# **Evaluation and Improvement Plan for Offline Voice AI Models**

**June 2025**

## **Abstract**

This document outlines an internal plan to evaluate and enhance our offline Voice AI models for bank call audio processing, covering three model groups: Transcription and Translation, Collection Status and Promise-to-Pay, and Call Insights. Each group includes temporary pretrained models and proprietary models under development, processing client-provided audio calls to deliver transcriptions, translations, collection classifications, promise-to-pay categorizations, and call quality insights, all operating offline on CPU. The plan focuses on internal evaluation using self-generated ground truth, generating performance reports, and improving models through iterative fine-tuning and deployment. Leveraging client audio calls and internal annotation, the approach ensures domain-specific accuracy for banking applications (e.g., loans, payments, customer inquiries) while maintaining offline compatibility. Official sources, including Hugging Face, spaCy, NLTK, scikit-learn, NumPy, and TextBlob documentation, guide the development process.

## **1. Introduction**

Our Voice AI models process bank call audio to deliver actionable outputs for banking applications:

* **Transcription and Translation Models**:
  + **Transcription**: Raw language text with timestamps, capturing banking terms (e.g., “utang” as “loan”).
  + **Diarization**: Speaker labels (spk1 = caller, spk2 = agent) using channel-based or neural separation.
  + **Translation**: English text from raw language transcription, preserving banking context.
* **Collection Status and Promise-to-Pay Models**:
  + **Collection Status**: Classifies calls into Predues Collection, Postdue Collections <30 Days, >30 Days, or Late Collections >60 Days.
  + **Collection Scoring Criteria**: Evaluates 22 true/false criteria (e.g., call\_open\_timely\_manner, friendly\_confident\_tone).
  + **Promise-to-Pay**: Classifies calls into Settlement, Partial Settlement, Promise Broken, or Denial, with two additional labels (e.g., customer name, payment date).
* **Call Insights Model**:
  + **Audio Metrics**: Measures signal energy, loudness, and pitch variance.
  + **Sentiment/Emotion Analysis**: Quantifies sentiment (POSITIVE/NEGATIVE) and emotion (e.g., neutral, surprise).
  + **Quality Metrics**: Scores criteria (e.g., timely\_greeting, polite\_tone as true/false).
  + **Speech Analytics**: Analyzes speech rate, pitch variance, and loudness.
  + **Critical Parameters**: Evaluates verification, compliance, and customer satisfaction (pass/fail).
  + **Final Score**: Overall call score (0–1).

Temporary models use pretrained tools (e.g., Whisper, SeamlessM4Tv2, spaCy, TextBlob, scikit-learn, NumPy) for interim solutions, while proprietary models employ transformer-based architectures for long-term customization. All models operate offline on CPU, supporting banking needs (e.g., intent prediction, compliance). This internal plan addresses:

* Setting up temporary and designing proprietary models.
* Building data pipelines for client audio calls.
* Generating ground truth for all model outputs.
* Evaluating performance using self-generated ground truth.
* Generating downloadable performance reports.
* Iteratively fine-tuning and deploying models via a model improvement cycle.

Clients provide audio calls, while all ground truth (transcriptions, translations, labels) and evaluations are handled internally by our data annotation team.

## **2. Internal Tasks**

Our team handles the following tasks to support evaluation and improvement across all model groups:

### **2.1. Model Setup and Design**

* **Temporary Models**:
  + **Transcription/Translation**: Configure Whisper large-v3, SeamlessM4Tv2, and AgglomerativeClustering for offline operation.
  + **Collection/Promise-to-Pay**: Configure spaCy for NLP, NLTK for text processing, and scikit-learn for classification.
  + **Call Insights**: Configure NumPy for audio metrics, TextBlob for sentiment/emotion, and scikit-learn for classification/regression.
* **Proprietary Models**: Define transformer-based architectures for transcription (ASR), diarization (speaker embedding), translation (NMT), collection status (text classification), collection criteria (multi-label classification), promise-to-pay (multi-output classification), and call insights components (regression/classification). Prototype on pilot data and benchmark against temporary models.

### **2.2. Data Pipeline Setup**

* Develop preprocessing pipeline for audio (e.g., 16kHz normalization with librosa, transcription via Whisper) and text (e.g., tokenization, code-switching handling).
* Automate data loading, splitting (training/validation/test, e.g., 80%/10%/10%), and annotation conversion to model-specific formats (e.g., JSON to inputs).
* Implement quality assurance (QA) checks for annotations (e.g., timestamp alignment, label consistency).

