# LexiLingo

AI-Powered English Learning Application


# MÔ TẢ KIẾN TRÚC

Mobile Application Architecture

**Phiên bản:** v1.2.7
**Ngày:** 14/01/2026
**Nền tảng:** Flutter (iOS/Android/Web)
**Trạng thái:** Development

Nguyen Huu Thang - Lead Developer
Email:nhthang312@gmail.com

# Mục lục

# 1 Tổng Quan Dự Án

## 1.1 Giới Thiệu

**LexiLingo** là ứng dụng học tiếng Anh thông minh sử dụng AI để hỗ trợ người học cải thiện kỹ năng ngôn ngữ một cách hiệu quả. Ứng dụng tập trung vào việc cung cấp trải nghiệm học tập cá nhân hóa thông qua:

- Hội thoại với AI Tutor
- Học từ vựng theo ngữ cảnh
- Kiểm tra ngữ pháp tự động
- Luyện phát âm với phản hồi thời gian thực
- Theo dõi tiến độ học tập

## 1.2 Công Nghệ Sử Dụng

| **Layer** | **Công Nghệ** | **Mục Đích** |
|---|---|---|
| Frontend | Flutter 3.29+ | Cross-platform UI |
| State Management | Provider | Quản lý trạng thái |
| Database | SQLite + Firestore | Local & Cloud storage |
| AI/ML | Qwen2.5 + Whisper + HuBERT | Language processing |
| Authentication | Firebase Auth | Xác thực người dùng |

Bảng 1: Stack công nghệ chính

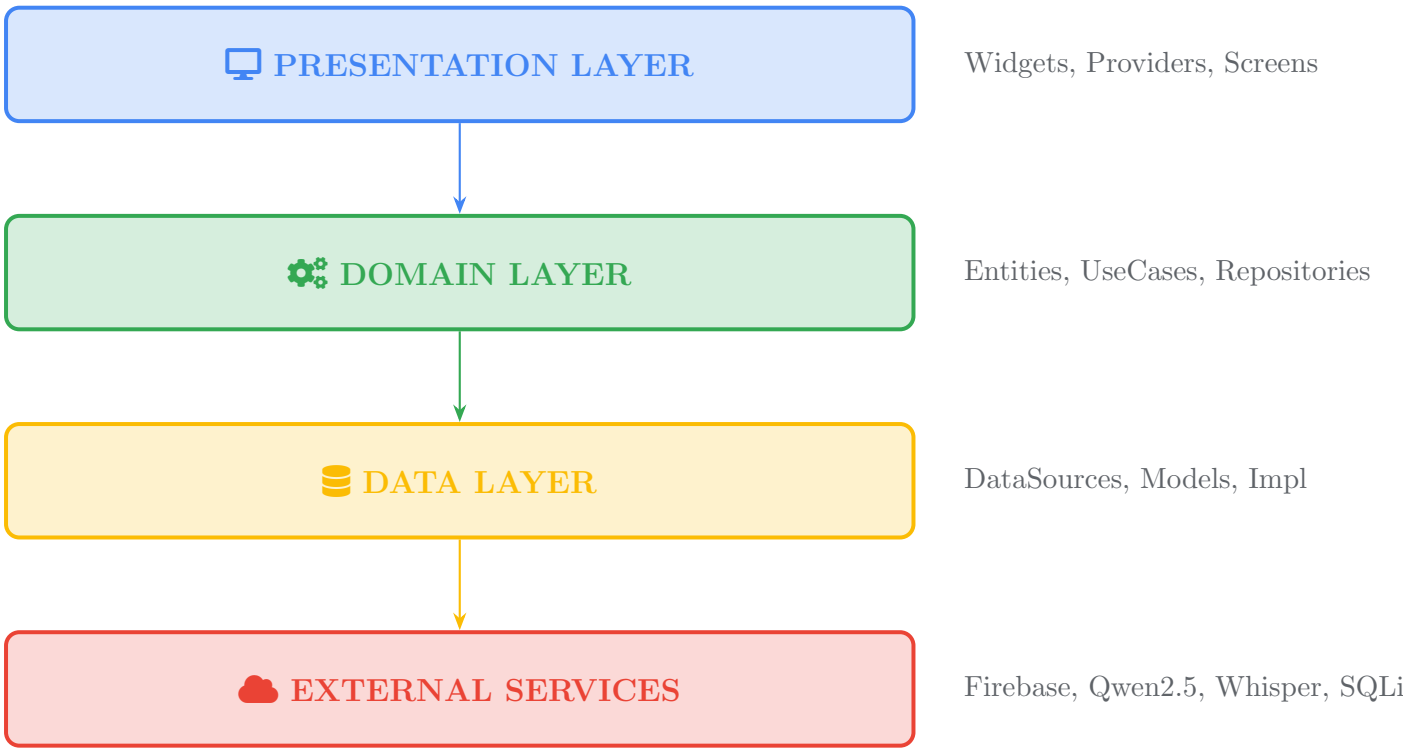
# 2 Kiến Trúc Tổng Quan

## 2.1 Clean Architecture

LexiLingo được xây dựng theo mô hình **Clean Architecture** kết hợp với **Feature-First Structure**, đảm bảo:

- **Separation of Concerns**: Tách biệt rõ ràng các tầng logic
- **Testability**: Dễ dàng viết unit test
- **Maintainability**: Bảo trì và mở rộng thuận tiện
- **Scalability**: Có thể scale khi cần thiết

