Task name

Soham Kotkar— Lightweight Online Adapter + RL Pipeline (MCP-enabled)

One-line goal

Enable fast, incremental multilingual quality improvements without big local downloads: stream corpora from remote MCP connectors, train tiny adapters/LoRA on the 4050 (8-bit/FP16 + gradient-accum), and run RL policy updates in cloud; expose simple inference + control endpoints.

Deliverables (3 days, lightweight)

1. adapter_service/ with scripts to train/apply LoRA-style adapters using streaming datasets (no full dataset download).
2. REST endpoints:
    - POST /adapter/train-lite — starts a small local adapter update job (uses streaming subset).
    - POST /generate-lite — inference endpoint using base LM + adapter.
    - GET /adapter/status/{job_id} — job progress + metrics.
3. Config: mcp_connectors.yml (S3/http/Qdrant stream sources) and adapter_config.yaml.
4. Lightweight RL hook scaffold: rl/collect.py logs episodes to NAS/cloud for remote trainer.
5. Smoke results: run 10 multilingual prompts (selected from MCP stream) and commit smoke_results.md.
6. Short how-to: commands to run locally and how to trigger cloud RL job.

Acceptance criteria

- Adapter fine-tune runs on 4050 with a small batch (batching + 8-bit) and completes within a few hours (not days).
- generate-lite returns sensible, language-correct output for 10 test prompts across languages present in MCP.
- No local corpus >100MB is required; streaming works.
- RL logs are being pushed to NAS / S3 for cloud trainer to consume.

High-level approach & technical choices

- Use PEFT / LoRA + bitsandbytes 8-bit (low VRAM) and accelerate for training.
- Stream corpora via Hugging Face datasets with streaming=True, or via MCP connectors that yield lines/samples. No full download.
- Use small adapter updates (few epochs, low batch) + gradient accumulation to fit 4050 VRAM.
- Inference uses the base decoder model + adapter merged at runtime.
- RL: local agent collects (prompt, output, auto-metric reward) and uploads episodes for cloud PPO/PPO2 trainer (runs on Yotta/office GPUs). Local 4050 only records and optionally runs tiny policy updates (bandit/Q-table) for immediate improvements.
- Tokenizer: reuse Soham's SentencePiece; allow on-the-fly tokenization via API.

Exact commands & snippets (copy-pasteable)

1. Create branch

```
git checkout -b task_adapter_mcp
git commit --allow-empty -m "start: lightweight adapter + MCP
streaming"
git push -u origin task_adapter_mcp
```

2. Install (local 4050)

```
python -m venv .venv && source .venv/bin/activate
pip install -r requirements-lite.txt
# requirements-lite.txt should include: accelerate,
transformers, bitsandbytes, peft, datasets
```

3. Run a tiny adapter training (streaming subset)

```
# example: train_adapt.py reads mcp_connectors.yml and
streams N samples
python adapter_service/train_adapt.py \
```

```
  --model_name_or_path gpt-small-base \
  --output_dir adapters/gurukul_lite \
  --num_epochs 3 \
  --per_device_train_batch_size 1 \
  --gradient_accumulation_steps 8 \
  --use_8bit True \
  --streaming_source "hf:your_remote_dataset" \
  --max_train_samples 2000
```

4. Launch inference API (FastAPI)

```
uvicorn adapter_service.api:app --host 0.0.0.0 --port 8100 --
reload
# POST /generate-lite -> {"prompt":"...", "lang":"hi"}
```

5. Trigger cloud RL trainer (manual)

```
# upload episodes to S3/NAS
python rl/collect.py --upload-path s3://gurukul-rl/episodes/
# then trigger cloud job (Vijay/Yotta) via provided script:
bash rl/trigger_cloud_trainer.sh --episodes s3://gurukul-rl/
episodes/
```

Minimal file plan (what Soham should push)

```
adapter_service/
  train_adapt.py           # streaming LoRA trainer
  api.py                   # FastAPI wrapper (generate-lite,
train-lite trigger)
  model_utils.py           # load base model + adapter merge
  requirements-lite.txt
mcp_connectors.yml         # remote data sources
adapter_config.yaml
rl/
  collect.py
  upload_helper.py
test_prompts/
  prompts_10.json
```

```
smoke_results.md
README.md
```

Who to coordinate with (quick)

- Vijay — grant access to Yotta for cloud RL trainer and NAS path for episodes.
- Nisarg — ensure /compose and BHIV trace_id can call generate-lite.
- Karthikeya — confirm language tag / TTS compatibility (audio format).
- Nipun — Qdrant/MCP connector endpoints for streaming KB chunks.

Timeline (aggressive, start now)

- Day 0 (today, 2–4 hrs): branch + repo scaffold, mcp_connectors.yml, requirements-lite.
- Day 1 (6–8 hrs): implement train_adapt.py streaming LoRA flow, local run on 4050 with small sample.
- Day 2 (4–6 hrs): FastAPI wrapper + generate-lite + smoke tests (10 prompts), push smoke_results.md.
- Day 3 (optional): RL collect + cloud trigger template + docs and PR.

Quick operational tips for Soham

- Use --use_8bit True (bitsandbytes), --gradient_accumulation_steps to emulate larger batch.
- Limit max_train_samples to a few thousand for quick adapter updates; iterate often.
- Use streaming splits with max_train_samples instead of full dataset.
- Persist adapters to NAS so others can pull & test.