**THE UNIVERSITY OF DANANG**
**DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY**
**FACULTY OF INFORMATION TECHNOLOGY**

# GRADUATION PROJECT THESIS

**MAJOR: Information technology**

**SPECIALTY: High-Quality Enterprise Collaboration Program**

**PROJECT TITLE:**

## Personalized Speech Generation Using Voice Cloning

**Instructor: Assoc. Prof., Ph.D. PHAM CONG THANG**
**Student: DO NHAT HOANG**
**Student ID: 102210057**
**Class: 21TCLC_DT1**

**Da Nang, 07/2026**

**THE UNIVERSITY OF DANANG**

**DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF INFORMATION TECHNOLOGY**

# GRADUATION PROJECT THESIS

**MAJOR: Information technology**

**SPECIALTY: High-Quality Enterprise Collaboration Program**

**PROJECT TITLE:**

## Personalized Speech Generation Using Voice Cloning

**Instructor: Assoc. Prof., Ph.D. PHAM CONG THANG**

**Student: DO NHAT HOANG**

**Student ID: 102210057**

**Class: 21TCLC_DT1**

**Da Nang, 07/2026**

DA NANG UNIVERSITY

**UNIVERSITY OF SICIENCE AND TECHNOLOGY**

FALCUTY OF INFORMATION TECHNOLOGY

**THE SOCIALIST REPUBLIC OF VIETNAM**

Independence - Freedom - Happiness

# GRADUATION PROJECT COMMENT

## I. General information:

1. Student name: Do Nhat Hoang
2. Class: 21TCLC_DT1 Student ID: 102210057
3. Topic title: Personalized Speech Generation Using Voice Cloning
4. Instructor: Pham Cong Thang Academic title/ degree: Assoc. Prof., Ph.D

## II. Reviews of graduation project

1. About the urgency, novelty, usability of the topic: (2 points)

……

……

2. About the results of solving the tasks required by the project: (4 points)

……

……

3. About the form, structure and layout of the graduation project: (2 points)

……

……

4. The topic includes scientific value/article/problem solving of the enterprise or school: (1 point)

……

……

5. Existing shortcoming need to be supplemented or modified:

……

……

## III. Spirit and attitude of the student (1 point):

……

……..

## IV. Evaluation:

1. Evaluation point: …./10
2. Suggest: Defense permitted/ Edit to defend/ Defense not permitted

Da Nang, date……month 07, year 2026

**Instructor**

# SUMMARY

Topic: Personalized Speech Generation Using Voice Cloning

Student name: Do Nhat Hoang

Student ID: 102210057 Class: 21TCLC_DT1

This capstone project titled "Personalized Speech Generation Using Zero-Shot Voice Cloning for Vietnamese" focuses on adapting a modern speech generation model to synthesize Vietnamese speech while preserving the speaker characteristics from a short reference audio sample. Voice cloning has become an important direction in Speech AI due to its potential applications in audiobooks, podcasts, virtual assistants, digital content creation, education, and accessibility tools.

Vietnamese zero-shot voice cloning is a challenging task because Vietnamese is a tonal language with complex pronunciation, rhythm, and prosodic patterns. In addition, many traditional voice cloning systems rely on static speaker embeddings, which may not fully represent the dynamic characteristics of a speaker's voice. Therefore, this project investigates an autoregressive audio language modeling approach as a more flexible direction for personalized Vietnamese speech generation.

The proposed approach is based on **NeuTTS-AIR**, an autoregressive audio language model that generates speech through discrete audio token prediction. To adapt the model to Vietnamese, this project constructs and processes a multi-source Vietnamese speech dataset from VIVOS, INFORE, viVoice, and PhoAudiobook. The data is processed through audio normalization, text preprocessing, grapheme-to-phoneme conversion, NeuCodec-based audio token extraction, and tokenization. A two-stage adaptation strategy is then applied, including initial lm_head fine-tuning and deeper adaptation using Global LoRA and full fine-tuning.

In addition to model adaptation, the project develops a complete end-to-end voice cloning application. The system includes a Flutter mobile application, a FastAPI backend, a database layer, cloud storage, asynchronous task processing, and a cloud GPU inference service. This demonstrates not only the technical feasibility of Vietnamese zero-shot voice cloning, but also its practical integration into a usable software system.

The adapted models are evaluated using objective metrics such as Word Error Rate, Character Error Rate, Speaker Embedding Cosine Similarity, and Mel Cepstral Distortion. Experimental results show that the proposed adaptation strategy improves Vietnamese speech generation quality compared with the initial adaptation stage.

Functional testing also confirms that the main application workflows, including user authentication, voice profile creation, speech generation, audio playback, and generated audio management, operate successfully.

**Structure of the Report:**

- **Chaper 1: PROJECT OVERVIEW**
  This chapter presents the overview of the topic, reasons for choosing the topic, project objectives, scope, methodology, technologies used, main contributions, and the overall structure of the report.
- **Chapter 2: THEORETICAL BACKGROUND**
  This chapter introduces the theoretical foundations related to speech synthesis, zero-shot voice cloning, neural audio codecs, autoregressive audio language models, LoRA, and the software architecture used in the system.
- **Chapter 3: PROPOSED METHOD**
  This chapter describes the proposed method for adapting NeuTTS-AIR to Vietnamese zero-shot voice cloning, including the baseline model, input-output representation, two-stage fine-tuning strategy, and training optimization techniques.
- **Chapter 4: EXPERIMENTAL RESULTS**
  This chapter presents the dataset preparation, experimental setup, evaluation metrics, training results, model comparison, and voice cloning evaluation on seen and unseen speakers.
- **Chapter 5: SYSTEM ANALYSIS AND DESIGN**
  This chapter presents the design of the end-to-end voice cloning system, including system architecture, database design, use case analysis, sequence diagrams, functional testing, and application demonstration.
- **CONCLUSION**
  This final section summarizes the achieved results, discusses the project limitations, and suggests future development directions.

DA NANG UNIVERSITY
**UNIVERSITY OF SICIENCE AND TECHNOLOGY**
FALCUTY OF INFORMATION TECHNOLOGY

**THE SOCIALIST REPUBLIC OF VIETNAM**
Independence - Freedom - Happiness

# GRADUATION PROJECT REQUIREMENTS

Student Name: Do Nhat Hoang Student ID : 102210057

Class: 21TCLC_DT1 Faculty: Information Technology

Major: High-Quality Enterprise Collaboration Program

*Topic title: Personalized Speech Generation Using Voice Cloning*

*Project topic : ☐has signed intellectual property agreement for final result*

*Initial figure and data:*

……..……..…….....……
……..……

*Content of the explanations and calculations:*

List of pictures
List of tables
List of symbol, acronym
Chapter 1: Project overview
Chapter 2: Theoretical background
Chapter 3: Proposed method
Chapter 4: Experimential results
Chapter 5: System analysis and design
Conclusion
References

*Drawings, charts (specify the types and sizes of drawings):*

……..……..…….....……
……..……

*Name of instructor:* Assoc. Prof., Ph.D. Pham Cong Thang

*Date of assignment : ……../……./2026*

*Date of completion : ……../……./2026*

*Da Nang, date month 07, year 2026*

**Head of the IT Department** **Instructor**

# PREFACE

During my study at the University of Science and Technology – The University of Danang, and throughout the process of completing this capstone project, I received valuable support and guidance from many lecturers, staff members, friends, and family.

First of all, I would like to express my sincere gratitude to all lecturers in the Faculty of Information Technology and other faculties for their dedicated teaching and guidance. Their knowledge, experience, and encouragement provided me with a strong foundation to complete this project.

In particular, I would like to express my deepest thanks to Assoc. Prof., Ph.D. Pham Cong Thang, my supervisor, for his dedicated guidance, valuable advice, and continuous support throughout the implementation of this capstone project. His comments and suggestions helped me better define the project objectives and improve the quality of my work.

I would also like to thank my friends and seniors for their support and encouragement during my study and project development.

Finally, I am deeply grateful to my parents and family for their unconditional love, encouragement, and spiritual support. Their support has been a great source of motivation for me to overcome difficulties and complete this project.

Although I have made great efforts, shortcomings are unavoidable. I sincerely hope to receive valuable feedback from lecturers and friends so that I can improve and gain more experience for future work.

Thank you sincerely.

# ASSURRANCE

I swear

1. The content presented in this thesis was carried out under the direct supervision of Assoc. Prof., Ph.D. Pham Cong Thang. All results and findings in the thesis were completed during my time studying at the Faculty of Information Technology, University of Science and Technology – The University of Danang**.**
2. All references used in the thesis have been clearly cited, including the author's name, title of the work, date and place of publication, along with valid URLs where applicable.
3. In the event of any improper copying or violation of academic regulations, I take full responsibility for my actions.

Student Performed

Do Nhat Hoang

# TABLE OF CONTENTS

LIST OF PICTURES ..... vi
LIST OF TABLES ..... viii
LIST OF SYMBOL, ACRONYM ..... ix
CHAPTER 1: PROJECT OVERVIEW ..... 1
1.1. Overview ..... 1
1.2. Reasons ..... 1
1.3. Objectives ..... 2
1.4. Scope of the project ..... 2
1.5. Methodology and technologies ..... 3
1.6. Contributions ..... 3
CHAPTER 2: THEORETICAL BACKGROUND ..... 5
2.1. Text-to-Speech (TTS) Systems ..... 5
2.2. Zero-shot Voice Coning (ZSVC) ..... 5
2.3. Neural Audio Codec ..... 6
2.4. Autoregressive Language Models for Audio ..... 7
2.5. Parameter-Efficient Fine-Tuning (PEFT) and LoRA ..... 7
2.6. Software architecture: Client-Server and RESTful APIs ..... 8
2.7. Related work ..... 8
CHAPTER 3: PROPOSED METHOD ..... 10
3.1. Overview of proposed method ..... 10
3.2. Baseline model: NeuTTS-AIR ..... 11
3.2.1. Input and output representation ..... 11
3.2.2. Model architecture ..... 12
3.3. Vietnamese adaptaiotn formulation ..... 13
3.4. Two-stage fine-tuning strategy ..... 14
3.4.1. Phase-1: lm_head fine-tuning ..... 16

3.4.2. Phase-2: Full fine-tuning ..... 17
3.4.3. Phase-2: Global LoRA adaptation ..... 18
3.5. Bucketing strategy for efficient training ..... 19
3.6. Chapter summary ..... 20
CHAPTER 4: EXPERIMENTAL RESULTS ..... 21
4.1. Chapter overview ..... 21
4.2. Vietnamese dataset and preprocessing results ..... 21
4.2.1. Dataset construction ..... 21
4.2.2. Preprocessing and tokenization results ..... 23
4.3. Experimental Setup ..... 25
4.3.1. Dataset Split ..... 25
4.3.2. Training environment ..... 26
4.3.3. Evaluation protocol and metrics ..... 27
4.4. Training results ..... 28
4.4.1. Phase-1 training results: lm_head fine-tuning ..... 28
4.4.2. Phase-2 Full fine-tuning results ..... 29
4.4.3. Phase-2 Global LoRA results ..... 30
4.5. Model comparison and discusion ..... 32
4.5.1. Comparison of baseline and best checkpoints ..... 32
4.5.2. Seen and unseen speaker analysis ..... 33
4.5.3. Qualitative Observation ..... 33
CHAPTER 5: SYSTEM ANALYSIS AND DESIGN ..... 35
5.1. System Architecture ..... 35
5.1.1. Overall System Architecture ..... 35
5.1.2. Mobile Application Layer ..... 35
5.1.3. Backend API Layer ..... 36
5.1.4. Model Inference Layer ..... 36
5.1.5. Database Design ..... 37

5.2. System Analysis and Design ..... 40
5.2.1. Use case diagram ..... 41
5.2.2. Use case specifications ..... 46
5.2.3. Sequence diagram ..... 52
5.3. Application evaluation ..... 54
5.3.1. Functional testing ..... 54
5.3.2. User interface demonstration ..... 56
5.3.3. End-to-End Voice Cloning workflow ..... 60
5.4. Discussion ..... 60
CONCLUSION ..... 62
REFERENCES ..... 65

# LIST OF PICTURES

Figure 2.1. Conventional neural TTS pipeline ..... 5
Figure 2.2. Speaker embedding-based zero-shot voice cloning framework ..... 5
Figure 2.3. Low-Rank Adaptation mechanism ..... 7
Figure 3.1. Overview of the proposed Vietnamese adaptation for NeuTTS-AIR ..... 10
Figure 3.2. Baseline architecture of NeuTTS-AIR based on Qwen2ForCausalLM ..... 13
Figure 3.3. Two-stage fine-tuning strategy for Vietnamese adaptation ..... 15
Figure 3.4. Global LoRA insertion into modules of NeuTTS-AIR ..... 18
Figure 3.5. Bucketing strategy for reducing padding during fine-tuning ..... 20
Figure 4.1. Vietnamese dataset contribution visualization ..... 22
Figure 4.2. Pipeline data preprocessing ..... 23
Figure 4.3. Vietnamese dataset contribution visualization after processing ..... 25
Figure 4.4. Train loss and eval loss graph of phase 1 ..... 29
Figure 4.5. Eval loss and trainloss graph of phase 2 Full fine-tuning ..... 30
Figure 4.6. Eval loss and trainloss graph of phase 2 Global LoRA ..... 31
Figure 5.1. Overall System Architecture ..... 35
Figure 5.2. Entity Relationship Diagram of the voice cloning system ..... 37
Figure 5.3. Overall use case diagram ..... 41
Figure 5.4. Authentication use case diagram ..... 42
Figure 5.5. Voice profile management use case diagram ..... 43
Figure 5.6. Voice clone and generated audio management use case diagram ..... 44
Figure 5.7. Premium features use case diagram ..... 45
Figure 5.8. Sequence diagram of User authentication ..... 52
Figure 5.9. Sequence diagram of creating voice profile ..... 52
Figure 5.10. Sequence diagram of voice cloning request ..... 53
Figure 5.11. Login screen of the application. ..... 56
Figure 5.12. Home screen for text input and voice selection. ..... 57

Figure 5.13. Voice profile management screen. ..... 57
Figure 5.14. Generated audio history screen. ..... 58
Figure 5.15. Audio playback and DSP adjustment screen. ..... 58
Figure 5.16. AI chatbot screen for script generation. ..... 59
Figure 5.17. Pro account upgrade screen. ..... 59

# LIST OF TABLES

Table 3.1. Baseline NeuTTS-AIR model configuration ..... 13
Table 3.2. Phase-1 lm_head fine-tuning configuration ..... 16
Table 3.3. Phase-2 full fine-tuning configuration ..... 17
Table 3.4. Phase-2 Global LoRA configuration ..... 19
Table 4.1. Vietnamese dataset construction ..... 22
Table 4.2. Vietnamese dataset construction after processing ..... 24
Table 4.3. Evaluation dataset split ..... 26
Table 4.4. Training environment ..... 26
Table 4.5. Evaluation metrics ..... 28
Table 4.6. Phase-1 evaluation ..... 29
Table 4.7. Phase-2 full fine-tuning evaluation results ..... 30
Table 4.8. Phase-2 Global LoRA results ..... 31
Table 4.9. Comparison of baseline and best checkpoints ..... 32
Table 5.1. User entity details ..... 38
Table 5.2. Voices profile entity details ..... 38
Table 5.3. Generated audio entity details ..... 39
Table 5.4. Chat session entity details ..... 40
Table 5.5. Chat message entity details ..... 40
Table 5.6. Use case specification for Register Account ..... 46
Table 5.7. Use case specification for Login ..... 47
Table 5.8. Use case specification for Create Voice Profile ..... 48
Table 5.9. Use case specification for Generate Cloned Speech ..... 49
Table 5.10. Use case specification for Manage Generated Audio ..... 50
Table 5.11. Functional testing results ..... 54

# LIST OF SYMBOL, ACRONYM

| Acronym | Full name |
|---|---|
| AI | Artificial Intelligence |
| TTS | Text-to-Speech |
| ZSVC | Zero-Shot Voice Cloning |
| NeuTTS-AIR | NeuTTS-AIR model |
| LLM | Large Language Model |
| G2P | Grapheme-to-Phoneme |
| MLP | Multi-Layer Perceptron |
| RMSNorm | Root Mean Square Layer Normalization |
| RoPE | Rotary Positional Embedding |
| SiLU | Sigmoid Linear Unit |
| LoRA | Low-Rank Adaptation |
| PEFT | Parameter-Efficient Fine-Tuning |
| GPU | Graphics Processing Unit |
| VRAM | Video Random Access Memory |
| CUDA | Compute Unified Device Architecture |
| BF16 | Brain Floating Point 16-bit |
| TF32 | TensorFloat-32 |
| API | Application Programming Interface |
| REST | Representational State Transfer |
| HTTP | Hypertext Transfer Protocol |
| JWT | JSON Web Token |
| JSON | JavaScript Object Notation |
| SQL | Structured Query Language |
| SQLite | Lightweight SQL Database Engine |

| | |
|---|---|
| Redis | Remote Dictionary Server |
| AWS | Amazon Web Services |
| S3 | Simple Storage Service |
| VPS | Virtual Private Server |
| UI | User Interface |
| DSP | Digital Signal Processing |
| WAV | Waveform Audio File Format |
| WER | Word Error Rate |
| CER | Character Error Rate |
| SECS | Speaker Embedding Cosine Similarity |
| MCD | Mel Cepstral Distortion |
| ECAPA-TDNN | Emphasized Channel Attention, Propagation and Aggregation Time Delay Neural Network |
| LUFS | Loudness Units relative to Full Scale |
| VIVOS | Vietnamese Speech Corpus |
| INFORE | INFORE Vietnamese Speech Dataset |

| **Symbol** | **Meaning** |
|---|---|
| x | Input speech waveform |
| E(·) | Encoder function in a neural audio codec |
| Q(·) | Quantization function in a neural audio codec |
| D(·) | Decoder function in a neural audio codec |
| S | Number of substitutions in error rate calculation |
| D | Number of deletions in error rate calculation |
| I | Number of insertions in error rate calculation |
| N | Total number of reference units, such as words or characters |
| R896 | Hidden representation space with dimension 896 |

| | |
|---|---|
| R217652 | Output vocabulary space with 217,652 tokens |
| r | LoRA rank |
| α | LoRA scaling factor |
| ↓ | Lower value is better |
| ↑ | Higher value is better |
| % | Percentage |
| kHz | Kilohertz |
| GB | Gigabyte |
| ms | Millisecond |
| 1e−4 | Scientific notation for learning rate value 0.0001 |
| 1e−5 | Scientific notation for learning rate value 0.00001 |

# CHAPTER 1: PROJECT OVERVIEW

## 1.1. Overview

Speech generation and voice cloning have become important research directions in modern Speech AI. These technologies allow a system to generate natural speech from text and, in the case of voice cloning, imitate the acoustic characteristics of a target speaker from a short reference audio sample. Such systems have many practical applications, including audiobooks, podcasts, virtual assistants, educational content, digital media production, and accessibility tools.

For Vietnamese, voice cloning remains a challenging task because Vietnamese is a tonal language with rich pronunciation patterns, regional accents, and prosodic variations. A small change in tone or rhythm can affect both speech intelligibility and naturalness. Therefore, adapting modern voice cloning models to Vietnamese requires not only model fine-tuning, but also careful dataset construction, text processing, audio normalization, and system integration.

This capstone project focuses on personalized Vietnamese speech generation using zero-shot voice cloning. The project adapts NeuTTS-AIR - an autoregressive audio language model - to Vietnamese speech and integrates the adapted model into a complete client-server application system.

## 1.2. Reasons

The demand for personalized speech generation is increasing in many real-world scenarios. Users may want to generate audio content using a specific voice for storytelling, video narration, audiobook production, language learning, or personal digital assistants. Compared with conventional text-to-speech systems, voice cloning provides a more personalized and flexible speech generation experience.

Recent advances in zero-shot voice cloning allow a system to synthesize speech in the voice of an unseen speaker using only a short reference audio sample, without training a separate model for each speaker. This makes voice cloning more practical and scalable.

