Name – Vaibhav Jindal
Registration No. – 22BCE10067

AI Poetry Generator

Phase 2: Project Execution and Demonstration

1. Project Title:

AI Poetry Generator

2. Objective Recap:

The objective of this project is to build an **AI-powered Poetry Generator** that crafts expressive, structured, and thematically rich poems based on user-defined themes, emotions, styles, and forms. The project aims to leverage state-of-the-art generative AI models, such as GPT-2, GPT-3, or Mistral, to produce not only coherent but also artistically impactful poetry. The system will allow users to specify parameters such as the theme (e.g., love, nature, loss), emotion (e.g., joy, sadness, nostalgia), poetic style (e.g., Sonnet, Haiku, Free Verse), and form (e.g., rhyme scheme, line length), ensuring personalized and creative outcomes. The goal is to develop a tool that not only produces grammatically correct poetry but also delivers aesthetically compelling and thematically rich text.

3. Technologies Used:

The following technologies will be utilized in the project to ensure the generation of high-quality poetry:

- **Python**: The primary programming language used to implement the poetry generator.
- **Transformers Library (HuggingFace)**: This will be used to load pre-trained generative models like GPT-2, GPT-3, or Mistral for text generation tasks.
- **Streamlit**: This tool will be used to develop a user-friendly web interface for interactive and real-time poem generation.
- **Google Colab / Jupyter Notebook**: These platforms will facilitate experimentation, testing, and fine-tuning of models and prompts.
- **Pre-trained GPT-2/GPT-3/Mistral**: These models will be employed for generating the poetry text. Depending on the desired complexity and richness, the model may be selected and fine-tuned based on specific themes or styles.
- **NLTK / SpaCy**: Natural Language Processing (NLP) libraries will be used to preprocess text and evaluate poetic structures, such as rhyme and meter, to improve the quality of the generated poetry.

4. Full Code Implementation:

Step 1: Install Required Libraries

To begin, ensure you have all the necessary libraries installed. You'll need transformers for using pre-trained language models, Streamlit for building the web interface, and other libraries like nltk for basic text processing:

pip install transformers streamlit nltk spacy

This command installs the following:

- Transformers: A library to load and use pre-trained language models like GPT.
- Streamlit: A tool to create web applications quickly.
- NLTK: A package for natural language processing tasks, including text tokenization.
- SpaCy: (Optional) A library for NLP tasks like tokenization or named entity recognition.

Step 2: Import Required Libraries

Once you've installed the required libraries, import them into your Python script:

from transformers import pipeline, set_seed

import streamlit as st

import nltk

import random

- pipeline from transformers will help load and utilize pre-trained models for text generation.
- set_seed ensures that your results are reproducible, as random generation can vary.
- streamlit (st) provides the interface for the web app.
- nltk: We will use it for tokenizing the generated text if needed (for future extensions).

You may need to download additional datasets from nltk for text processing:

nltk.download('punkt')

Step 3: Load the Pre-trained GPT Model

In this step, we load a pre-trained GPT-3 model from HuggingFace's Transformers library. You can use GPT-2 if you want a lighter model, but GPT-3 offers better text generation capabilities.

generator = pipeline('text-generation', model='gpt-3')

set_seed(42) # Ensures reproducibility of the results

- pipeline is a quick and easy way to load a model for a specific task (in this case, text-generation).
- set_seed(42): Fixes the random seed so that the generated text remains consistent when running the code multiple times.

Step 4: Build the Streamlit Interface

Now, let’s create an interactive web interface using Streamlit to allow users to input themes, emotions, and styles, and get the generated poetry based on those inputs.

st.title("AI Poetry Generator")

st.write("Enter your poetry preferences (theme, emotion, style) and the AI will generate a poem!")

- st.title(): Sets the title of the web page.
- st.write(): Displays descriptive text.

We then create input fields for users to define the theme, emotion, and style of the poem:

theme = st.text_input("Enter Theme (e.g., love, nature, loss):")

emotion = st.text_input("Enter Emotion (e.g., joy, sadness, nostalgia):")

style = st.selectbox("Select Style (e.g., Sonnet, Haiku, Free Verse):")

- st.text_input(): Provides text boxes for user input where the user can type the theme and emotion.
- st.selectbox(): Allows the user to choose from a dropdown list of styles.