## 2.2 Sơ Đồ Kiến Trúc Tổng Quan



# 3 Chi Tiết Kiến Trúc Các Layer

## 3.1 Presentation Layer

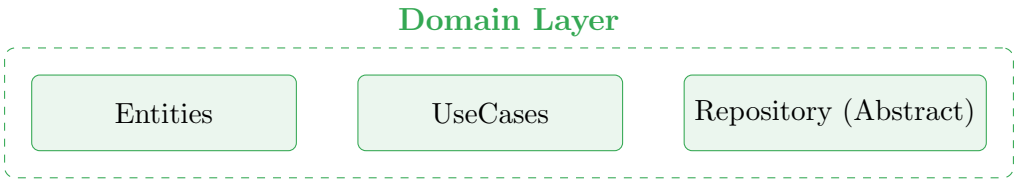
Tầng này chịu trách nhiệm về giao diện người dùng và quản lý trạng thái UI.


**Thành phần chính:**

- **Screens**: Các màn hình chính (ChatScreen, HomeScreen, CourseScreen...)
- **Widgets**: UI components tái sử dụng (MessageBubble, CourseCard...)
- **Providers**: State management với ChangeNotifier pattern

## 3.2 Domain Layer

Tầng nghiệp vụ chứa business logic thuần túy, không phụ thuộc vào framework.

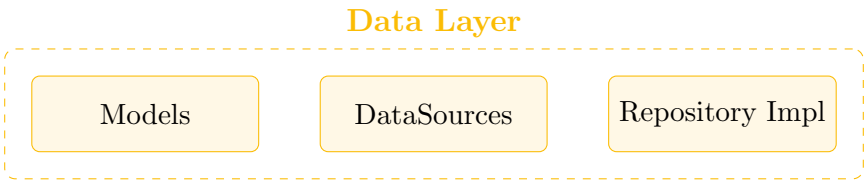



**Thành phần chính:**

- **Entities**: Business objects (ChatMessage, User, Course, Vocabulary)
- **UseCases**: Các use case cụ thể (SendMessageUseCase, GetCoursesUseCase)
- **Repository**: Interface định nghĩa contract cho data layer

## 3.3 Data Layer

Tầng dữ liệu xử lý việc lấy và lưu trữ dữ liệu từ các nguồn khác nhau.




**Thành phần chính:**

- **Models**: Data Transfer Objects với JSON serialization
- **DataSources**: Local (SQLite) và Remote (API, Firebase)
- **Repository Impl**: Triển khai cụ thể của Repository interface

# 4 Cấu Trúc Module (Feature-First)

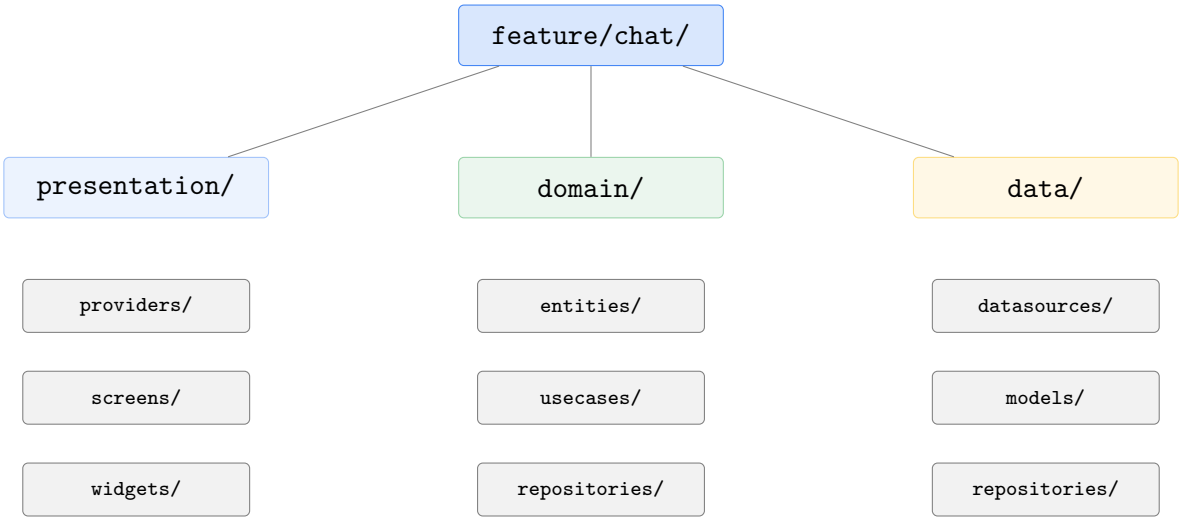
## 4.1 Tổ Chức Thư Mục

Project Structure

```
lib/
 core/                    # Shared utilities
    di/                  # Dependency Injection
    services/            # Core services
    theme/               # App theming
    utils/               # Utilities

 features/                # Feature modules
    auth/                # Authentication
    chat/                # AI Chat (NEW)
    course/              # Courses
    home/                # Home dashboard
    user/                # User profile
    vocabulary/          # Vocabulary

 main.dart                # Entry point
```

## 4.2 Cấu Trúc Một Feature Module

Mỗi feature module được tổ chức theo 3 layer:



# 5 Module AI Chat

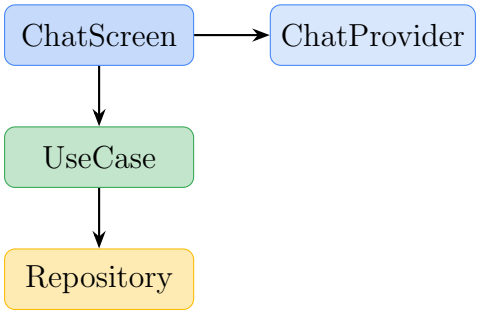
## 5.1 Tổng Quan Module

Module **Chat** là tính năng cốt lõi cho phép người dùng tương tác với AI Tutor để học tiếng Anh.