However, building a complete Vietnamese voice cloning system is still a complex problem. It involves multiple components, including speech dataset preparation, model adaptation, inference deployment, backend services, storage,

authentication, and mobile user interaction. Therefore, this topic was chosen to combine both AI model research and practical software system development.

## 1.3. Objectives

The main objective of this project is to build an end-to-end Vietnamese zero-shot voice cloning system that allows users to input text and provide a short reference audio sample to generate personalized speech.

The specific objectives are:

- To study the principles of text-to-speech synthesis, zero-shot voice cloning, neural audio codecs, and autoregressive audio language models.
- To construct and process a Vietnamese multi-speaker speech dataset from VIVOS, INFORE, viVoice, and PhoAudiobook.
- To adapt NeuTTS-AIR to Vietnamese speech generation using a two-stage fine-tuning strategy.
- To evaluate the adapted model using objective metrics such as WER, CER, SECS, and MCD.
- To develop a practical voice cloning application with a Flutter mobile app, FastAPI backend, cloud storage, asynchronous task processing, and Modal GPU inference service.

## 1.4. Scope of the project

The subject of this project is a Vietnamese zero-shot voice cloning system based on NeuTTS-AIR. The system targets users who need to generate personalized Vietnamese speech from input text and a short reference audio sample.

The input of the system includes Vietnamese text and a short reference audio sample representing the target speaker. The output is a generated speech audio file that follows the input text while preserving the speaker characteristics from the reference audio.

The scope of this project includes Vietnamese speech generation, zero-shot voice cloning from short reference audio, adaptation of an existing pretrained NeuTTS-AIR model, and development of a functional mobile application and backend system.

This project does not include training a speech model from scratch, real-time streaming voice cloning, large-scale commercial deployment, or full production-level distributed serving.

## 1.5. Methodology and technologies

The project follows a research-and-development methodology. First, theoretical foundations related to TTS, zero-shot voice cloning, neural audio codecs, autoregressive language models, and LoRA-based adaptation are studied. Then, a Vietnamese dataset is constructed and processed into the format required by NeuTTS-AIR. After that, the model is adapted through a two-stage fine-tuning strategy and evaluated using both model-level and application-level criteria.

The main technologies used in this project include:

- AI model: NeuTTS-AIR, Qwen2ForCausalLM backbone, NeuCodec.
- Fine-tuning: PyTorch, Transformers, PEFT/LoRA.
- Backend: FastAPI, SQLAlchemy, JWT authentication.
- Task queue: Redis and background worker.
- Cloud inference: Modal GPU service.
- Storage: AWS S3 or MinIO-compatible object storage.
- Mobile application: Flutter.
- Evaluation: Whisper large-v3, jiwer, SpeechBrain ECAPA-TDNN, and MCD toolbox.

## 1.6. Contributions

The main contributions of this project are summarized as follows:

- Constructing a Vietnamese multi-speaker speech corpus from multiple sources, including VIVOS, INFORE, viVoice, and PhoAudiobook, to support Vietnamese zero-shot voice cloning adaptation.
- Designing a Vietnamese data processing pipeline for NeuTTS-AIR, including audio normalization, text preprocessing, grapheme-to-phoneme conversion, NeuCodec-based audio token extraction, and precomputed tokenized training data generation.
- Proposing a two-stage adaptation strategy for NeuTTS-AIR. In the first stage, only the lm_head layer is fine-tuned to provide an initial adaptation to Vietnamese audio tokens. In the second stage, deeper adaptation is investigated using Global LoRA and full fine-tuning.
- Evaluating the adapted models using objective metrics, including Word Error Rate, Character Error Rate, Speaker Embedding Cosine Similarity, and Mel Cepstral Distortion.

- Developing a complete client-server voice cloning system, including a Flutter mobile application, FastAPI backend, asynchronous task queue, cloud storage, and cloud GPU inference service.

# CHAPTER 2: THEORETICAL BACKGROUND

## 2.1. Text-to-Speech (TTS) Systems

Text-to-Speech (TTS) synthesis is the process of artificially generating human-like speech from written text. The evolution of TTS systems has transitioned through several paradigms, from early concatenative synthesis - which stitched together pre-recorded speech fragments - to Statistical Parametric Speech Synthesis (SPSS) based on Hidden Markov Models (HMMs) [1].

In recent years, Neural TTS has become the industry standard, driven by breakthrough architectures such as Tacotron 2 [2] , FastSpeech [3] , and VITS [4] . Modern neural pipelines typically consist of two stages: an acoustic model that maps text or phonemes into intermediate acoustic representations (such as mel-spectrograms), and a vocoder (e.g., HiFi-GAN) that converts these representations into raw audio waveforms [5].




*Figure 2.1. Conventional neural TTS pipeline*

While these conventional neural models produce highly intelligible speech, they are fundamentally deterministic and often struggle to generate diverse, highly expressive, and contextual prosody without explicit conditioning.

## 2.2. Zero-shot Voice Coning (ZSVC)

Zero-shot Voice Cloning refers to the capability of a TTS system to synthesize speech in the voice of an unseen speaker usig only a brief reference audio sample (usually 3 to 10 seconds), without requiring any fine-tuning or model retraining for that special user [6].

Traditional ZSVC methodologies rely heavily on Speaker Verification (SV) models to extract speaker embedding from the reference audio. This embedding is then concatenated with the linguistic features derived from the input text [6].

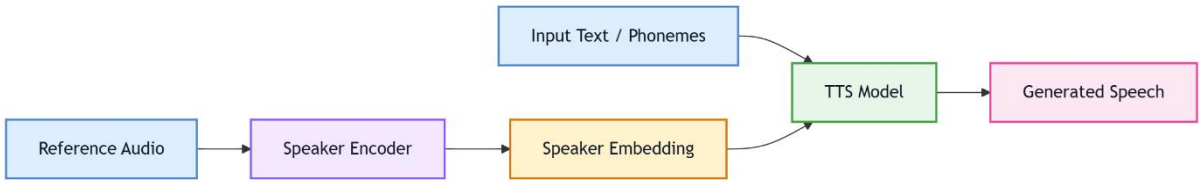



*Figure 2.2. Speaker embedding-based zero-shot voice cloning framework*

As shown in Figure 2.2, the reference audio is first processed by a speaker encoder to obtain a fixed speaker embedding. This embedding represents the vocal identity of the target speaker and is used together with the input text or phoneme sequence to guide the TTS model in generating speech. In this way, the model can synthesize a new utterance while attempting to preserve the acoustic characteristics of the unseen speaker.

While this approach is computationally efficient and practical, extracting a single fixed vector inherently averages out the speaker's dynamic acoustic characteristics. Consequently, this bottleneck limits the model's ability to replicate emotional variation, speaking rhythm, natural breathing patterns, and subtle prosodic details, especially in complex tonal languages such as Vietnamese.

## 2.3. Neural Audio Codec

Traditional speech synthesis systems using continuous acoustic features such as mel-spectrograms. However, these continuous representations are not directly compatible with autoregressive language models, which operate on discrete tokens. To address this limitation, mordern speech generation systems employ Neural Audio Codec to convert speech waveforms into discrete audio tokens.

A Neural Audio Codec consists of an encoder, a quantization module, and a decoder. The encoder compresses the input speech signal into latent representations, which are then discretized into a sequence of tokens through vector quantization. The decoder reconstructs the waveform from these tokens [7].

The overall process can be summarized as:

$$\begin{cases} z = E(x) \\ \hat{z} = Q(z) = \mathrm{VQ}\big(E(x)\big) \\ \hat{x} = D(\hat{z}) \end{cases} \qquad (1)$$

Where $x \in \mathbb{R}^T$ denotes the continuous raw input speech waveform in the time domain, $E(\cdot)$ is the encoder, $Q(\cdot)$ or $\mathrm{VQ}(\cdot)$ represents the Vector Quantization module, and $D(\cdot)$ is the reconstructed waveform.

By transforming speech into discrete tokens, speech generation can be reformulated as a sequence prediction task. This enables autoregressive language models to generate audio in a manner similar to next-token prediction in natural language processing. Such a paradigm serves as the foundation of modern Neural Codec Language Models, including NeuTTS-AIR used in this study [8].

## 2.4. Autoregressive Language Models for Audio

To overcome the limitations of static speaker embeddings, recent breakthroughs have redefined audio generation as a language modeling task. This paradigm shift is made possible by Neural Audio Codecs (such as NeuCodec), which compress continuous audio waveforms into discrete, quantized audio tokens [7], [8].

Once audio is discretized, Large Language Models (LLMs) featuring a Causal Transformer architecture - similar to the Qwen2 backbone utilized in modern frameworks - can be trained to predict the next audio token. Mathematically, given a text prompt condition $c$ and previous audio tokens $x_{<t}$, the autoregressive model predicts the probability of the next token $x_t$:

$$P(X|c) = \prod_{t=1}^{T} P\left(x_t|x_{<t}, c\right) \qquad (2)$$

This token-by-token prediction mechanism allows the model to deeply understand the acoustic context, enabling highly natural zero-shot cloning where the generated speech seamlessly inherits the acoustic environment, emotion, and prosody of the reference prompt [8].

## 2.5. Parameter-Efficient Fine-Tuning (PEFT) and LoRA

Adapting large-scale autoregressive audio models to new linguistic domains requires substantial computational resources. Full-parameter fine-tuning often leads to catastrophic forgetting and exceeds standard GPU memory constraints (VRAM) [9].

Parameter-Efficient Fine-Tuning (PEFT) mitigates by freezing the majority of the pre-trained model's parameters and only training a small number of special weights. Low-Rank Adaptation (LoRA) is a highly effective PEFT technique. Instead of updating a dense pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA hypothesizes that the changes in weights have a low intrinsic rank. It approximates the weight update $\Delta W$ by factorizing it into two smaller matrices, $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, where the rank $r \ll \min(d, k)$ [9].

The forward pass is modified as follows:

$$h = W_0 x + \Delta W x = W_0 x + BAx \qquad (3)$$

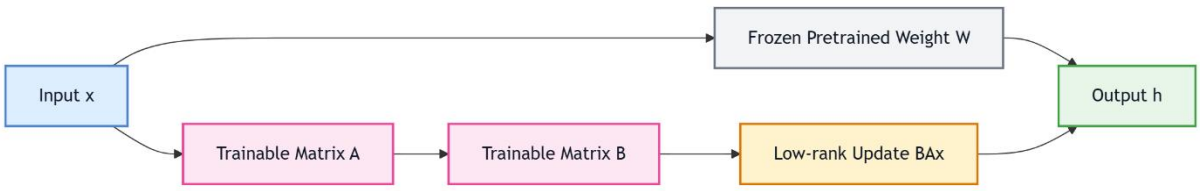



*Figure 2.3. Low-Rank Adaptation mechanism*

As shown in Figure 2.3, the original pretrained weight $W_0$ remains frozen during training, while only the low-rank matrices A and B are updated. The output is obtained by combining the original transformation $W_0x$ with the additional low-rank update $BAx$. This design allows the model to adapt to a new task without updating the full parameter set.

During training, only a small number of parameters need to be optimized, which significantly reduces memory usage and training cost. At the same time, LoRA helps preserve the pretrained model's generalization ability, making it a practical adaptation method for large audio language models and other Transformer-based architectures.

## 2.6. Software architecture: Client-Server and RESTful APIs

To make the voice cloning model accessible to end users, the system is designed based on a Client-Server architecture [10]. In this architecture, the mobile application serves as the client, providing an interface for users to upload reference audio, enter text, and receive synthesized speech. The backend server handles request processing, user management, and communication with the NeuTTS-AIR model.

The client and server communicate through HTTP-based APIs developed using the FastAPI framework. When a user submits a voice cloning request, the application sends the input text and reference audio to the server. The server then processes the request, invokes the speech synthesis model, and returns the generated audio result to the client.

To ensure secure access to system resources, JSON Web Token (JWT) authentication is employed. After a successful login, the server generates an access token for the user. This token is attached to subsequent requests, allowing the server to verify user identity while maintaining a stateless communication mechanism [11].

## 2.7. Related work

Early neural text-to-speech systems mainly focused on improving the mapping from text to acoustic representations. Tacotron 2 introduced an end-to-end neural pipeline that predicts mel-spectrograms from text and uses a neural vocoder to synthesize waveform audio [2]. Later, FastSpeech improved the efficiency and stability of speech synthesis by using a non-autoregressive architecture [3]. VITS further integrated variational inference and adversarial learning into an end-to-end speech synthesis framework [4]. These approaches significantly improved the naturalness and intelligibility of synthetic speech. However, most conventional neural TTS systems still rely on intermediate acoustic features such as mel-spectrograms and

usually require explicit speaker conditioning for multi-speaker or voice cloning scenarios.

In zero-shot voice cloning, speaker embedding based approaches are widely used. A common method is to extract a fixed speaker representation from a short reference audio sample using a speaker verification model, and then condition the TTS model on this embedding [6]. This approach is computationally efficient and practical for unseen speakers. However, a single speaker embedding may not fully capture dynamic speech characteristics such as prosody, rhythm, emotion, and speaking style. This limitation is especially important for Vietnamese, where tonal pronunciation and prosodic patterns strongly affect speech naturalness and intelligibility.

Compared with previous speaker embedding based methods, neural codec language models such as VALL-E provide a promising direction for zero-shot voice cloning because they model speech as a sequence of discrete audio tokens rather than relying only on a fixed speaker vector [8]. However, adapting such models to a new language requires careful dataset preparation and efficient fine-tuning. Full-parameter fine-tuning can be computationally expensive and may affect the pretrained model's generalization ability. Therefore, parameter-efficient fine-tuning methods such as LoRA are useful for adapting large pretrained models while updating only a subset of trainable parameters [9].

Based on these related studies, this thesis focuses on adapting NeuTTS-AIR, an autoregressive audio language model, to Vietnamese zero-shot voice cloning. Unlike works that only evaluate model performance, this thesis also develops a complete end-to-end application system, including data processing, model adaptation, cloud GPU inference, backend services, and a Flutter mobile application. This makes the project both a model adaptation study and a practical system implementation for personalized Vietnamese speech generation.

# CHAPTER 3: PROPOSED METHOD

## 3.1. Overview of proposed method

This chapter presents the proposed method for adapting NeuTTS-AIR to Vietnamese zero-shot voice cloning. Instead of designing a speech generation model from scratch, this thesis uses NeuTTS-AIR as the baseline model and focuses on adapting it to Vietnamese speech through a staged fine-tuning strategy.

The proposed method consists of three main components. First, the baseline NeuTTS-AIR model is analyzed to understand how it represents speech generation as an autoregressive audio token prediction task. Second, the Vietnamese adaptation problem is formulated by converting Vietnamese text and speech into the input-output format required by NeuTTS-AIR. Third, a two-stage fine-tuning strategy is designed to gradually adapt the model to Vietnamese speech while preserving its pretrained zero-shot voice cloning capability.

The proposed workflow can be summarized as follows:

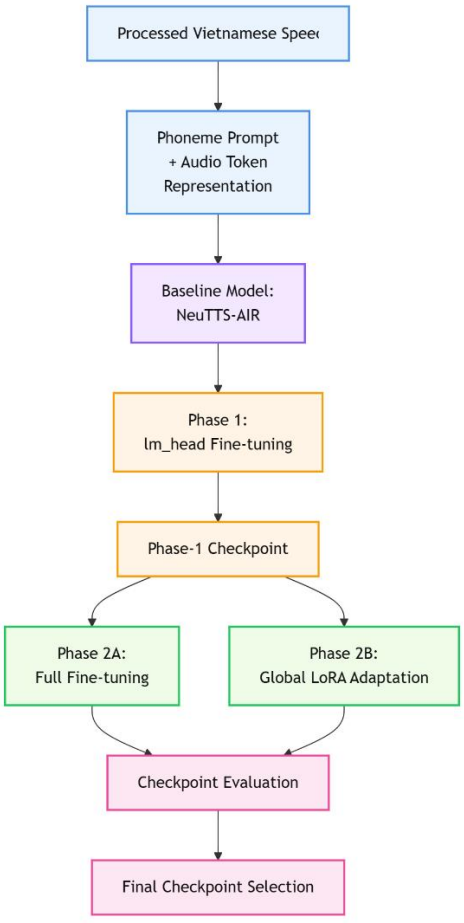



*Figure 3.1. Overview of the proposed Vietnamese adaptation for NeuTTS-AIR*

As shown in Figure 3.1, the processed Vietnamese training samples are first represented as phoneme prompts and discrete audio tokens. These samples are then used to adapt the baseline NeuTTS-AIR model. Phase 1 performs lightweight adaptation by fine-tuning only the output projection layer. Phase 2 further adapts the model using either full fine-tuning or Global LoRA. The trained checkpoints are finally evaluated and selected for application deployment. The baseline model provides the pretrained speech generation ability, while the fine-tuning stages gradually adjust the model to Vietnamese pronunciation, rhythm, and acoustic characteristics.

## 3.2. Baseline model: NeuTTS-AIR

The baseline model used in this thesis is NeuTTS-AIR, an autoregressive audio language model for speech generation. Unlike conventional neural TTS systems that predict continuous acoustic features such as mel-spectrograms, NeuTTS-AIR generates speech by predicting discrete audio tokens. These tokens are later decoded into waveform audio by a neural audio codec.

NeuTTS-AIR is built on top of a Qwen2ForCausalLM backbone. The model follows a causal Transformer structure, where each output token is predicted based on the previous tokens and the input text prompt. This makes the speech generation process similar to next-token prediction in natural language processing.

In this thesis, NeuTTS-AIR is treated as the baseline model before Vietnamese adaptation. The purpose of the proposed method is to investigate how this pretrained autoregressive audio language model can be adapted to Vietnamese speech using a limited but diverse Vietnamese speech corpus.

### 3.2.1. Input and output representation

The input of NeuTTS-AIR is organized in a conversational prompt format. The text or phoneme prompt is placed between special text prompt tokens, while the target speech is represented as a sequence of discrete audio tokens. During training, the model receives the complete input sequence, but the loss is computed only on the target audio token sequence.

The training sample format can be summarized as follows:

```
user: Convert the text to speech:
<|TEXT_PROMPT_START|>{phoneme_sequence}<|TEXT_PROMPT_END|>

assistant:
<|SPEECH_GENERATION_START|>{audio_tokens}<|SPEECH_GENERATION_END|>
```

In this format, the phoneme sequence provides the linguistic content, while the audio tokens represent the target speech. The prompt tokens and text condition are masked during training by setting their labels to -100. Therefore, the model is optimized to predict only the speech token sequence.

This representation is important because it allows Vietnamese speech synthesis to be formulated as an autoregressive token prediction problem. Instead of directly generating waveform samples, the model learns to generate discrete audio tokens conditioned on the Vietnamese phoneme prompt.

### 3.2.2. Model architecture

The architecture of the baseline NeuTTS-AIR model is shown in Figure 3.2. The model receives tokenized input prompts through an embedding layer. The hidden representations are then processed by multiple Qwen2 decoder layers. Each decoder layer contains a self-attention block and a feed-forward MLP block. Finally, the output hidden states are passed through a final normalization layer and the language modeling head to predict the next audio token.

