**Project Proposal: Poet-Persona Generative AI: Fine-tuning a Language Model for Interactive Style-Specific Poetry Generation**

**1. Project Title:** Poet-Persona Generative AI: Fine-tuning a Language Model for Interactive Style-Specific Poetry Generation

**2. Project Focus:** Natural Language Processing (NLP) - Generative AI Models

**3. Project Goal:** The primary goal of this project is to develop an interactive web-based application that allows users to engage with a generative AI model that roleplays as a specific poet or adheres to a particular poetic genre. The core of the system will involve fine-tuning a pre-trained Large Language Model (LLM) to generate new poetry in the distinctive style of the chosen persona/genre, based on user prompts.

**4. Problem Statement & Motivation:** Generating creative text, particularly poetry, with stylistic control is a challenging and exciting frontier in generative AI. While modern LLMs can produce impressive text, maintaining a consistent and nuanced poetic style (e.g., the lyrical depth of Emily Dickinson, the free verse of Walt Whitman, or the structure of a Shakespearean sonnet) based on user interaction remains an area for focused exploration. This project aims to bridge the gap between general text generation and precise stylistic imitation in poetry, offering an engaging and educational tool for exploring literary styles and the capabilities of generative AI.

**5. Project Objectives:**

* **Objective 1: Dataset Curation & Preparation:** Select and prepare a high-quality dataset of poetry from specific authors or genres, ensuring proper formatting and metadata for fine-tuning.
* **Objective 2: LLM Fine-tuning:** Fine-tune a suitable pre-trained language model (e.g., GPT-2, LLaMA-2 7B) on the curated poetry datasets to instill the stylistic characteristics of chosen poets/genres.
* **Objective 3: Interactive User Interface (UI) Development:** Design and implement a user-friendly web interface that allows users to:
  + Select a "poet persona" (e.g., "Emily Dickinson," "Haiku Master").
  + Input a prompt or topic for the poem.
  + Receive generated poetry in the chosen style.
  + (Optional) Provide feedback on the generated output.
* **Objective 4: Persona-driven Prompt Engineering:** Explore and integrate prompt engineering techniques within the UI to guide the fine-tuned model's generation, reinforcing the chosen persona's style and responding meaningfully to user input.
* **Objective 5: Evaluation & Analysis:** Qualitatively evaluate the generated poetry for stylistic coherence, creativity, and adherence to the chosen persona/genre. Analyze the model's performance and identify areas for improvement.

**6. Technical Approach & Methodology:**

* **6.1. Data Collection & Preprocessing (Objective 1):**
  + **Source:** Utilize publicly available poetry datasets from sources like Project Gutenberg, Poetry Foundation archives (if legally permissible, or pre-scraped public domain datasets from Kaggle/Hugging Face), or the Hugging Face merve/poetry dataset.
  + **Selection:** Identify 2-3 distinct poet personas (e.g., a Romantic poet, a Modernist poet, a structured form like Haiku/Sonnet) or genres based on data availability and clear stylistic differences.
  + **Preprocessing:**
    - Clean text: Remove boilerplate, normalize whitespace, handle punctuation.
    - Format: Structure poems into a consistent format suitable for LLM input (e.g., <startofpoem> <line1> <line2> ... <endofpoem>).
    - Splitting: Divide data into training, validation, and test sets.
* **6.2. Language Model Selection & Fine-tuning (Objective 2):**
  + **Base Model:** Select a suitable pre-trained LLM.
    - **Option A (Accessible):** GPT-2 (small/medium) or DistilGPT-2 for easier local fine-tuning and faster iteration.
    - **Option B (More Powerful):** LLaMA-2 7B (via Hugging Face and requiring sufficient GPU resources) for potentially higher quality, though fine-tuning will be more demanding.
  + **Fine-tuning Strategy:**
    - **LoRA (Low-Rank Adaptation):** This will be the primary fine-tuning method to efficiently adapt the LLM to the new poetic style without retraining the entire model, reducing computational requirements.
    - **Training Loop:** Utilize the Hugging Face transformers library for training. Employ techniques like gradient accumulation and mixed-precision training if resources are limited.
    - **Hyperparameters:** Experiment with learning rates, batch sizes, and training epochs to optimize generation quality.
* **6.3. Interactive UI Development (Objective 3 & 4):**
  + **Framework:** Python-based web framework (e.g., Flask or Streamlit) for rapid prototyping and deployment.
  + **Backend:** The fine-tuned LLM will be loaded on the server-side. User prompts will be sent to the backend, and the generated poetry returned.
  + **Frontend:** Simple HTML/CSS/JavaScript, or Streamlit's built-in components for a clean user experience.
  + **Features:**
    - Dropdown/buttons for selecting poet personas.
    - Text area for user input.
    - Display area for generated poetry.
    - (Optional) "Generate again" button, prompt refinement suggestions.
  + **Persona-driven Prompting:**
    - Initial system prompt for each persona (e.g., "You are Emily Dickinson. Write a short poem about [topic] in your characteristic style. Use dashes frequently and explore themes of nature, death, and immortality.").
    - Combine system prompt with user input for model inference.
* **6.4. Evaluation & Analysis (Objective 5):**
  + **Qualitative Assessment:**
    - Human evaluation of generated poems by project members and potentially peers for:
      * **Stylistic Fidelity:** How well does it match the target poet/genre? (e.g., vocabulary, meter, rhyme scheme, thematic elements).
      * **Coherence & Creativity:** Is the poem meaningful and original?
      * **Fluency:** Is the language natural and grammatically correct?
  + **Quantitative (if applicable):**
    - Perplexity on a held-out test set (for training stability, not direct stylistic quality).
    - (Advanced/Stretch Goal): Develop simple metrics for stylistic elements like average line length, specific word frequencies, or rhyme scheme consistency if enough ground truth exists for a specific style.
  + **Ablation Studies:** Briefly analyze the impact of fine-tuning versus using only prompt engineering on the base model.

