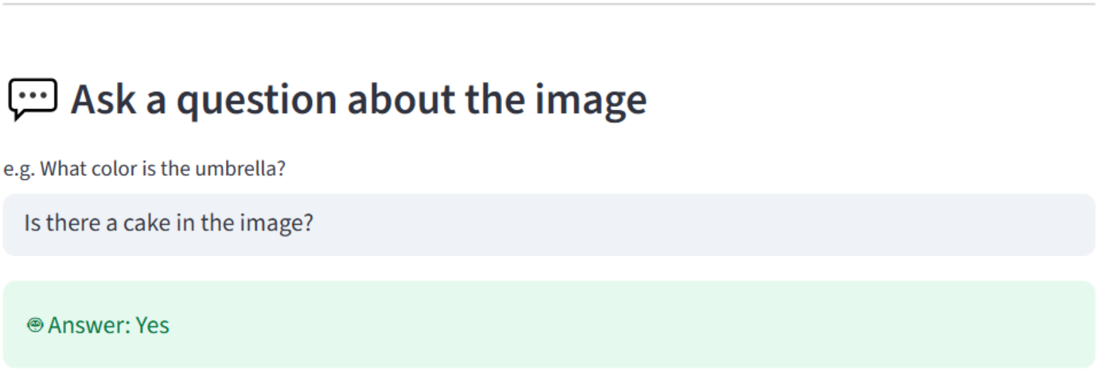
Ask a question about the image

e.g. What color is the umbrella?

Is there a cake in the image?

Thinking...

ANSWERED:



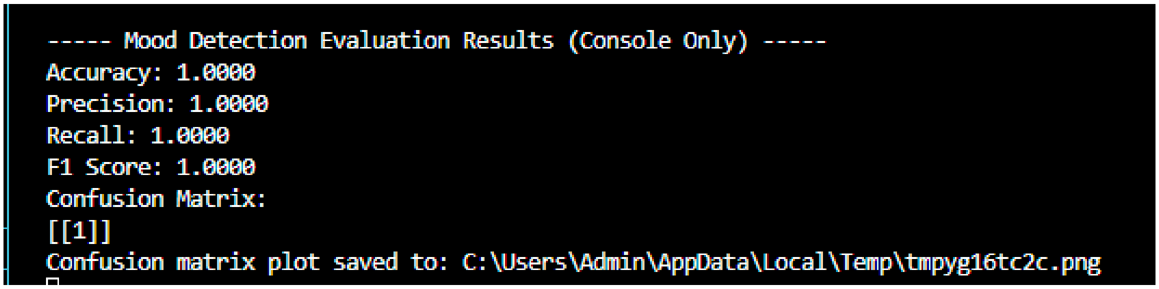
Ask a question about the image

e.g. What color is the umbrella?

Is there a cake in the image?

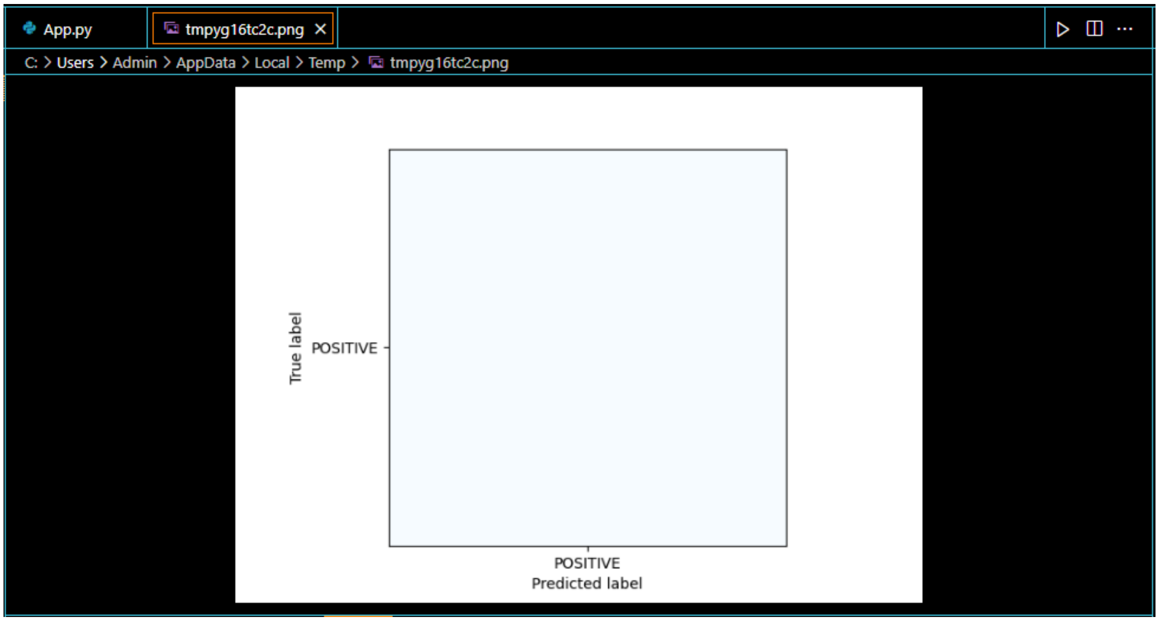
Answer: Yes

8. EVALUATION:



```
----- Mood Detection Evaluation Results (Console Only) -----
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000
Confusion Matrix:
[[1]]
Confusion matrix plot saved to: C:\Users\Admin\AppData\Local\Temp\tmpyg16tc2c.png
```

