CMPE 491
Nounce: Language Platform for Phonetic Analysis & Pronunciation Assessment

Ümit Can Evleksiz

Advisor:
Prof. Lale Akarun
Prof. Murat Saraçlar

Contents

Chapter 1

INTRODUCTION

1.1 Broad Impact

1.1.1 Learning Impact

Nounce is a web-based pronunciation assessment application that provides learners with practical how-to knowledge for speaking standard English, focusing on pronunciation accuracy and clarity. By introducing users to the International Phonetic Alphabet (IPA) and phonetic concepts, the system builds domain familiarity, enabling learners to discuss and understand sound units systematically. The platform incorporates gamification elements and provides fast, friendly feedback to create a safe learning environment where users can practice without fear of judgment.

It is important to note that the objective is not to replace human-to-human interactions in language learning, but rather to provide accessible tools for individuals who may lack access to human tutors, native speaker friends, or formal language education resources. The system serves as a supplementary tool that empowers self-directed learning while acknowledging the irreplaceable value of human interaction in language acquisition.

1.1.2 Academic Impact

This project contributes to the academic community by demonstrating the practical application of fine-tuning deep learning models for specialized phonetic transcription tasks. The work combines core signal processing techniques with modern machine learning approaches, integrating multiple moving parts including audio preprocessing, phonetic transcription, forced alignment, and pronunciation assessment into a cohesive full-stack application. The system showcases how research-grade models can be adapted and deployed in real-world educational contexts, providing a reference implementation for similar pronunciation assessment systems.

1.2 Ethical Considerations

1.2.1 User Speech Data

The system handles sensitive user speech data, which raises important privacy and usage concerns. Audio recordings are used for inference during pronunciation assessment, but users must explicitly opt-in for any potential use of their data in model fine-tuning, voice cloning, or correct pronunciation generation features. Clear consent mechanisms and transparent data usage policies are essential to protect user privacy and maintain trust.

1.2.2 Prescriptive Approach in Linguistics

The application focuses on US English pronunciation, which may appear prescriptive. However, it is crucial to emphasize that all dialects, accents, and language variants are linguistically valid. US English is chosen not to minimize or devalue other accents, but because it is widely adopted and in high demand globally, particularly in professional and academic contexts. The system aims to improve comprehensibility and communication effectiveness rather than eliminate linguistic diversity. Users are encouraged to understand that accent reduction is optional and that the primary goal is clear communication, not accent elimination.

1.2.3 Monetization and Accessibility

Commercialization of educational technology raises concerns about potentially gatekeeping education from economically disadvantaged populations. While the system may include premium features or subscription models, careful consideration must be given to ensuring that core pronunciation assessment capabilities remain accessible. Free tiers, educational discounts, and open-source components can help mitigate barriers to access, ensuring that pronunciation learning tools do not become exclusive to those who can afford them.

Chapter 2

PROJECT DEFINITION AND PLANNING

2.1 Project Definition

This project develops Nounce, a web-based application to assist English language learners in improving pronunciation through automated speech analysis and personalized feedback. The system enables users to record speech, receive detailed pronunciation assessments, and gain insights into English phonology.

Core functionality includes: (i) capturing and processing user speech recordings, (ii) analyzing pronunciation accuracy at the phonetic level using machine learning models, (iii) comparing actual pronunciation against target acoustic patterns, and (iv) providing educational guidance on English phonology.

The system focuses on English with US accent (potentially including UK), with initial development targeting Turkish-native English speakers. Future milestones will leverage users' native language to better detect phonetic units. The application is designed to be accessible and pedagogically sound, emphasizing comprehensibility over accent elimination.

2.2 Project Planning

2.2.1 Project Time and Resource Estimation

The project is developed as a solo effort with the following resource allocation:

Time Commitment:

- Development time: 15 hours per week
- Project duration: One academic semester (approximately 12 weeks)

- Total estimated effort: 180 hours

Infrastructure Costs:

- Full-stack server hosting: $5 per month
- Serverless PostgreSQL database: $5 per month
- Serverless GPU for ML job queue: $20-30 per month (moderate user activity)
- Total monthly infrastructure cost: $30-40 per month

Additional Resources:

- Development tools and frameworks (open-source)
- ML model training and evaluation datasets
- Audio processing libraries and APIs

2.2.2 Success Criteria

The project will be considered successful upon achieving the following objectives:
Technical Requirements:

- Full-stack secure web application with authentication and authorization
- System uptime of 99% or higher
- Acoustic ML model producing accurate phonetic-level transcriptions independent of prosodic variations
- Difference algorithm comparing actual pronunciation against target patterns with actionable feedback
- Scalable architecture supporting concurrent sessions and asynchronous audio processing

User Experience Requirements:

- Industry-grade UI/UX following modern web standards
- Intuitive workflow for recording, uploading, and reviewing feedback
- Clear presentation of pronunciation analysis results
- Responsive design supporting multiple devices

- Accessible interface adhering to web accessibility guidelines

Functional Requirements:

- Accurate real-time or near-real-time pronunciation assessment
- Educational content integration for English phonology
- Progress tracking and historical analysis
- Support for multiple audio formats and quality levels

2.2.3 Risk Analysis

Several risks have been identified that may impact project success:

Technical Risks:

- **ML Model Performance:** The machine learning model may not achieve desired accuracy, particularly with low-quality audio, background noise, or non-standard accents. Mitigation includes extensive training on diverse datasets, robust preprocessing, and fallback mechanisms for low-confidence predictions.
- **Audio Quality Variability:** Users may submit recordings with varying quality, device capabilities, and environmental conditions. The system must handle this through adaptive preprocessing and quality assessment.
- **Computational Resource Constraints:** Processing requirements may strain serverless GPU resources during peak usage. Load balancing, queue management, and resource scaling will address this.
- **Integration Complexity:** Coordinating multiple system components may introduce integration challenges. Comprehensive testing and modular architecture will mitigate these risks.

Pedagogical and Ethical Risks:

- **Language and Accent Limitations:** The application focuses on English with US accent (potentially UK). Other languages are not supported. Initial development targets Turkish-native English speakers, with future milestones leveraging native language for better phonetic detection. This limitation must be clearly communicated to users to avoid misconceptions about supported languages and accents.

- **Overly Prescriptive Approach:** The system may promote misconceptions about a single "correct" pronunciation, marginalizing linguistic variation. This is addressed by emphasizing comprehensibility over accent matching, acknowledging linguistic diversity, and transparently communicating system limitations.

