Project Overview

Objective: Start from an open-source neural text-to-speech (TTS) checkpoint and customize it to produce authentic Bangladeshi Bangla speech. You may choose the model family that best fits your experience—XTTS v2 is strongly preferred, XTTS v1 is a close second, and VITS-based checkpoints (for example, bangla-speech-processing/bangla_tts_female) are acceptable if you already rely on them.

Duration: 5 Days

Claude Code Integration: You may use Claude Code or any similar assistant to speed up prototyping and debugging.

Technical Requirements

Prerequisites
- Strong Python skills and experience with PyTorch or TensorFlow
- Familiarity with speech synthesis or ASR pipelines
- Understanding of transfer learning and fine-tuning
- Working knowledge of Bangladeshi vs Indian Bengali phonetic differences

Core Tasks

Task 1: Environment Setup & Baseline Exploration
- Work in a lightweight environment (Google Colab, Kaggle, or similar). Free GPU tiers usually offer limited VRAM, so plan for batch sizes of 1–2, gradient accumulation, and short training sessions.
- Install only the packages you truly need (for example: torch, torchvision, torchaudio, coqui-tts or xtts, librosa, datasets, phonemizer, accelerate, wandb/tensorboard, numpy, matplotlib, seaborn).
- Load your chosen checkpoint (you could use XTTS v2/v1 or VITS Multilingual) and generate short baseline samples to understand pronunciation, timbre, and latency. Document model architecture, expected inputs, tokenizer/phonemizer behaviour, and any constraints you observe.

Task 2: Data Acquisition & Curation
- Smaller-duration clips (2–8 seconds) are easier to fit within limited GPU memory and tend to stabilize training faster. Prefer corpora that already provide short, well-trimmed utterances.
- You may use any open-source dataset. OpenSLR Bangla corpora (for example OpenSLR 53 and OpenSLR 54, particularly the Bangladesh-focused subsets) are a good place to start. You

may also explore Common Voice Bangla, Bengali.AI, or other publicly available Bangladeshi Bangla resources.
- Create a simple ingestion pipeline that checks sampling rate, duration distribution, speaker metadata, and text quality. Normalise text (punctuation, numerals, English loanwords) and ensure audio meets your model's requirements.
- If you mix datasets, keep track of data provenance, licensing, and accent labels so you can reason about results later.

Task 3: Accent Classification & Similarity Evaluation
- Build a lightweight system that helps you compare accents before and after fine-tuning. At a minimum, design a pipeline that extracts features (for example, MFCCs, mel-spectrogram stats, or embeddings from a small pretrained model) and groups data by accent or speaker.
- Use this pipeline to (a) curate Bangladeshi-accent training data and (b) evaluate generated audio. You may start with heuristics or clustering, then iterate toward a supervised classifier once you have labels. The ability to re-run evaluation and compare multiple training runs is a key deliverable.

Task 4: Training / Fine-Tuning Strategy
- Start with simple experiments in Colab/Kaggle; most free GPUs will let you train for 1–2 hours. Expect to pause/resume sessions. Checkpoints should be written frequently to cloud storage (for example, Google Drive, Hugging Face Hub, or GitHub LFS).
- Use tiny batches (1–2) with gradient accumulation. Mixed precision and smaller audio segments (trim, chunk, or random crop) will help you stay within memory limits.
- Prioritize getting a stable training loop running over exhaustive hyperparameter sweeps. Show us how you approach the problem: curriculum you tried, layers you chose to freeze/unfreeze, how you balanced learning rates for the acoustic model vs vocoder, etc.
- Document any difficulty you encounter (for example, GPU memory, convergence issues, or phoneme coverage) and how you mitigated it.

Task 5: Evaluation & Iteration
- Your scoring will be based primarily on approach and evaluation depth. We want to see how you reason about accent quality, intelligibility, and similarity to Bangladeshi references.
- Use your accent/similarity pipeline to compare checkpoints. For example, quantify mel cepstral distortion against validation samples, compute speaker embedding distances, or run AB tests with labelled listeners (if available). Highlight which run you would ship and why.
- Capture qualitative observations (mispronunciations, prosody issues, noise) and suggest targeted fixes for the next iteration.

Task 6: Deployment Considerations
- Describe how you would package the model for inference. This could involve exporting to ONNX, running streamlined inference with XTTS/Coqui, or exposing a FastAPI/Gradio/Streamlit service.

- Outline latency expectations, GPU/CPU requirements, and how you might handle streaming or low-resource environments. Even a short paragraph with trade-offs is valuable.

Deliverables

1. Fine-Tuned Model Artefacts
   - Training/evaluation scripts and notebooks (Colab links are acceptable if reproducible)
   - Checkpoints or Hugging Face Hub repo links (if upload is feasible)
   - Logs/metrics demonstrating at least two training attempts

2. Data & Preprocessing Notes
   - List of datasets used, with rationale for inclusion
   - Statistics on duration, speaker mix, and accent coverage
   - Preprocessing scripts (text normalisation, audio trimming, alignment helpers)

3. Evaluation Package
   - Accent/similarity pipeline code or notebooks
   - Objective metrics (e.g., MCD, F0 correlation, embedding distances) or clear reasons why a proxy was chosen
   - Subjective evaluation plan or results (listener feedback, AB tests, qualitative table)

4. Technical Report
   - Summary of experiments, model choices, training settings, and results
   - What worked, what failed, and how you would iterate with more time or compute
   - Thoughts on deployment and productisation

How We Evaluate

- Approach & Reasoning (35%): How you frame the problem, choose models/datasets, and adapt to resource limits
- Accent Evaluation Strategy (30%): Quality of your accent/similarity pipeline and how you compare runs
- Implementation & Reproducibility (25%): Code quality, use of Colab/Kaggle/free GPU best practices, clarity of documentation
- Deployment Insight (5%): Practical understanding of how to serve the model in production
- Polish & Communication (5%): Clarity of reporting, structure, and storytelling

Guidelines & Tips

- Start with short experiments to validate the pipeline before running longer fine-tunes.
- Keep audio samples short (<8 seconds) when possible; trim silence to help models converge.

- Use small validation sets that capture diverse accents and phonetic coverage so you can iterate quickly.
- If training diverges, fall back to a smaller learning rate, freeze more layers, or revisit data cleaning.
- You are welcome to utilize any helper notebooks or open-source tools, provided you explain their functions.

Outcome

By the end of the five days, you should have:
- A clearly documented path for adapting an open-source XTTS/VITS-style model to Bangladeshi Bangla
- Evidence of progress toward better accent fidelity, even if the model is not perfect
- A reproducible evaluation framework to judge future iterations
- Practical notes on packaging the work for production deployment

We are most interested in your problem-solving skills, your understanding of speech synthesis trade-offs, and how you assess your own work under tight compute constraints. Good luck, and enjoy the building process!