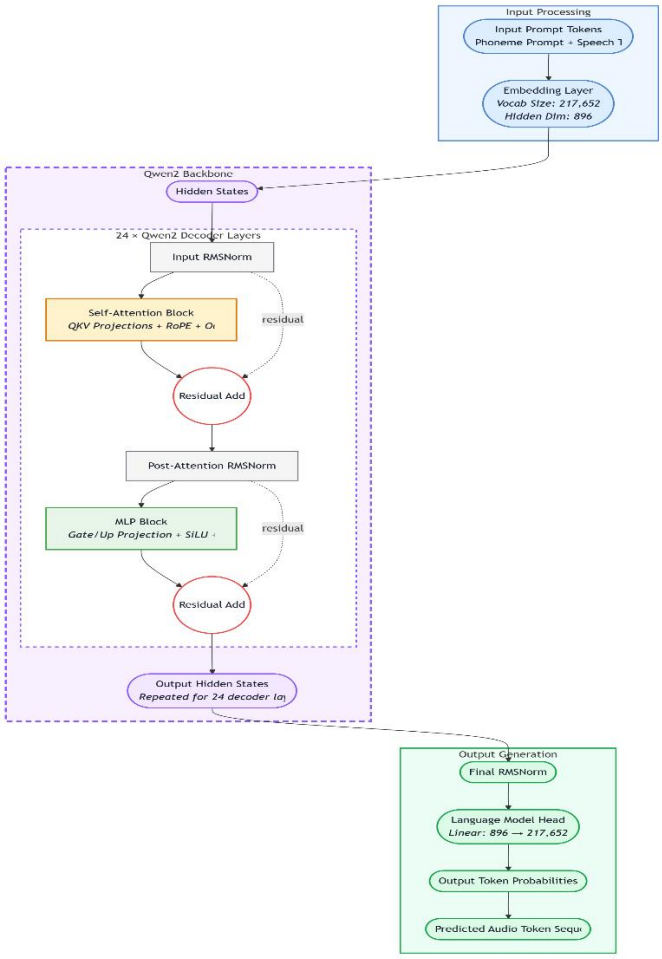


*Figure 3.2. Baseline architecture of NeuTTS-AIR based on Qwen2ForCausalLM*

As illustrated in Figure 3.2, the model architecture follows the standard causal language modeling design. However, in this task, the predicted tokens correspond to audio tokens rather than text tokens. This allows the model to generate speech in an autoregressive manner.

The main configuration of the baseline model is summarized in Table 3.1.

*Table 3.1. Baseline NeuTTS-AIR model configuration*

| **Parameter** | **Value** | **Unit / Description** |
|---|---|---|
| Base model | NeuTTS-AIR | Pretrained autoregressive audio language model |
| Backbone | Qwen2ForCausalLM | Causal Transformer backbone |
| Number of decoder layers | 24 | layers |
| Hidden size | 896 | dimensions |
| Vocabulary size | 217,652 | tokens |
| Audio codebook size | 65,536 | codec tokens |
| Maximum sequence length | 2,048 | tokens |
| Attention modules | q_proj, k_proj, v_proj, o_proj | projection layers |
| MLP modules | gate_proj, up_proj, down_proj | feed-forward layers |

Table 3.1 shows that NeuTTS-AIR is a relatively compact but deep autoregressive model. The large vocabulary size includes both text-related tokens and speech-related tokens. The maximum sequence length of 2,048 tokens limits the length of the input prompt and generated speech token sequence used during training.

## 3.3. Vietnamese adaptaiotn formulation

The objective of Vietnamese adaptation is to make NeuTTS-AIR generate intelligible and natural Vietnamese speech while preserving the speaker characteristics from a short reference audio sample. Since Vietnamese is a tonal language, the model must learn not only the textual content but also pronunciation, tone, rhythm, and prosodic patterns.

Based on the autoregressive audio language modeling formulation (2) described in Section 2.4, this thesis adapts NeuTTS-AIR by treating Vietnamese speech synthesis as a conditional audio token prediction task. In this formulation, the condition (c) corresponds to the Vietnamese text or phoneme prompt, while the target sequence (X) corresponds to the discrete audio token sequence extracted from the speech waveform.

During training, the model receives the Vietnamese phoneme prompt and previous audio tokens as context, then predicts the next audio token step by step. The generated token sequence is later decoded into waveform audio by the neural audio codec. Therefore, the adaptation process does not require training a speech model from scratch. Instead, the pretrained autoregressive representation of NeuTTS-AIR is reused and gradually adapted to Vietnamese through fine-tuning.

In the context of this thesis, the adaptation process focuses on three main aspects. First, the model must learn the relationship between Vietnamese phoneme sequences and corresponding audio tokens. Second, it must preserve the speaker characteristics provided by the reference audio. Third, it must improve Vietnamese pronunciation and prosody while maintaining the zero-shot voice cloning ability of the pretrained model.

This formulation provides the basis for the two-stage fine-tuning strategy described in the following section. Phase 1 performs lightweight adaptation on the output projection layer, while Phase 2 further adapts the model using full fine-tuning and Global LoRA.

## 3.4. Two-stage fine-tuning strategy

Directly fine-tuning the whole NeuTTS-AIR model on Vietnamese data may lead to unstable training, high memory usage, or degradation of the pretrained zero-shot capability. Therefore, this thesis proposes a two-stage adaptation strategy.

Before applying the two-stage strategy, the pretrained NeuTTS-AIR model is first considered as the baseline model. Although the model already has the ability to generate speech through autoregressive audio token prediction, it has not been specifically adapted to Vietnamese. Therefore, directly using the original checkpoint may lead to unstable Vietnamese pronunciation, incorrect tonal patterns, and limited naturalness in generated speech.

To address this issue, the adaptation process is divided into two stages. Phase 1 is designed as a safe and lightweight adaptation step, where only the lm_head layer is updated to adjust the output token distribution to Vietnamese speech tokens. After

obtaining a more suitable starting checkpoint, Phase 2 performs deeper adaptation using two different strategies: full fine-tuning and Global LoRA. This design makes the training process more controlled and allows the experimental chapter to compare the effectiveness of different adaptation methods.

The overall fine-tuning strategy is shown in Figure 3.3.

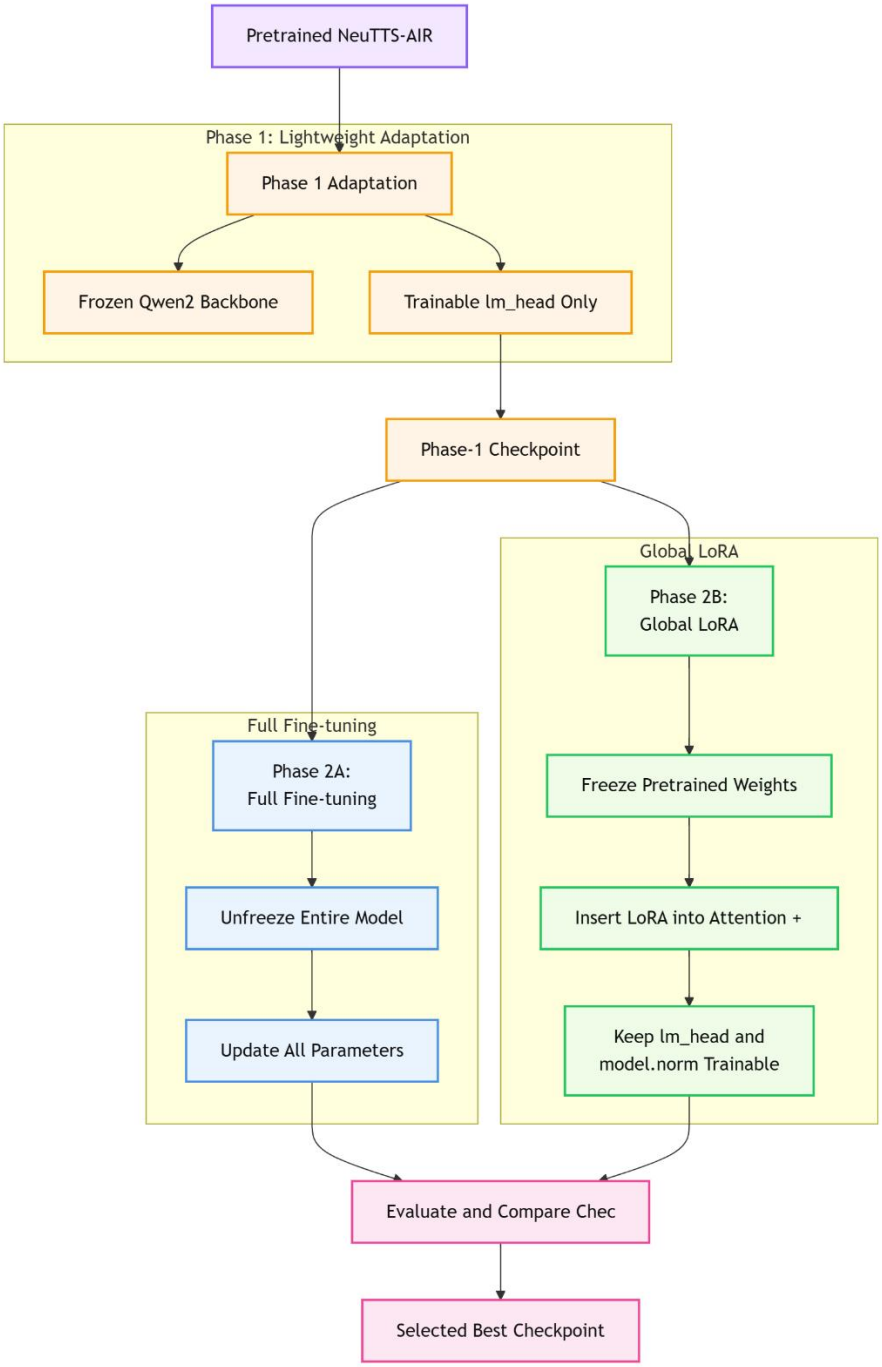



*Figure 3.3. Two-stage fine-tuning strategy for Vietnamese adaptation*

As shown in Figure 3.3, Phase 1 performs lightweight adaptation by updating only the lm_head layer. This stage helps the model adjust its output token distribution to Vietnamese speech tokens. Phase 2 then performs deeper adaptation using either

full fine-tuning or Global LoRA. Full fine-tuning updates the entire model, while Global LoRA adapts the model through low-rank trainable matrices inserted into selected Transformer modules.

This staged design provides a controlled adaptation process. The first stage creates a Vietnamese-adapted starting checkpoint, while the second stage investigates different strategies for improving speech intelligibility, speaker similarity, and audio quality.

### 3.4.1. Phase-1: lm_head fine-tuning

In Phase 1, only the lm_head layer is fine-tuned, while the Transformer backbone remains frozen. The lm_head is responsible for projecting hidden states into the output vocabulary distribution. By updating only this layer, the model can initially adapt its output distribution to Vietnamese audio tokens without changing the pretrained internal representations.

The configuration of Phase 1 is summarized in Table 3.2.

*Table 3.2. Phase-1 lm_head fine-tuning configuration*

| **Parameter** | **Value** | **Unit / Description** |
|---|---|---|
| Base model | NeuTTS-AIR | pretrained checkpoint |
| Trainable parameters | lm_head only | output projection layer |
| Frozen parameters | Transformer backbone | pretrained layers |
| Maximum sequence length | 2,048 | tokens |
| Maximum training steps | 5 | steps |
| Learning rate | 1,00E-04 | optimizer learning rate |
| Warmup ratio | 0.03 | ratio of total steps |
| Batch size per device | 2 | samples / GPU |
| Gradient accumulation steps | 16 | steps |
| Effective batch size | 32 | samples / update |
| Validation ratio | 0.01 | ratio |

Table 3.2 shows that Phase 1 is designed as a lightweight adaptation stage. Since only the output projection layer is updated, this phase requires less memory and provides a stable starting point for deeper adaptation in Phase 2.

### 3.4.2. Phase-2: Full fine-tuning

Full fine-tuning is the most direct strategy for adapting NeuTTS-AIR to Vietnamese. In this approach, the entire model is unfrozen and all trainable parameters are updated using the Vietnamese training data. Since every layer can be optimized, full fine-tuning provides the highest adaptation capacity.

However, this strategy also has several limitations. It requires more GPU memory and computational resources. In addition, updating all pretrained parameters may increase the risk of overfitting or catastrophic forgetting, especially when the target dataset is smaller than the original pretraining data. Therefore, a smaller learning rate is used to reduce the risk of damaging the pretrained representations.

The configuration of full fine-tuning is summarized in Table 3.3.

*Table 3.3. Phase-2 full fine-tuning configuration*

| **Parameter** | **Value** | **Unit / Description** |
|---|---|---|
| Base model | NeuTTS-AIR | pretrained checkpoint |
| Trainable parameters | lm_head only | output projection layer |
| Frozen parameters | Transformer backbone | pretrained layers |
| Maximum sequence length | 2,048 | tokens |
| Maximum training steps | 5 | steps |
| Learning rate | 1,00E-04 | optimizer learning rate |
| Warmup ratio | 0.03 | ratio of total steps |
| Batch size per device | 2 | samples / GPU |
| Gradient accumulation steps | 16 | steps |

As shown in Table 3.3, full fine-tuning updates the entire model with a relatively small learning rate. This configuration is used to investigate whether direct full-model adaptation can improve Vietnamese speech generation quality.

### 3.4.3. Phase-2: Global LoRA adaptation

To reduce training cost while still enabling deep adaptation, this thesis also investigates Global LoRA adaptation. In this strategy, the original pretrained model weights are frozen, and trainable low-rank matrices are inserted into selected Transformer modules.

Unlike partial LoRA, which adapts only a limited number of layers or modules, Global LoRA is applied across all 24 decoder layers of NeuTTS-AIR. LoRA modules are inserted into both the self-attention and MLP components, including:

- Attention modules: q_proj, k_proj, v_proj, o_proj
- MLP modules: gate_proj, up_proj, down_proj

The insertion of Global LoRA modules is illustrated in Figure 3.4.

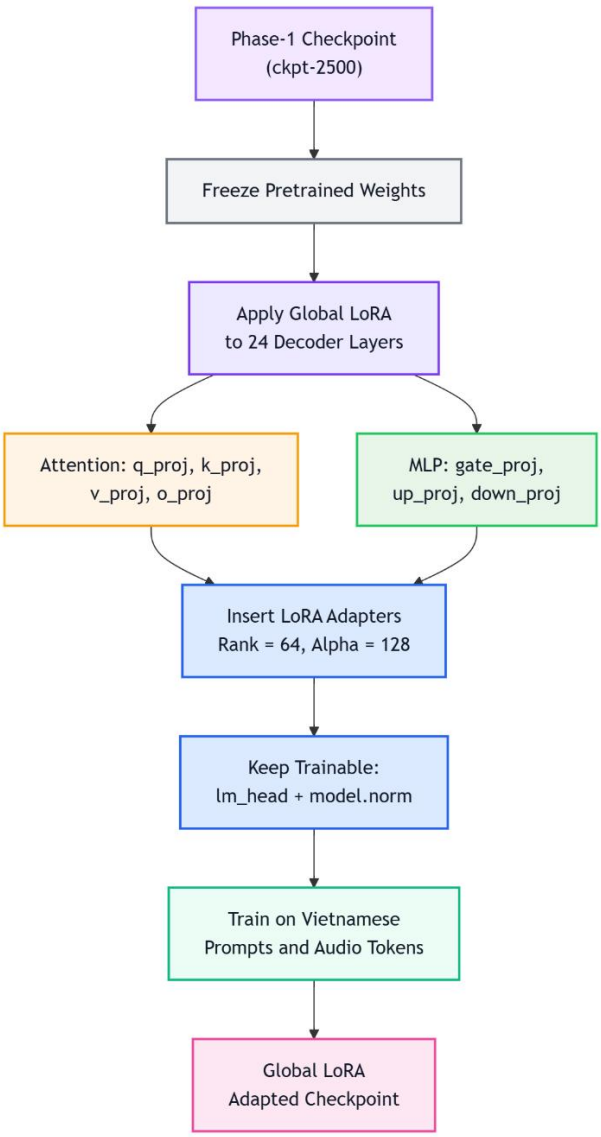



*Figure 3.4. Global LoRA insertion into modules of NeuTTS-AIR*

As shown in Figure 3.4, LoRA introduces additional trainable low-rank matrices into selected projection layers while keeping the original pretrained weights frozen. This allows the model to adapt to Vietnamese linguistic and acoustic characteristics without updating all model parameters.

The configuration of Global LoRA adaptation is summarized in Table 3.4.

*Table 3.4. Phase-2 Global LoRA configuration*

| **Parameter** | **Value** | **Unit / Description** |
|---|---|---|
| Base checkpoint | Phase-1 checkpoint-2500 | starting checkpoint |
| LoRA rank | 64 | rank |
| LoRA alpha | 128 | scaling factor |
| LoRA dropout | 0.05 | dropout probability |
| Target attention modules | q_proj, k_proj, v_proj, o_proj | projection layers |
| Target MLP modules | gate_proj, up_proj, down_proj | projection layers |
| Trainable saved modules | lm_head, model.norm | additional trainable modules |
| Maximum sequence length | 2,048 | tokens |
| Training steps | 15,000 | steps |
| Learning rate | 5e-5 | optimizer learning rate |
| Warmup ratio | 0.05 | ratio of total steps |

The motivation behind Global LoRA is to balance adaptation ability and training efficiency. By updating only a smaller set of additional trainable parameters, the model can adapt to Vietnamese speech while preserving most of the pretrained zero-shot voice cloning capability. This makes Global LoRA a practical alternative to full fine-tuning when computational resources are limited.

## 3.5. Bucketing strategy for efficient training

During fine-tuning, input sequences have different lengths because each training sample contains both text tokens and audio tokens. If all samples are padded to the same maximum length, a large amount of GPU memory may be wasted. This problem

becomes more serious in autoregressive speech generation because audio token sequences can be much longer than text sequences.

To address this issue, a bucketing sampler is used. Samples with similar sequence lengths are grouped into the same batch. This reduces unnecessary padding and improves GPU utilization. In addition, length-consistent batches help stabilize training because the model processes samples with more similar sequence structures in each update.

The bucketing process is illustrated in Figure 3.5.

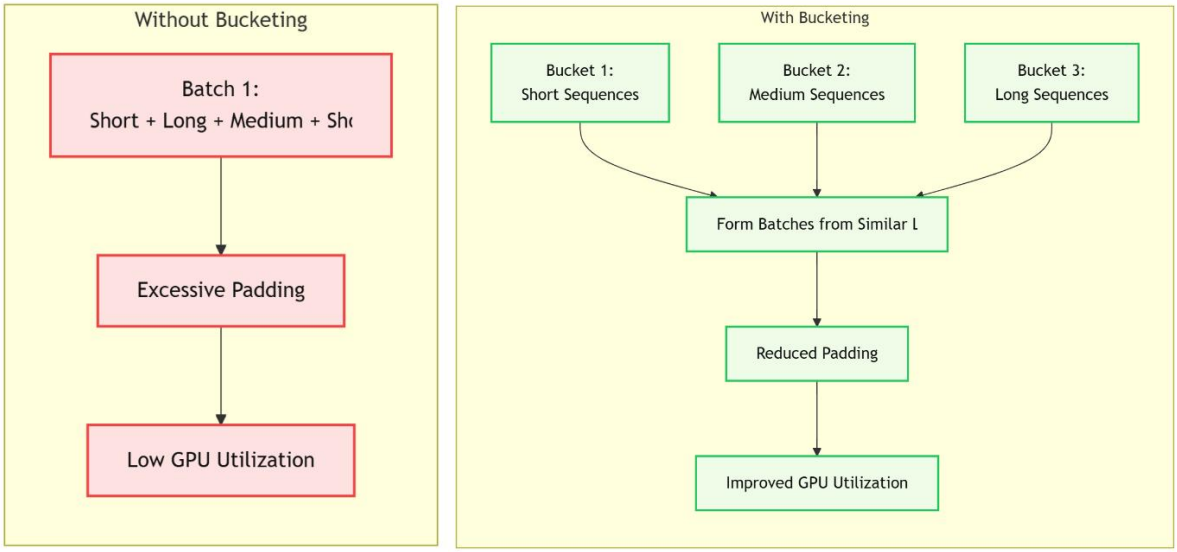



*Figure 3.5. Bucketing strategy for reducing padding during fine-tuning*

As shown in Figure 3.5, samples are first sorted or grouped by sequence length. Each batch is then formed from samples with similar lengths. This strategy improves training efficiency without changing the model architecture or training objective.

## 3.6. Chapter summary

This chapter presented the proposed method for adapting NeuTTS-AIR to Vietnamese zero-shot voice cloning. First, NeuTTS-AIR was introduced as the baseline autoregressive audio language model. Then, the Vietnamese adaptation problem was formulated as a conditional audio token prediction task. After that, a two-stage fine-tuning strategy was described, including Phase-1 lm_head fine-tuning and Phase-2 adaptation using full fine-tuning and Global LoRA. Finally, the bucketing strategy was introduced to improve training efficiency.

The next chapter presents the experimental setup and evaluation results, including the Vietnamese dataset, preprocessing results, training outcomes, model comparison, and voice cloning evaluation.