- **User Expectations vs. Reality:** Users may have unrealistic expectations about pronunciation transformation. Clear communication about the tool's purpose as a learning aid is essential.

- **Data Privacy and Security:** Handling sensitive user audio data requires robust security measures and clear privacy policies. Users must explicitly opt-in for any use of their data in model fine-tuning, voice cloning, or pronunciation generation. Compliance with data protection regulations will be prioritized.

- **Accessibility and Monetization:** Commercial features may create barriers for economically disadvantaged users. Free tiers and educational discounts should be considered to ensure accessibility.

Project Management Risks:

- **Single Developer Constraints:** Limited capacity for parallel development and peer review. Careful time management and feature prioritization are essential.

- **Scope Creep:** The project may expand beyond initial scope. Potential scope creep includes voice cloning and generating correct pronunciation using the user's voice, which are explicitly out of scope. A clear feature prioritization framework and milestone-based development approach will maintain focus.

2.2.4 Team Work

This project is being developed as an individual effort. All aspects including requirements analysis, system design, implementation, testing, and documentation are handled by a single developer.

Chapter 3

RELATED WORK

3.1 Speech Foundation Models and Phonetic Transcription

Several open-source deep learning models are available for phonetic transcription. Wav2Vec 2.0 [1] and WavLM [2] provide self-supervised speech representations that can be fine-tuned for phonetic recognition, though they require significant adaptation. For this application, POWSM (Phonetic Open Whisper-Style Speech Model) [11] is best suited, as it is specifically engineered for phonetic transcription with textual transcription context. The model is open to fine-tuning, which is valuable for adapting to Turkish-native English speakers. Connectionist Temporal Classification (CTC) [10] is commonly used for sequence alignment in phonetic transcription tasks.

3.2 Forced Alignment and Assessment

The Montreal Forced Aligner (MFA) [12] is an industry-standard tool for phoneme-level time alignment, supporting English out-of-the-box. Pronunciation assessment employs metrics such as Phoneme Error Rate (PER) and goodness of pronunciation (GOP) scores [14] to evaluate speech quality.

3.3 Commercial Applications

Commercial pronunciation assessment platforms demonstrate practical applications of these technologies. ELSA Speak [3] provides real-time pronunciation feedback for English language learners with personalized AI coaching. Pronounce AI [13] offers instant speech feedback, pronunciation practice, and AI speaking partners for professionals and language learners, supporting both American and British English accents.

Chapter 4

METHODOLOGY

4.1 Application Architecture

Nounce is built as an industry-standard web-based application with a focus on user experience. The full-stack architecture is dockerized for consistent deployment and development environments. The frontend utilizes React with TanStack Start for server-side rendering and routing, providing a modern, responsive user interface built with Shadcn UI components for a sleek, accessible design. The backend leverages TanStack Start's server functions for API endpoints and business logic.

Authentication is handled through Clerk (free tier), supporting email/password and social sign-in options (Google, GitHub). Authorization is implemented using user metadata stored in Clerk, enabling role-based access control for administrative features.

Data persistence is handled through Neon, a serverless PostgreSQL database, providing scalable and reliable storage for user recordings and metadata with automatic scaling and backups. Audio file storage is implemented using Cloudflare R2 object storage solution. The entire stack is containerized using Docker, enabling reproducible deployments across development, staging, and production environments.

Modern server state management and caching are implemented using TanStack Query, which provides efficient data fetching, automatic background refetching, and intelligent caching mechanisms. This ensures optimal performance and user experience by minimizing unnecessary network requests and maintaining responsive UI updates.

4.1.1 Production Deployment Architecture

The production deployment follows a serverless, cloud-native architecture:

Domain and DNS: The application is served at `https://nounce.pro`, with the domain purchased and configured for production use. DNS management is handled through Cloudflare, which provides reliable DNS resolution and domain management capabilities.

Application Hosting: The TanStack Start application is deployed on Fly.io, providing global edge deployment with automatic scaling based on traffic. The application is containerized using Docker and deployed through automated CI/CD pipelines. The application is accessible via the `nounce.pro` domain, with Cloudflare's global network providing DDoS protection and enhanced security.

Security and Protection: Cloudflare's global network provides comprehensive DDoS protection, protecting the application from distributed denial-of-service attacks. The Cloudflare infrastructure also offers additional security features including SSL/TLS termination, rate limiting, and web application firewall capabilities.

Database: Neon serverless PostgreSQL provides the database layer with automatic scaling, point-in-time recovery, and branching capabilities for development and testing.

ML Services: Machine learning inference is handled by RunPod serverless GPU endpoints, which provide on-demand GPU resources for model inference. Two separate endpoints are deployed:

- **Assessment Endpoint:** Handles pronunciation assessment with POWSM [11] PR and G2P models, MFA alignment, and error detection

- **IPA Generation Endpoint:** Generates IPA transcriptions for reference speeches using POWSM [11] G2P model

Model Caching: RunPod network volumes are used to cache large model files (POWSM [11] models, MFA dictionaries and acoustic models), significantly reducing cold-start times. The maximum observed cold-start time is 30 seconds, while average task execution times are 3 seconds for Phone Recognition (PR) and 2 seconds for Grapheme-to-Phoneme (G2P) tasks.

Reference Speech Generation: ElevenLabs API is integrated for generating reference speech audio files. Currently, the system uses US English with two different speaker configurations to provide variety in target pronunciations.

Deployment Automation: GitHub Actions CI/CD pipelines handle building Docker images, running tests, linting, and automated deployment to Fly.io. The build process includes diagram generation from PlantUML source files, ensuring documentation stays synchronized with code changes.

4.2 Machine Learning Approach

The machine learning pipeline follows an experimental and iterative approach. After evaluating multiple open-source phonetic transcription models, including Wav2Vec 2.0, WavLM, and POWSM [11], POWSM was selected as the best-performing solution for the specific use case. The selection was based on accuracy, inference speed, and compatibility with the target user population (Turkish-native English speakers).

POWSM [11] is integrated for both Phone Recognition (PR) and Grapheme-to-Phoneme (G2P) tasks, providing accurate phonetic transcriptions. The implementation required extensive experimentation and custom implementation due to the lack of comprehensive documentation and standardized inference providers for many open-source phonetic transcription models. Fine-tuning on domain-specific data remains a future enhancement opportunity to further improve performance for Turkish-native English speakers.