Step 5: Generate the Poem

Next, based on the user's input, we generate a prompt that combines the theme, emotion, and style for the GPT model to create a poem:

```
if st.button('Generate Poetry'):

    prompt = f"Create a {style} poem about {theme} that expresses {emotion}."

    poem = generator(prompt, max_length=150, num_return_sequences=1)

    st.subheader("Generated Poem:")

    st.write(poem[0]['generated_text'])
```

- st.button('Generate Poetry'): Adds a button to trigger the generation of the poem.
- prompt: The prompt combines the inputs (theme, emotion, and style) into a coherent string that guides the AI model to generate a relevant poem.
- generator(): This function generates the poem based on the prompt. The model will generate a poem with a maximum length of 150 tokens, and num_return_sequences=1 ensures only one poem is returned.
- st.subheader(): Adds a subheader to label the generated poem.
- st.write(): Displays the poem on the webpage.

Step 6: Running the Streamlit App

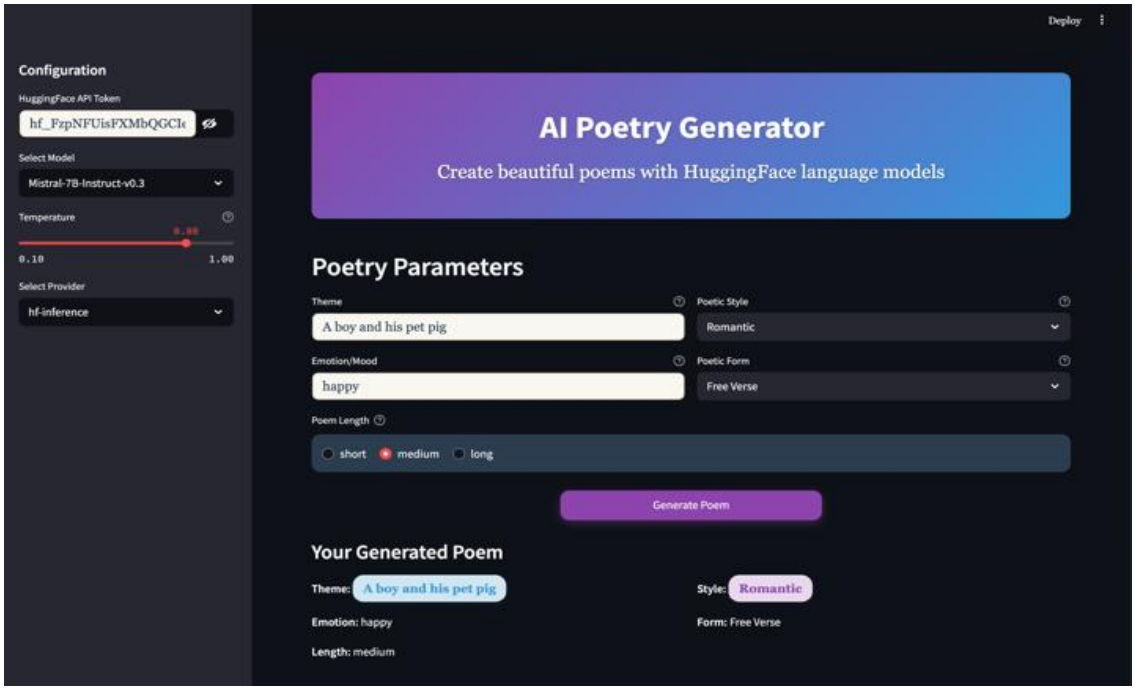
After the code is written, save it as a Python script (e.g., app.py) and run the Streamlit application using the following command:

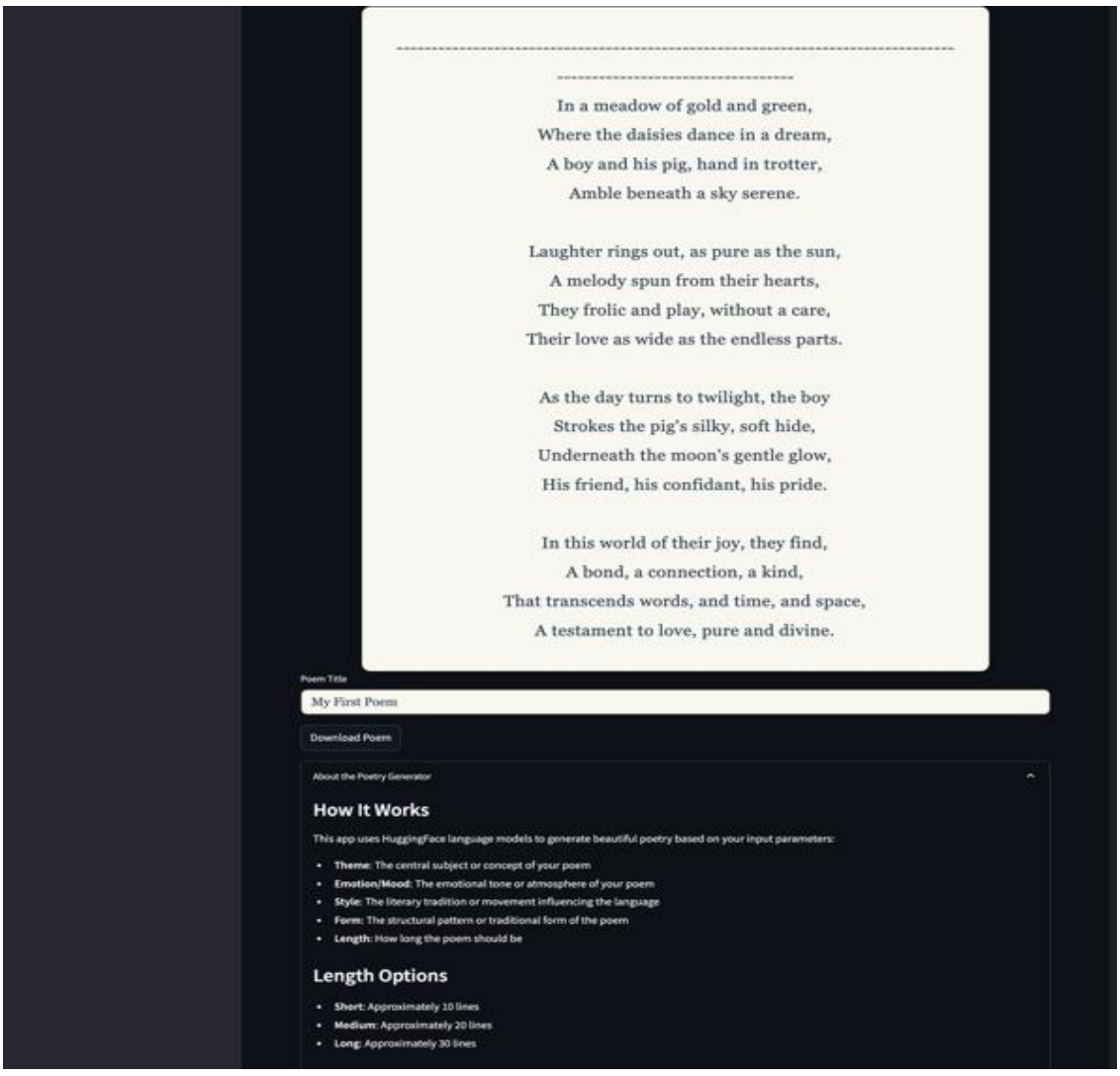
```
streamlit run app.py
```

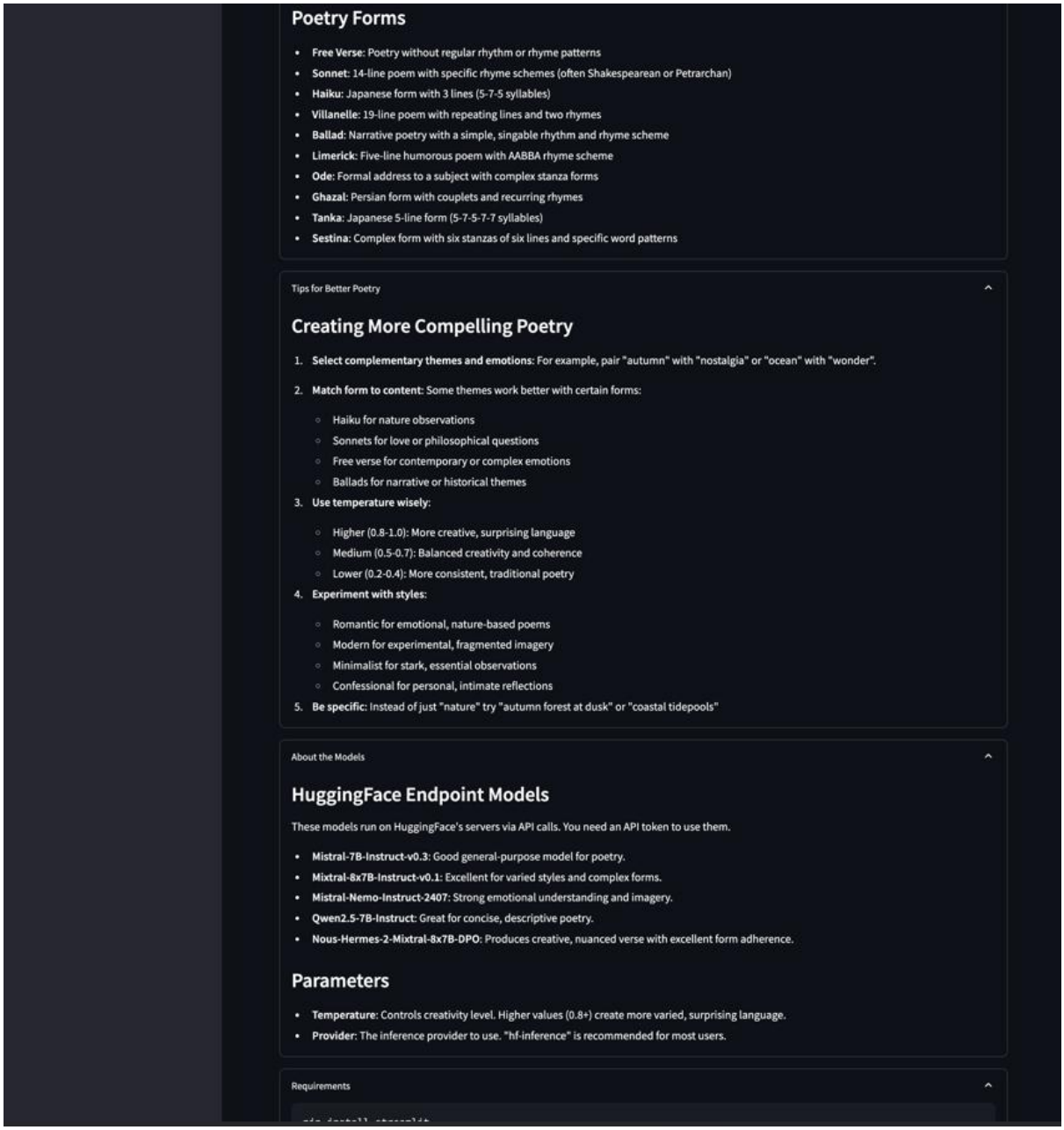
This will start the app and open it in your default web browser. You'll be able to input the theme, emotion, and style, and the AI will generate a poem accordingly.