# CHAPTER 4: EXPERIMENTAL RESULTS

## 4.1. Chapter overview

This chapter presents the experimental results of the proposed Vietnamese adaptation framework for NeuTTS-AIR. The evaluation focuses on both the training behavior and the voice cloning performance of the adapted models. First, the Vietnamese dataset used in the experiments is summarized, including the selected data sources and the final preprocessed dataset. Next, the experimental setup, dataset split, training environment, and evaluation metrics are described. After that, the training loss curves of each adaptation phase are analyzed to observe the learning behavior of the model. Finally, the quantitative results are reported using WER, CER, SECS, and MCD, followed by a comparison between Phase-1 adaptation, full fine-tuning, and Global LoRA.

The main objective of this chapter is to evaluate whether the proposed fine-tuning strategy can improve Vietnamese speech generation quality while preserving zero-shot voice cloning capability. Since the target task requires both accurate speech content and speaker similarity, the results are analyzed from multiple perspectives, including intelligibility, speaker preservation, acoustic quality, and checkpoint stability.

## 4.2. Vietnamese dataset and preprocessing results

### 4.2.1. Dataset construction

The Vietnamese speech dataset used in this thesis was constructed from four sources: VIVOS, INFORE, viVoice, and PhoAudiobook. These datasets provide different recording conditions, speaker characteristics, and speaking styles, which are useful for adapting NeuTTS-AIR to Vietnamese zero-shot voice cloning.

VIVOS and INFORE are used as clean core datasets because they contain relatively clear Vietnamese speech under controlled recording conditions. viVoice provides a larger amount of speech data and contributes to linguistic and acoustic diversity. PhoAudiobook is added to increase speaker variation and reading-style speech characteristics.

Since viVoice and PhoAudiobook contain a large number of recordings, not all available samples are used directly. Instead, a speaker-level selection strategy is applied to reduce dataset imbalance and control the contribution of each source. For VIVOS, INFORE, and viVoice, approximately 300 utterances per speaker are selected

when available. In contrast, PhoAudiobook is sampled more conservatively, with about 60–70 utterances per speaker, because audiobook recordings are usually longer and contain a different reading-style speech pattern. This strategy prevents speakers or datasets with excessive recordings from dominating the training data while still preserving speaker diversity and speaking-style variation.

*Table 4.1. Vietnamese dataset construction*

| Dataset | Number of speakers | Duration | Number of selected samples | Sampling rate |
|---|---|---|---|---|
| VIVOS | 46 | 10 hours | 11,221 | 16 kHz |
| INFORE | 1 | 0.2 hours | 300 | 16 kHz |
| viVoice | Không xác định | 75 hours | 52,094 | 24 kHz |
| PhoAudioBook | 708 | 6 hours | 4,937 | 24 kHz |
| Total | 935 | ~91.2 hours | 68,552 | 16/24 kHz |

Table 4.1 summarizes the selected dataset before preprocessing. The dataset contains approximately 935 speakers and 68,552 selected utterances, with a total duration of about 91.2 hours. The sampling rates are not uniform at this stage because the data was collected from multiple sources. Therefore, audio normalization and resampling are required before token extraction and fine-tuning.

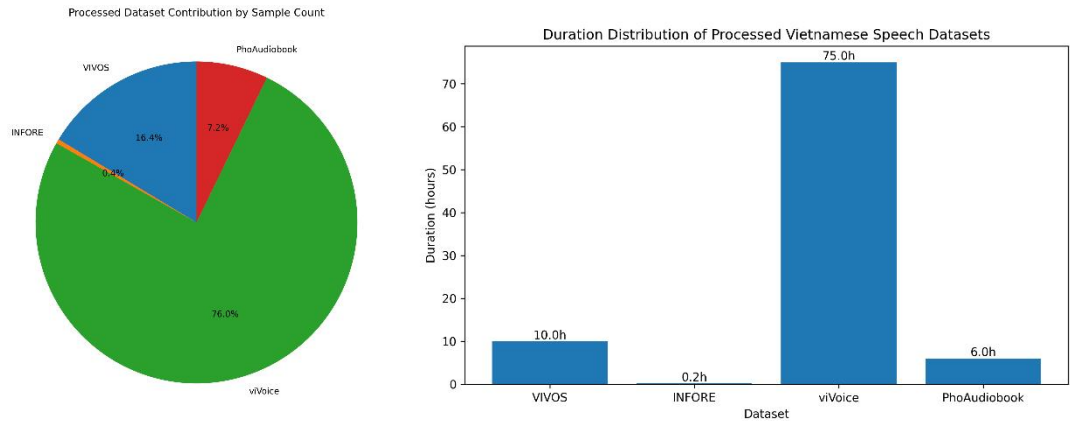



*Figure 4.1. Vietnamese dataset contribution visualization*

Figure 4.1 visualizes the contribution of each dataset before preprocessing. viVoice accounts for the largest portion of the selected samples, while VIVOS and INFORE provide cleaner core speech data. PhoAudiobook contributes additional speaker diversity and reading-style speech. This distribution shows that the constructed dataset combines both clean speech and diverse speech sources.

### 4.2.2. Preprocessing and tokenization results

The data preprocessing pipeline is designed to convert raw Vietnamese speech data into a training-ready format for NeuTTS-AIR. This stage is important because autoregressive speech models are sensitive to noisy text, inconsistent audio volume, abnormal duration, and misalignment between text and audio.

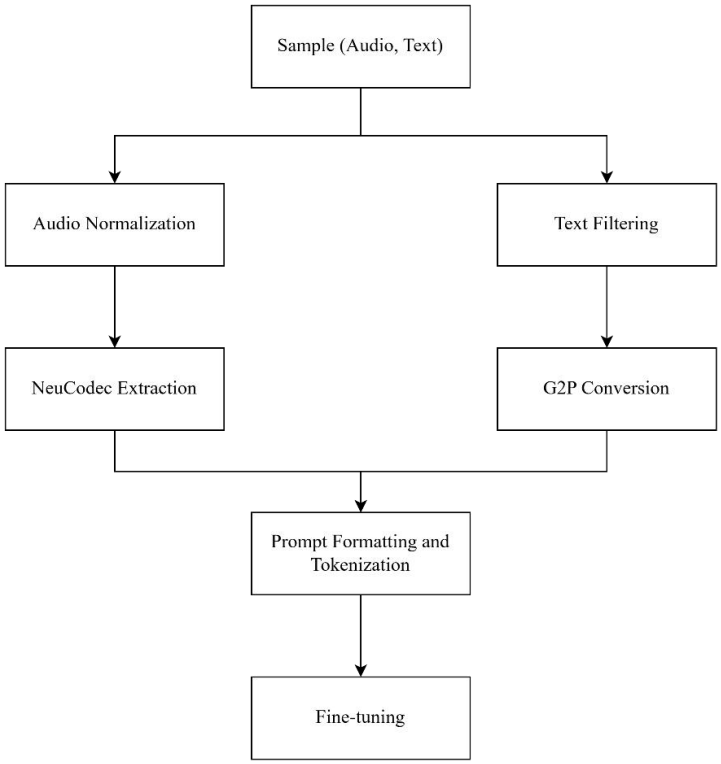



*Figure 4.2. Pipeline data preprocessing*

Figure 4.2 illustrates the preprocessing pipeline used in the experiments. The raw speech data is first normalized and filtered, then converted into phoneme and audio token representations. This step is important because NeuTTS-AIR does not directly learn from raw waveform audio during fine-tuning. Instead, it learns to predict discrete audio tokens conditioned on the Vietnamese phoneme prompt.

**Audio normalization:** Since the collected datasets originate from different recording environments and sampling rates, all audio files are first normalized into a unified format of 16 kHz and mono channel. The normalization process includes silence trimming, padding, noise reduction, loudness normalization, and anti-clipping. This step helps reduce acoustic variations across datasets and provides cleaner audio for subsequent token extraction.

**NeuCodec extraction:** After audio normalization, each sample is transformed into the conversational format required by NeuTTS-AIR. The corresponding speech signal is represented by discrete audio tokens extracted using NeuCodec. During

training, the loss is computed only on the generated audio token sequence, allowing the model to focus on speech generation rather than prompt reconstruction.

**Text filtering:** To improve data quality, unsuitable samples are removed before training. Text samples that are empty, excessively long, or contain unsupported patterns are filtered out. Audio samples outside the target duration range are also discarded. In addition, a ratio-based filtering strategy is applied to detect abnormal text-audio pairs, helping eliminate samples with excessive silence, incorrect transcriptions, or alignment issues.

**Grapheme-to-Phoneme conversion:** Vietnamese text is converted into phoneme sequences using the Espeak backend through the phonemizer library. Since NeuTTS-AIR is trained with phoneme-level text prompts, this step provides a more consistent pronunciation representation and helps the model learn Vietnamese pronunciation more effectively.

**Prompt Formatting and Tokenization:** After G2P conversion and audio code extraction, each sample is formatted into the prompt structure required by NeuTTS-AIR. The phoneme sequence is inserted between <|TEXT_PROMPT_START|> and <|TEXT_PROMPT_END|>, while the audio codes are converted into speech tokens and placed after <|SPEECH_GENERATION_START|>. The complete prompt is then tokenized using the original NeuTTS-AIR tokenizer to obtain input_ids. During training, the prompt part is masked with -100, so the loss is computed only on the target audio token sequence.

*Table 4.2. Vietnamese dataset construction after processing*

| Dataset | Number of speakers | Duration | Number of selected samples | Sampling rate |
|---|---|---|---|---|
| VIVOS | 46 | 10 hours | 11,221 | 16 kHz |
| INFORE | 1 | 0.2 hours | 300 | 16 kHz |
| viVoice | Không xác định | 55 hours | 38,284 | 16 kHz |
| PhoAudioBook | 708 | 6 hours | 4,937 | 16 kHz |
| Total | 935 | ~11.2 hours | 54,742 | 16 kHz |

Table 4.2 shows the final dataset after preprocessing and filtering. The final training set contains 54,742 tokenized samples with a total duration of approximately 71.2 hours. Compared with the selected dataset before preprocessing, some samples were removed due to filtering conditions such as abnormal duration, unsupported text,

or mismatched text-audio ratio. All remaining samples were resampled to 16 kHz and converted into precomputed input_ids and labels, which were directly used during fine-tuning.

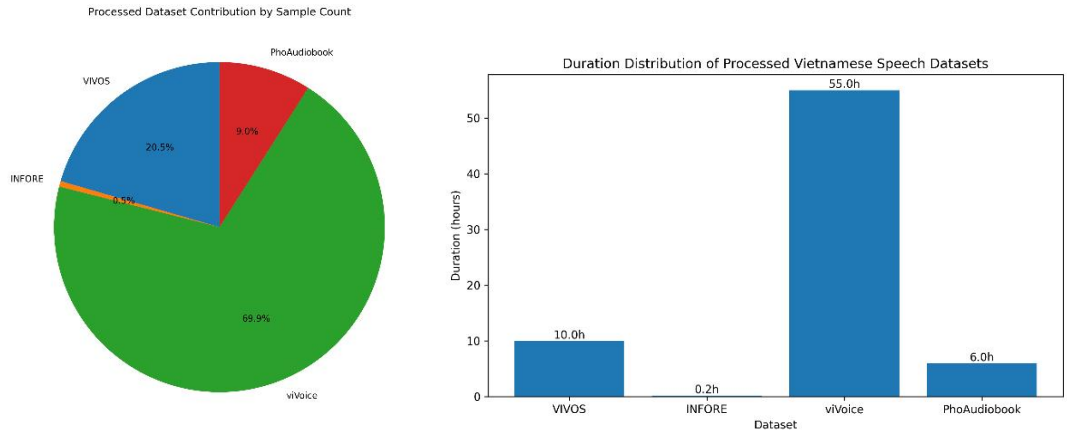



*Figure 4.3. Vietnamese dataset contribution visualization after processing*

Figure 4.3 shows the dataset contribution after preprocessing. Although viVoice remains the largest data source, the filtering process reduces the number of samples and produces a cleaner training set. This final dataset is used for both Phase-1 and Phase-2 Vietnamese adaptation experiments.

## 4.3. Experimental Setup

This section presents the experimental setup used for evaluating the Vietnamese adaptation of NeuTTS-AIR. It describes the dataset split, training environment, and evaluation metrics used in the experiments. The evaluation is designed to analyze both speech intelligibility and speaker similarity, which are the two main requirements of Vietnamese zero-shot voice cloning.

. The experiments use the preprocessed Vietnamese dataset described in Section 4.2. A validation subset is used during training to monitor checkpoint quality, while separate seen-speaker and unseen-speaker test sets are used for model evaluation. The generated speech is evaluated using WER, CER, SECS, and MCD to measure transcription accuracy, speaker similarity, and acoustic quality.

### 4.3.1. Dataset Split

The model was trained using the Vietnamese speech dataset constructed from VIVOS, INFORE, viVoice, and PhoAudiobook, as described in Chapter 3. The combined dataset contains approximately 935 speakers and 71.2 hours of Vietnamese speech. During training, 1% of the training data was separated as the validation set for monitoring training loss and checkpoint quality.

For model evaluation, two test subsets were prepared: a seen-speaker set and an unseen-speaker set. The seen-speaker set contains 300 test utterances from speakers within the training distribution. The unseen-speaker set contains 300 test utterances from 19 speakers that were not used during training. This separation is important because the main objective of the system is zero-shot voice cloning, where the model must synthesize speech for speakers not observed during fine-tuning.

*Table 4.3. Evaluation dataset split*

| **Subset** | **Number of utterances** | **Number of speakers** | **Purpose** |
|---|---|---|---|
| Seen speakers | 300 | Multi spakers | Evaluate zero-shot voice cloning ability |
| Unseen speakers | 300 | 19 speakers | Evaluate zero-shot voice cloning ability |

Table 4.3 summarizes the evaluation dataset split. The use of both seen and unseen speakers allows the experiment to analyze not only Vietnamese adaptation quality, but also the model's ability to preserve speaker characteristics for speakers not observed during fine-tuning.

### 4.3.2. Training environment

All experiments were conducted on the Modal cloud platform using an NVIDIA RTX 4090 GPU with 24 GB of VRAM. Mixed precision training was enabled using bfloat16 and TF32 to improve training efficiency. Gradient accumulation, gradient checkpointing, and bucketing were applied to reduce memory usage and improve batch efficiency for variable-length speech-token sequences.

*Table 4.4. Training environment*

| **Configuration** | **Value** | **Unit / Description** |
|---|---|---|
| Cloud platform | Modal | cloud GPU service |
| GPU | NVIDIA RTX 4090 | training GPU |
| GPU memory | 24GB | CUDA version |

| | | |
|---|---|---|
| CUDA version | 12.8 | version |
| Precision | bf16 / TF32 | mixed precision |
| Batch size per device | 2 | samples / GPU |
| Gradient accumulation steps | 16 | steps |
| Effective batch size | 32 | samples / update |
| Gradient checkpointing | Enabled | memory optimization |
| Bucketing sampler | Enabled | length-based batching |
| Maximum sequence length | 2,048 | tokens |
| PyTorch version | 2.3.0 | version |
| Transformers version | 4.56.1 | version |
| PEFT version | 0.11.1 | version |

Table 4.4 presents the training environment and key training configuration. Since speech token sequences can be long and variable in length, memory optimization techniques such as gradient checkpointing and bucketing were necessary to fit the experiments into the available GPU memory.

### 4.3.3. Evaluation protocol and metrics

To evaluate the generated speech, four objective metrics were used: Word Error Rate (WER), Character Error Rate (CER), Speaker Embedding Cosine Similarity (SECS), and Mel Cepstral Distortion (MCD).

WER and CER were used to measure speech intelligibility. The generated audio was first transcribed using Whisper large-v3, and the transcription was compared against the input text used for synthesis. Both WER and CER were computed using the jiwer library. Lower WER and CER values indicate better pronunciation accuracy and higher intelligibility.

SECS was used to evaluate speaker similarity. Speaker embeddings were extracted from both the reference audio and the generated audio using the SpeechBrain ECAPA-TDNN speaker recognition model. The cosine similarity between the two embeddings was then computed. A higher SECS value indicates that the generated speech is more similar to the target speaker.

MCD was used to estimate acoustic distortion between speech signals. In this experiment, MCD was computed using Calculate_MCD with MCD_mode="plain". A lower MCD value indicates better acoustic similarity and lower spectral distortion.

*Table 4.5. Evaluation metrics*

| Metric | Meaning | Tool/ Model | Direction |
|---|---|---|---|
| WER | Word-level transcription error | Whisper large-v3 + jiwer | Lower is better |
| CER | Character-level transcription error | Whisper large-v3 + jiwer | Lower is better |
| SECS | Speaker similarity between reference and generated audio | SpeechBrain ECAPA-TDNN | Higher is better |
| MCD | Acoustic distortion | Calculate_MCD, plain mode | Lower is better |

Table 4.5 summarizes the evaluation metrics used in this thesis. WER and CER mainly measure whether the generated speech follows the input text, while SECS measures whether the generated speech preserves the reference speaker identity. MCD provides an additional acoustic quality indicator based on spectral distortion.

## 4.4. Training results

### 4.4.1. Phase-1 training results: lm_head fine-tuning

In Phase 1, only the lm_head layer was fine-tuned while the transformer backbone was kept frozen. The objective of this stage was to perform a lightweight adaptation of the output token distribution to Vietnamese speech tokens.

The training loss decreased steadily from 8.2915 at step 250 to approximately 7.54 near the end of training. However, the validation loss quickly plateaued around 7.80 after approximately 2000–2500 steps. This indicates that although the model adapted slightly to the Vietnamese audio token distribution, updating only the output layer was not sufficient for high-quality Vietnamese speech generation.

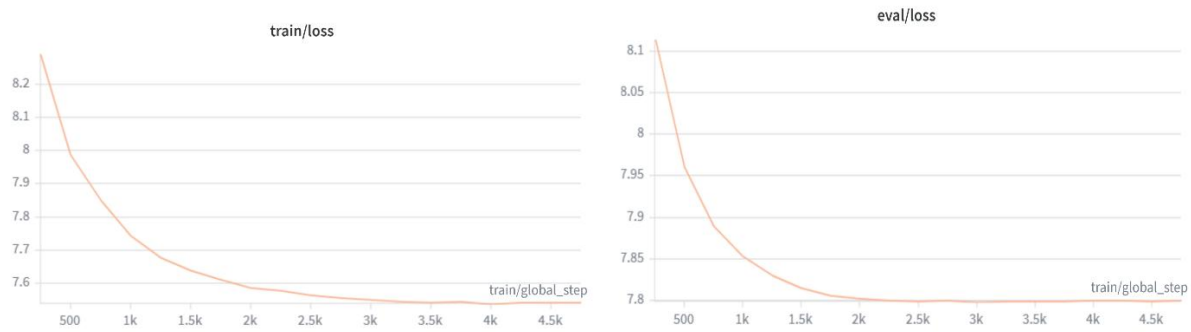



*Figure 4.4. Train loss and eval loss graph of phase 1*

As shown in Figure 4.4, the training loss decreased steadily during the first stage, indicating that the output layer gradually adapted to the Vietnamese audio token distribution. However, the validation loss quickly reached a plateau after approximately 2,000 to 2,500 steps. This suggests that updating only the lm_head layer was not sufficient to fully adapt the model to Vietnamese speech generation. Therefore, Phase 1 was mainly used as an initialization stage for deeper adaptation.

The quantitative evaluation of Phase 1 is shown in Table 4.6.

*Table 4.6. Phase-1 evaluation*

| **Dataset** | **Model** | **WER (%) ↓** | **CER (%) ↓** | **SECS ↑** | **MCD ↓** |
|---|---|---|---|---|---|
| Unseen speakers | ckpt-2250-lmhead | 135.16 | 116.99 | 0.4186 | 20.89 |
| | **ckpt-2500-lmhead** | **101.53** | **72.53** | **0.4287** | **20.74** |
| | ckpt-3000-lmhead | 136.88 | 104.45 | 0.4333 | 21.09 |
| | ckpt-3500-lmhead | 102.84 | 76.02 | 0.4471 | 20.53 |

Table 4.6 shows that Phase 1 alone did not produce reliable Vietnamese zero-shot voice cloning quality. Although checkpoint 2500 achieved the best balance among the evaluated Phase-1 checkpoints, WER and CER remained very high. This result confirms that deeper model adaptation is necessary.