4.3 Development Methodology

The project employed parallel development tracks, leveraging prior professional software development expertise while building new knowledge in signal processing and deep learning domains. Signal processing components, including audio preprocessing, quality assessment, and feature extraction, were developed alongside deep learning model integration and assessment pipelines.

Experimentation and prototyping were conducted using Python notebooks (Jupyter/IPython), enabling rapid iteration on digital signal processing (DSP) techniques, visualization of audio features, and deep learning model evaluation. These notebooks served as both development tools and documentation of the experimentation process, capturing insights and learnings that informed the final implementation.

The development process emphasizes modularity and separation of concerns, with clear boundaries between frontend, backend, database, and machine learning components. This architecture facilitates independent development and testing of each component while maintaining system integration.

4.3.1 Local Development and RunPod Simulation

For local development, a custom RunPod serverless simulator was implemented to replicate the production RunPod API behavior without requiring cloud GPU resources. The simulator architecture consists of:

Proxy Server: A FastAPI-based proxy server (`mod/dev/runpod_proxy.py`) that mimics the RunPod Cloud API, providing endpoints for job submission (`POST /v2/{endpoint_id}/run`) and status polling (`GET /v2/{endpoint_id}/status/{job_id}`). The proxy maintains in-memory job state and manages worker queues per endpoint.

Worker Containers: Docker containers for assessment and IPA generation workers, running the same code as production endpoints but accessible locally via Docker Compose networking.

Job Processing: Background async worker loops pull jobs from per-endpoint queues, ensuring sequential processing that mimics single GPU pod behavior. The simulator handles retries, timeouts, and webhook delivery, providing a faithful representation of RunPod's serverless architecture.

Configuration: Worker endpoints are mapped via `WORKER_MAP` environment variable, allowing flexible configuration of endpoint-to-worker mappings. The simulator is orchestrated using Docker Compose (`docker-compose.dev.yml`) and can be started using the `scripts/runpod.py` utility script.

This local simulation environment enables rapid development and testing of ML pipeline integration without incurring cloud GPU costs or dealing with cold-start delays during development iterations.

Chapter 5

REQUIREMENTS SPECIFICATION

5.1 Functional Requirements

5.1.1 User Authentication and Authorization

- **FR-1.1: User Registration** The system shall allow new users to register accounts with email and password authentication or social sign-in (Google, GitHub, etc.) through Clerk authentication service.
- **FR-1.2: User Login** The system shall provide secure login functionality for registered users through Clerk authentication service.
- **FR-1.3: Session Management** The system shall maintain user sessions and support automatic session expiration after periods of inactivity.
- **FR-1.4: User Profile** The system shall associate all recordings and progress data with authenticated user accounts.
- **FR-1.5: Admin Access** The system shall provide restricted administrative access for content management functions such as creating, editing, and deleting target texts and practice materials.

5.1.2 Speech Recording and Audio Input

- **FR-2.1: Browser-Based Recording** The system shall enable users to record speech directly through the web browser using the device microphone.
- **FR-2.2: Real-Time Audio Visualization** The system shall display real-time waveform visualization during recording to provide visual feedback.

- **FR-2.3: Recording Controls** The system shall provide start, stop, and pause controls for audio recording.
- **FR-2.4: Audio Format Support** The system shall accept audio recordings in WebM/Opus format from browsers and convert them to WAV format (16-bit, 16kHz, mono) for processing.
- **FR-2.5: Recording Duration Limits** The system shall enforce reasonable duration limits to prevent excessive resource consumption.

5.1.3 Audio Quality Assessment

- **FR-3.1: Basic Quality Check** The system shall perform basic audio quality assessment to ensure recordings meet minimum standards for analysis.
- **FR-3.2: Quality Feedback** The system shall provide feedback to users when recordings fail quality checks, suggesting improvements when possible.

5.1.4 Phonetic Analysis and Transcription

- **FR-4.1: Phonetic Transcription** The system shall generate International Phonetic Alphabet (IPA) transcriptions of user speech using acoustic ML models.
- **FR-4.2: Prosody Independence** The system shall produce phonetic transcriptions that are independent of prosodic variations (stress, intonation, rhythm).
- **FR-4.3: Phone-Level Alignment** The system shall align phonetic transcriptions with audio timestamps at the phone level.
- **FR-4.4: Target Comparison** The system shall compare user phonetic transcriptions against canonical IPA transcriptions of target pronunciations.
- **FR-4.5: Error Classification** The system shall classify pronunciation errors into categories: substitution, deletion, and insertion.

5.1.5 Pronunciation Feedback and Assessment

- **FR-5.1: Error Identification** The system shall identify and highlight specific phoneme-level pronunciation errors.
- **FR-5.2: Error Visualization** The system shall present pronunciation errors in a clear format showing the target vs. actual pronunciation.

- **FR-5.3: Overall Score** The system shall calculate and display an overall pronunciation accuracy score based on error rate.

5.1.6 Educational Content and Phonology Guidance

- **FR-6.1: Basic Guidance** The system shall provide basic guidance on English phonology concepts relevant to identified errors.
- **FR-6.2: Target Text Management** The system shall allow administrators to create, edit, and manage target texts for pronunciation practice.

5.1.7 Progress Tracking and History

- **FR-7.1: Recording History** The system shall maintain a history of user recordings with timestamps and metadata.
- **FR-7.2: Basic Progress Tracking** The system shall allow users to view their recording history and basic progress information.

5.1.8 Data Storage and Management

- **FR-8.1: Audio Storage** The system shall store processed audio files (16-bit, 16kHz, mono WAV) in persistent storage for long-term access.
- **FR-8.2: Metadata Storage** The system shall store recording metadata (user ID, file path, timestamps, analysis results) in a PostgreSQL database.

5.1.9 Asynchronous Processing

- **FR-9.1: Job Queue** The system shall process audio analysis tasks asynchronously using a job queue system to handle concurrent requests.
- **FR-9.2: Processing Status** The system shall provide status updates to users during audio processing.
- **FR-9.3: Error Handling** The system shall handle processing failures gracefully, providing clear error messages.

5.1.10 User Interface and Experience

- **FR-10.1: Responsive Design** The system shall provide a responsive user interface that works effectively on desktop and mobile devices.

