Software Documentation

Release 0.0.

Ai Dev Challenge — Emotional Speech Generation

Amsal Pardamean

03 November 2025

list of contents

Part A — System Design

1. **Pipeline overview** — what are the main steps from text input to final audio?
   a. Text Input & Preprocessing
      User action: User input raw text from the user.
      Process:
      - Normalize raw test input, convert numbers to words, fix punctuation
      - Segmentasi teks panjang menjadi kalimat pendek agar hasil TTS lebih stabil.
      - Extraction of basic linguistic information: phonemes, punctuation, intonation, and sentence context (e.g. interrogative vs. narrative or imperative sentences).

      Output: clean text ready to be converted into phonetic representation.

   b. Step 2. Linguistic & Acoustic Feature Extraction
      Convert text into the smallest sound unit.
      Add prosodic features such as::
      - Target pitch (intonation)
      - Duration (speech speed)
      - Energy (loudness/softness)

      Utilizes a grapheme-to-phoneme (G2P) converter as a model

      Graphemes are letters or groups of letters (like "sh" or "ai") that represent a sound, while phonemes are the individual sounds in a language. For example, the word "cat" has three graphemes ("c", "a", "t") but three phonemes ("k", "æ", "t").

      Output: phoneme sequence + basic prosodic features

   c. Emotional Conditioning / Control
      At this stage, the system determines the target emotion based on the desired output and controls it using several methods (e.g., neutral, sad, excited).
      - Add an emotion embedding vector to the model input.
      - Use style tokens for prosody and tone control.
      - Or mimic the prosody of a reference audio (emotion transfer)

      Output**:** fonem + prosody + emotion conditioningDddd

   d. Acoustic Model (Text → Spectrogram)

      Major TTS models (e.g., FastSpeech2, VITS, or Tacotron2) generate a mel-spectrogram from text and emotion input. A spectrogram is a visual representation of audio frequencies over time.

      Here, the model learns to combine linguistic content and emotional expression into an acoustic form.

      Output: mel-spectrogram emosional.

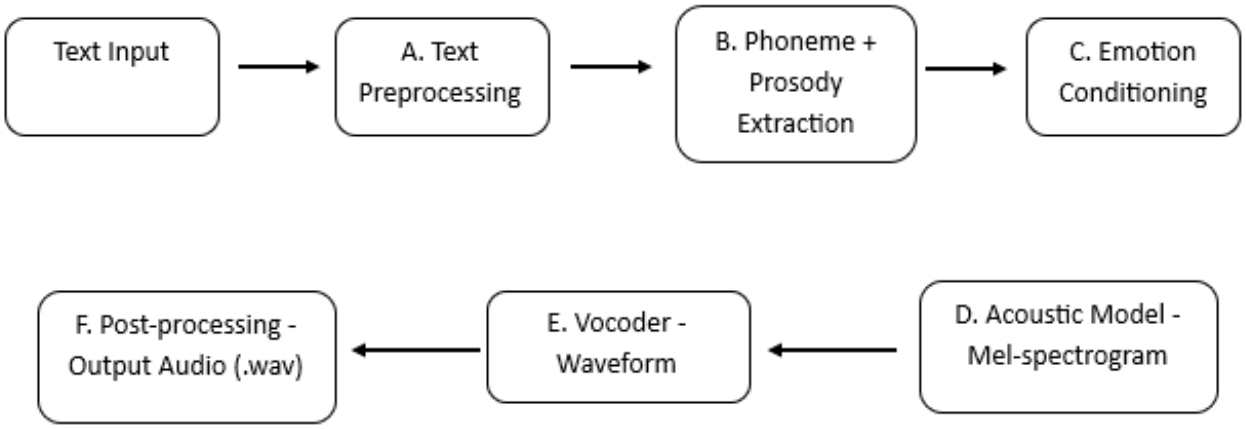
e. Vocoder (Spectrogram → Waveform)
The vocoder converts the mel-spectrogram into an audio signal (sound wave).
Common models:
HiFi-GAN — fast and realistic.
WaveGlow, Parallel WaveGAN, or DiffWave.
This stage determines the final naturalness of the voice.

