College Name: VIT Bhopal University
Student Name: Ishan Peshkar

GEN AI PROJECT PHASE 3 SUBMISSION DOCUMENT

Phase 3: Final Report and Submission

1. Project Title:

AI-Powered Poem Generator

2. Summary of Work Done

- ***Phase 1 – Proposal and Idea Submission (10 Marks):***

  In this initial phase, we identified the opportunity to create a tool for generating creative poetic text. The problem statement focused on assisting users with poetic inspiration and content creation. We proposed the development of an AI-Powered Poem Generator leveraging pre-trained language models. Key objectives defined were:
  - To understand and apply generative models for creative text generation in NLP.
  - To utilize pre-trained models (like GPT-2) from the Hugging Face transformers library to generate thematically relevant poems.
  - To design and implement an interactive user interface for topic input and poem display.

    A detailed proposal was submitted, outlining the problem, objectives, scope, technology stack, and expected outcomes.

- ***Phase 2 – Execution and Demonstration (20 Marks):***

  The second phase involved the core implementation of the AI Poem Generator. Key accomplishments included:
  - **Setting up the Python development environment and installing necessary libraries (Hugging Face transformers, PyTorch, Streamlit).**
  - **Developing the core poem generation logic using a pre-trained GPT-2 model, including prompt engineering and parameter tuning (max_length, temperature, num_beams, etc.).**
  - **Building an interactive web-based interface using Streamlit, allowing users to input a poem topic, select a model (e.g., GPT-2, GPT-2 Medium), and adjust generation parameters like poem length and creativity.**
  - **Implementing model caching (@st.cache_resource) in Streamlit for improved performance on subsequent model loads.**
  - **Testing the application with various topics to assess the quality, coherence, and creativity of the generated poems.**

    The complete code, alongside sample outputs and UI screenshots, was documented as part of this phase.

College Name: VIT Bhopal University
Student Name: Ishan Peshkar

3. GitHub Repository Link

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

🔗 **GitHub Repository – AI-Powered Poem Generator**

4. Testing Phase

4.1 Testing Strategy

The AI Poem Generator was tested to ensure functionality, quality of output, and user experience. The testing strategy focused on:

- **Input Handling: Ensuring the system gracefully handles various topic inputs (single words, short phrases, longer themes).**
- **Output Quality & Relevance: Evaluating the generated poems for coherence, thematic relevance to the input topic, creativity, and poetic structure (even if basic).**
- **Parameter Responsiveness: Verifying that changes to parameters like "Max Poem Length" and "Creativity (Temperature)" demonstrably affect the output.**
- **User Interface (UI) & User Experience (UX): Assessing the intuitiveness and responsiveness of the Streamlit interface.**
- **Error Handling: Checking how the system responds to potential issues (e.g., model loading errors, though caching mitigates this after the first load).**

College Name: VIT Bhopal University
Student Name: Ishan Peshkar

4.2 Types of Testing Conducted

1. **Functional Testing:**
    - **Verified that each UI element (text input, sliders, buttons, model selector) functions as expected.**
    - **Ensured that poem generation is triggered correctly and output is displayed.**
2. **Qualitative Evaluation (Manual Testing):**
    - **A variety of topics were input (e.g., "the ocean," "a lonely star," "winter silence," "a joyful memory," "the concept of time").**
    - **Generated poems were manually reviewed for:**
        - **Thematic consistency with the input topic.**
        - **Coherence and readability.**
        - **Elements of poetic language (e.g., imagery, rhythm, occasional rhyme).**
        - **Impact of different temperature and max_length settings.**
3. **Usability Testing:**
    - **Informal testing by interacting with the Streamlit application to assess ease of use, clarity of controls, and overall user satisfaction.**
4. **Performance Testing (Basic):**
    - **Observed the initial model loading time (for gpt2 and gpt2-medium).**
    - **Noted the poem generation speed after the model was loaded/cached.**

4.3 Results

- **Functionality: The application consistently generated poems based on user inputs and selected parameters. The Streamlit interface proved responsive.**
- **Output Quality:**
    - **The GPT-2 model (and gpt2-medium) generally produced poems that were thematically relevant.**
    - **Higher temperature values led to more abstract or unexpected (sometimes less coherent) outputs, while lower values produced more focused poems.**
    - **max_length effectively controlled the approximate length of the poems.**

- **The poetic quality varied; some outputs were surprisingly evocative, while others were more prosaic or occasionally nonsensical, which is expected from general-purpose LLMs without specific fine-tuning on poetry.**

- **Response Time: Initial model loading took some time (especially for gpt2-medium), but subsequent generations were quick due to Streamlit's caching. Poem generation itself was typically within a few seconds.**
- **User Interface: The Streamlit UI was found to be intuitive and easy to navigate.**

5. Future Work

While the AI Poem Generator successfully meets its initial objectives, several exciting avenues exist for future enhancements:

1. **Fine-Tuning on Poetry Corpora:**
   - Fine-tune a base model (like GPT-2 or a smaller, efficient model) on a large dataset of diverse poetry. This could significantly improve the stylistic quality, adherence to poetic forms (e.g., haikus, sonnets if desired), and understanding of poetic devices.
2. **Style and Mood Selection:**
   - Allow users to select a desired poetic style (e.g., "Limerick," "Free Verse," "Haiku") or a mood (e.g., "Joyful," "Melancholic," "Mysterious") to further guide the generation process. This might involve conditional prompting or different fine-tuned models.
3. **Enhanced Output Controls & Editing:**
   - Provide options for users to regenerate specific lines or stanzas they dislike.
   - Incorporate basic editing tools or suggestions (e.g., rhyme suggestions, synonym finders).
4. **Integration of Rhyme and Meter Constraints:**
   - Explore techniques to more explicitly control rhyme schemes and metrical patterns, which is a complex NLP challenge but would greatly enhance poetic output. This might involve constraint-based decoding or reinforcement learning.
5. **User Accounts and Poem Saving:**
   - Allow users to create accounts, save their favorite generated poems, and perhaps share them within the platform or externally.
6. **Deployment to a Public Platform:**
   - Deploy the application using services like Streamlit Cloud or Hugging Face Spaces to make it publicly accessible.

6. Conclusion

This AI-Powered Poem Generator project successfully demonstrates the application of Generative AI, specifically transformer-based language models, for creative text generation. Through the three phases, the project progressed from an initial concept to a functional

prototype with an interactive web interface. The system showcases the potential of LLMs to assist in creative endeavors like poetry writing, offering users a tool for inspiration and exploration. While the generated output reflects the capabilities of general pre-trained models, the project lays a solid foundation for future work involving specialized fine-tuning and more sophisticated poetic control.