- **FR-10.2: Intuitive Navigation** The system shall provide clear navigation and workflow for recording and viewing results.
- **FR-10.3: Visual Feedback** The system shall provide immediate visual feedback for user actions, including loading states and error notifications.
- **FR-10.4: Language Support** The system interface shall clearly communicate that it supports English with US accent (potentially UK), targeting Turkish-native speakers, and that other languages are not supported.

Chapter 6

DESIGN

6.1 Information Structure

6.1.1 Entity-Relationship Diagram

The entity-relationship diagram represents the complete database schema for Nounce. All tables and relationships shown in the diagram are fully implemented and operational, including:

- `practice_texts` - Practice materials with difficulty and category classification
- `authors` - Reference voice/author management
- `reference_speeches` - Target pronunciation audio files
- `user_recordings` - User-submitted audio recordings
- `audio_quality_metrics` - Quality assessment results
- `analyses` - Pronunciation assessment results
- `alignments` - Phone-level time alignment data
- `phoneme_errors` and `word_errors` - Error tracking
- `jobs` and `ipa_generation_jobs` - Asynchronous job tracking
- `user_preferences` - User settings

The schema supports the complete ML pipeline and assessment workflow, with proper indexing for efficient queries and relationships ensuring data integrity.

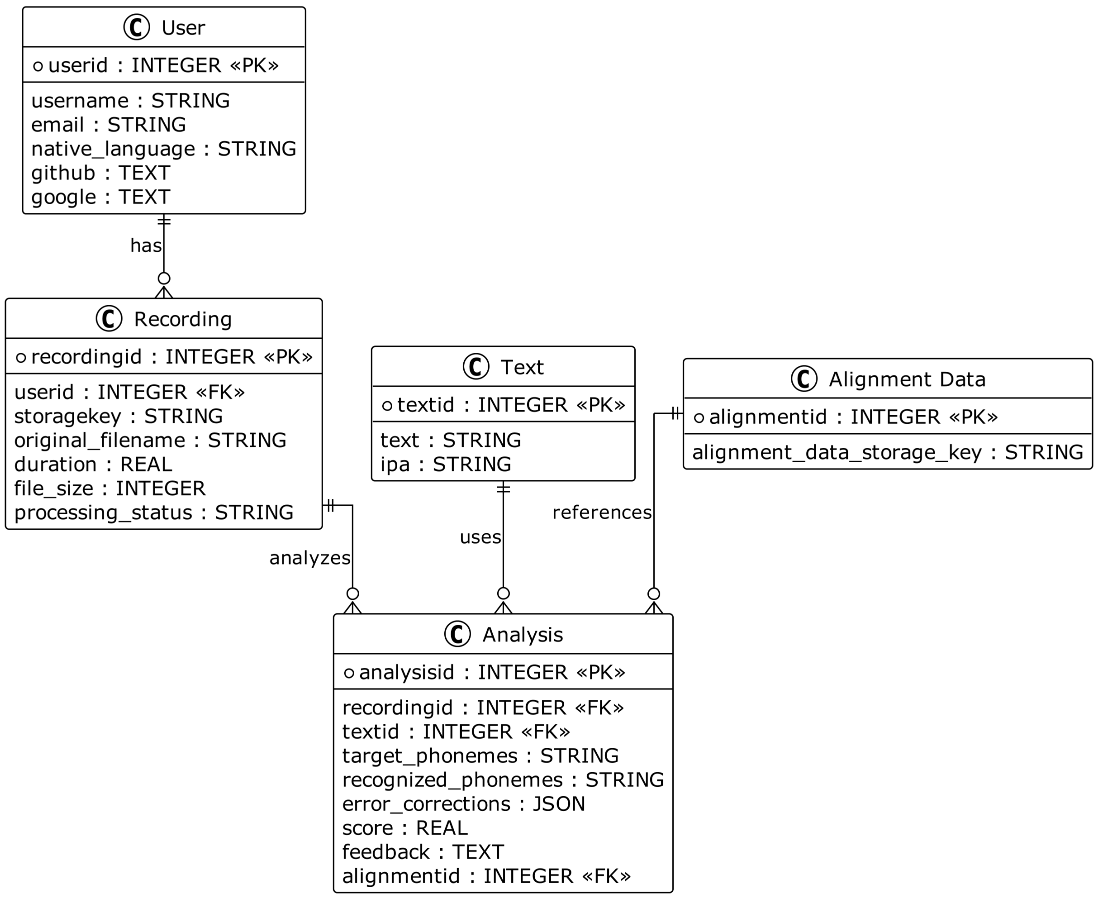


Figure 6.1: Entity-Relationship Diagram for Nounce (Planned Schema)