### 4.4.2. Phase-2 Full fine-tuning results

In the full fine-tuning strategy, all trainable parameters of NeuTTS-AIR were updated using the Vietnamese dataset. This approach provides the model with

maximum adaptation capacity, but it also requires careful learning rate control to avoid damaging the pretrained representations.

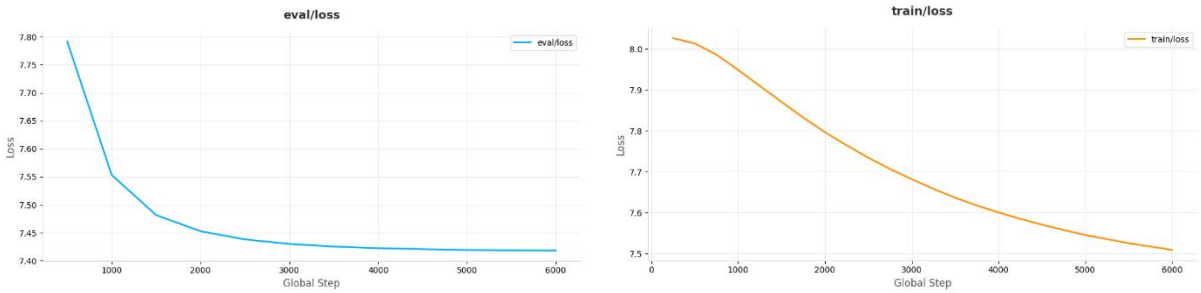



*Figure 4.5. Eval loss and trainloss graph of phase 2 Full fine-tuning*

Figure 4.5 shows the loss behavior during full fine-tuning. Compared with Phase 1, the model was able to adapt more deeply because all model parameters were updated. The loss curves are used mainly to monitor training stability, while the final checkpoint selection is based on the objective evaluation metrics reported in Table 4.7.

*Table 4.7. Phase-2 full fine-tuning evaluation results*

| **Dataset** | **Model** | **WER (%) ↓** | **CER (%) ↓** | **SECS ↑** | **MCD ↓** |
|---|---|---|---|---|---|
| Seen speakers | ckpt-3000 | 44.27 | 25.25 | 0.4967 | 20.57 |
| | ckpt-4000 | 35.22 | 22.55 | 0.4955 | 20.35 |
| | ckpt-5000 | 29.43 | 17.66 | 0.5088 | 20.39 |
| Unseen speakers | ckpt-3000 | 40.44 | 22.04 | 0.5795 | 17.97 |
| | ckpt-4000 | 39.50 | 21.67 | 0.5734 | 17.89 |
| | **ckpt-5000** | **37.60** | **19.65** | **0.5888** | **17.78** |

As shown in Table 4.7, full fine-tuning consistently improved over Phase 1. On the unseen-speaker set, WER decreased from 40.44% at checkpoint 3000 to 37.60% at checkpoint 5000, while CER decreased from 22.04% to 19.65%. SECS also remained stable and reached 0.5888 at checkpoint 5000. However, the performance was still lower than the best Global LoRA checkpoint, suggesting that full-parameter adaptation did not provide the best zero-shot generalization in this experiment.

### 4.4.3. Phase-2 Global LoRA results

In the Global LoRA strategy, low-rank adaptation modules were inserted into all 24 decoder layers of NeuTTS-AIR, covering both attention and MLP projections. This strategy updated only a subset of parameters while preserving the majority of the

pretrained model. The number of trainable parameters was 230,209,920 out of 783,124,224 total parameters, corresponding to approximately 29.40% of the model.

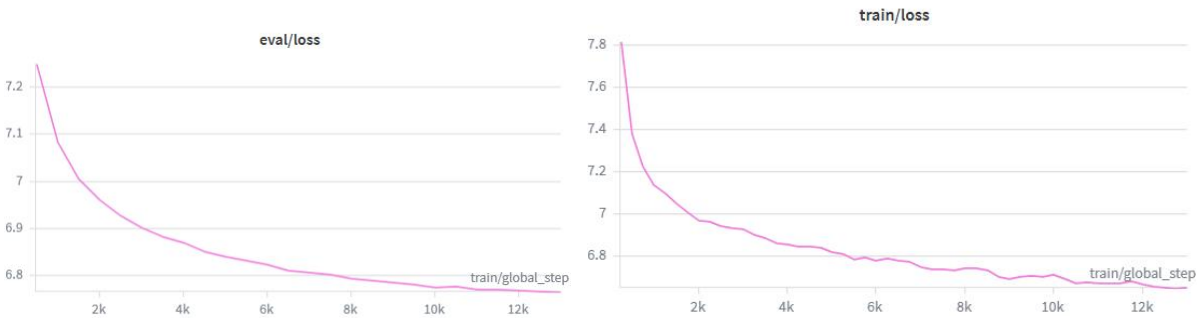



*Figure 4.6. Eval loss and trainloss graph of phase 2 Global LoRA*

Figure 4.6 presents the loss curves of Global LoRA training. The loss graph is used to observe training stability and detect possible degradation during later checkpoints. However, as with the other phases, the final decision is based primarily on the objective evaluation metrics because lower training loss does not always guarantee better speech intelligibility or speaker similarity.

*Table 4.8. Phase-2 Global LoRA results*

| **Dataset** | **Model** | **WER (%) ↓** | **CER (%) ↓** | **SECS ↑** | **MCD ↓** |
|---|---|---|---|---|---|
| Seen speakers | ckpt-10000 | 10.67 | 7.05 | 0.5917 | 18.67 |
| | ckpt-11000 | 8.51 | 5.59 | 0.5941 | 18.92 |
| | ckpt-12000 | 8.31 | 5.35 | 0.5915 | 19.1 |
| Unseen speakers | ckpt-10000 | 13.35 | 7.42 | 0.6165 | 17.1 |
| | **ckpt-11000** | **12.04** | **6.24** | **0.6138** | **17.17** |
| | ckpt-12000 | 12.51 | 6.58 | 0.6158 | 17.22 |

Based on the results, checkpoint 11000 was selected as the final Global LoRA checkpoint because it provides the best overall balance across intelligibility, speaker similarity, and acoustic quality. Although checkpoint 12000 achieved slightly lower WER and CER on the seen-speaker set, checkpoint 11000 produced better performance on the unseen-speaker set in terms of WER and CER, which is more important for the zero-shot voice cloning objective of this thesis. In addition, checkpoint 11000 maintained a high SECS score and a competitive MCD value, showing that it could preserve speaker similarity while generating clear Vietnamese

speech. Therefore, checkpoint 11000 was chosen as the most suitable checkpoint for the final application prototype.

## 4.5. Model comparison and discusion

### 4.5.1. Comparison of baseline and best checkpoints

To evaluate the contribution of the proposed adaptation strategy, the original NeuTTS-AIR model was also evaluated as the baseline model before Vietnamese fine-tuning. The baseline results are compared with the selected checkpoints from Phase 1, full fine-tuning, and Global LoRA in Table 4.9.

*Table 4.9. Comparison of baseline and best checkpoints*

| **Dataset** | **Strategy** | **Best model** | **WER(%)↓** | **CER(%)↓** | **SECS ↑** | **MCD ↓** |
|---|---|---|---|---|---|---|
| Seen speakers | Baseline NeuTTS-Air | Original | 265.12 | 257.61 | 0.2983 | 20.42 |
| | Phase 1: lm_head fine-tuning | ckpt-2500-lmhead | 123.13 | 92.95 | 0.3859 | 21.53 |
| | Phase 2: Full fine-tuning | ckpt-5000 | 29.43 | 17.66 | 0.5088 | 20.39 |
| | **Phase 2: Global LoRA** | **ckpt-11000** | **8.51** | **5.59** | **0.5941** | **18.92** |
| Unseen speakers | Baseline NeuTTS-Air | Original | 285.85 | 223.34 | 0.3065 | 20.17 |
| | Phase 1: lm_head fine-tuning | ckpt-2500-lmhead | 101.53 | 72.53 | 0.4287 | 20.74 |
| | Phase 2: Full fine-tuning | ckpt-5000 | 37.60 | 19.65 | 0.5888 | 17.78 |
| | **Phase 2: Global LoRA** | **ckpt-11000** | **12.04** | **6.24** | **0.6138** | **17.17** |

Table 4.9 compares the original baseline model with the selected checkpoints from each adaptation stage. The baseline NeuTTS-AIR model produced very high WER and CER on both seen and unseen speakers, showing that the original model was not suitable for Vietnamese speech generation without adaptation. After applying the proposed fine-tuning strategies, the performance improved significantly.

On the unseen-speaker set, the baseline model achieved 285.85% WER and 223.34% CER, while the selected Global LoRA checkpoint reduced these values to 12.04% and 6.24%, respectively. Speaker similarity also improved from 0.3065 to 0.6138, and MCD decreased from 20.17 to 17.17. These results demonstrate that the proposed Vietnamese adaptation pipeline substantially improves speech intelligibility, speaker similarity, and acoustic quality compared with the original baseline model.

Among the adaptation strategies, Global LoRA achieved the best overall performance on both seen and unseen speakers. Although full fine-tuning also improved the model compared with the baseline, it did not outperform Global LoRA. This suggests that Global LoRA provides a better balance between adapting the model to Vietnamese speech and preserving the pretrained zero-shot voice cloning capability.

### 4.5.2. Seen and unseen speaker analysis

On the seen-speaker evaluation set, Global LoRA achieved 8.51% WER and 5.59% CER, while full fine-tuning achieved 29.43% WER and 17.66% CER at its best checkpoint. This indicates that Global LoRA learned the Vietnamese speech distribution more effectively while maintaining stable generation quality.

On the unseen-speaker set, Global LoRA also achieved the best speaker similarity score with SECS of 0.6138. This result is important because the main goal of the system is zero-shot voice cloning, where the model must preserve the speaker characteristics from a short reference audio sample. The higher SECS score suggests that Global LoRA better preserves speaker identity compared with full fine-tuning and Phase-1-only adaptation.

### 4.5.3. Qualitative Observation

Qualitative listening also confirmed the objective results. Phase-1 outputs were often unstable, with frequent mispronunciations, hallucinated content, and unclear Vietnamese pronunciation. Full fine-tuning improved intelligibility and reduced hallucination, but the synthesized speech was still less stable than the best Global LoRA checkpoint. Global LoRA at checkpoint 11000 produced clearer Vietnamese

speech, better pronunciation, and stronger speaker similarity, making it the selected checkpoint for the application prototype.

# CHAPTER 5: SYSTEM ANALYSIS AND DESIGN

## 5.1. System Architecture

After fine-tuning NeuTTS-AIR for Vietnamese voice cloning, the model is integrated into a complete client-server system to make it accessible to end users. The proposed system is designed as a multi-layer architecture consisting of a mobile application, backend API server, asynchronous task queue, cloud-based inference server, database, and object storage.

### 5.1.1. Overall System Architecture

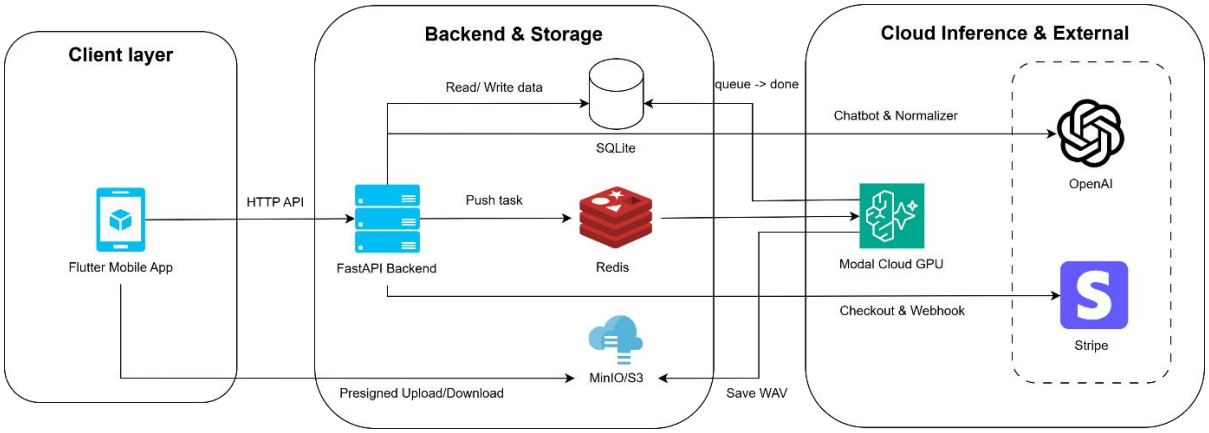



*Figure 5.1. Overall System Architecture*

Figure 5.1 shows the overall architecture of the proposed voice cloning system. The architecture separates the system into several layers, including the Flutter mobile application, FastAPI backend, Redis task queue, Modal GPU inference service, SQLite database, and MinIO-compatible object storage. This separation helps keep the mobile application lightweight, allows the backend to remain responsive, and moves heavy AI inference tasks to the cloud GPU layer.

The main voice cloning workflow is asynchronous. First, the mobile application uploads reference audio and sends a generation request. The backend records the request and pushes a task to Redis. The background worker processes the task on Modal GPU, stores the generated WAV file in cloud storage, updates the database, and allows the mobile application to retrieve the result through polling.

### 5.1.2. Mobile Application Layer

The mobile application is implemented using Flutter and serves as the main user interface of the system. It provides authentication screens for login and registration, a

home screen for text input and speech generation, a library screen for managing voice profiles and generated audios, and a profile screen for account and token management.

The application also supports local audio interaction, including recording, playback, and basic audio processing. For generated speech, users can listen to the result, manage their audio history, and download or share audio files. By separating the mobile layer from the inference layer, the application remains lightweight while still providing access to a large-scale voice cloning model.

### 5.1.3. Backend API Layer

The backend is implemented using FastAPI and acts as the central coordination layer of the system. It exposes HTTP-based APIs for authentication, voice profile management, audio generation, payment handling, and chatbot-related functions. JWT authentication is used to protect user-specific resources and maintain stateless communication between the client and server.

The backend also manages the interaction between the mobile application, database, object storage, Redis queue, and external services. For example, it generates presigned URLs for direct audio upload, confirms uploaded reference audio, creates voice cloning tasks, and tracks the status of generated audio. This design allows the backend to remain responsive while long-running AI tasks are executed asynchronously.

The main API groups include authentication APIs, voice APIs, payment APIs, and chatbot APIs. Authentication APIs handle user registration, login, and profile retrieval. Voice APIs manage text normalization, voice profile creation, generated audio history, and voice cloning requests. Payment APIs support account upgrade through Stripe, while chatbot APIs provide AI-assisted text generation for users.

### 5.1.4. Model Inference Layer

The model inference layer is deployed on Modal cloud GPU infrastructure. This layer is responsible for executing computationally intensive AI tasks that cannot be efficiently handled by the backend server. The inference pipeline includes reference audio verification, voice feature extraction, NeuCodec-based audio code processing, and speech generation using the selected fine-tuned NeuTTS-AIR checkpoint.

The inference process is divided into two stages. In the first stage, the system extracts and caches information from the reference audio, including audio codes and phoneme representations. In the second stage, when the user requests speech

generation, the system receives the input text, cached reference voice information, and phoneme sequence, then generates a WAV audio output.

Before inference, the input text is normalized using a hybrid text normalization approach. Rule-based normalization handles common patterns such as units, currencies, abbreviations, and symbols. For more complex cases such as long numbers, dates, and context-dependent spoken forms, an LLM-based normalizer is used to convert the input into a more natural Vietnamese spoken form. This improves the pronunciation quality of the generated speech.

### 5.1.5. Database Design

The system uses SQLite as the relational database for managing users, voice profiles, generated audios, and chatbot sessions. The database stores metadata rather than large audio files. Actual audio files, including reference audio and generated audio, are stored in cloud object storage, while the database keeps their corresponding file paths.

Before describing each table in detail, the overall entity relationship diagram of the database is shown in Figure 5.2.

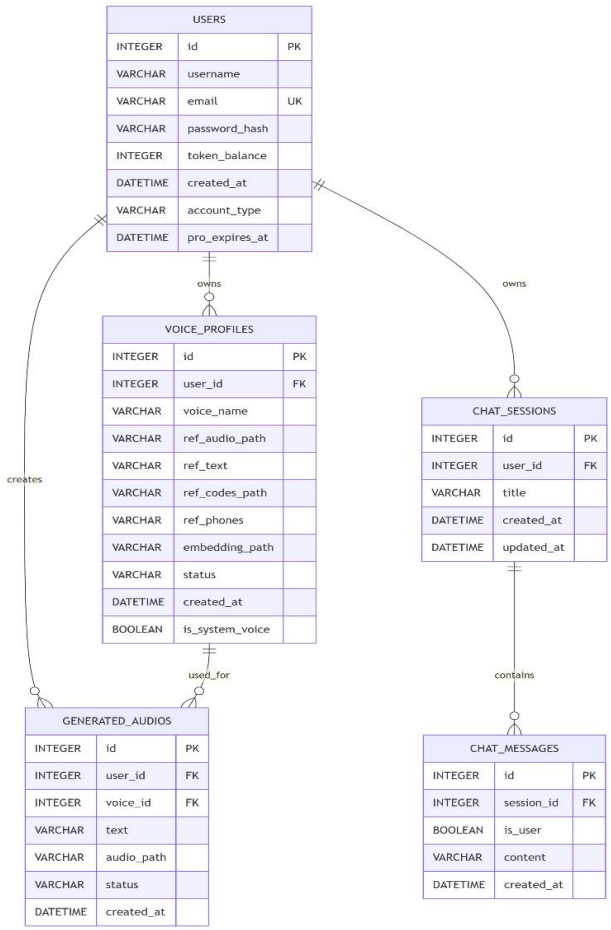



*Figure 5.2. Entity Relationship Diagram of the voice cloning system*

Figure 5.2 illustrates the main entities and relationships in the system database. A user can own multiple voice profiles and generate multiple audio records. Each generated audio record is associated with one selected voice profile and one user. In addition, each user can create multiple chatbot sessions, and each chatbot session can contain multiple chat messages. This structure separates user management, voice profile management, generated audio history, and chatbot interaction into different entities, making the system easier to maintain and extend.

*Table 5.1. User entity details*

| **Attributes** | **Datatype** | **Null** | **Key** | **Default** |
|---|---|---|---|---|
| id | INTEGER | No | PK | |
| username | VARCHAR | No | | |
| email | VARCHAR | No | UNIQUE | |
| password_hash | VARCHAR | No | | |
| token_balance | INTEGER | No | | 500 |
| created_at | DATETIME | No | | CURRENT_TIMESTAMP |
| account_type | VARCHAR(50) | No | | basic |
| pro_expires_at | DATETIME | Yes | | |

Table 5.1 presents the structure of the users table. This table is the central entity of the database because most user-specific resources, such as voice profiles, generated audios, and chatbot sessions, are linked to a user account through foreign keys. The token_balance, account_type, and pro_expires_at fields are used to support usage control and premium account features.

*Table 5.2. Voices profile entity details*

| **Attributes** | **Datatype** | **Null** | **Key** | **Default** |
|---|---|---|---|---|
| id | INTEGER | No | PK | |
| user_id | INTEGER | No | FK | |
| voice_name | VARCHAR | No | | |
| ref_audio_path | VARCHAR | No | | |
| ref_text | VARCHAR | No | | |
| ref_codes_path | VARCHAR | Yes | | |

| | | | | |
|---|---|---|---|---|
| ref_phones | VARCHAR | Yes | | |
| embedding_path | VARCHAR | Yes | | |
| status | VARCHAR | No | | processing |
| created_at | DATETIME | No | | CURRENT_TIMESTAMP |
| is_system_voice | BOOLEAN | No | | false |

Table 5.2 describes the voice_profiles table. Each voice profile belongs to a user and stores the metadata required for reusable voice cloning. Instead of storing large extracted features directly in the database, the system stores paths such as ref_codes_path and embedding_path. This design keeps the database lightweight while allowing the inference layer to reuse cached voice information.

