Live Voice Language Tutor - Product Requirements Document (PRD)

Version: 1.1
Author: Aiman (on behalf of product team)
Date: 2025-08-21

1. Overview

Product summary: This product is a web-based live voice language tutoring platform that provides interactive language learning through multi-turn conversations. The AI tutor teaches sentences in the target language, explains their meaning, demonstrates pronunciation, and provides real-time feedback on user attempts. The system uses Deepgram for Speech-to-Text and Text-to-Speech, Google Cloud Translation API for translation, GPT-5 as the LLM for tutoring, and Supabase for authentication, database, and file storage. The frontend is a Next.js application deployed on Vercel.

2. Goals & Success Metrics

Primary goals:

- Demonstrate a smooth, end-to-end language learning session: AI teaches → user practices → AI provides feedback → multiple turns per session
- Provide actionable pronunciation and grammar feedback after each user attempt
- Show gamification progress through badges and streaks; track user engagement metrics

Success metrics (quantitative targets for hackathon/prototype):

- Live transcription latency ≤ 1.5 seconds for target test network conditions
- Session save + transcript viewable within 5 seconds after session end
- Badge awarded correctly for first completed session, streaks tracked over 7-day window
- At least 80% of demo testers report the feedback is 'useful' in a 5-point survey during demo

3. Scope (MVP - strictly implemented)

The Minimum Viable Product will implement the following features exactly:

- **Interactive Learning Flow:** AI tutor presents sentences in target language, explains meaning, demonstrates pronunciation, then coaches user attempts through multiple turns
- Real-time speech-to-text (Deepgram streaming) for user pronunciation attempts
- AI Tutor personas driven by GPT-5 prompt templates; persona output delivered as text and synthesized audio using Deepgram TTS

- **Session Structure:** Multiple teaching turns per session (3-5 sentences), with each turn containing: AI teaches → AI explains → User practices → AI provides feedback
- Session recording: complete audio file and full transcript of all turns saved to Supabase Storage
- **Explainable feedback:** automated pipeline using Deepgram word-level confidences and GPT-5 to produce feedback on pronunciation accuracy and grammar understanding
- Badges & Rewards: server-side rules for completed sessions, streaks, and learning milestones

Non-goals (explicitly excluded):

- Single-turn interactions without teaching component
- Phoneme-level articulatory graphics or forced-alignment visualizers
- On-device custom phonetic models and phoneme-level scoring

4. User Personas & Core Flows

Primary user personas:

- **Language Learner:** Student wanting structured learning with pronunciation practice
- **Product Demo Judge / Stakeholder:** Evaluates end-to-end learning experience

Core flows (exact sequence each must follow):

4.1 Start Learning Session

1. User signs in using Supabase Auth (email/password or OAuth)
2. User clicks 'Start Session' and chooses: target learning language, difficulty level, and AI Persona
3. Frontend requests microphone access, establishes streaming connection to backend
4. Backend initializes session context and selects appropriate sentences for the user's level

4.2 Multi-Turn Learning Interaction

Per Turn (repeated 3-5 times per session):

1. **AI Teaches:** Backend sends sentence in target language to GPT-5 with persona context. GPT-5 returns: sentence text, pronunciation guide, meaning explanation
2. **AI Demonstrates:** Backend requests TTS from Deepgram. Frontend displays sentence text and plays pronunciation audio
3. **AI Explains:** Frontend shows meaning/context explanation from GPT-5
4. **User Practices:** User attempts to pronounce the sentence. Frontend streams audio to backend via WebSocket/WebRTC
5. **AI Provides Feedback:** Backend processes user audio through Deepgram ASR, compares with target sentence, sends to GPT-5 for pronunciation and grammar feedback

6. **Optional Repetition:** User can request AI to repeat demonstration or attempt pronunciation again

4.3 End Session

1. User clicks 'End Session' or completes target number of turns
2. Backend finalizes session: combines all turn audio into single file, stores in Supabase Storage, writes complete transcript and learning progress to Postgres, triggers badge evaluation
3. User views session summary with overall progress and can replay individual turns

5. Functional Requirements (APIs, Data & Behavior)

5.1 Authentication & Authorization

Requirement: All API endpoints validate Supabase JWT. Implement Row Level Security (RLS) policies for user session access.

5.2 Learning Content & Session Management

Requirement: Backend maintains a structured sentence database organized by language and difficulty. Session context tracks current turn, user progress, and selected sentences.

5.3 Streaming & ASR (User Input Only)

Requirement: Backend accepts user audio during practice phases and forwards to Deepgram streaming ASR with word-level confidence scores for pronunciation analysis.