6.2 Information Flow

6.2.1 Pronunciation Assessment Workflow

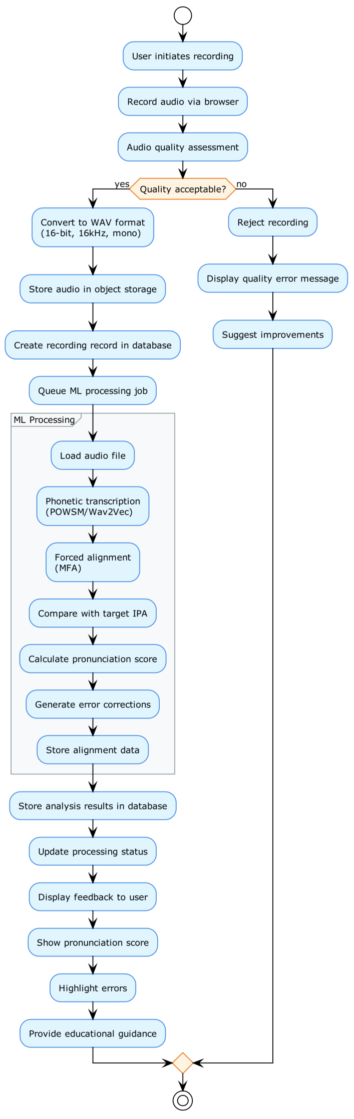


Figure 6.2: Activity Diagram: Pronunciation Assessment Workflow

6.3 System Design

6.3.1 System Architecture Diagram

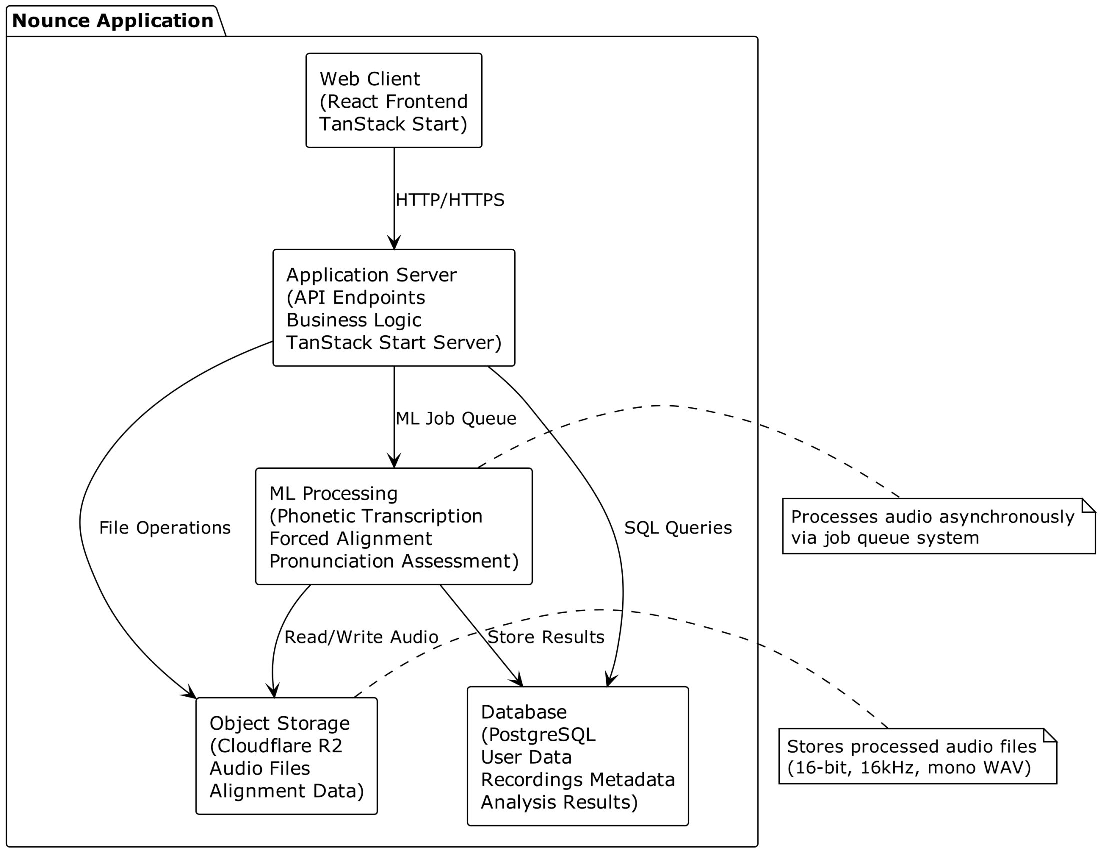

Nounce Application
Web Client (React Frontend TanStack Start)
HTTP/HTTPS
Application Server (API Endpoints Business Logic TanStack Start Server)
ML Job Queue
File Operations
SQL Queries
ML Processing (Phonetic Transcription Forced Alignment Pronunciation Assessment)
Processes audio asynchronously via job queue system
Read/Write Audio
Store Results
Object Storage (Cloudflare R2 Audio Files Alignment Data)
Database (PostgreSQL User Data Recordings Metadata Analysis Results)
Stores processed audio files (16-bit, 16kHz, mono WAV)

Figure 6.3: System Architecture Diagram for Nounce

6.3.2 Deployment Architecture Diagram

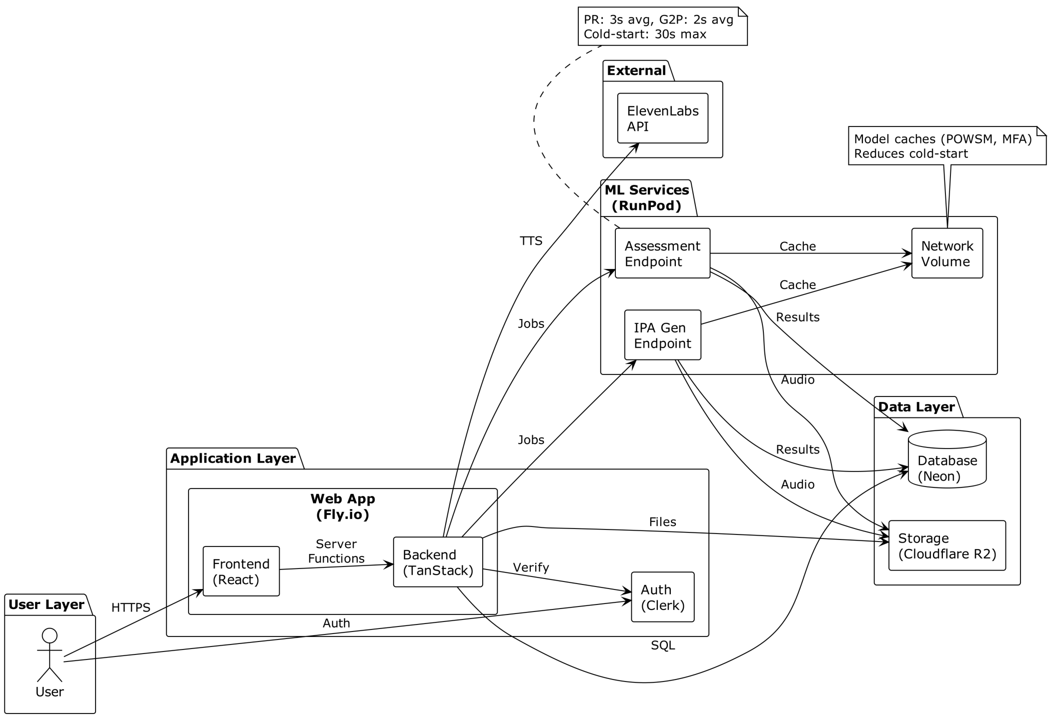
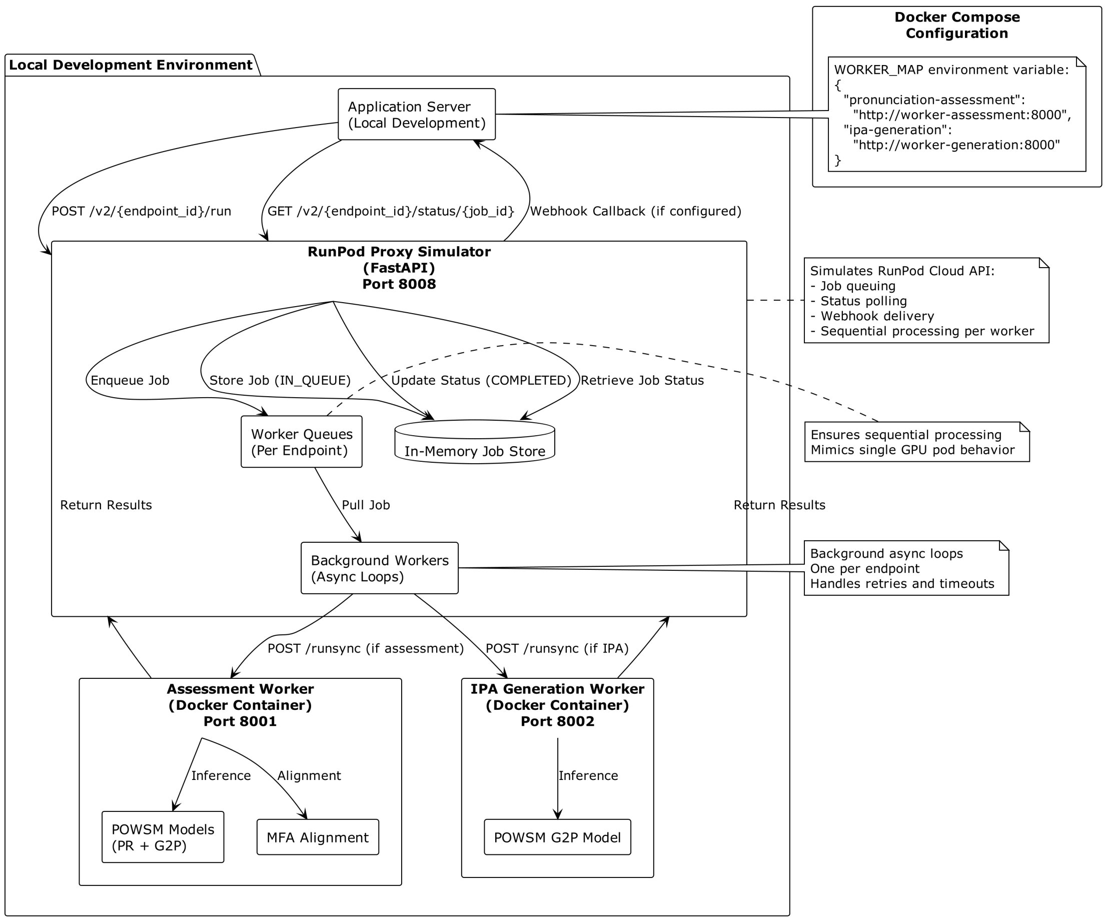


