Karthikeya Task 5: Multilingual Personalisation with LLM + TTS Integration (Phase 1)

Duration: 6 Days

Objective:

Upgrade the Vaani Sentinel-X system from simulated multilingual previews to actual AI-powered content adaptation using LLM translation APIs, metadata-driven voice tagging, and first-pass TTS synthesis simulation.

Components to Build:

1. Translation Agent

- Implement translation using an LLM or API (use OpenAI / Gemini / huggingface transformers).
- Input: content_metadata.json with content text.
- Output: translated_content.json (generate for 10 Indian + 10 global languages).
- Add confidence score to translation output.

2. Personalization Agent

- Read user_profiles.json.
- Modify content tone based on user preferences (e.g., more formal, devotional, casual).
- Use prompt engineering with LLM to simulate tone variation.

3. Voice Tag Selector (Advanced)

- Extend language_voice_map.json.
- Add tone-voice mapping rules.
  Example:
  If tone = “devotional” → voice_tag = “hindi_female_devotional_1”
- Allow fallback to default tone-neutral voice.

4. TTS Simulation Layer

- Do not generate actual audio yet.
- Instead, create tts_simulation_output.json that includes:
    - language
    - tone
    - voice_tag
    - dummy audio URL/path (simulate only)

5. Platform Publisher Update

- Extend publisher_sim.py to read translated_content.json + tts_simulation_output.json.
- For each post preview:
    - Include language, tone, voice_tag, dummy audio path.

6. Weekly Adaptive Hook Prototype

- Read engagement signals from analytics_db.json.
- Adjust tone/language suggestions for next batch based on top 3 performing posts.
- Output: weekly_strategy_recommendation.json.

File/Folder Additions:

vaani-sentinel-x/

├── agents/

│ ├── translation_agent.py

│ ├── personalization_agent.py

│ └── tts_simulator.py

├── data/

│ ├── translated_content.json

│ ├── tts_simulation_output.json

│ └── weekly_strategy_recommendation.json

├── config/

│ └── updated_language_voice_map.json

Deliverables:

- 20 translated content outputs (translated_content.json).
- Personalized tone variations for each post.
- Voice-tag assignments per post with tone mapping.
- Simulated TTS JSON output.
- Weekly strategy JSON showing adaptation logic.
- Demo walkthrough video showing the full flow.
- Code and updated repo structure.

Notes:

- Use existing metadata pipeline as base.
- Keep translation/tone prompts simple, not full NLP yet.
- Prioritize clean modular code to support later RL fine-tuning.