Bien sûr, voici un plan récapitulatif que tu peux archiver pour répliquer cette architecture MoE dans un autre domaine.

Plan MoE (Phi-2 + LoRA) déjà réalisé
Préparer le socle
Répo ~/projects/soleam-photo
Venv .venv avec transformers, peft, datasets, accelerate
Script générique train_expert_lora.py (bf16, Trainer standard, tokenisation manuelle)
Script d'automatisation automate_expert_training.py
Script d'inférence moe_router_inference.py
Créer les datasets agents
30 fichiers agent_*.jsonl
Format commun : {"instruction", "response", "metadata":
{"agent","theme","gear","location","difficulty","keywords"}}
Balises utiles pour le routage (catégorie primaire + agent précis)
Former les adaptateurs LoRA
Depuis ~/projects/soleam-photo :source .venv/bin/activate
python automate_expert_training.py

Résultat : models/phi2_lora_agent_<nom>/final_lora_adapters
Configurer le routeur MoE
moe_router_inference.py
AGENT_MAP et LORA_BASE_PATH
route_query_hierarchical() → catégorie (prise de vue, matériel… ) puis agent (expo_iso, portrait…)
Chargement/caching du modèle base en bf16, attachement dynamique de l'adaptateur
Fonction generate_response(query) (prompt ### Instruction/Response)
Tester l'ensemble
Une fois 1-4 terminés :source .venv/bin/activate
python moe_router_inference.py

Vérifier que les logs affichent l'agent correctement routé et que la réponse exploite l'adaptateur.

Tu peux recycler ce plan pour un Smart Expert "métier X" : il suffit de reconstituer la table des agents (datasets, keywords) puis relancer les étapes 1-5.

Tu as désormais un plan reproductible pour n'importe quel autre métier :
Créer les JSONL d'agents
Par spécialité, fais 20–30 entrées instruction/response + metadata (agent, thème, difficulté, keywords…).
Pour les agents « culture/critique » de SOLÉAM, injecte les époques, artistes, maximes, etc.
Entraîner les LoRA automatiquement
Scripts : train_expert_lora.py (bf16, Hugging Face Trainer) et automate_expert_training.py.
Commande :cd ~/projects/<ton_nouveau_metier>
source .venv/bin/activate
python automate_expert_training.py

Chaque adaptateur est enregistré sous models/phi2_lora_agent_<nom>/final_lora_adapters.
Configurer le routeur hiérarchique
moe_router_inference.py: mappe les mots-clés (niveau 1 catégorie, niveau 2 agent).
Ajoute une post-logique response_clean += … pour rappeler ton ADN de marque (ex. coupler à un agent Studio/Produit).
Tester
python moe_router_inference.py → vérifie que chaque question est routée vers le bon expert et qu'un adaptateur LoRA est chargé.
Ce canevas s'applique à toute nouvelle architecture « métier » : il suffit d'écrire les datasets spécialisés, de lancer l'automatisation et de définir le routage + tonalité dans l'inférence.