Figure 6.4: Production Deployment Architecture

6.3.3 RunPod Simulator Architecture Diagram



Docker Compose Configuration
WORKER_MAP environment variable:
{
"pronunciation-assessment":
"http://worker-assessment:8000",
"ipa-generation":
"http://worker-generation:8000"
}
Local Development Environment
Application Server (Local Development)
POST /v2/{endpoint_id}/run
GET /v2/{endpoint_id}/status/{job_id}
Webhook Callback (if configured)
RunPod Proxy Simulator (FastAPI) Port 8008
Simulates RunPod Cloud API:
- Job queuing
- Status polling
- Webhook delivery
- Sequential processing per worker
Enqueue Job
Store Job (IN_QUEUE)
Update Status (COMPLETED)
Retrieve Job Status
Worker Queues (Per Endpoint)
In-Memory Job Store
Ensures sequential processing
Mimics single GPU pod behavior
Return Results
Pull Job
Return Results
Background Workers (Async Loops)
Background async loops
One per endpoint
Handles retries and timeouts
POST /runsync (if assessment)
POST /runsync (if IPA)
Assessment Worker (Docker Container) Port 8001
Inference
Alignment
POWSM Models (PR + G2P)
MFA Alignment
IPA Generation Worker (Docker Container) Port 8002
Inference
POWSM G2P Model

Figure 6.5: Local RunPod Serverless Simulator Architecture

6.4 User Interface Design

6.4.1 Application Logo

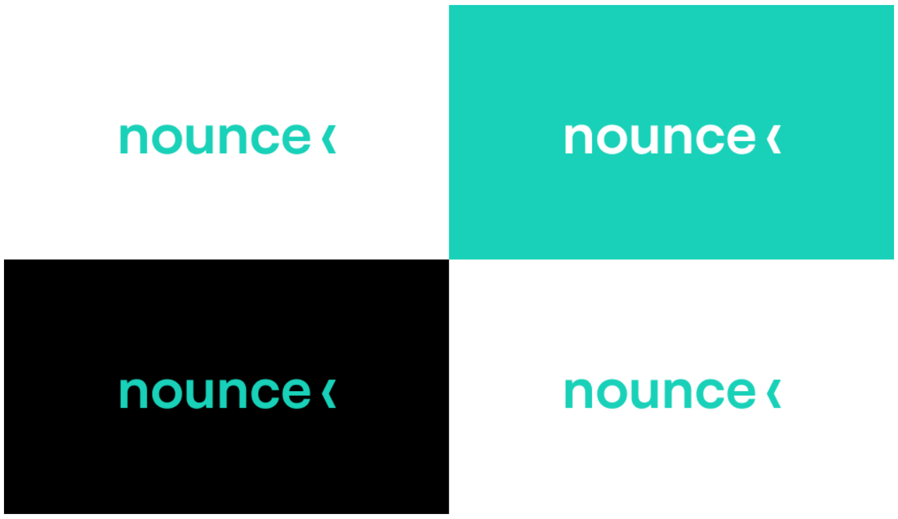
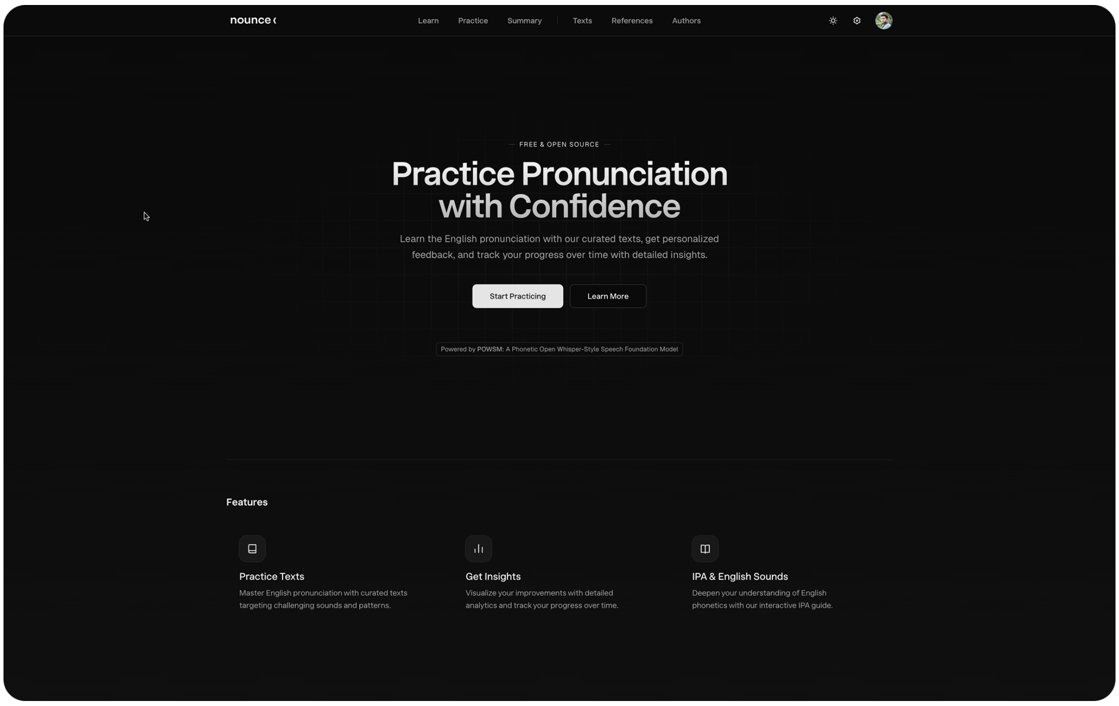