Output: raw audio file (waveform) with target emotion.

f. Post-processing & Output
Normalize the volume, remove noise, and apply enhancements (e.g., dynamic range compression).
Save the result to .wav or another audio format.
(Optional) Add small effects to enhance the atmosphere (e.g., a light ambiance for documentaries).

Final Output: .wav audio file with emotional and natural speaking style.

Table Process:



2. **Models** — which approaches/models could be used for expressive TTS?

   FastSpeech 2 due to its faster architecture, explicit prediction of duration, pitch, and energy, so it is fast and stable, has the advantage of real time and can be controlled directly for expression with prosody parameters, even though it uses additional parameters, it is more stable for long texts.

3. **Emotion control** — how to make the voice sound *neutral*, *excited*, *sad*, etc.?
   Parameters tuning:
   Pitch: determines the pitch (higher → happier)

Energy: determines the strength of the voice (higher → more enthusiastic)
Duration: determines the speech rate (slower → sadder/calmer)

FastSpeech2 has a pitch predictor and an energy predictor, which can be modified during inference:

pitch_scale = 1.2 → higher → happier
energy_scale = 0.8 → softer → sadder

4. Data requirements — what dataset(s) would you need?
   emotional speech corpus or emotional speech audio because it has various audio with various emotions

5. **Evaluation** — how would you measure success?
   1. MOS: Target: MOS ≥ 4.0 for results closest to human voice.
   2. Emotion Classification Accuracy (Human Judgment), Target: ≥ 80% of evaluators recognize the correct emotion.
   3. Duration Consistency: Increasingly long but remains stable

6. **Deployment** — how would you make it usable as a “one button” tool?

   API + simple frontend

   Example API:

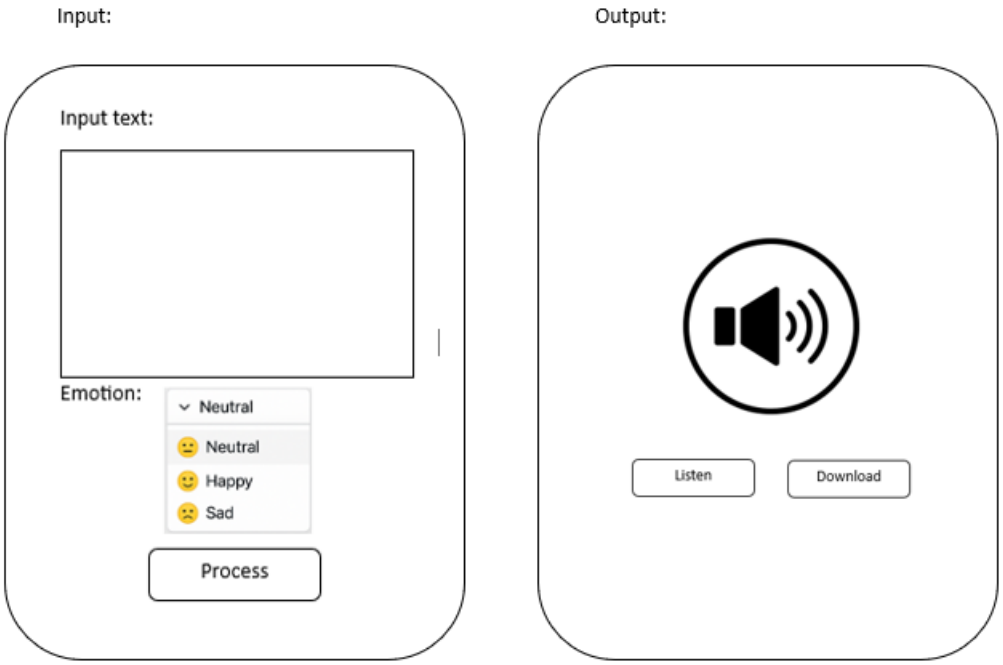
   {

   "text": "Hello world!",

   "emotion": "happy/sad/neutral"

   }

   Example Frontend:



7. **Challenges & limitations** — what are the hardest parts, and how would you mitigate them?
   Limited Emotional Speech Data

   High-quality emotional speech datasets (especially for documentary/narrative styles) are still limited.
   Public datasets (e.g., EmoV-DB, RAVDESS) are small and rarely acted upon.
   Emotional diversity is lacking.