**Tính năng chính:**

- Hội thoại với AI tutor bằng văn bản
- Kiểm tra và sửa lỗi ngữ pháp tự động
- Giải thích từ vựng theo ngữ cảnh
- Đánh giá độ trôi chảy (fluency scoring)
- Lưu trữ lịch sử hội thoại

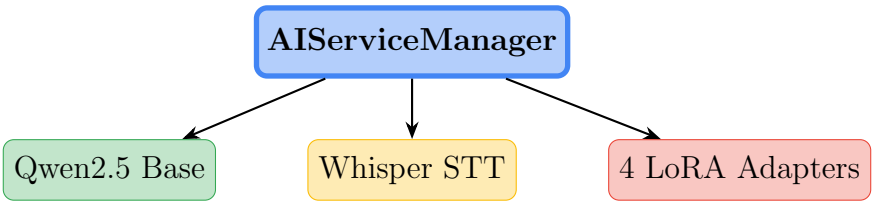
## 5.2 Kiến Trúc AI Chat Module




**Thành phần:** ChatScreen (UI) → ChatProvider (State) → UseCase (Logic) → Repository (Data)

## 5.3 AI Service Integration

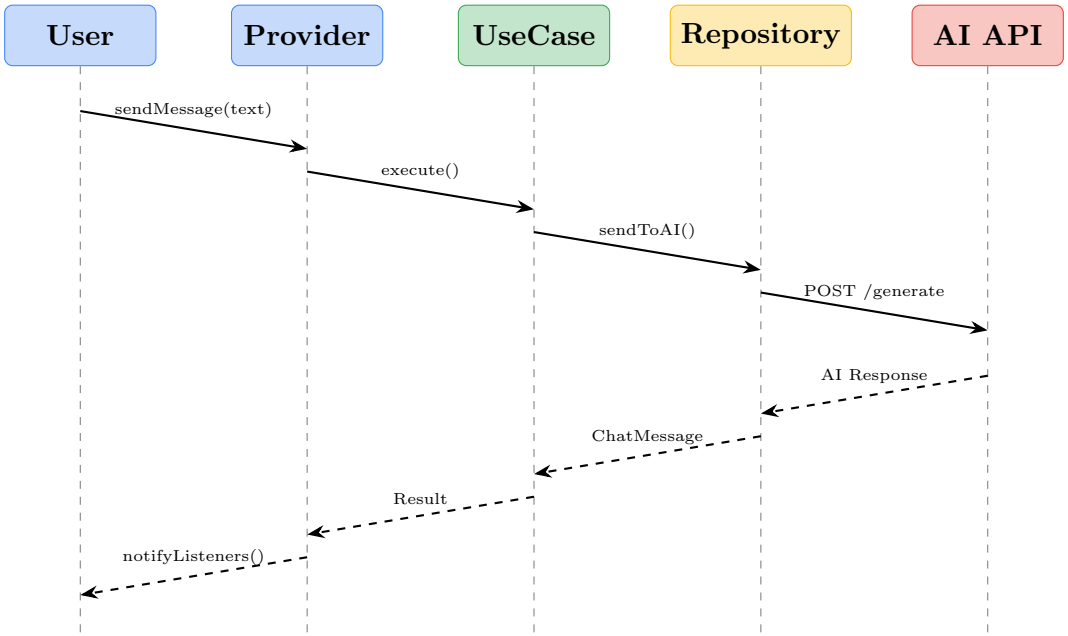
Module Chat tích hợp với 3 AI services chính:



# 6 Luồng Dữ Liệu (Data Flow)

## 6.1 Gửi Tin Nhắn - Send Message Flow

## 6.2 Chi Tiết Sequence Diagram



# 7 Dependency Injection

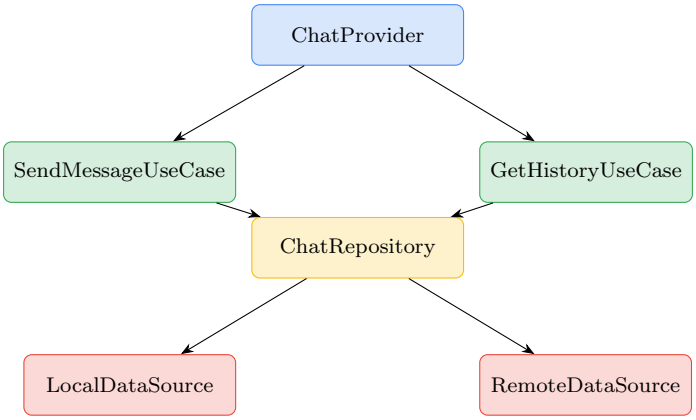
## 7.1 GetIt Service Locator

Dự án sử dụng **GetIt** làm Service Locator để quản lý dependencies:

injection_container.dart

```
final sl = GetIt.instance;

Future<void> initializeDependencies() async {
  // Core Services
  sl.registerLazySingleton<FirestoreService>(
    () => FirestoreService.instance
  );

  // DataSources
  sl.registerLazySingleton<ChatRemoteDataSource>(
    () => ChatRemoteDataSource(modelPath: qwenModelPath)
  );

  // Repositories
  sl.registerLazySingleton<ChatRepository>(
    () => ChatRepositoryImpl(remote: sl(), local: sl())
  );

  // UseCases
  sl.registerLazySingleton(
    () => SendMessageUseCase(sl())
  );

  // Providers
  sl.registerFactory(
    () => ChatProvider(sendMessage: sl(), getHistory: sl())
  );
}
```

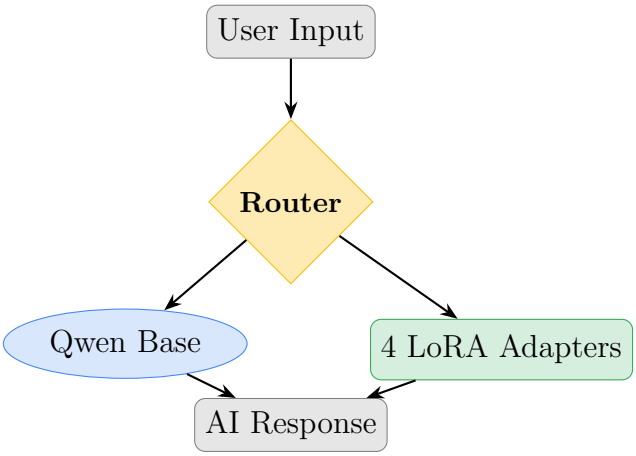
## 7.2 Dependency Graph



# 8 Tích Hợp AI Models

## 8.1 Kiến Trúc AI Hybrid

LexiLingo sử dụng kiến trúc Hybrid AI kết hợp Cloud và On-device models:



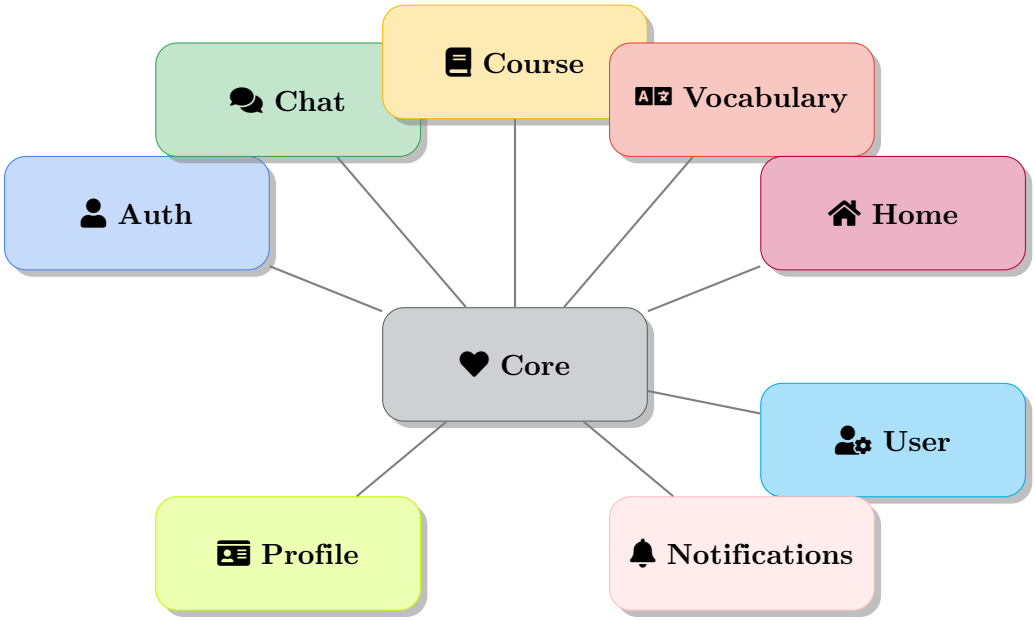

## 8.2 LoRA Adapters (Fine-tuned Models)

Hệ thống sử dụng 4 LoRA adapters được fine-tune từ Qwen2.5-1.5B:

| **Adapter** | **Task** | **Size** | **Loss** |
|---|---|---|---|
| Grammar | Sửa lỗi ngữ pháp | 151 MB | 0.77 |
| Vocabulary | Giải thích từ vựng | 151 MB | 0.71 |
| Dialogue | Hội thoại | 151 MB | 1.89 |
| Fluency | Đánh giá độ trôi chảy | 151 MB | 1.78 |