Figure 6.6: Nounce Application Logo

6.4.2 User Interface Screenshots

The Nounce application provides a modern, intuitive user interface built with Shadcn UI and ElevenLabs UI components. The following screenshots demonstrate key features and workflows of the application. All screenshots are captured in dark mode.

Home Page



— FREE & OPEN SOURCE —

Practice Pronunciation with Confidence

Learn the English pronunciation with our curated texts, get personalized feedback, and track your progress over time with detailed insights.

Powered by POWSM: A Phonetic Open Whisper-Style Speech Foundation Model

Features

Practice Texts

Master English pronunciation with curated texts targeting challenging sounds and patterns.

Get Insights

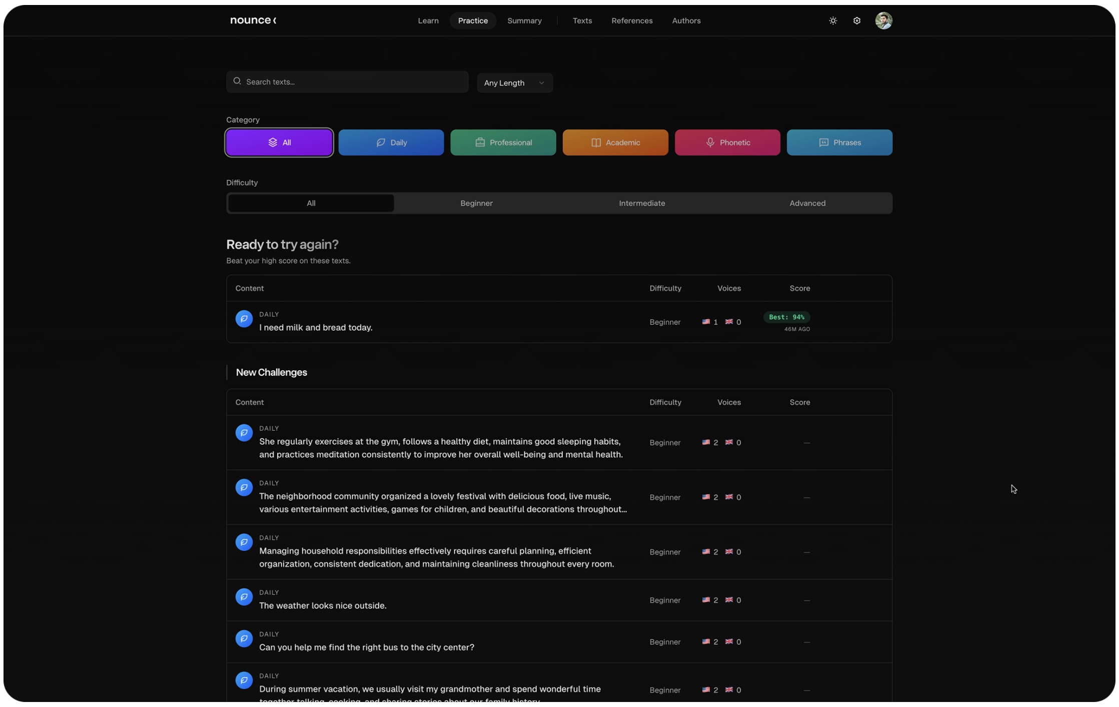
Visualize your improvements with detailed analytics and track your progress over time.

IPA & English Sounds

Deepen your understanding of English phonetics with our interactive IPA guide.

Figure 6.7: Home Page - Main landing page with feature overview and navigation

Practice Text Selection



Ready to try again?

Beat your high score on these texts.