In addition to basic metadata, the table stores paths to preprocessed voice information, such as ref_codes_path, ref_phones, and embedding_path. These fields help the system reuse extracted voice features without processing the same reference audio repeatedly. The status field tracks the processing state of the profile, while is_system_voice indicates whether the profile is a default system voice or a user-created voice.

*Table 5.3. Generated audio entity details*

| **Attributes** | **Datatype** | **Null** | **Key** | **Default** |
|---|---|---|---|---|
| id | INTEGER | No | PK | |
| user_id | INTEGER | No | FK | |
| voice_id | INTEGER | No | FK | |
| text | VARCHAR | No | | |
| audio_path | VARCHAR | Yes | | |
| status | VARCHAR | No | | queued |
| created_at | DATETIME | No | | CURRENT_TIMESTAMP |

Table 5.3 shows the generated_audios table. This table records each voice cloning request submitted by a user. Each record contains the input text, the selected voice profile, the output audio path, and the generation status. The status field is important for asynchronous processing because a generation request may be queued, processing, completed, or failed.

*Table 5.4. Chat session entity details*

| Attributes | Datatype | Null | Key | Default |
|---|---|---|---|---|
| id | INTEGER | No | PK | |
| user_id | INTEGER | No | FK | |
| title | VARCHAR | No | | Đoạn chat mới |
| created_at | DATETIME | No | | CURRENT_TIMESTAMP |
| updated_at | DATETIME | No | | CURRENT_TIMESTAMP |

Table 5.4 presents the chat_sessions table. This table groups chatbot messages into separate conversations. Each session belongs to one user and stores basic metadata such as the session title, creation time, and update time.

*Table 5.5. Chat message entity details*

| Attributes | Datatype | Null | Key | Default |
|---|---|---|---|---|
| id | INTEGER | No | PK | |
| session_id | INTEGER | No | FK | |
| is_user | BOOLEAN | No | | |
| content | VARCHAR | No | | |
| created_at | DATETIME | No | | CURRENT_TIMESTAMP |

Table 5.5 describes the chat_messages table. Each message belongs to a chat session and stores whether the message was sent by the user or generated by the chatbot. This design allows the application to reconstruct the full conversation history for script generation and user interaction.

## 5.2. System Analysis and Design

This section presents the system analysis and design of the proposed Vietnamese voice cloning application. The purpose is to describe how users interact with the system and how the main components collaborate to complete key business workflows. Instead of focusing on low-level user interface actions, the analysis emphasizes the core system-level functions, including user authentication, voice profile creation, speech generation, audio history management, and optional premium features such as AI chatbot support and account upgrade.

The system is modeled using use case and sequence diagrams. The use case diagram provides an overview of the main actors and their interactions with the system, while the sequence diagrams describe the internal communication flow between the mobile application, backend server, database, message queue, cloud GPU inference service, and object storage.

### 5.2.1. Use case diagram

#### a. Overall use case

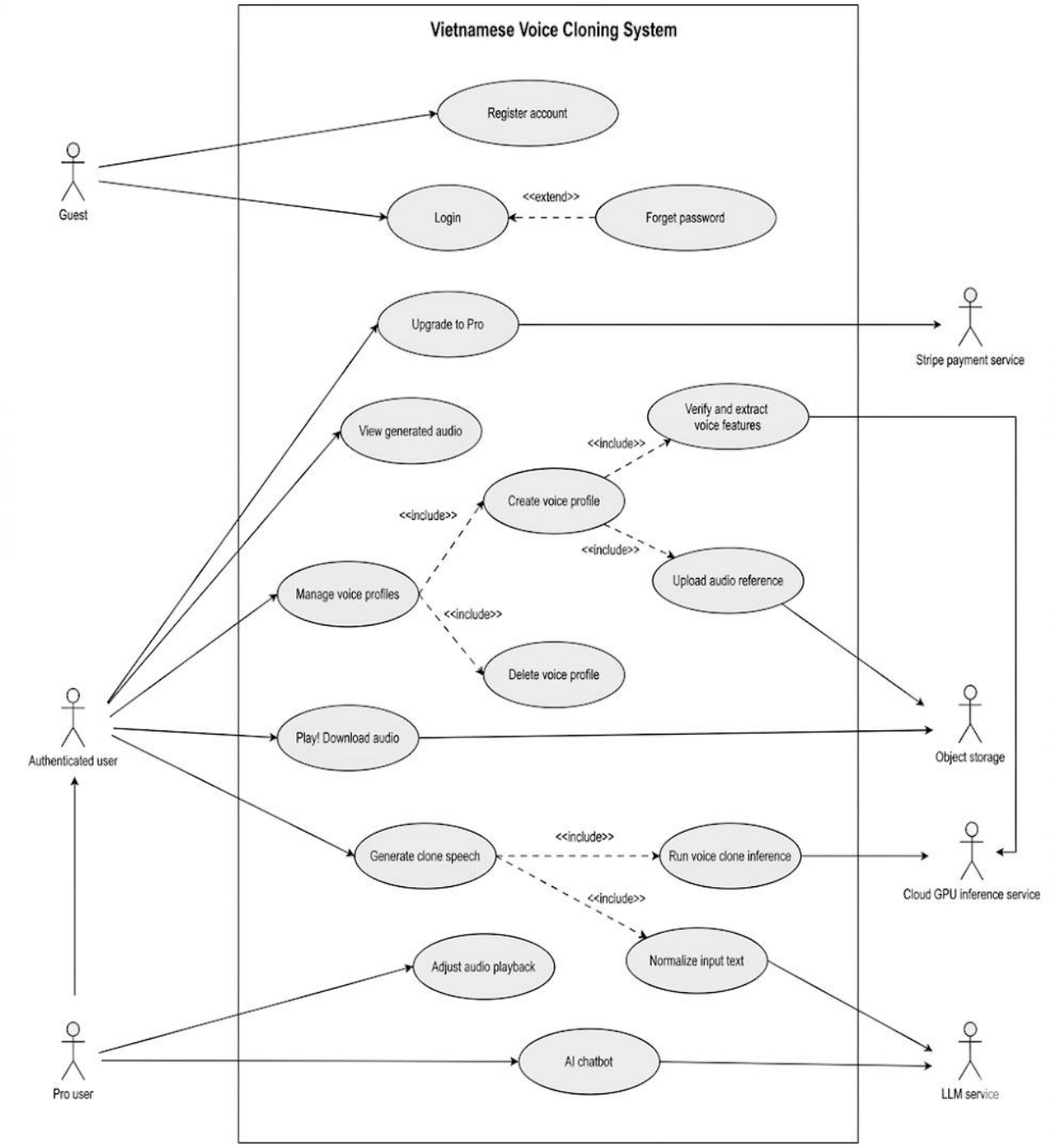



*Figure 5.3. Overall use case diagram*

Figure 5.3 presents the main use cases of the system. The core users include guests, authenticated users, and Pro users. Guests can register and log in, while

authenticated users can create voice profiles, generate cloned speech, manage generated audio, and use chatbot support. Pro users have access to additional premium features. External services such as Stripe, object storage, and cloud GPU inference are also involved in supporting the complete workflow.

The overall use case diagram provides a high-level view of how different actors interact with the Vietnamese voice cloning system. It shows that the system is centered around the authenticated user, who can manage voice profiles, submit voice cloning requests, and access generated audio results. The diagram also separates basic user functions from premium features, making it clear which functions are available to normal users and which functions are reserved for Pro users. In addition, the connection with external services shows that the application depends not only on the mobile and backend layers, but also on cloud storage, GPU inference, payment processing, and LLM-based support to complete the full workflow.

**b. Authentication use case**

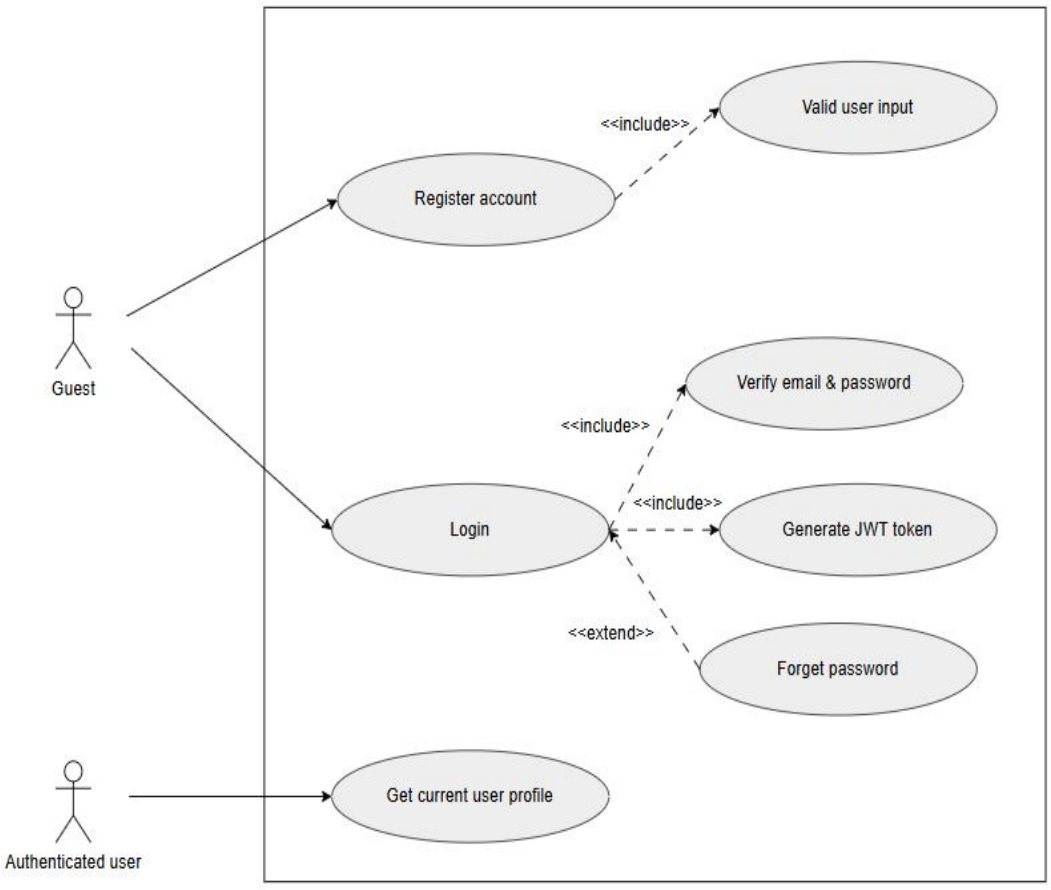



*Figure 5.4. Authentication use case diagram*

Figure 5.4 focuses on the authentication-related use cases. The authentication module includes account registration, user login, forgot password, profile retrieval, and

protected access through JWT tokens. The Forgot Password use case is modeled as an optional recovery flow that allows users to regain access when they cannot log in with their current password. This module is important because most system functions, such as voice profile creation and speech generation, require a valid authenticated user.

**c. Voice profile management use case**

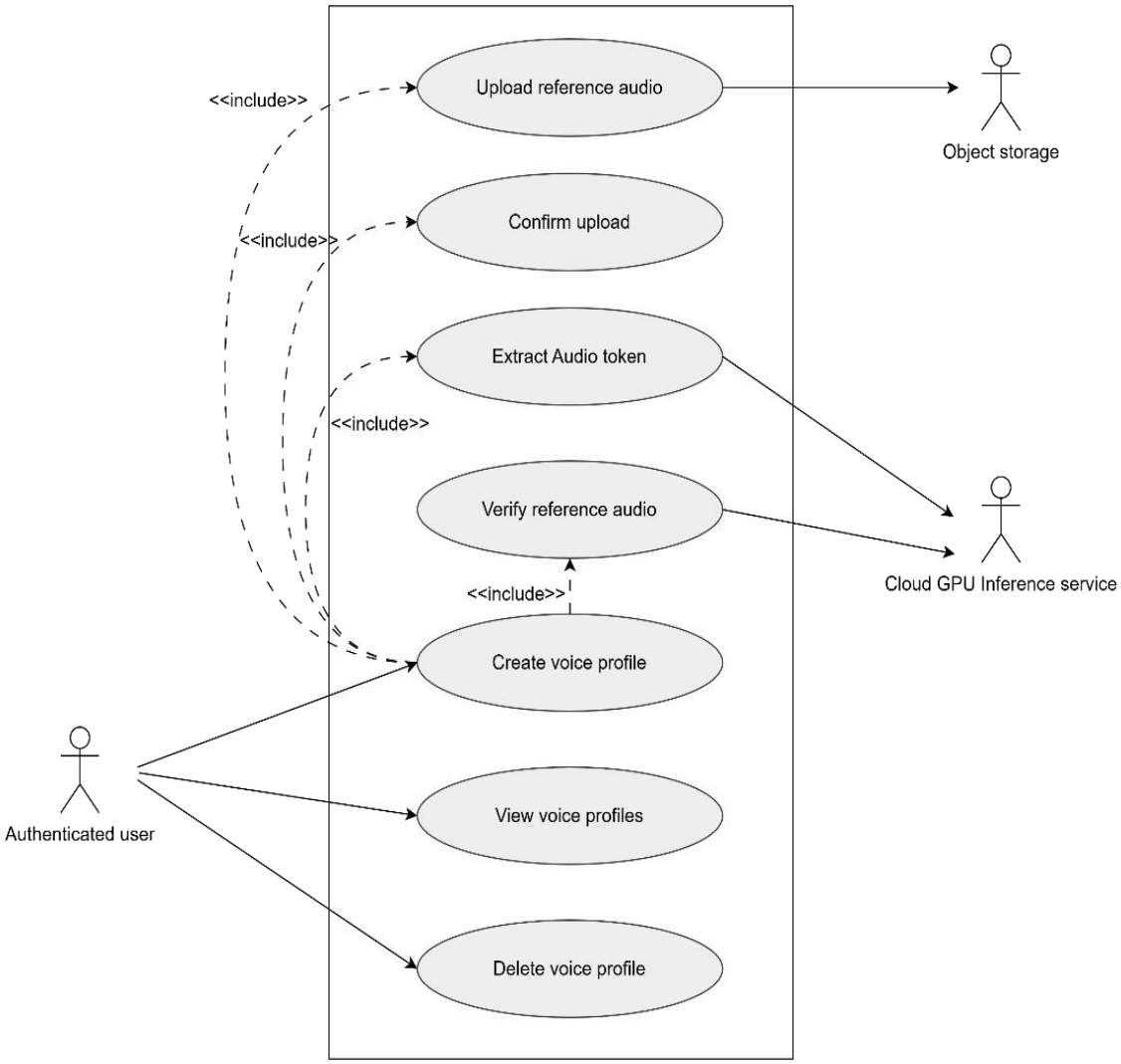



*Figure 5.5. Voice profile management use case diagram*

Figure 5.5 shows how users create and manage voice profiles. A voice profile is created from a reference audio sample and its corresponding reference text. After the reference audio is uploaded, the system extracts reusable voice information and updates the profile status. This allows users to reuse the same voice profile for multiple speech generation requests.

**d. Voice cloning and generated audio management use case**

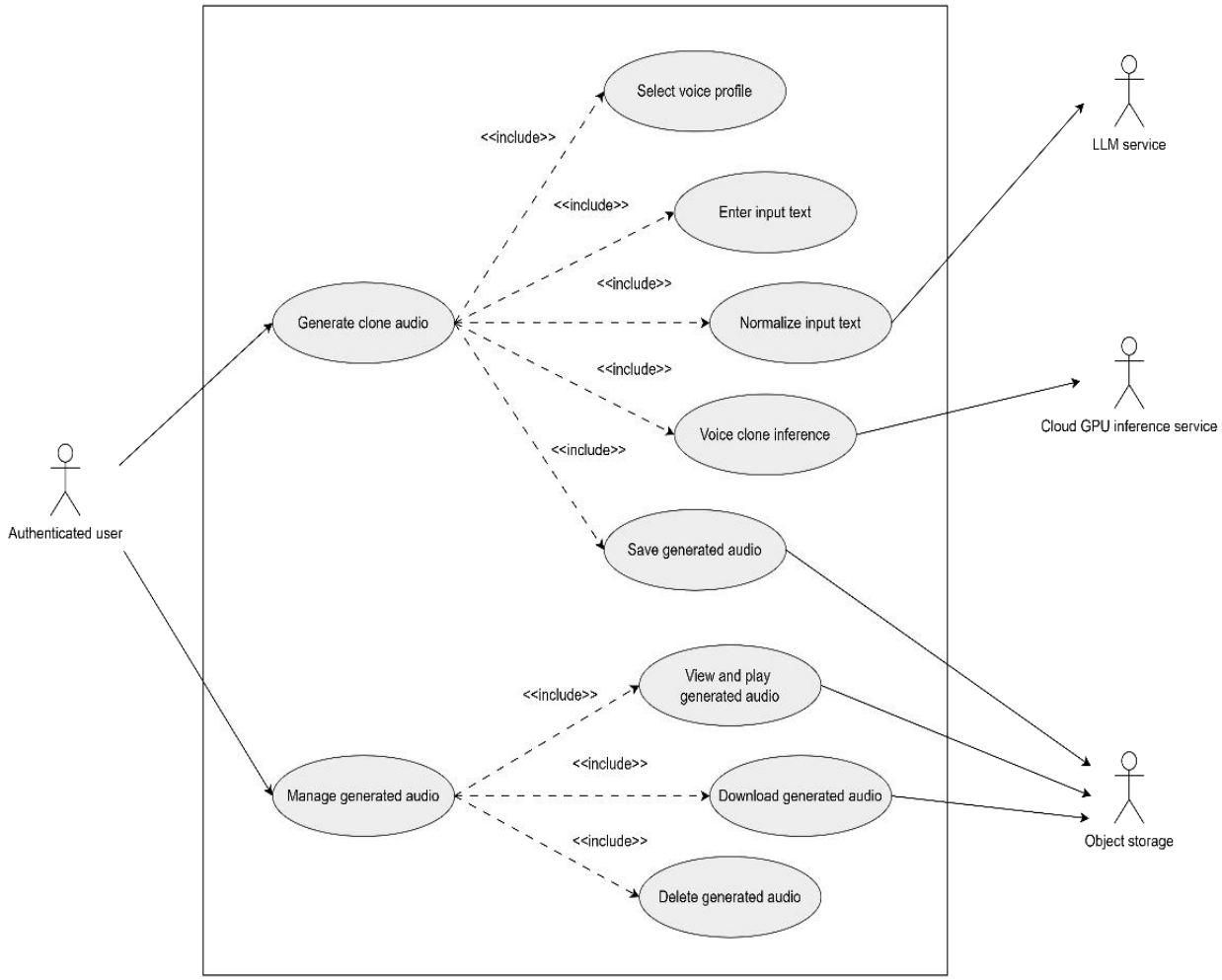



*Figure 5.6. Voice clone and generated audio management use case diagram*

Figure 5.6 describes the main voice cloning and generated audio management use cases from the user perspective. In this group of functions, the authenticated user first selects an available voice profile and enters the text to be synthesized. The system then validates the request, checks whether the selected voice profile is ready, creates a generation task, and sends it to the inference pipeline. After the backend and inference service complete the task, the generated audio is stored and made available in the user's audio library.

This diagram also shows that generated audio management is an important part of the complete workflow. After speech generation is completed, the user can view the generated audio history, play the audio, download the result, share it, or delete unused records. Therefore, this use case group covers not only the speech generation request itself, but also the management of generated results after inference. This design helps users reuse, organize, and access their generated speech files more conveniently.

### e. Premium features use case diagram




*Figure 5.7. Premium features use case diagram*

Figure 5.7 presents the optional premium features of the system. These features include chatbot-assisted script generation, audio playback adjustment, and account upgrade through Stripe. Although these functions are not part of the core voice cloning model, they improve the usability and practical value of the application by helping users prepare more suitable input text, customize generated audio playback, and access extended system capabilities. The chatbot feature supports users in creating or refining TTS-friendly scripts before speech generation, while the audio playback adjustment feature allows Pro users to modify playback speed and pitch of generated speech. In addition, the Pro upgrade feature provides a payment-based mechanism for expanding user access and managing premium usage.