Bảng 2: LoRA Adapters Configuration

# 9 Tổng Quan Các Feature Modules




## 9.1 Mô Tả Các Module

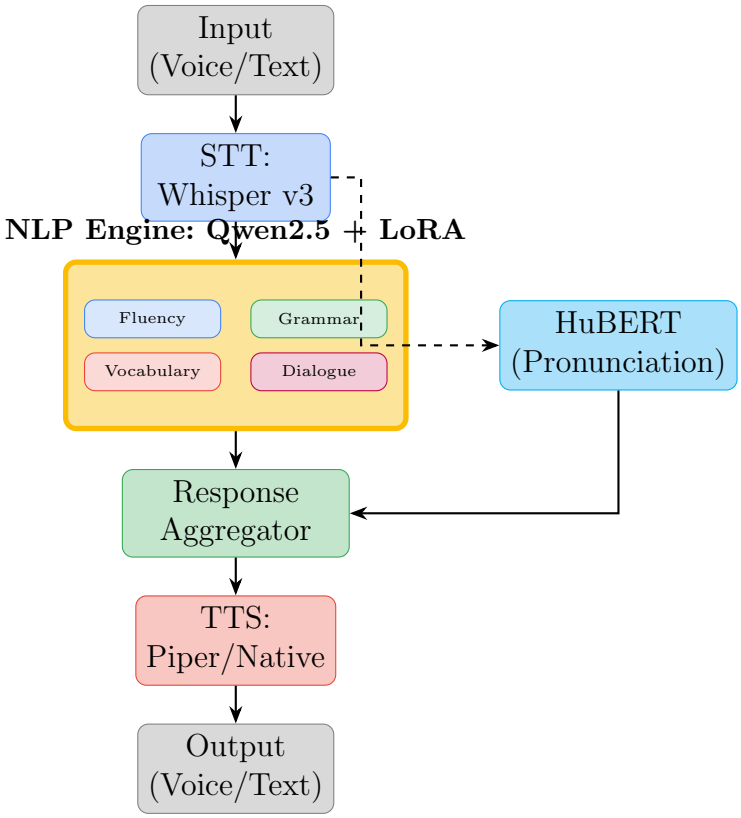
| Module | Mô Tả | Trạng Thái |
|---|---|---|
| Auth | Xác thực với Google Sign-In, Firebase Auth | Done |
| Chat | AI Tutor conversation, grammar check | Done |
| Course | Quản lý khóa học, bài học | Done |
| Vocabulary | Từ vựng, flashcards, spaced repetition | Done |
| Home | Dashboard, quick stats, recent activity | Done |
| User | Profile, settings, preferences | Done |
| Notifications | Push notifications, reminders | In Progress |
| Profile | User stats, achievements | Done |

Bảng 3: Feature Modules Status

# 10 Kiến Trúc Deep Learning (DL)

## 10.1 Tổng Quan Pipeline Xử Lý AI

Khi người dùng tương tác với LexiLingo, dữ liệu đi qua một **pipeline xử lý đa tầng** như sau:




## 10.2 Tầng 1: Speech-to-Text (STT) - Nhận Dạng Giọng Nói

### 10.2.1 Mục đích và Vị trí

Module STT là **điểm vào đầu tiên** khi người dùng nói tiếng Anh. Nó chuyển đổi audio thành text để các module NLP có thể xử lý.

### 10.2.2 Model sử dụng: OpenAI Whisper v3

**Tại sao chọn Whisper?**

- Pre-trained trên **680,000 giờ** dữ liệu đa ngôn ngữ
- Hỗ trợ tốt cho **non-native speakers** (Vietnamese accent)
- Cung cấp **word-level timestamps** cho phân tích phát âm
- Open-source, có thể chạy offline hoàn toàn

| Variant | Params | WER | Latency | Use Case |
|---|---|---|---|---|
| Whisper Large v3 | 1.5B | 3-5% | 200-300ms | Development |
| Whisper Medium | 769M | 5-7% | 100-200ms | High-end mobile |
| Whisper Small | 244M | 8-10% | 50-100ms | **Production mobile** |

Bảng 4: Whisper Model Variants

### 10.2.3 Quy trình xử lý chi tiết

STT Processing Pipeline

```
Bước 1: Audio Preprocessing
 Resample → 16kHz mono (Whisper requirement)
 Normalize → Peak -3dB (consistent volume)
 VAD (Silero) → Loại bỏ silence segments
 Chunking → 30s segments với 1s overlap

Bước 2: Whisper Inference
 Encoder: Mel spectrogram → 1500 frames
 Decoder: Auto-regressive text generation
 Beam search: width=5 for accuracy
 Output: Text + confidence per word

Bước 3: Post-processing
 Normalize text (lowercase, punctuation)
 Filter low-confidence words (< 0.6)
 Extract timestamps for pronunciation

Output Example:
{
  "text": "I like learning English",
  "confidence": 0.94,
  "words": [
    {"word": "I", "start": 0.0, "end": 0.2, "conf": 0.98},
    {"word": "like", "start": 0.3, "end": 0.6, "conf": 0.95}
  ]
}
```

## 10.3 Tầng 2: NLP Processing Engine - Xử Lý Ngôn Ngữ

Đây là **trái tim của hệ thống**, sử dụng kiến trúc **Unified Multi-Task** với 1 base model + 4 LoRA adapters.