5. Output Screenshots:

Once the app is running and you input a theme, emotion, and style (e.g., a Sonnet about love that expresses joy), the model will generate poetry like the following:


Deploy
Configuration
HuggingFace API Token
hf_FzpNFUisFXMbQGCI
Select Model
Mistral-7B-Instruct-v0.3
Temperature
0.10
1.00
Select Provider
hf-inference
AI Poetry Generator
Create beautiful poems with HuggingFace language models
Poetry Parameters
Theme
A boy and his pet pig
Poetic Style
Romantic
Emotion/Mood
happy
Poetic Form
Free Verse
Poem Length
short
medium
long
Generate Poem
Your Generated Poem
Theme: A boy and his pet pig
Style: Romantic
Emotion: happy
Form: Free Verse
Length: medium


In a meadow of gold and green,
Where the daisies dance in a dream,
A boy and his pig, hand in trotter,
Amble beneath a sky serene.
Laughter rings out, as pure as the sun,
A melody spun from their hearts,
They frolic and play, without a care,
Their love as wide as the endless parts.
As the day turns to twilight, the boy
Strokes the pig's silky, soft hide,
Underneath the moon's gentle glow,
His friend, his confidant, his pride.
In this world of their joy, they find,
A bond, a connection, a kind,
That transcends words, and time, and space,
A testament to love, pure and divine.
Poem Title
My First Poem
Download Poem
About the Poetry Generator
How It Works
This app uses HuggingFace language models to generate beautiful poetry based on your input parameters:
Theme: The central subject or concept of your poem
Emotion/Mood: The emotional tone or atmosphere of your poem
Style: The literary tradition or movement influencing the language
Form: The structural pattern or traditional form of the poem
Length: How long the poem should be
Length Options
Short: Approximately 10 lines
Medium: Approximately 20 lines
Long: Approximately 30 lines



6. Conclusion:

The **AI Poetry Generator** project successfully implements an application that allows users to generate poetry based on their input. By leveraging **HuggingFace's GPT models**, the system is able to produce stylistically diverse, coherent, and thematically rich poems. The integration with **Streamlit** provides an intuitive interface for users to experiment with various themes, emotions, and poetic forms. The potential applications for this tool are vast, ranging from creative writing assistants to tools for lyric generation, poetry education, and even personalized greeting cards.

7. References:

- **HuggingFace Transformers Documentation**: Provides insights on using pre-trained models for text generation tasks.
- **OpenAI GPT Models Documentation**: Documentation for the GPT family of models, which serve as the core generative model for this project.
- **Streamlit Documentation**: A guide for creating interactive web applications with Streamlit.
- **Research papers on AI-driven creative writing and generative poetry**: Explore the advancements and methodologies in the realm of AI-generated literature and poetry.
- **Projects on AI-generated literature and poetry from GitHub**: Community projects and collaborations showcasing AI-generated creative works.