### 5.2.2. Use case specifications

#### a. Use case specification for Register Account

*Table 5.6. Use case specification for Register Account*

| **Field** | **Description** |
|---|---|
| Use case ID | UC-01 |
| Use Case Name | Register Account |
| Actor | Guest |
| Description | Allows a new user to create an account in the system. |
| Pre-condition | The user has not logged in and provides valid registration information. |
| Trigger | The user submits the registration form. |
| Main Flow | 1. The user enters username, email, and password.<br>2. The mobile application sends the registration request to the backend.<br>3. The backend validates the input data.<br>4. The backend checks whether the email already exists.<br>5. The password is hashed before being stored.<br>6. A new user record is created with the default token balance and basic account type.<br>7. The system returns the registration result to the mobile application. |
| Alternative Flow | A1. The user cancels the registration process before submission.<br>A2. The user selects the Forgot Password option. The system asks the user to enter the registered email address. The backend verifies whether the email exists and sends a password recovery link or verification code to the user. After successful verification, the user can set a new password and return to the login screen. |
| Exception Flow | E1. If the email already exists, the system returns a |

| | |
|---|---|
| | registration error.<br>E2. If the input data is invalid, the system requests the user to correct the information.<br>E3. If the password recovery email or verification code cannot be sent, the system displays a recovery failure message. |
| Post-condition | The user is authenticated and can access protected features if the login is successful. If the Forgot Password flow is completed, the user can log in again using the new password. |
| Priority | High |

**b. Use case specification for Login**

*Table 5.7. Use case specification for Login*

| **Field** | **Description** |
|---|---|
| Use case ID | UC-02 |
| Use Case Name | Login |
| Actor | Guest |
| Description | Authenticates the user and provides an access token for using protected system functions. |
| Pre-condition | The user already has a registered account |
| Trigger | The user submits email and password on the login screen |
| Main Flow | 1. The user enters email and password.<br>2. The mobile application sends the login request to the backend.<br>3. The backend searches for the user by email.<br>4. The backend verifies the password.<br>5. If the credentials are valid, the backend generates a JWT access token.<br>6. The token and user information are returned to the mobile |

| | |
|---|---|
| | application.<br>7. The application stores the token locally and navigates the user to the main screen. |
| Alternative Flow | A1. The user chooses to go to the registration screen instead. |
| Exception Flow | E1. If the email does not exist, the system returns an authentication error.<br>E2. If the password is incorrect, the system rejects the login request. |
| Post-condition | The user is authenticated and can access protected features. |
| Priority | High |

**c. Use case specification for Create Voice Profile**

*Table 5.8. Use case specification for Create Voice Profile*

| **Field** | **Description** |
|---|---|
| Use case ID | UC-03 |
| Use Case Name | Create Voice Profile |
| Actor | Authenticated user, |
| Description | Allows the user to create a reusable voice profile from a reference audio sample and its corresponding text. |
| Pre-condition | The user is logged in and has a valid reference audio sample. |
| Trigger | The user submits a request to create a new voice profile. |
| Main Flow | 1. The user records or selects a reference audio file.<br>2. The user provides the reference text and voice profile name.<br>3. The mobile application requests a presigned upload URL from the backend.<br>4. The reference audio is uploaded directly to object storage.<br>5. The mobile application confirms the upload with the backend.<br>6. The backend creates a voice profile record with “processing” status.<br>7. The backend pushes an “extract_profile” task to Redis. |

| | |
|---|---|
| | 8. The background worker sends the task to the Modal GPU inference service.<br>9. The inference service verifies the reference audio and extracts reusable voice features.<br>10. The extracted audio codes and phoneme information are stored.<br>11. The backend updates the voice profile status to “ready”. |
| Alternative Flow | A1. The user cancels the process before confirming the upload. |
| Exception Flow | E1. If the audio upload fails, the voice profile is not created.<br>E2. If the reference audio is invalid, the system updates the status to rejected.<br>E3. If feature extraction fails, the system keeps the profile in a failed or rejected state. |
| Post-condition | A new voice profile is available for speech generation if processing succeeds. |
| Priority | Very high |

**d. Use case specification for Generate Cloned Speech**

*Table 5.9. Use case specification for Generate Cloned Speech*

| **Field** | **Description** |
|---|---|
| Use case ID | UC-04 |
| Use Case Name | Generate Cloned Speech |
| Actor | Authenticated User |
| Description | Allows the user to generate speech from input text using a selected voice profile. |
| Pre-condition | The user is logged in and has at least one voice profile with “ready” status |
| Trigger | The user enters text, selects a voice profile, and submits a generation request. |
| Main Flow | 1. The user enters the text to be synthesized.<br>2. The user selects an available voice profile. |

| | |
|---|---|
| | 3. The mobile application sends the generation request to the backend.<br>4. The backend validates the JWT token and request data.<br>5. The backend normalizes the input text.<br>6. The backend creates a generated audio record with queued status.<br>7. The backend pushes a generate_audio task to Redis.<br>8. The background worker sends the task to the Modal GPU inference service.<br>9. The inference service generates speech using the selected fine-tuned NeuTTS-AIR checkpoint.<br>10. The generated WAV file is uploaded to object storage.<br>11. The database is updated with the output audio path and done status.<br>12. The mobile application retrieves the result and displays it to the user. |
| Alternative Flow | A1. The user cancels the request before submission.<br>A2. The user edits the input text before generating audio. |
| Exception Flow | E1. If the user is not authenticated, the system returns an authorization error.<br>E2. If the selected voice profile is not ready, the request is “rejected”.<br>E3. If inference fails, the generated audio status is updated to “failed”. |
| Post-condition | A generated audio record is created and can be played or downloaded by the user. |
| Priority | Very high |

**e. Use case specification for Manage Gẽnated Audio**

*Table 5.10. Use case specification for Manage Generated Audio*

| **Field** | **Description** |
|---|---|
| Use case ID | UC-05 |
| Use Case Name | Manage Generated Audio |

| | |
|---|---|
| Actor | Authenticated User |
| Description | Allows the user to view, play, download, share, or delete generated audio files. |
| Pre-condition | The user is logged in and has at least one generated audio record. |
| Trigger | The user opens the generated audio library. |
| Main Flow | 1. The user opens the generated audio history screen.<br>2. The mobile application requests the list of generated audios from the backend.<br>3. The backend validates the JWT token.<br>4. The backend retrieves generated audio records belonging to the user.<br>5. The system returns metadata such as text, status, creation time, and audio path.<br>6. The user selects an audio item.<br>7. The application plays the audio or downloads it through the storage URL. |
| Alternative Flow | A1. The user deletes a generated audio record.<br>A2. The user shares or saves the generated audio locally. |
| Exception Flow | E1. If the audio file is not available, the system shows an error message.<br>E2. If the generation status is still queued or processing, the application shows the current status instead of playing the audio. |
| Post-condition | The user can manage generated audio results from the library. |
| Priority | High |

### 5.2.3. Sequence diagram

#### a. User authentication

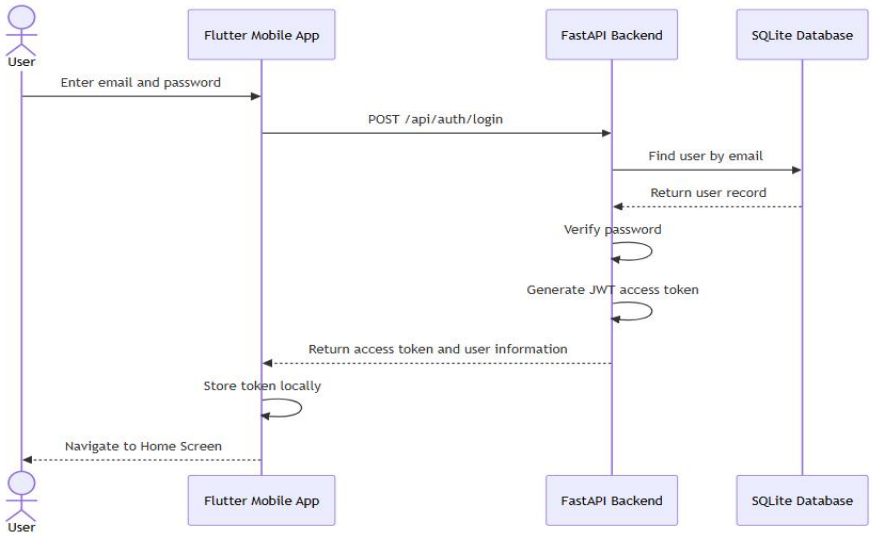



*Figure 5.8. Sequence diagram of User authentication*

Figure 5.8 illustrates the interaction flow during user authentication. The mobile application sends the login request to the FastAPI backend. The backend verifies the user credentials against the database and returns a JWT access token if the login is successful. This token is then used by the mobile application to access protected APIs.

#### b. Create voice profile

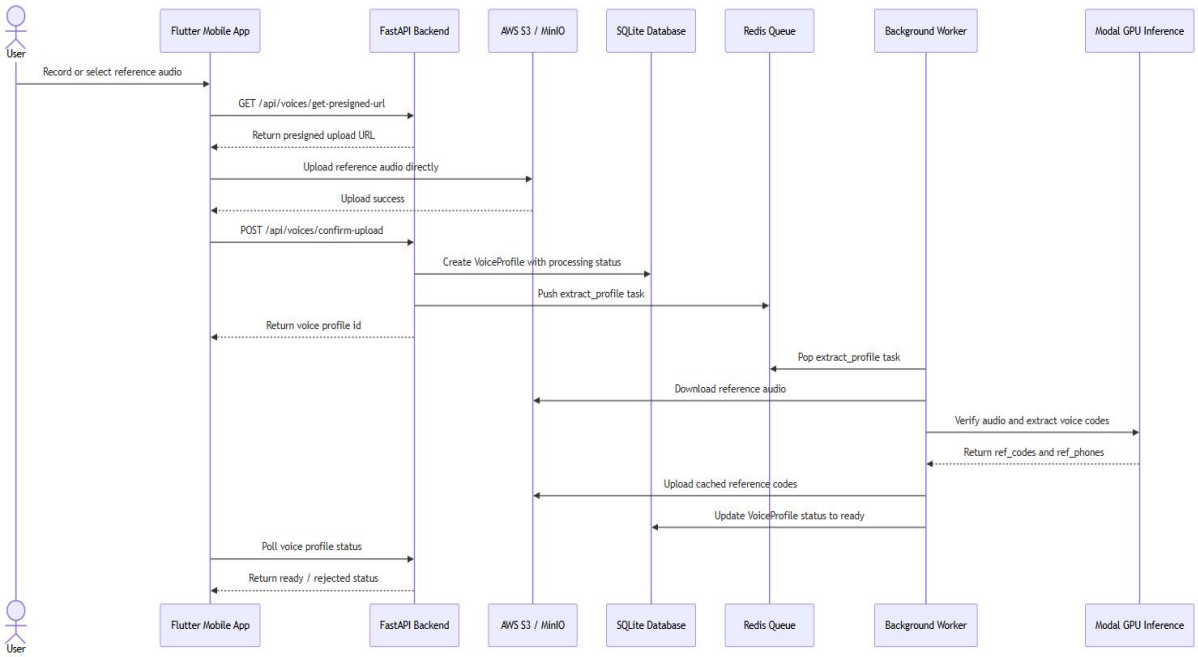



*Figure 5.9. Sequence diagram of creating voice profile*

Figure 5.9 shows the sequence diagram of the voice profile creation process. The workflow starts when the user records or selects a reference audio sample from the mobile application. The application requests a presigned upload URL from the FastAPI backend and then uploads the reference audio directly to object storage such as AWS S3 or MinIO. After the upload is completed, the mobile application confirms the upload with the backend. The backend creates a new voice profile record in the SQLite database with an initial processing status and pushes an extract_profile task to the Redis queue.

The background worker retrieves the task, downloads the reference audio from object storage, and sends it to the Modal GPU inference service. The inference service verifies the reference audio and extracts reusable voice information, including reference audio codes and phoneme-related information. These extracted features are stored as cached reference data, and the corresponding voice profile status is updated in the database. If the process succeeds, the profile status becomes ready; otherwise, it may be marked as rejected or failed. The mobile application periodically polls the backend to display the final status to the user.

**c. Voice cloning request**

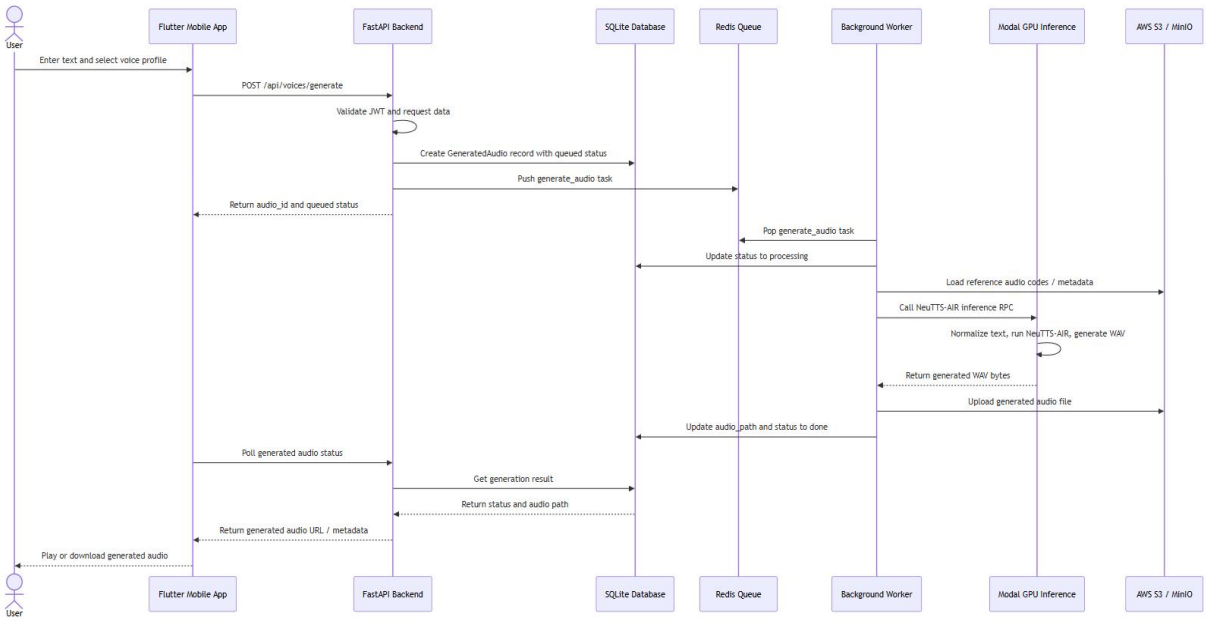



*Figure 5.10. Sequence diagram of voice cloning request*

Figure 5.10 describes the asynchronous voice cloning request. After the user submits text and selects a voice profile, the backend validates the request, creates a generated audio record, and pushes a task to Redis. The background worker calls the Modal GPU inference service to generate speech. The generated WAV file is uploaded

to object storage, and the database is updated so the mobile application can retrieve and display the result.

This sequence diagram also highlights why asynchronous processing is necessary for the voice cloning workflow. Since speech generation requires GPU inference and may take several seconds to complete, the backend immediately returns a queued status to the mobile application instead of blocking the request. The mobile application then polls the backend to check the generation status until the output audio is ready. This design keeps the application responsive while the heavy inference task is handled by the background worker and Modal GPU service.

## 5.3. Application evaluation

In addition to model evaluation, the developed application was tested to verify whether the main user workflows function correctly. The evaluation focuses on authentication, reference audio upload, voice profile creation, speech generation, audio history management, chatbot support, and Pro account upgrade.

### 5.3.1. Functional testing

Functional testing was conducted to verify whether the main workflows of the system operate correctly from the mobile application to the backend, database, object storage, and inference service. The test cases focus on the most important user-facing functions, including registration, login, reference audio upload, voice profile creation, speech generation, audio history management, chatbot support, and Pro account upgrade.

*Table 5.11. Functional testing results*

| Function | Expected Result | Actual Result | Status |
|---|---|---|---|
| Register | User account is created successfully in the database with a hashed password. | Account created and saved successfully. | Pass |
| Login | User authenticates successfully and receives a valid JWT token. | Token returned successfully, user logged in. | Pass |
| Upload reference audio | Audio file is uploaded to Cloud Storage (S3/MinIO) and returns a valid presigned URL. | File uploaded successfully, file key & URL returned. | Pass |

| | | | |
|---|---|---|---|
| Create voice profile | Voice features are extracted (NeuCodec), verified by Whisper, and saved with "ready" status. | Profile created, features extracted and saved in DB. | Pass |
| Generate cloned speech | AI generates audio from text, deducts tokens, and saves the result to the database & storage. | WAV file generated, tokens deducted, file appears in library. | Pass |
| View generated audio history | System retrieves and displays a list of all previously generated audios for the user. | History fetched from DB and displayed correctly on UI. | Pass |
| Play audio | Audio plays smoothly; users can adjust DSP settings (pitch, speed) if applicable. | Audio playback works perfectly without latency. | Pass |
| Download/share audio | Audio file is successfully downloaded to local device storage or shared via OS dialog. | File downloaded and shared successfully. | Pass |
| Chatbot | AI evaluates prompt, generates a TTS-friendly script (or asks to clarify), and saves the session. | Script generated successfully, session saved in DB. | Pass |
| Upgrade Pro | User is redirected to Stripe, completes payment, and account status updates to 'pro'. | Payment processed successfully, account upgraded to Pro. | Pass |

Table 5.11 shows that all tested functions passed successfully. The results confirm that the developed system can support the complete voice cloning workflow, from user authentication and reference audio upload to cloud inference and generated audio playback. In particular, the successful speech generation test verifies that the asynchronous pipeline involving FastAPI, Redis, Modal GPU inference, database update, and object storage works correctly.

### 5.3.2. User interface demonstration

This section presents the main screens of the developed mobile application. The screenshots should demonstrate the practical usability of the system and show how users interact with the voice cloning workflow.

The user interface is organized around the main voice cloning workflow. After logging in, users can access the home screen to enter text, select a voice profile, and submit a speech generation request. The library screen allows users to manage created voice profiles and review previously generated audio files. In addition, the playback interface provides basic audio controls, while the chatbot and Pro upgrade screens support advanced features such as script generation and account enhancement.

The following screenshots demonstrate the main screens of the developed mobile application. These screens are selected to represent the complete user journey, including authentication, text input, voice profile management, generated audio history, audio playback, chatbot support, and account upgrade.

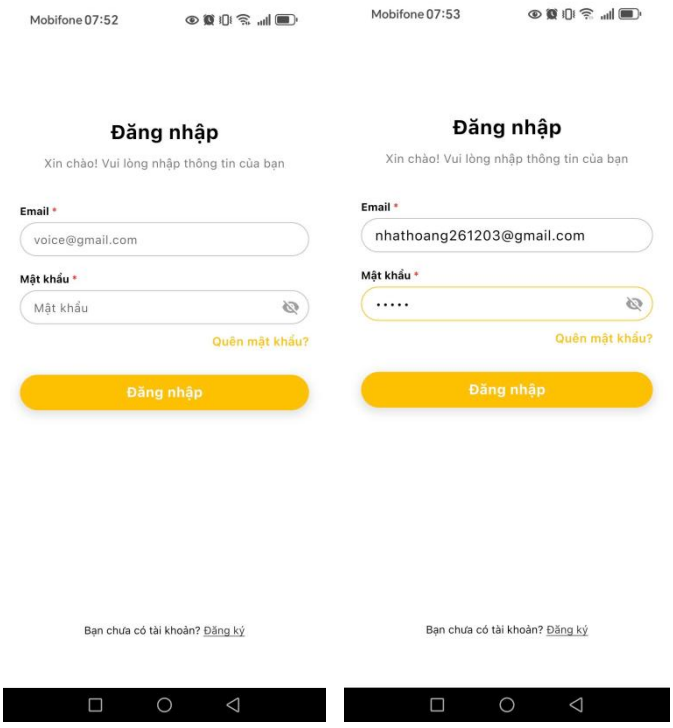



*Figure 5.11. Login screen of the application.*

Figure 5.11 shows the login screen of the application. This screen allows registered users to authenticate using their account credentials. After successful login, the application stores the JWT token locally and uses it for subsequent protected API requests.