### **2.3. Ground Truth Generation**

* Label client audio calls using semi-automated tools (e.g., Whisper for transcription, manual correction). Data formats:
  + **Transcription/Translation**:
    - Audio, transcription with timestamps, speaker labels (spk1/spk2), English translation.
  + **Collection Status**:
    - Audio, transcription, translation, collection status (Predues, Postdue <30, >30, >60).
  + **Collection Scoring Criteria**:
    - Audio, transcription, translation, 22 true/false criteria (e.g., call\_open\_timely\_manner).
  + **Promise-to-Pay**:
    - Audio, transcription, translation, category (Settlement, Partial Settlement, Promise Broken, Denial), two labels (e.g., customer name, payment date).
  + **Call Insights** (per segment):
    - Audio, transcription, energy (float), loudness (dB), pitch\_variance (float), sentiment (POSITIVE/NEGATIVE), sentiment\_score (0–1), emotion (e.g., neutral, surprise), emotion\_score (0–1), quality metrics (true/false), speech\_rate (words/second), critical parameters (pass/fail), final score (0–1).
* Reserve validation and test sets (e.g., 10%/10% of calls) separate from training data.
* Ensure ground truth captures banking terms (e.g., “utang,” “balance”) and handles code-switching or accents.

### **2.4. Continuous Integration and Testing**

* Set up a continuous integration (CI) pipeline with automated tests for model components (e.g., transcription, classification, sentiment) and data pipeline (e.g., JSON parsing).
* Conduct regression testing during fine-tuning to detect performance drops.

### **2.5. Evaluation Pipeline**

* Build a Python-based pipeline to compare model outputs to ground truth, computing metrics using offline-compatible libraries (e.g., jiwer, sacrebleu, scikit-learn, NumPy, TextBlob).

### **2.6. Performance Reporting**

* Generate downloadable reports (PDF via reportlab, CSV) summarizing metrics, error analysis, and improvement recommendations.

### **2.7. Model Improvement Cycle**

* Fine-tune models iteratively, using validation sets for hyperparameter tuning and regularization to prevent overfitting.
* Optimize models for CPU deployment (e.g., quantization, pruning).
* Deploy improved models with updated code and documentation.

## **3. Evaluation Metrics**

Metrics assess performance across model groups:

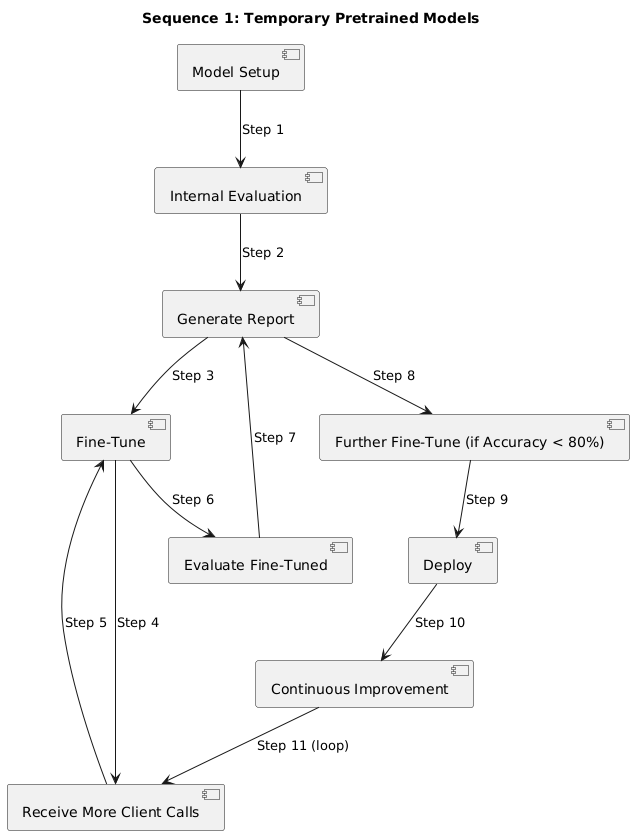
* **Transcription and Translation**:
  + **Transcription**:
    - Word Error Rate (WER): Incorrect words (target < 20%) [1].
    - Character Error Rate (CER): Character-level errors.
    - Banking Term Accuracy: Accuracy for key terms (e.g., “balanse”).
  + **Diarization**:
    - Channel Accuracy (Stereo): Correct assignments (target > 95%).
    - Diarization Error Rate (DER) (Mono): Errors in speaker/timing (target < 15%) [2].
  + **Translation**:
    - BLEU Score: N-gram overlap (target > 0.30) [3].
    - METEOR Score: Semantic accuracy (target > 0.30) [4].
    - chrF: Character-based metric.
  + **Latency**: Processing times.
* **Collection Status and Promise-to-Pay**:
  + **Collection Status**:
    - Classification Accuracy: Correct type\_of\_collection (target > 80%) [5].
    - Confusion Matrix: Misclassified types.
    - F1-Score: Per-class performance.
  + **Collection Scoring Criteria**:
    - Per-Criterion Accuracy: Correct true/false values (target > 85%).
    - Average Criteria Accuracy: Mean across 22 criteria.
    - Hamming Loss: Incorrect labels.
  + **Promise-to-Pay**:
    - Classification Accuracy: Correct category (target > 80%) [5].
    - Confusion Matrix: Misclassified categories.
    - Precision, Recall, F1-Score: Per-category performance.
    - Label Accuracy: Correct additional labels.
  + **Latency**: Processing times.
* **Call Insights**:
  + **Audio Metrics**:
    - Relative Error: Energy, loudness, pitch\_variance (target < 10%).
    - Mean Absolute Error (MAE): Predicted vs. ground truth.
  + **Sentiment/Emotion Analysis**:
    - Correlation: Sentiment/emotion scores (target > 0.8).
    - Classification Accuracy: Correct labels (target > 85%).
    - MAE: Scores (0–1).
  + **Quality Metrics**:
    - Per-Criterion Accuracy: True/false values (target > 85%).
    - Hamming Loss: Incorrect labels.
  + **Speech Analytics**:
    - Relative Error: Speech\_rate, pitch\_variance, loudness (target < 10%).
    - MAE: Predicted vs. ground truth.
  + **Critical Parameters**:
    - Accuracy: Pass/fail predictions (target > 90%).
  + **Final Score**:
    - MAE: Predicted vs. ground truth scores (target < 0.1).
  + **Latency**: Processing times.

## **4. Evaluation and Improvement Plans**

Two sequences are outlined for each model group: Sequence 1 for temporary pretrained models and Sequence 2 for proprietary models. Both incorporate a model improvement cycle.

### **4.1. Sequence 1: Temporary Pretrained Models**

#### **Flow Diagram**



#### **Step 1: Model Setup**

* **Transcription/Translation**: Configure Whisper large-v3, SeamlessM4Tv2, AgglomerativeClustering (e.g., TRANSFORMERS\_OFFLINE=1).
* **Collection/Promise-to-Pay**: Configure spaCy for NLP, NLTK for text processing, scikit-learn for classification (e.g., logistic regression, random forest).
* **Call Insights**: Configure NumPy for audio metrics, TextBlob for sentiment/emotion, scikit-learn for classification/regression.

#### **Step 2: Internal Evaluation**

* Evaluate on internal dataset (e.g., 5 calls, 25 minutes) with self-generated ground truth.
* Compute metrics (e.g., WER, accuracy, MAE) and analyze errors (e.g., misrecognized “balanse,” incorrect polite\_tone).

#### **Step 3: Generate Performance Report**

* Summarize metrics, errors, and fine-tuning recommendations.
* Note internal ground truth and temporary model usage.
* Formats: PDF, CSV.

#### **Step 4: Fine-Tune Model**

* Fine-tune using internal dataset, tuning hyperparameters (e.g., learning rate) on validation set (e.g., 5 calls).
* Update keyword lists (e.g., “pay today” for Settlement), adjust thresholds (e.g., fuzzy match for criteria), recalibrate audio processing.
* Apply regularization (e.g., dropout, L2) and optimize for CPU (e.g., quantization).

#### **Step 5: Receive Additional Client Calls**

* Obtain large number of calls (e.g., 50–100, 250–500 minutes).

#### **Step 6: Fine-Tune Model (Model Improvement Cycle)**

* Label new calls for training, reserving validation/test sets (e.g., 10%/10%).
* Fine-tune on training data, using validation set.
* Conduct regression testing (e.g., compare metrics to Step 4).

