College Name: VIT Bhopal University
Student Name: Ishan Peshkar

Project Title

Phase 2: Project Execution and Demonstration

1. Project Title:

AI-Powered Poem Generator

2. Objective Recap:

The primary objective of this project was to develop an AI-Powered Poem Generator capable of creating short, creative poems based on user-provided topics. Key goals included leveraging pre-trained language models via the Hugging Face transformers library, implementing an interactive user interface (initially CLI, then enhanced with Streamlit), and experimenting with prompt engineering and generation parameters to produce coherent and thematically relevant poetic outputs.

3. Technologies Used:

- **Programming Language:** Python
- **Core AI Library:** Hugging Face transformers
- **Deep Learning Backend:** PyTorch
- **Web Interface:** Streamlit
- **Development Environment:** Local Python environment with venv, VS Code (or similar IDE)
- **Pre-trained Model:** GPT-2 (with potential to use gpt2-medium or others)

4. Full Code Implementation:

Step 1: Install Required Libraries

```
pip install transformers streamlit torch
```

Step 2: Import Required Libraries

```
import streamlit as st
from transformers import pipeline, set_seed
import torch
```

College Name: VIT Bhopal University
Student Name: Ishan Peshkar

Step 3: Load the Pretrained GPT-2 Model

```
@st.cache_resource
def load_generator_pipeline(model_name="gpt2"):
    device = 0 if torch.cuda.is_available() else -1
    return pipeline('text-generation', model=model_name, device=device)
```

Step 4: Define Poem Generation Function

```
def generate_poem(generator, topic, max_len=60, num_poems=1,
temperature=0.7, seed_value=42):
    set_seed(seed_value)
    prompt = f"Compose a short, creative and evocative poem about
{topic}:\n\n"
    outputs = generator(
        prompt,
        max_length=max_len + len(prompt.split()),
        num_return_sequences=num_poems,
        temperature=temperature,
        top_k=50,
        top_p=0.95,
        no_repeat_ngram_size=2,
        early_stopping=True
    )
    return [out['generated_text'][len(prompt):].strip() for out in outputs]
```

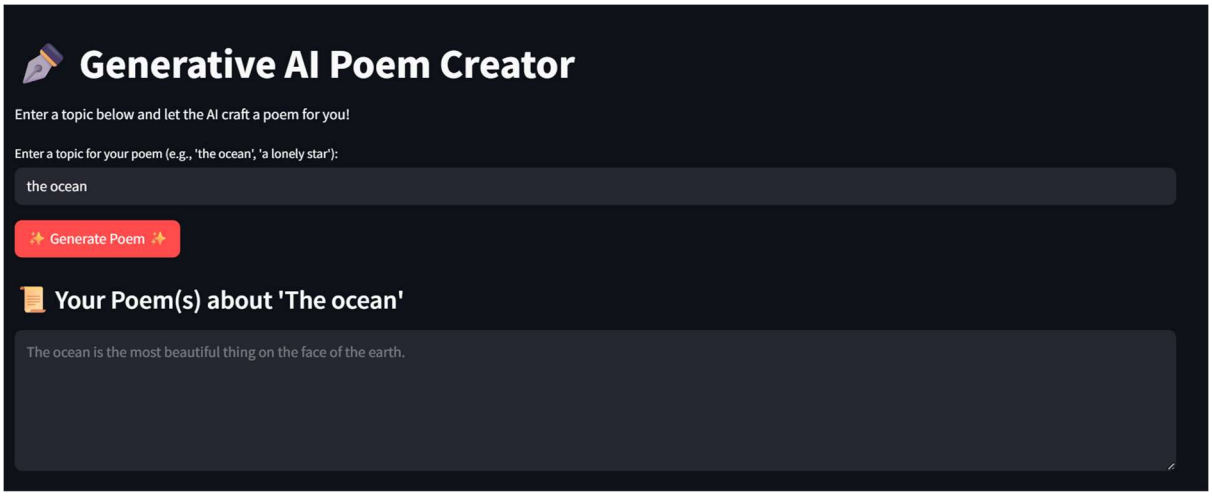
Step 5: Build the Streamlit Interface

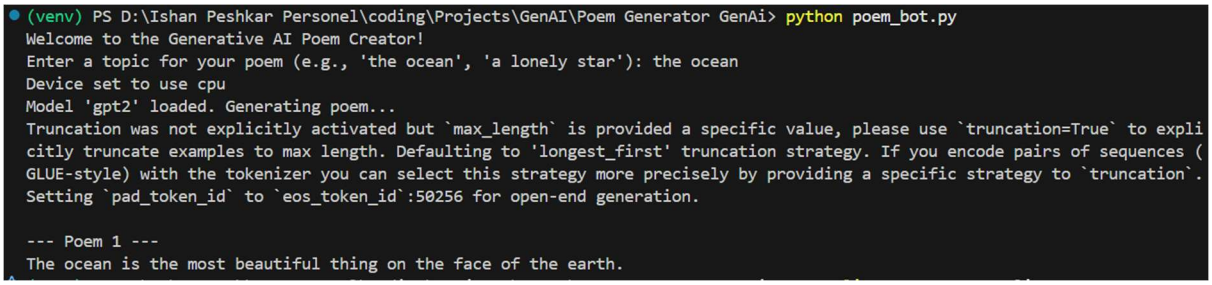
```
st.set_page_config(page_title="AI Poem Generator", page_icon="✍️")
st.title("🖋️ AI-Powered Poem Generator")

# Sidebar Controls
with st.sidebar:
    st.header("🎛️ Poem Settings")
    model = st.selectbox("Choose Model", ["gpt2", "gpt2-medium"])
    topic = st.text_input("Poem Topic", "a forgotten dream")
    max_len = st.slider("Max Poem Length", 30, 150, 70, 10)
    temperature = st.slider("Creativity (Temperature)", 0.5, 1.0, 0.75,
0.05)
    num_poems = st.number_input("Number of Poems", 1, 3, 1)
    seed = st.number_input("Seed", 0, 1000, 42)
    generate = st.button("Generate Poem")

# Generate Output
if generate and topic:
    generator_pipeline = load_generator_pipeline(model)
    poems = generate_poem(generator_pipeline, topic, max_len, num_poems,
temperature, seed)
    st.subheader(f"📜 Poem(s) about '{topic}'")
    for i, poem in enumerate(poems):
        if num_poems > 1:
            st.markdown(f"--- **Poem {i+1}** ---")
        st.text_area("", value=poem, height=200, disabled=True)
elif generate and not topic:
    st.warning("Please enter a topic for your poem.")
```

Step 6: Run the Streamlit App

```
streamlit run app.py
```

5. Output Screenshots:





6. Conclusion:

This project phase successfully executed the development of an AI-Powered Poem Generator. The application, built using Python, Hugging Face transformers (specifically the GPT-2 model), and Streamlit, provides an interactive platform for users to generate creative poetry based on specified topics and parameters. The system demonstrates the capability of Generative AI to produce contextually relevant and often imaginative poetic text, fulfilling the core objectives. The use of Streamlit significantly enhanced the user experience compared to a simple CLI.

7. References:

- Hugging Face Transformers Documentation: https://huggingface.co/docs/transformers
- OpenAI GPT-2 Research: https://openai.com/research/better-language-models
- Streamlit Documentation: https://docs.streamlit.io/
- PyTorch Documentation: https://pytorch.org/docs/stable/index.html