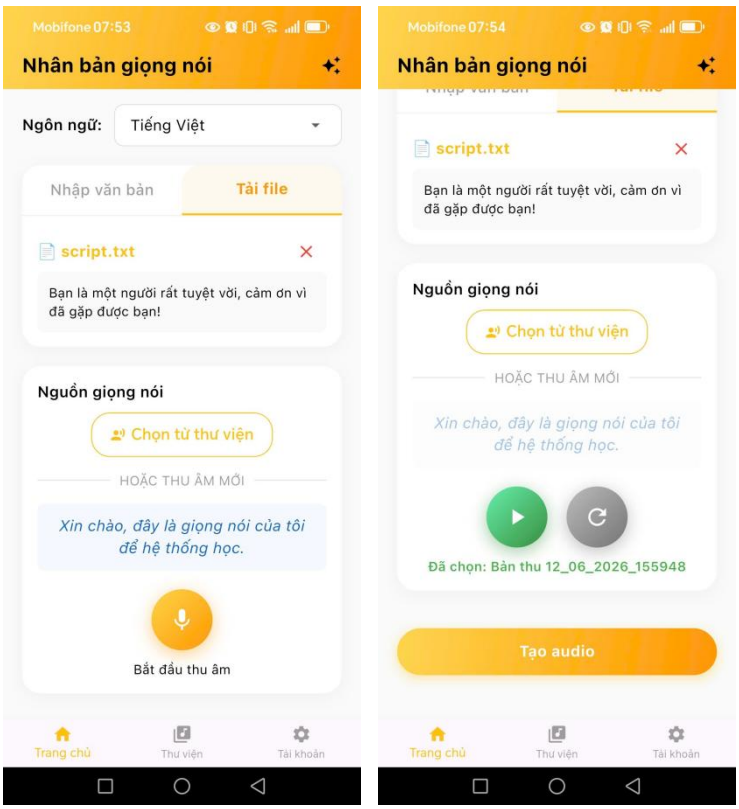



*Figure 5.12. Home screen for text input and voice selection.*

Figure 5.12 presents the home screen, where users enter the text to be synthesized and select an available voice profile. This screen is the main entry point for the voice cloning workflow.

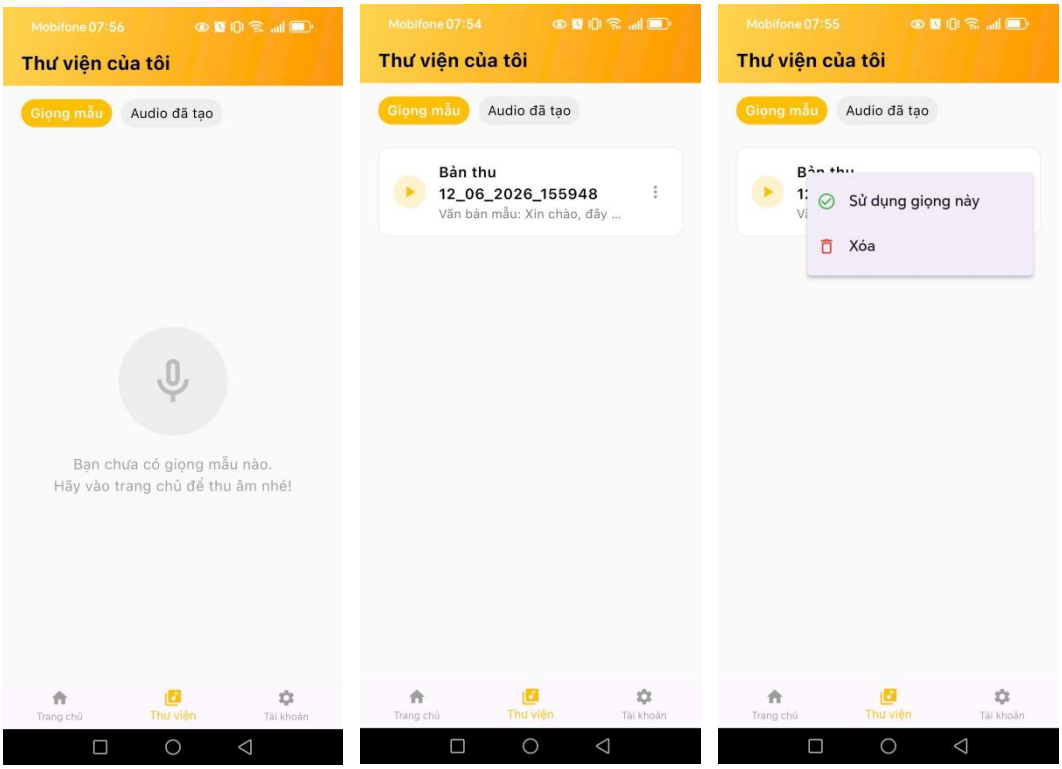



*Figure 5.13. Voice profile management screen.*

Figure 5.13 shows the voice profile management screen. Users can create new voice profiles from reference audio, view existing profiles, and select a profile for speech generation. The status of each profile indicates whether it is ready to use.

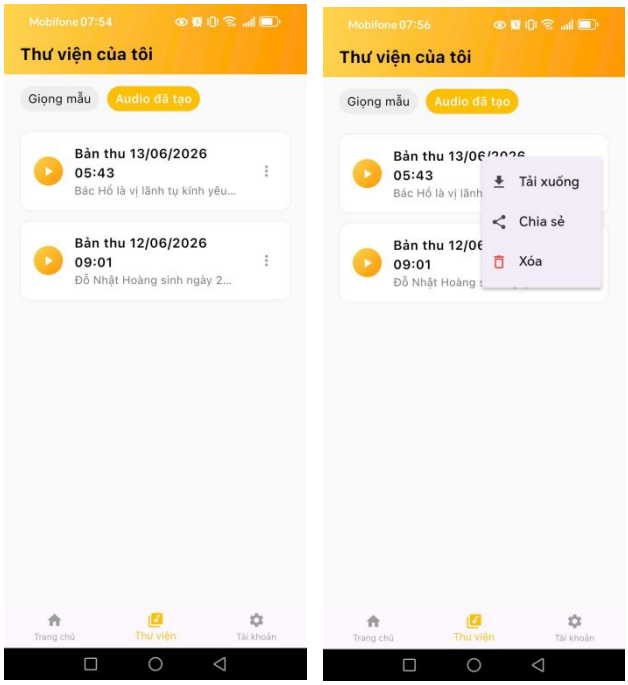



*Figure 5.14. Generated audio history screen.*

Figure 5.14 presents the generated audio history screen. This screen allows users to review previous voice cloning results, check generation status, and access completed audio files.




*Figure 5.15. Audio playback and DSP adjustment screen.*

Figure 5.15 shows the audio playback interface. Users can listen to generated speech and apply simple playback adjustments such as speed or pitch control if supported by the application.

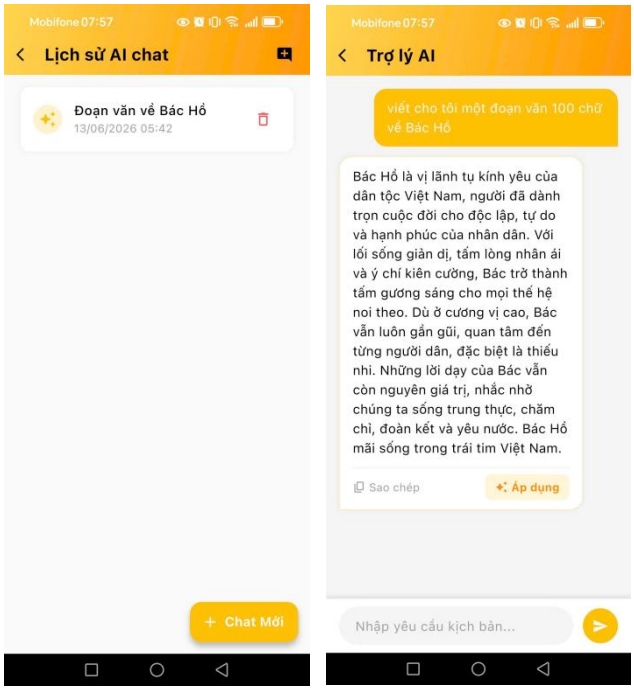



*Figure 5.16. AI chatbot screen for script generation.*

Figure 5.16 illustrates the chatbot screen. This feature assists users in generating or refining TTS-friendly text before submitting it to the voice cloning module.

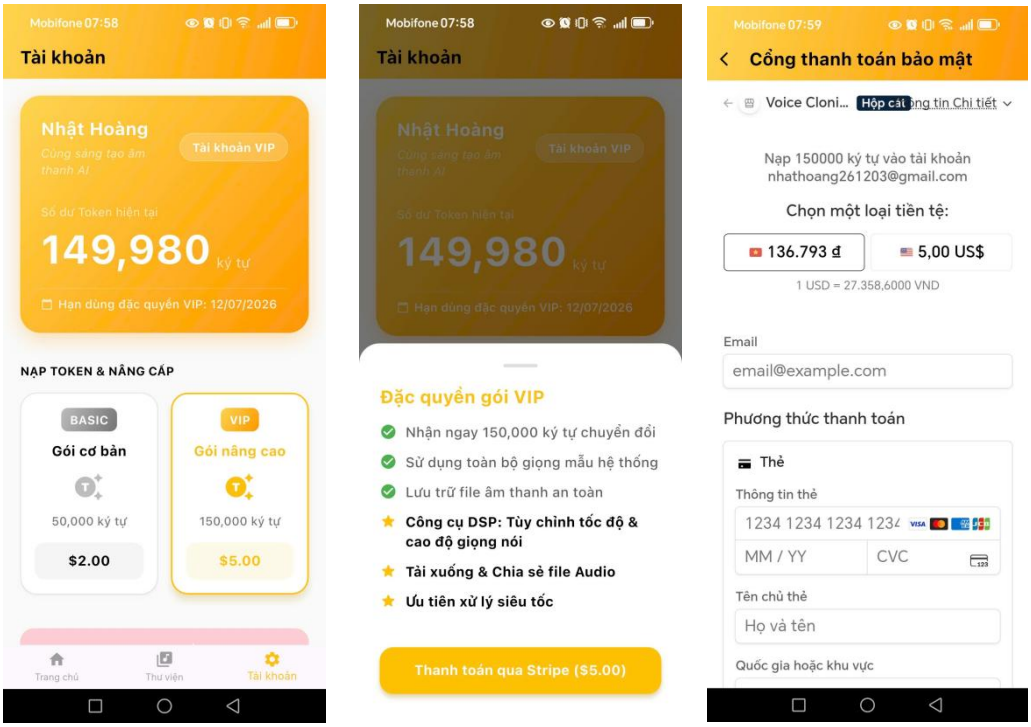



*Figure 5.17. Pro account upgrade screen.*

Figure 5.17 shows the Pro account upgrade screen. This screen connects the user to the payment workflow and allows account status to be upgraded after successful payment.

### 5.3.3. End-to-End Voice Cloning workflow

The end-to-end workflow was tested from the user interface to the cloud inference layer. First, the user logs in and uploads a reference audio sample. The system extracts reusable voice information and creates a voice profile. Next, the user enters text and selects the created voice profile. The backend normalizes the text, creates a generation task, and sends it to the background worker. The worker calls the Modal GPU inference service, which generates the speech using the selected fine-tuned NeuTTS-AIR checkpoint. Finally, the generated audio is stored in cloud storage and displayed in the mobile application.

This confirms that the proposed system is not only a fine-tuned speech model, but also a complete voice cloning application that connects model inference, backend services, cloud storage, and a mobile user interface.

## 5.4. Discussion

The system design demonstrates that the fine-tuned NeuTTS-AIR model can be integrated into a practical end-to-end voice cloning application. By separating the system into mobile, backend, task queue, inference, database, and storage layers, the architecture avoids placing heavy AI workloads directly on the mobile device or backend server. This separation improves maintainability and allows each component to focus on a specific responsibility.

One important design decision is the use of asynchronous processing for voice cloning requests. Since speech generation requires GPU inference and may take several seconds to complete, the backend pushes generation tasks to Redis instead of processing them synchronously. The background worker then invokes the Modal GPU inference service and updates the database when the generated audio is ready. This design keeps the backend responsive and allows the mobile application to monitor the task status through polling.

The use of object storage also improves system scalability. Instead of storing reference audio and generated audio files directly in the database, the system stores them in MinIO-compatible object storage and keeps only metadata and file paths in the database. This reduces database size and makes audio file management more flexible.

The database design supports the main business entities of the application, including users, voice profiles, generated audios, chatbot sessions, and chatbot messages. The relationship between users, voice profiles, and generated audios allows each user to manage multiple reusable voices and generated results. The chatbot tables provide additional support for script generation and conversation history.

However, the current system still has several limitations. First, SQLite is suitable for prototyping but may not be ideal for a large-scale production environment. A future version should migrate to PostgreSQL or another production-ready relational database. Second, inference latency can become a bottleneck when many users generate speech at the same time. Third, the current task processing mechanism can be improved with better retry handling, monitoring, and failure recovery. Finally, the quality of generated speech still depends on the quality of the reference audio and the robustness of the text normalization pipeline.

Overall, Chapter 5 shows that the proposed work is not only a model adaptation experiment but also a complete application system. The developed architecture successfully connects the mobile user interface, backend services, cloud GPU inference, object storage, and database into a functional Vietnamese zero-shot voice cloning application.

# CONCLUSION

## 1. Result

This thesis presented a Vietnamese adaptation framework for NeuTTS-AIR and developed an end-to-end zero-shot voice cloning application. The work focused on two main objectives: adapting a pretrained autoregressive speech generation model to Vietnamese and integrating the adapted model into a practical mobile application.

First, a Vietnamese multi-speaker speech dataset was constructed from multiple sources, including VIVOS, INFORE, viVoice, and PhoAudiobook. The collected data was processed through audio normalization, text preprocessing, grapheme-to-phoneme conversion, and tokenization. A precomputed training dataset was then created to improve training efficiency and reduce CPU bottlenecks during fine-tuning.

Second, the NeuTTS-AIR model was adapted to Vietnamese through a two-stage training strategy. Phase 1 fine-tuned only the lm_head layer to provide an initial adaptation to Vietnamese audio tokens. However, the results showed that this stage alone was not sufficient for stable Vietnamese speech generation. Therefore, Phase 2 further investigated two deeper adaptation strategies: full fine-tuning and Global LoRA.

The experimental results showed that Global LoRA achieved the best overall performance. On the unseen-speaker evaluation set, the selected Global LoRA checkpoint achieved 12.04% WER, 6.24% CER, 0.6138 SECS, and 17.17 MCD. Compared with Phase 1 and full fine-tuning, Global LoRA provided a better balance between speech intelligibility, speaker similarity, and acoustic quality. These results indicate that Global LoRA is an effective adaptation strategy for Vietnamese zero-shot voice cloning while still preserving the pretrained model's speaker generalization ability.

Finally, the fine-tuned model was integrated into a complete application system. The system includes a Flutter mobile application, FastAPI backend, Redis task queue, Modal GPU inference service, SQLite database, and cloud object storage. The application supports user authentication, reference voice upload, voice profile creation, speech generation, generated audio history, playback, chatbot-based script generation, and Pro account upgrade. Functional testing confirmed that the main workflows of the application worked successfully.

## 2. Restriction

Although the proposed system achieved promising results, several limitations remain.

First, the quality of the generated speech still depends strongly on the reference audio. Noisy recordings, unclear pronunciation, long silence, or mismatched reference text can reduce speaker similarity and speech naturalness. Therefore, reference audio verification and preprocessing are still important for maintaining stable output quality.

Second, although Global LoRA significantly improved the model compared with Phase 1 and full fine-tuning, the generated speech is not yet perfect. Some samples may still contain pronunciation errors, unnatural rhythm, or unstable prosody, especially for long or complex Vietnamese sentences. This shows that Vietnamese prosody modeling remains a challenging problem for autoregressive speech generation.

Third, the evaluation mainly relied on objective metrics such as WER, CER, SECS, and MCD. While these metrics are useful for measuring intelligibility, speaker similarity, and acoustic distortion, they cannot fully represent human perception of naturalness and expressiveness. A large-scale subjective listening test was not conducted in this thesis due to time and resource limitations.

Fourth, the current application prototype still depends on cloud GPU inference. Although this design allows the system to run a large speech generation model, inference latency and service cost may become limitations when the number of users increases. In addition, the current database is based on SQLite, which is suitable for prototyping but may not be ideal for a large-scale production environment.

## 3. Development

Several directions can be considered for future development.

First, the Vietnamese speech dataset can be expanded with more speakers, more diverse accents, and more recording conditions. A larger and more balanced dataset would help the model learn Vietnamese pronunciation, rhythm, and speaker characteristics more effectively. Additional data cleaning and speaker balancing strategies could also improve training stability.

Second, the voice cloning model can be further improved by investigating better adaptation strategies. For example, future work can explore different LoRA ranks, layer selection strategies, learning rate schedules, or adapter-based methods. In

addition, improving the neural audio codec or using a more suitable codec for Vietnamese speech may further enhance audio quality.

Third, the text normalization and pronunciation processing pipeline can be improved. Vietnamese contains many numbers, abbreviations, symbols, foreign words, and context-dependent readings. A more robust Vietnamese text normalization module would help reduce pronunciation errors and improve the naturalness of generated speech.

Fourth, the system can be optimized for production deployment. Future versions may replace SQLite with PostgreSQL, improve task monitoring, add retry mechanisms for failed inference jobs, and optimize cloud GPU usage to reduce latency and cost. Caching frequently used voice features and optimizing audio chunking can also improve the end-to-end generation speed.

Finally, the application can be extended with more user-oriented features, such as advanced voice management, better audio editing tools, real-time generation progress, more detailed user feedback, and subjective rating collection. These improvements would make the system more practical and provide additional data for evaluating and improving Vietnamese zero-shot voice cloning in real-world usage.

# REFERENCES

[1] H. Zen, K. Tokuda and A. W. Black, "Statistical Parametric Speech Synthesis," *Speech Communication,* vol. 51, no. 11, pp. 1039-1064, 2009.

[2] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis và Y. Wu, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," trong *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[3] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao và T.-Y. Liu, "FastSpeech: Fast, robust and controllable text to speech," trong *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[4] J. Kim, J. Kong và J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," trong *2021*, International Conference on Machine Learning (ICML).

[5] J. Kong, J. Kim và J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," trong *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[6] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno và Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," trong *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] A. Défossez, J. Copet, G. Synnaeve và Y. Adi, "High fidelity neural audio compression," *Transactions on Machine Learning Research,* 2023.

[8] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, L. He, S. Zhao và F. Wei, "Neural codec language models are zero-shot text to speech synthesizers," 2023.

[9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang và W. Chen, "LoRA: Low-rank adaptation of large language models," trong *International Conference on Learning Representations (ICLR)*, 2022.

[10] R. T. Fielding, "Architectural styles and the design of network-based software

architectures," Irvine, 2000.

[11] M. B. Jones, J. Bradley và N. Sakimura, "JSON Web Token (JWT)," 2015.

# APPENDIX

## 1. Source code

- Github:
  - https://github.com/NhatHoang26122003/voice_clone_backend
  - https://github.com/NhatHoang26122003/voice-clone-app
- Status: Private
- Description: The source code was developed as part of this graduation thesis and is still under active development. Therefore, only the core components necessary for academic evaluation and research purposes are publicly available. Other components are currently withheld and will continue to be improved and extended in future work.

## 2. Dataset

- Github: https://github.com/NhatHoang26122003/voice-cloning-vietnamese-dataset
- Status: Public
- Description: This repository contains the Vietnamese speech dataset prepared for the graduation thesis "Personalized Speech Generation Using Voice Cloning". The dataset is constructed by combining multiple publicly available Vietnamese speech corpora, including VIVOS, INFORE, PhoAudiobook, and viVoice, together with the corresponding preprocessing pipeline and training metadata. It is intended for academic research and educational purposes.