### 10.3.1 Tại sao chọn kiến trúc này?

**So sánh: 4 models riêng vs 1 base + 4 adapters**
**4 Models riêng lẻ:**

- RAM: 4 × 2GB = 8GB
- Storage: 4 × 900MB = 3.6GB
- Switching time: 2-5 giây (reload model)

**1 Base + 4 LoRA Adapters (Our Choice):**

- RAM: 2GB (base) + 100MB (adapters) = **2.1GB**
- Storage: 900MB + 4×25MB = **1GB**
- Switching time: $<$ **1ms** (chỉ swap adapter weights)

⇒ **Tiết kiệm 72% RAM, 72% storage**

### 10.3.2 Base Model: Qwen2.5-1.5B-Instruct

**Tại sao chọn Qwen2.5?**

- **Instruction-tuned**: Hiểu trực tiếp các câu lệnh như "Check grammar", "Rate fluency"
- **Multilingual**: Pre-trained trên 18T tokens, mạnh về English
- **Efficient**: 1.5B params nhưng hiệu năng ngang GPT-3.5 trên nhiều tasks
- **Open-source**: Apache 2.0 license, không phụ thuộc API

| Specification | Development (1.5B) | Production (0.5B) |
|---|---|---|
| Architecture | Decoder-only Transformer | Same |
| Layers | 28 | 24 |
| Hidden Size | 1536 | 896 |
| Attention Heads | 12 | 14 |
| Context Window | 32,768 tokens | 32,768 tokens |
| Vocab Size | 151,936 (BPE) | 151,936 |
| Model Size (Q4) | 900MB | 300MB |
| Inference Time | 100ms/sentence | 50ms/sentence |

Bảng 5: Qwen2.5 Model Architecture

### 10.3.3 LoRA (Low-Rank Adaptation) - Kỹ thuật Fine-tuning

**LoRA hoạt động như thế nào?**

Thay vì fine-tune toàn bộ 1.5B parameters, LoRA chỉ train **low-rank matrices** được inject vào attention layers:

$$W_{new} = W_{base} + \Delta W = W_{base} + BA$$

Trong đó: $B \in R^{d \times r}$, $A \in R^{r \times k}$, với $r \ll \min(d, k)$

| **Parameter** | **Value** | **Giải thích** |
|---|---|---|
| Rank (r) | 32 | Số chiều của low-rank decomposition |
| Alpha ($\alpha$) | 64 | Scaling factor: $\alpha/r = 2$ |
| Target Modules | 7 | q, k, v, o, gate, up, down_proj |
| Trainable Params | 25M | Chỉ **1.7%** của 1.5B base |
| Dropout | 0.05 | Regularization |

Bảng 6: LoRA Configuration

## 10.4 Chi Tiết 4 LoRA Adapters

### 10.4.1 Adapter 1: Fluency Scoring - Đánh Giá Độ Trôi Chảy

**Vị trí trong pipeline:** Được gọi **đầu tiên** sau khi nhận text từ STT để đánh giá tổng quan chất lượng câu nói.

**Nhiệm vụ:** Cho điểm từ 0.0 đến 1.0 dựa trên:

- Độ tự nhiên của câu (naturalness)
- Độ phức tạp ngữ pháp (grammatical complexity)
- Sự mạch lạc (coherence)
- Phù hợp ngữ cảnh (contextual appropriateness)

Fluency Adapter - Quy Trình

```
INPUT FORMAT (Instruction-tuning):
<|im_start|>user
Rate the fluency of this English sentence from 0.0 to 1.0:
"Yesterday I go to library for study English"
Provide score and brief reasoning.
<|im_end|>
<|im_start|>assistant

MODEL PROCESSING:
1. Tokenize input → 45 tokens
2. Load fluency adapter weights (~25MB)
3. Forward pass through Qwen2.5 + LoRA
4. Generate score + reasoning

OUTPUT:
{
  "fluency_score": 0.45,
  "reasoning": "Multiple grammar errors: verb tense
               (go→went), missing article (the library),
               preposition (for→to). Sentence is
               understandable but not fluent.",
  "confidence": 0.88
}

TRAINING DATA: 1,500 samples from EFCAMDAT corpus
LOSS: MSE(predicted_score, human_score)
METRICS: MAE=0.12, Pearson r=0.91
```

### 10.4.2 Adapter 2: Grammar Correction - Sửa Lỗi Ngữ Pháp

**Vị trí trong pipeline:** Chạy **song song** với Fluency, sử dụng **2-tier architecture**:

1. **Tier 1 - ERRANT**: Rule-based detection (nhanh, <5ms)

2. **Tier 2 - Qwen + LoRA**: Deep correction + explanation

**Tại sao cần 2 tiers?**

- ERRANT phát hiện lỗi nhanh, chính xác cho lỗi đơn giản
- LLM xử lý lỗi phức tạp, cung cấp giải thích chi tiết
- Kết hợp: Nhanh + Chính xác + Có giải thích

Grammar Adapter - Two-Tier Pipeline

```
TIER 1: ERRANT (Rule-based, <5ms)
Input: "Yesterday I go to library"
Output: [
  {type: "VERB:TENSE", orig: "go", corr: "went", pos: 2},
  {type: "DET", orig: "", corr: "the", pos: 4}
]

TIER 2: Qwen + LoRA (Deep correction, ~150ms)
Input: Original text + ERRANT hints
Output:
{
  "original": "Yesterday I go to library",
  "corrected": "Yesterday I went to the library",
  "errors": [
    {
      "type": "VERB_TENSE",
      "original": "go",
      "corrected": "went",
      "explanation": "Use past tense 'went' with time
                      marker 'yesterday'. Simple past
                      indicates completed action."
    },
    {
      "type": "MISSING_ARTICLE",
      "original": "library",
      "corrected": "the library",
      "explanation": "Use definite article 'the' for
                      specific places known to both
                      speaker and listener."
    }
  ]
}

TRAINING DATA: 9,200 samples (BEA-2019, CoNLL-2014, FCE)
LOSS: CrossEntropy + Confidence score
METRICS: F0.5=68, Precision=78%, Recall=68%
```

### 10.4.3 Adapter 3: Vocabulary Classification - Phân Loại Từ Vựng

**Vị trí trong pipeline:** Chạy **song song** để xác định trình độ CEFR của người học, giúp Dialogue adapter điều chỉnh response phù hợp.