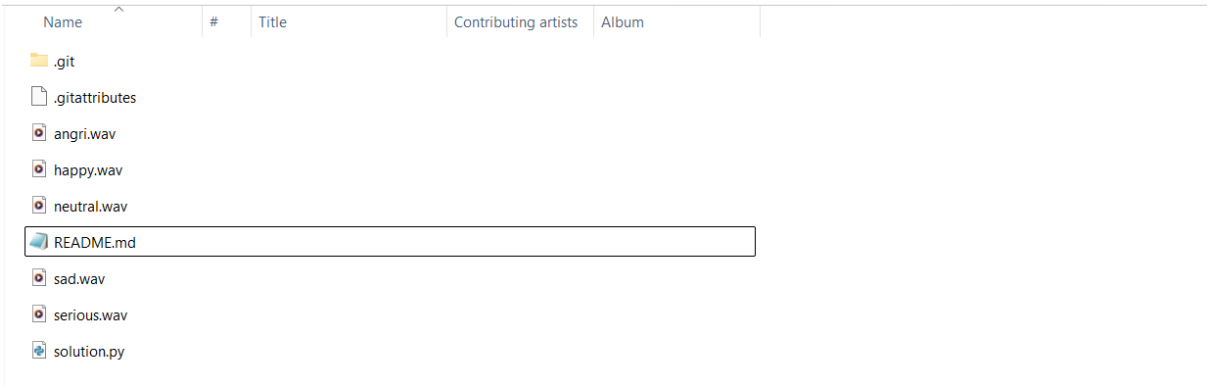
   Mitigation:
   Collecting and combining datasets is the best choice because the more training data there is, the greater the success rate in testing or implementation.

2. Part B — Prototype Coding Task

1. Github Link: https://github.com/amsalpardamean/ESG_Test
2. Setup instruction:
   Installed Python 3.9+ is recommended.

   Requirements:
   pip install pyttsx3 librosa soundfile

3. Directory:



4. How to run and logging on command line:
   a. Neutral speech(default)

      python solution.py "Hello world" neutral.wav

```
(Data Amsal Pardamean) D:\Data Amsal Pardamean\Test_interview\ESG_test>python solution.py "Hello world" neutral.wav
Convert success and audio saved: neutral.wav
```

   b. Happy

      python solution.py "Hello world" happy.wav happy

```
(Data Amsal Pardamean) D:\Data Amsal Pardamean\Test_interview\ESG_test>python solution.py "Hello world" happy.wav happy
Convert success and audio saved: happy.wav
Check input voice emotion: happy
Convert to happy voice
Applied 'happy' style effect
```

   c. Sad

      python solution.py "Hello world" sad.wav sad

```
(Data Amsal Pardamean) D:\Data Amsal Pardamean\Test_interview\ESG_test>python solution.py "Hello world" sad.wav sad
Convert success and audio saved: sad.wav
Check input voice emotion: sad
Convert to sad voice
Applied 'sad' style effect
```

   d. Angri

      python solution.py "Hello world" angri.wav angri

```
(Data Amsal Pardamean) D:\Data Amsal Pardamean\Test_interview\ESG_test>python solution.py "Hello world" angri.wav angri
Convert success and audio saved: angri.wav
Check input voice emotion: angri
Applied 'angri' style effect
```

   e. Serious

      python solution.py "Hello world" serious.wav serious

```
(Data Amsal Pardamean) D:\Data Amsal Pardamean\Test_interview\ESG_test>python solution.py "Hello world" serious.wav seri
ous
Convert success and audio saved: serious.wav
Check input voice emotion: serious
Convert to serious voice
Applied 'serious' style effect
```