#### **Step 7: Evaluate Fine-Tuned Model (Model Improvement Cycle)**

* Evaluate on fixed test set (5 calls) with self-generated ground truth.
* Compute updated metrics and analyze errors.

#### **Step 8: Generate Performance Report (Model Improvement Cycle)**

* Summarize improvements (e.g., WER: 25% → 15%, accuracy: 70% → 85%) and errors.
* Recommend further fine-tuning if metrics are below targets (e.g., WER > 15%, accuracy < 80%).
* Formats: PDF, CSV.

#### **Step 9: Further Fine-Tune (Conditional, Model Improvement Cycle)**

* Fine-tune if metrics are below targets, using validation set and regularization.
* Optimize for CPU (e.g., pruning, ONNX).

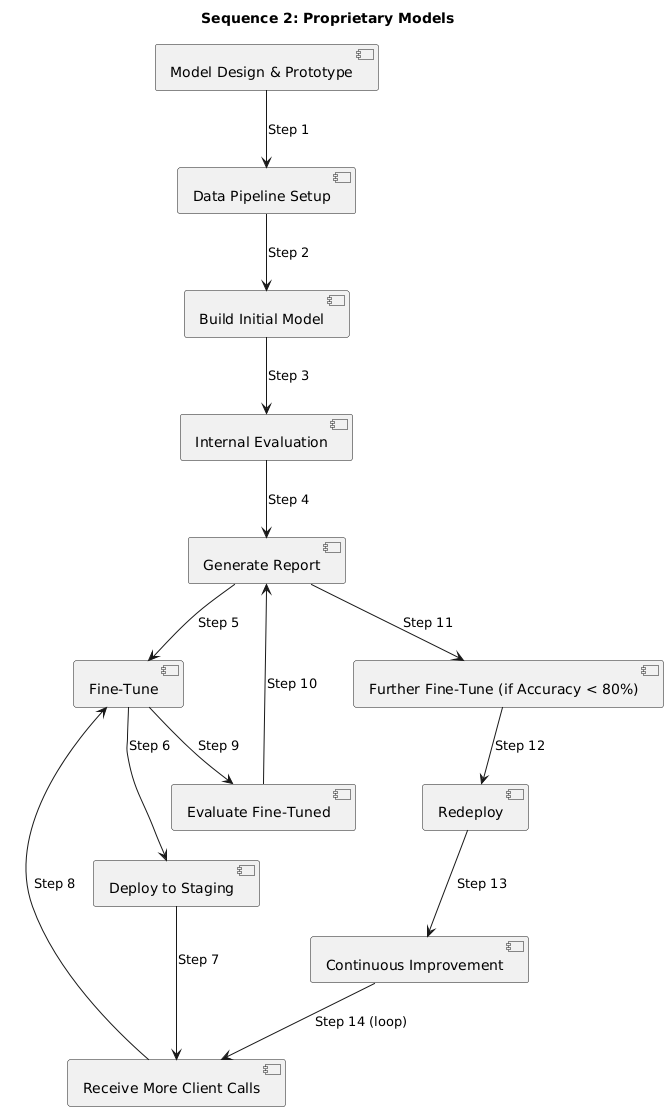
#### **Step 10: Deploy Model**

* Deploy to production with offline installation instructions.
* Document fine-tuning and evaluation results.

#### **Step 11: Continuous Improvement**

* Request more calls if metrics are suboptimal.
* Re-run model improvement cycle.
* Transition to proprietary models as they mature.

### **4.2. Sequence 2: Proprietary Models**



#### **Step 1: Model Design and Prototyping**

* Define transformer-based architectures:
  + **Transcription/Translation**: ASR for transcription, neural speaker embedding for diarization, NMT for translation.
  + **Collection Status**: Text classification for type\_of\_collection (4 classes).
  + **Collection Scoring Criteria**: Multi-label classification for 22 criteria.
  + **Promise-to-Pay**: Multi-output classification for category (4 classes) and two labels.
  + **Call Insights**: Regression for audio/speech metrics, classification for sentiment/emotion, quality metrics, critical parameters, regression for final score.
* Prototype on pilot dataset (e.g., 5 calls, 25 minutes) with annotations.
* Benchmark against temporary models (e.g., WER < 25%, accuracy > 75%).
* Document architecture and results.

