Task-Specific Generative Text API

1. **Project Title:** Fine-Tuned Generative Text API for [Specific Task] *(Students should replace "[Specific Task]" with their chosen application, e.g., "Product Descriptions", "Marketing Snippets", "Simple Email Drafting")*
2. **Project Goal/Objective:** To fine-tune a pre-trained generative language model for a specific, narrow text generation task and deploy this specialized model as a REST API service.
3. **Problem Statement/Background:** While large pre-trained language models are powerful, they often lack the specific style, tone, or domain knowledge required for specialized business tasks. Fine-tuning allows adapting these models to generate more relevant and useful text for targeted applications, automating content creation.
4. **Scope:**
    - **In Scope:** Defining a narrow text generation task; selecting an appropriate pre-trained model (e.g., GPT-2, T5-small, DistilBERT adapted for generation); sourcing or creating a suitable fine-tuning dataset; performing the fine-tuning process; evaluating the generated text quality; building a REST API endpoint that accepts a prompt/input data and returns generated text; containerizing the application; deploying the containerized API.
    - **Out of Scope:** Fine-tuning very large models (due to computational cost); building a sophisticated UI; real-time generation for chat applications; guaranteeing factual accuracy of generated text (focus is on style/format).
5. **Key Deliverables:**
    - Fine-tuning dataset (or documentation of source).
    - Fine-tuned generative model (code, saved model files).
    - Documented API specification (e.g., OpenAPI/Swagger).
    - A containerized (Docker) REST API application.
    - Deployed API endpoint accessible via URL, capable of generating text based on input.
    - Final technical report covering task definition, data, model selection, fine-tuning process, evaluation, API design, deployment, and example outputs.
    - Final presentation demonstrating the API service and discussing results.
6. **Potential Datasets to Get Started:**
    - **Task-Dependent:** Data needs to match the chosen task.
        - *Product Descriptions:* Use subsets of Amazon review/product data (linked previously), web scrapes of e-commerce sites.
        - *Marketing Copy:* Public advertising datasets (if available), company website text, marketing email examples.
        - *Email Drafting:* Public email datasets (e.g., Enron dataset - use ethically), synthetic examples.
    - **Hugging Face Datasets:** Explore the `datasets` library/hub for text data suitable for conditional text generation.
        - *Link:* https://huggingface.co/datasets

    - **Note:** Data quality and relevance to the specific task are crucial for successful fine-tuning. Creating a small, high-quality custom dataset might be necessary.
7. **Potential Technical Approach:**
    - **Model Selection:** Use Hugging Face `transformers` library. Choose a model feasible to fine-tune and serve (e.g., GPT-2, DistilGPT2, T5-small/base). Consider Canadian AI providers like Cohere if API access is feasible/available for experimentation (though fine-tuning might be via their platform).
    - **Fine-tuning:** PyTorch or TensorFlow. Requires careful setup of training loop and data loaders. May require GPU resources (consider cloud VMs with GPUs like AWS EC2 P/G instances, GCP Compute Engine GPUs, Azure NC-series VMs, or platforms like Google Colab Pro for experimentation).
    - **API Framework:** Flask or FastAPI.
    - **Containerization:** Docker. Must include model files and all dependencies. Model size is a key consideration.
    - **Cloud Platform:** Needs to support container deployment. Consider services that offer CPU or GPU options for inference depending on model size/speed requirements (e.g., SageMaker multi-model endpoints, Vertex AI Prediction, Azure ML Endpoints with appropriate instance types, or standard container services like Cloud Run/Fargate/Azure Container Apps possibly with GPU support if needed).
8. **Deployment Requirements & Tips:**
    - **Requirement:** The fine-tuned model must be deployed as a containerized REST API on AWS, GCP, or Azure. The API should accept input (e.g., a prompt or structured data) and return the generated text (JSON response). The endpoint URL must be functional.
    - **Tips for Students:**
        - **Separate Training & Deployment:** Fine-tuning often needs GPUs, while inference might work on CPUs (depending on model/latency needs). Plan resource usage accordingly. Fine-tune on a GPU instance, save the model, then build a *separate* inference container (potentially CPU-only) for deployment.
        - **Model Size vs. Resources:** Be realistic about model choice. Larger models generate better text but are harder/costlier to fine-tune and deploy. Test inference speed/memory usage locally.
        - **Hugging Face Hub:** Use the Hub to store fine-tuned models, making it easier to load them in the deployment container.
        - **Dockerizing Large Models:** Ensure model files are efficiently included in the Docker image (e.g., download during build or runtime). Be aware of image size limits on some platforms.
        - **Inference Optimization:** Explore techniques like model quantization or optimized inference runtimes (ONNX Runtime, TensorRT) if performance is critical (advanced topic).

        - **Cloud Inference Options:** Evaluate cloud options: CPU vs. GPU endpoints, serverless vs. provisioned instances based on cost, latency needs, and model size.
        - **Resource:** Leverage Hugging Face documentation/courses, cloud provider docs for GPU instances and inference endpoints. The `Deep Learning`, `NLP`, and `Cloud Computing for ML` courses are key.
9. **Potential Challenges:** Finding/creating good fine-tuning data; high computational cost/time for fine-tuning; managing large model files; optimizing inference speed/cost; evaluating generative output quality; potential for generating biased or harmful text (ethics).
10. **Success Metrics:** Deployed API functionality; quality/relevance of generated text for the defined task (qualitative evaluation, potentially automated metrics); successful fine-tuning process; clear documentation/presentation.
11. **Team Size / Duration:** 3-4 Students / 14 Weeks