**CEFR Levels được hỗ trợ:**

- **A2 (Elementary)**: Từ vựng cơ bản - go, like, friend, school
- **B1 (Intermediate)**: Từ vựng trung cấp - discuss, environment, improve
- **B2 (Upper-Intermediate)**: Từ vựng nâng cao - comprehensive, demonstrate

Vocabulary Adapter - Classification

```
INPUT:
"I want to discuss the environmental problems
 that affect our community"

PROCESSING:
1. Word-level CEFR lookup (Trie-based, O(n)):
   - discuss → B1
   - environmental → B2
   - problems → A2
   - affect → B1
   - community → B1

2. LLM refinement (context-aware):
   - Sentence structure complexity → B1-B2
   - Topic sophistication → B1

OUTPUT:
{
  "overall_level": "B1",
  "distribution": {"A2": 0.15, "B1": 0.55, "B2": 0.30},
  "key_words": {
    "B2": ["environmental"],
    "B1": ["discuss", "affect", "community"],
    "A2": ["want", "problems"]
  },
  "recommendation": "User demonstrates solid B1 level
                     with some B2 vocabulary. Ready for
                     more complex topics."
}

TRAINING DATA: 2,500 samples (Oxford Graded Readers)
LOSS: CrossEntropy with class weights [0.9, 1.0, 1.1]
METRICS: Accuracy=90%, Macro F1=0.89
```

#### 10.4.4 Adapter 4: Dialogue Response - Sinh Phản Hồi AI Tutor

**Vị trí trong pipeline:** Chạy **cuối cùng**, tổng hợp kết quả từ 3 adapters trước để sinh response phù hợp.

**Tại sao cần AutoTutor Dialogue Corpus?**

- **Socratic Tutoring**: Dạy qua câu hỏi gợi mở thay vì đưa đáp án trực tiếp
- **Scaffolding**: Hỗ trợ từng bước, điều chỉnh theo trình độ học sinh
- **Pedagogical Strategies**: Khuyến khích, phản hồi xây dựng, duy trì động lực
- **Conversational Coherence**: Duy trì hội thoại tự nhiên, không máy móc

**Context Variables được sử dụng:**

- Fluency score từ Adapter 1
- Grammar errors từ Adapter 2
- CEFR level từ Adapter 3
- Conversation history (3 turns gần nhất)

Dialogue Adapter - Response Generation

```
CONTEXT INPUT:
{
  "user_input": "Yesterday I go to library",
  "fluency_score": 0.45,
  "cefr_level": "A2",
  "errors": ["VERB_TENSE", "MISSING_ARTICLE"],
  "history": ["Hi, how are you?", "I'm fine, thanks!"]
}

PERSONA: Friendly, encouraging English tutor
TONE ADAPTATION:
- A2: Simple words, short sentences, more encouragement
- B1: Natural conversation, moderate complexity
- B2: Near-native interaction, subtle corrections

GENERATION CONFIG:
- Temperature: 0.7 (balanced creativity)
- Top-p: 0.9 (nucleus sampling)
- Max tokens: 100
- Stop tokens: ["<|im_end|>"]

OUTPUT:
{
  "response": "Good effort!  You should say 'Yesterday
              I went to the library.' Remember: use
              'went' for past actions, and 'the' before
              'library'. What did you do there?",
  "corrections_included": true,
  "follow_up_question": true,
  "tone": "encouraging",
  "adapted_for_level": "A2"
}

TRAINING DATA: 5,200 samples
 AutoTutor Dialogue Corpus: 1,800 samples
   (Tutorial dialogues, Socratic questioning)
 Intel/orca-dpo-pairs: 1,500 samples
   (High-quality instruction following)
 Custom tutoring conversations: 1,900 samples
   (English learning scenarios)
 Total: 5,200 tutoring-style interactions

LOSS: CrossEntropy on response tokens
METRICS: Quality=96%, Appropriateness=94%
Tutoring Style Score: 89% (human evaluation)
```

## 10.5 Tầng 3: Pronunciation Analysis - Phân Tích Phát Âm

### 10.5.1 Model: HuBERT-large (facebook/hubert-large-ls960)

**Vị trí:** Chạy **song song** với NLP Engine, nhận audio trực tiếp từ STT.

**HuBERT (Hidden-Unit BERT) hoạt động như thế nào?**

1. **Self-supervised pre-training**: Học representations từ 960h LibriSpeech
2. **Phoneme recognition**: CTC decoding để nhận dạng 44 ARPAbet phonemes
3. **Forced alignment**: So sánh với native speaker reference