#### **Step 2: Data Pipeline Setup**

* Develop preprocessing scripts (e.g., 16kHz normalization with librosa, text tokenization).
* Automate data splits (training/validation/test) and JSON conversion.
* Implement QA checks (e.g., timestamp overlaps, label consistency).

#### **Step 3: Build Initial Model**

* Train on client calls (e.g., 50 calls, 250 minutes), using labeled training data.
* Reserve validation (5 calls) and test sets (5 calls).
* Tune hyperparameters (e.g., learning rate) on validation set.
* Apply domain adaptation (e.g., banking terms, weighted loss).

#### **Step 4: Internal Evaluation**

* Evaluate on fixed test set (5 calls) using self-generated ground truth.
* Compute metrics and analyze errors.

#### **Step 5: Generate Performance Report**

* Summarize metrics, per-file errors, and fine-tuning recommendations.
* Note internal ground truth usage.
* Formats: PDF, CSV.

#### **Step 6: Fine-Tune Model**

* Fine-tune based on report (e.g., target high WER, low accuracy).
* Use validation set for hyperparameter tuning and early stopping.
* Apply regularization and quantization.

#### **Step 7: Deploy to Staging**

* Deploy to internal staging environment.
* Validate on new calls (e.g., 5 calls).
* Collect informal client feedback.

#### **Step 8: Receive Additional Client Calls**

* Obtain large number of calls (e.g., 50–100, 250–500 minutes).

#### **Step 9: Fine-Tune Model (Model Improvement Cycle)**

* Label new calls for training, reserving validation/test sets.
* Fine-tune on training data, tuning hyperparameters.
* Conduct regression testing.

#### **Step 10: Evaluate Fine-Tuned Model (Model Improvement Cycle)**

* Evaluate on fixed test set (5 calls).
* Compute updated metrics and analyze errors.

#### **Step 11: Generate Performance Report (Model Improvement Cycle)**

* Summarize improvements and errors.
* Recommend further fine-tuning if metrics are below targets.
* Formats: PDF, CSV.

#### **Step 12: Further Fine-Tune (Conditional, Model Improvement Cycle)**

* Fine-tune if metrics are below targets, using validation set and regularization.
* Optimize for CPU (e.g., pruning, ONNX).

#### **Step 13: Redeploy Model**

* Deploy to production with offline instructions.
* Document architecture, hyperparameters, and results.

#### **Step 14: Continuous Improvement**

* Request more calls if metrics are suboptimal.
* Re-run model improvement cycle.
* Conduct knowledge transfer workshops.

## **6. Conclusion**

This consolidated internal plan ensures robust evaluation and enhancement of our Voice AI models for transcription and translation, collection status and promise-to-pay, and call insights. Sequence 1 leverages pretrained models for immediate results, while Sequence 2 drives long-term customization with proprietary models. By using client audio calls, internal ground truth, and iterative model improvement cycles, we achieve high accuracy for banking applications while maintaining offline compatibility. Immediate next steps include setting up data pipelines and annotating new client calls to initiate both sequences across all model groups.

## **References**

[1] V. Panayotov et al., “LibriSpeech: An ASR Corpus Based on Public Domain Audio Books,” INTERSPEECH, 2015.  
[2] H. Bredin, “pyannote.metrics: A Toolkit for Reproducible Evaluation, Diagnostics, and Error Analysis of Speaker Diarization Systems,” 2017.  
[3] K. Papineni et al., “BLEU: A Method for Automatic Evaluation of Machine Translation,” ACL, 2002.  
[4] S. Banerjee and A. Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments,” ACL Workshop, 2005.  
[5] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, 2011.  
[6] S. Bird et al., “Natural Language Processing with Python,” O’Reilly Media, 2009.  
[7] M. Honnibal et al., “spaCy: Industrial-strength Natural Language Processing in Python,” [https://spacy.io](https://spacy.io/), 2023.  
[8] S. Loria, “TextBlob: Simplified Text Processing,” [https://textblob.readthedocs.io](https://textblob.readthedocs.io/), 2023.  
[9] Hugging Face, “Whisper Model Documentation,”<https://huggingface.co/docs/transformers/model_doc/whisper>, 2023.  
[10] Hugging Face, “SeamlessM4T Model Documentation,”<https://huggingface.co/docs/transformers/model_doc/seamless_m4t>, 2023.  
[11] Hugging Face, “Transformers Model Documentation,”<https://huggingface.co/docs/transformers>, 2023.