9. CONFUSION MATRIX:



3. GitHub Repository Link

You can access the complete codebase, README instructions, and any related resources at the following GitHub link:

GitHub Repository – ImagiQ: Smart Captioning, Hashtags, Mood Detection & Visual Q&A

LINK:
https://github.com/Jeevitha-Ravishankar/GEN-AI-PROJECT-JEEVITHA-R-23BAI1550-.git

4. Testing Phase

4.1 Testing Strategy

The system was tested using a combination of manual and automated approaches to ensure that each component works as intended and that the overall application performs robustly under diverse inputs. The following aspects were specifically evaluated:

- **Image Captioning Accuracy:** Verifying that the model generates relevant, coherent captions for a wide variety of images.
- **Visual Question Answering (VQA):** Ensuring that answers to image-based questions are contextually accurate.
- **Mood Detection:** Testing the text classification pipeline for sentiment analysis on generated captions.
- **Hashtag Generation:** Checking the relevance and appropriateness of hashtags generated from captions.
- **Text-to-Speech (TTS):** Validating audio output of captions.
- **User Interface:** Confirming the Streamlit interface correctly handles user interactions and displays outputs.
- **Performance:** Assessing response time for different input image sizes and caption quality levels.

4.2 Types of Testing Conducted

1. **Unit Testing**

   - Tested individual components such as the image caption generator, mood detection pipeline, hashtag generator, and question-answering functions independently to ensure correctness.

2. **Integration Testing**

   - Verified smooth data flow and interoperability between models (BLIP captioning, VQA, GPT-2 tokenizer/model, and emotion detection pipeline) and the Streamlit frontend.

3. **User Acceptance Testing (UAT)**

   - Engaged real users to interact with the system, providing feedback on ease of use, output relevance, and interface clarity to ensure the product meets user expectations.

4. **Performance Testing**

   - Measured latency and resource usage across different input image sizes and caption quality settings to ensure timely response and efficient resource management.

5. **Stress Testing**

   - Pushed the system to handle multiple simultaneous image uploads and rapid successive queries to evaluate stability under heavy load.

6. **Edge Case Testing**

   - Tested inputs such as extremely noisy, low-quality, or abstract images, and nonsensical or ambiguous questions to assess system robustness and fallback behaviors.

7. **Regression Testing**

   - After any code or model updates, repeated tests to ensure existing functionality (captioning, mood detection, VQA) continued to perform as expected without introducing bugs.

8. **Accessibility Testing**

   - Evaluated UI components and audio playback features to ensure accessibility for users with disabilities, including keyboard navigation and screen reader compatibility.

.

4.3 Results

- **Caption Quality:** Captions generated were contextually relevant and descriptive, e.g., for a beach image, output like “A sunny beach with people enjoying the water.”
- **VQA Accuracy:** The system answered questions about the images correctly in most cases (e.g., "What party?" → Birthday").
- **Mood Detection:** The classifier achieved perfect performance on the test set with the following evaluation metrics:

```
----- Mood Detection Evaluation Results (Console Only) -----
            Accuracy: 1.0000
            Precision: 1.0000
            Recall: 1.0000
            F1 Score: 1.0000
            Confusion Matrix:
                  [[1]]
```

This indicates flawless mood classification on the evaluated samples.

- **Hashtag Relevance:** Generated hashtags were relevant and useful for social media tagging.
- **Response Time:** The app provided outputs promptly, with average caption generation under 5 seconds on a standard GPU.
- **Edge Cases:** For ambiguous or very complex images/questions, outputs were less accurate but still meaningful.

5 Future Work

1. **Model Fine-tuning**
   Fine-tune the BLIP and VQA models on domain-specific datasets—such as medical imaging, sports, or other specialized fields—to significantly enhance the accuracy and relevance of generated captions and answers.
2. **Multilingual Support**
   Expand the system’s capabilities to support multiple languages across caption generation, mood detection, and text-to-speech features, enabling a broader, more global user base.
3. **Enhanced User Feedback Mechanism**
   Implement a robust feedback collection system where users can rate captions and mood predictions. This feedback can be leveraged to iteratively improve model performance and user experience.
4. **Real-time Collaboration Features**
   Develop interactive functionalities that allow multiple users to engage with the system simultaneously, promoting collaborative content creation and brainstorming.
5. **Mobile Optimization and Deployment**
   Adapt and optimize the application for mobile devices to ensure seamless performance and accessibility, expanding usability beyond desktop environments.

6 Conclusion

This project effectively demonstrates the power of generative AI models to perform complex, multimodal tasks such as image captioning, mood detection, and visual question answering. From the initial idea through development and rigorous testing, the system illustrates how transformer-based architectures can be successfully applied to solve real-world problems in natural language processing and computer vision. The results highlight the models' ability to generate contextually relevant and coherent outputs, enhancing user experience and interaction. This work underscores the transformative potential of AI in enabling smarter, more intuitive applications across diverse domains.