Pronunciation Analysis Pipeline

```
STEP 1: Phoneme Recognition (HuBERT + CTC)
Audio: "think" → // // // /k/
User pronounced: /s/ // // /k/

STEP 2: Forced Alignment (DTW Algorithm)
Compare user phonemes vs native reference:
| Position | Expected | User    | Status      |
|----------|----------|---------|-------------|
| 1        | //       | /s/     | SUBSTITUTION|
| 2        | //       | //      | CORRECT     |
| 3        | //       | //      | CORRECT     |
| 4        | /k/      | /k/     | CORRECT     |

STEP 3: Error Analysis
{
  "phoneme_accuracy": 0.75,
  "errors": [
    {
      "type": "SUBSTITUTION",
      "expected": "// (voiceless dental fricative)",
      "actual": "/s/ (voiceless alveolar fricative)",
      "word": "think",
      "tip": "Place tongue between teeth for 'th' sound"
    }
  ],
  "prosody": {
    "stress_pattern": "correct",
    "intonation": "slightly flat"
  }
}

MODEL: facebook/hubert-large-ls960 (960M params)
LATENCY: <500ms per utterance
```

## 10.6 Tầng 4: Text-to-Speech (TTS) - Chuyển Văn Bản Thành Giọng Nói

**Hệ thống sử dụng 3-tier TTS** để cân bằng chất lượng và tốc độ:

| **Tier** | **Model** | **Size** | **Latency** | **Use Case** |
|---|---|---|---|---|
| 1 | Native OS TTS | 0MB | <100ms | Quick feedback |
| 2 | Piper TTS (VITS) | 30-60MB | 100-300ms | Pronunciation demos |
| 3 | Cloud TTS | 0MB | 300-800ms | Premium quality |

Bảng 7: TTS Tier System

## 10.7 Training Pipeline Chi Tiết

LoRA Training Pipeline

```
PHASE 1: Data Preparation
 Collect domain-specific data:
   • Fluency: 1,500 (EFCAMDAT)
   • Grammar: 9,200 (BEA-2019, CoNLL-2014, FCE)
   • Vocabulary: 2,500 (Oxford Graded Readers)
   • Dialogue: 5,200 (AutoTutor, Intel/orca, Custom)
 Format: Instruction-tuning template (<|im_start|>...)
 Split: 70% train / 15% val / 15% test
 Augmentation: Back-translation, paraphrase

PHASE 2: Environment Setup
 Hardware: Mac M1/M2 32GB or NVIDIA GPU
 Framework: PyTorch + Transformers + PEFT
 Precision: BFloat16 (faster on Apple Silicon)
 Batch: 8 (gradient accumulation 4 → effective 32)

PHASE 3: Training
 Optimizer: AdamW (lr=2e-4, weight_decay=0.01)
 Scheduler: Cosine with 3% warmup
 Epochs: 5-7
 Gradient clipping: 1.0
 Early stopping: patience=3

PHASE 4: Evaluation
 Fluency: MAE, Pearson correlation
 Grammar: F0.5, Precision, Recall
 Vocabulary: Accuracy, Macro F1
 Dialogue: Human evaluation (quality, appropriateness)

PHASE 5: Production Optimization
 Quantization: INT8 / INT4
 Knowledge distillation: 1.5B → 0.5B
 Export: ONNX / TensorRT for mobile
```

## 10.8 So Sánh Development vs Production

| Component | Development | Production |
|---|---|---|
| **Speech-to-Text** | | |
| Model | Whisper Large v3 | Whisper Small |
| Size | 1.5GB | 500MB |
| WER | 3-5% | 8-10% |
| RAM | 4GB | 1.5GB |
| **NLP Engine (Unified)** | | |
| Base Model | Qwen2.5-1.5B | Qwen2.5-0.5B |
| LoRA Adapters | 4 × 25MB | 4 × 8MB |
| RAM | 2GB | 600MB |
| Quality | 96% | 91% |
| **Pronunciation** | | |
| Model | HuBERT-large | Server-side |
| Params | 960M | API call |
| **TTS** | | |
| Model | Native + Piper | Native only |
| Size | 0-60MB | 0MB |
| **TOTAL** | | |
| RAM | 6-7GB | 2.4GB |
| Storage | 3GB | 1GB |
| Latency | ~600ms | ~400ms |

Bảng 8: Development vs Production Comparison

# 11 Kết Luận

## 11.1 Tóm Tắt Kiến Trúc

**Điểm Nổi Bật**

- **Clean Architecture**: Tách biệt rõ ràng Presentation - Domain - Data
- **Feature-First**: Tổ chức code theo feature, dễ maintain
- **Dependency Injection**: Sử dụng GetIt cho loose coupling
- **Hybrid AI**: Kết hợp Cloud + On-device models
- **Cross-Platform**: Flutter cho iOS, Android, Web

## 11.2 Roadmap Phát Triển

1. **Phase 1 (Current)**: Core features - Chat, Course, Vocabulary
2. **Phase 2**: Voice interaction - STT/TTS integration
3. **Phase 3**: Advanced AI - On-device inference với LoRA adapters
4. **Phase 4**: Social features - Leaderboard, community

## 11.3 Thông Tin Dự Án

| Thông Tin | Chi Tiết |
|---|---|
| Version | v0.2.0 |
| Flutter SDK | 3.29+ |
| Dart SDK | 3.8.1+ |
| Platforms | iOS, Android, Web |
| Repository | github.com/InfinityZero3000/LexiLingo |
| Branch | feature |

*Document generated on Ngày 15 tháng 1 năm 2026*

**LexiLingo** - Learn English with AI