Soham Kotkar – Multilingual Tokenization & Model Integration (56 hours)

Goal:

Build and integrate a multilingual tokenizer and inference API for Hindi, Sanskrit, Marathi, and English, ensuring smooth plug-in with the Black Hole DB and Vaani (Blackhole TTS) systems. By Day 7, the language model must process user input in any of the four languages and output grammatically correct text in the chosen language, without any external knowledge baked in.

Day-by-Day Plan

Day 1 – Understanding & Setup (8 hrs)

- Review Hugging Face transformers docs for tokenizer customization and multilingual model integration.
- Study how tokenization impacts different scripts (Devanagari vs Latin).
- Learning Resources:
    - Hugging Face Tokenizers Course
    - YouTube: "Custom Tokenizers with HuggingFace" – by AssemblyAI (15 min)
    - Wikipedia: Devanagari script rules for Hindi, Sanskrit, Marathi.
- Install and test tokenizers, sentencepiece, transformers.

Day 2 – Multilingual Tokenizer Creation (8 hrs)

- Create a SentencePiece tokenizer that supports all 4 languages with proper handling of ligatures.
- Ensure proper Unicode normalization for Devanagari.
- Train tokenizer on provided parallel corpora (clean text in 4 languages).
- Save vocab and merge files in a sharable format.
- Learning Resource: Google SentencePiece GitHub

Day 3 – Base Model Integration (8 hrs)

- Select a light, open-source decoder-only model (e.g., GPT-NeoX, BLOOM-560M) with no domain knowledge.

- Replace its tokenizer with the custom multilingual tokenizer.
- Test with dummy sentences in all 4 languages.
- Verify that tokenization + detokenization is lossless.

Day 4 – Language Identification & Routing (8 hrs)

- Implement a simple language detection module (fastText or langdetect).
- Route detected language to correct tokenizer encoding flow.
- Begin designing an inference wrapper to accept API calls from KB.
- Learning Resource: fastText Language Identification

Day 5 – API Development & Integration Hooks (8 hrs)

- Build REST API endpoints for:
    - /tokenize – returns token IDs.
    - /generate – returns generated text.
    - /language-detect – returns detected language.
- Ensure stateless design so KB can call it anytime.
- Test with Gurukul sample queries in all 4 languages.

Day 6 – Fine-Tuning & RLHF Prep (8 hrs)

- Apply small fine-tuning to improve grammar and sentence structure using bilingual corpora.
- Prepare model for RLHF phase (hand over to Abhishek for reward model connection).
- Test switching between languages mid-conversation.

Day 7 – Full Integration & QA (8 hrs)

- Connect API with Alpha/Core knowledge base mock endpoint.
- Test multilingual Q&A loop:

    - User → Multilingual LM → KB → LM → Response in user language.
- Fix tokenization errors, latency issues.
- Deliver full code + integration docs.

Deliverables by Day 7

- Multilingual tokenizer (Hindi, Sanskrit, Marathi, English).
- Integrated base LM (decoder-only) with tokenizer.
- REST API for inference and tokenization.
- Language detection and routing module.
- Tested integration with Gurukul/Uniguru KB endpoints.

Contact Details for Integration with other members of this task will be provided on day 4 .

Learning & References

- Model Architecture Basics: YouTube – The Illustrated Transformer
- Fine-tuning Hugging Face models: HF Course – Fine Tuning
- Indic NLP Library: GitHub
- AI4Bharat Corpora: ai4bharat.org
- Language Detection: fastText Language ID
- Dockerizing NLP Models: Docker + FastAPI Guide

(Note: if links are broken please GPT the terms for more links.)