5.4 GPT-5 Integration (Teaching + Feedback)

Requirement: Backend calls GPT-5 with different prompt templates for:

- **Teaching Mode:** Generate/select appropriate sentence, provide pronunciation guidance and meaning
- **Feedback Mode:** Analyze user pronunciation attempt against target sentence

Teaching Payload:

```json
{
  "session_id": "<UUID>",
  "persona_profile_id": "<persona_id>",
  "user_target_language": "<lang_code>",
  "user_level": "<beginner|intermediate|advanced>",
  "turn_number": "<int>",
  "session_context": "<previous_sentences>"
}
```

Feedback Payload:

```json
{
  "session_id": "<UUID>",
  "target_sentence": "<sentence_user_should_say>",
  "user_transcript": "<what_user_actually_said>",
  "confidence_scores": ["<per_word_scores>"],
  "persona_profile_id": "<persona_id>"
}
```

5.5 TTS (Deepgram)

Requirement: All tutor demonstrations use Deepgram's Aura-2 TTS. Cache common teaching sentences to minimize cost.

5.6 Session Persistence and Metadata

Data model (exact table names and essential columns):

Table: sessions

- id: uuid (primary key)
- user_id: uuid (foreign key -> users.id)
- persona_id: text
- target_language: text (ISO code)
- difficulty_level: text
- started_at: timestamptz
- ended_at: timestamptz
- total_turns: integer
- completed_turns: integer
- audio_url: text (Supabase storage URL)
- session_transcript_json: jsonb (all turns with timestamps)
- overall_progress_json: jsonb (session learning metrics)

Table: learning_turns

- id: uuid (primary key)
- session_id: uuid (foreign key -> sessions.id)
- turn_number: integer
- target_sentence: text

- sentence_meaning: text
- user_transcript: text
- pronunciation_feedback_json: jsonb
- grammar_feedback_json: jsonb
- turn_completed: boolean
- created_at: timestamptz

Table: sentence_bank

- id: text primary key
- language: text (ISO code)
- difficulty_level: text
- sentence_text: text
- meaning_english: text
- pronunciation_guide: text
- category: text (greetings, food, travel, etc.)

5.7 Badge Rules (server-side)

Requirement: SQL functions/triggers for:

- first_session: award when sessions count = 1
- perfect_pronunciation: award when turn has >90% pronunciation accuracy
- 7_day_streak: award for sessions on 7 distinct days in last 7 days
- sentences_mastered_50: award when 50 sentences completed with good scores

6. Non-functional Requirements

Performance: Live transcription latency ≤ 1.5 seconds. Session turn transition < 2 seconds. **Reliability:** 99% uptime during demo window **Scalability:** Handle 10 concurrent learning sessions

Security & Privacy:

- Encrypt audio at rest in Supabase Storage
- Apply Supabase RLS for user session isolation
- Explicit recording consent for learning sessions
- Store API keys as encrypted environment variables

7. Infrastructure & Deployment

Hosting: Next.js app on Vercel (frontend + API routes). Supabase for Postgres + Storage + Auth.
Networking: WSS for real-time audio streaming during practice phases.

8. 3-Day Development Plan (exact schedule)

Day 1 — Foundation (8 hours):

- Initialize Next.js, configure Vercel, provision Supabase
- Implement Supabase Auth and basic session management
- Build sentence bank and learning turn data models
- Implement frontend mic capture for practice phases

Day 2 — Core Learning Flow (10 hours):

- Wire GPT-5 teaching mode (sentence selection + explanation)
- Implement Deepgram TTS for AI demonstrations
- Build user practice audio capture and Deepgram ASR processing
- Implement GPT-5 feedback mode for pronunciation analysis
- Create multi-turn session orchestration

Day 3 — Polish & Gamification (8 hours):

- Implement badge system for learning milestones
- Build session replay functionality
- Add learning progress visualization
- Demo preparation and bug fixes

9. Risks & Mitigations

Risk: Multi-turn complexity may introduce latency issues **Mitigation:** Implement turn state management and audio chunking for smooth transitions

Risk: Teaching content quality varies with GPT-5 prompts
Mitigation: Curate sentence bank and use structured prompt templates for consistent teaching

10. Documentation Links (essential)

- Deepgram - API Overview & Streaming ASR: https://developers.deepgram.com/reference/deepgram-api-overview
- Deepgram - Text-to-Speech (Aura-2): https://developers.deepgram.com/docs/text-to-speech
- Google Cloud Translation API: https://cloud.google.com/translate/docs
- OpenAI - GPT-5 model docs: https://platform.openai.com/docs/guides/latest-model

- Supabase - Storage and Auth: https://supabase.com/docs/guides/storage
- Next.js documentation: https://nextjs.org/docs
- Vercel - Deploying Next.js: https://vercel.com/docs/frameworks/full-stack/nextjs