**7. Expected Outcomes & Deliverables:**

* **Fine-tuned LLM Model(s):** Saved model checkpoints for each poet/genre persona.
* **Interactive Web Application:** A functional web interface demonstrating the project's capabilities.
* **Generated Poetry Samples:** A portfolio of generated poems from various personas and prompts.
* **Project Report:**
  + Detailed explanation of methodology, challenges, and solutions.
  + Analysis of generated poetry, including qualitative evaluation results.
  + Discussion of the strengths and limitations of the approach.
  + Suggestions for future work.
* **Presentation/Demonstration:** A public presentation of the project, including a live demo of the application.

**8. Project Timeline (Semester-based, approximately 12-14 weeks):**

* **Weeks 1-2: Research & Setup:**
  + In-depth literature review on style transfer in NLP and poetry generation.
  + Selection of specific poets/genres and LLM.
  + Initial environment setup (Python, libraries, GPU access).
* **Weeks 3-4: Data Acquisition & Preprocessing:**
  + Collecting and cleaning poetry datasets.
  + Data formatting for fine-tuning.
* **Weeks 5-8: Model Fine-tuning:**
  + Experimenting with different fine-tuning parameters (LoRA ranks, learning rates).
  + Iterative training and preliminary qualitative assessment.
  + (Mid-semester check-in/progress report)
* **Weeks 9-11: UI Development & Integration:**
  + Designing and implementing the web application.
  + Integrating the fine-tuned models with the UI.
  + Initial testing of user interaction and generation.
* **Weeks 12-13: Evaluation & Refinement:**
  + Comprehensive qualitative evaluation of generated poetry.
  + Bug fixing and UI improvements.
  + Final model adjustments.
* **Week 14: Documentation & Presentation:**
  + Writing the final project report.
  + Preparing the project presentation and demonstration.

**9. Resources Required:**

* **Computational:** Access to a GPU (e.g., NVIDIA RTX 3090/4090, or cloud GPU instances like Google Colab Pro/AWS EC2) for fine-tuning LLMs, especially if using LLaMA-2.
* **Software:** Python 3.x, Hugging Face Transformers, PyTorch/TensorFlow, Streamlit/Flask, standard NLP libraries (NLTK, SpaCy for data cleaning if needed).
* **Data:** Publicly available poetry datasets.

**10. Potential Challenges & Mitigation:**

* **Data Scarcity for Niche Styles:** Some specific poets may not have enough readily available text for robust fine-tuning. *Mitigation: Focus on poets with larger bodies of work or generalize to broader genres.*
* **Computational Resources:** Fine-tuning even smaller LLMs can be memory and time-intensive. *Mitigation: Utilize LoRA, efficient training practices, or leverage cloud resources.*
* **Stylistic Nuance Capture:** It can be hard for models to perfectly replicate subtle poetic nuances beyond superficial features. *Mitigation: Extensive qualitative evaluation, iterative fine-tuning, and robust prompt engineering.*
* **Evaluation Subjectivity:** Assessing "poetic style" is inherently subjective. *Mitigation: Clearly define qualitative criteria, use multiple human evaluators, and focus on observable stylistic features.*

**11. Future Work (Beyond Semester Scope):**

* Integration of more advanced stylistic control parameters (e.g., rhyme scheme, meter, specific themes).
* Allowing users to "mix" styles from different poets.
* Developing a feedback loop where user ratings help refine the model.
* Deploying the application to a public server for wider access.
* Exploring interpretability techniques to understand *how* the model captures and applies poetic styles.