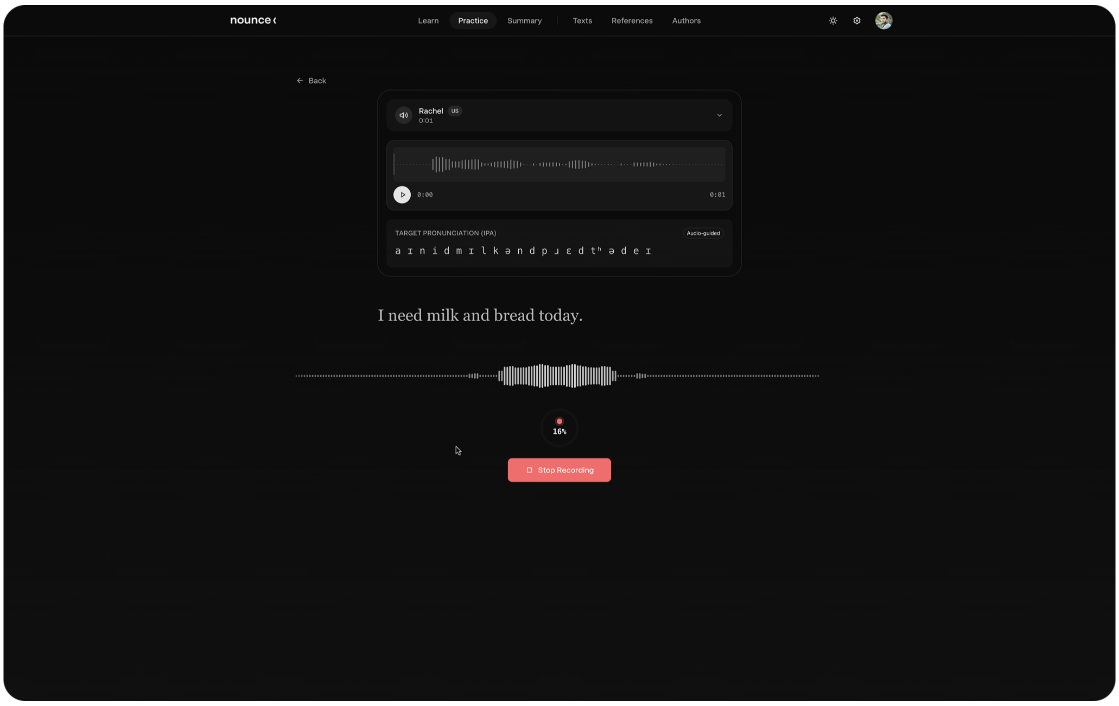
Content	Difficulty	Voices	Score
DAILY I need milk and bread today.	Beginner	🇺🇸 1 🇬🇧 0	Best: 94% 46M AGO

New Challenges

Content	Difficulty	Voices	Score
DAILY She regularly exercises at the gym, follows a healthy diet, maintains good sleeping habits, and practices meditation consistently to improve her overall well-being and mental health.	Beginner	🇺🇸 2 🇬🇧 0	—
DAILY The neighborhood community organized a lovely festival with delicious food, live music, various entertainment activities, games for children, and beautiful decorations throughout...	Beginner	🇺🇸 2 🇬🇧 0	—
DAILY Managing household responsibilities effectively requires careful planning, efficient organization, consistent dedication, and maintaining cleanliness throughout every room.	Beginner	🇺🇸 2 🇬🇧 0	—
DAILY The weather looks nice outside.	Beginner	🇺🇸 2 🇬🇧 0	—
DAILY Can you help me find the right bus to the city center?	Beginner	🇺🇸 2 🇬🇧 0	—
DAILY During summer vacation, we usually visit my grandmother and spend wonderful time together talking, cooking, and sharing stories about our family history.	Beginner	🇺🇸 2 🇬🇧 0	—

Figure 6.8: Practice Text Selection - Browse and filter practice texts by difficulty and category

Recording Interface

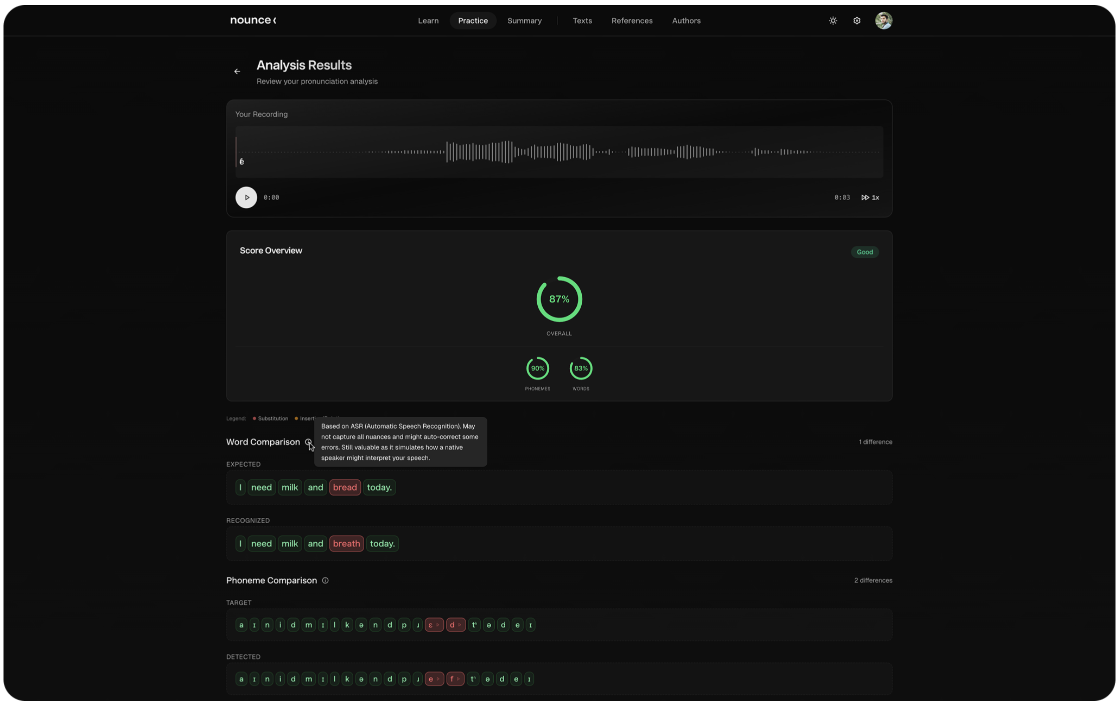


a ɪ n i d m ɪ l k ə n d p ɹ ɛ d tʰ ə d e ɪ

I need milk and bread today.

Figure 6.9: Recording Interface - Browser-based audio recording with real-time waveform visualization (dark mode)

Analysis Results



Analysis Results

Review your pronunciation analysis

Your Recording

0:00 0:03 1x

Score Overview

Good

87% OVERALL

90% PHONEMES

83% WORDS

Legend: ● Substitution ● Inserti

Based on ASR (Automatic Speech Recognition). May not capture all nuances and might auto-correct some errors. Still valuable as it simulates how a native speaker might interpret your speech.

Word Comparison — 1 difference

EXPECTED: I need milk and bread today.

RECOGNIZED: I need milk and breath today.

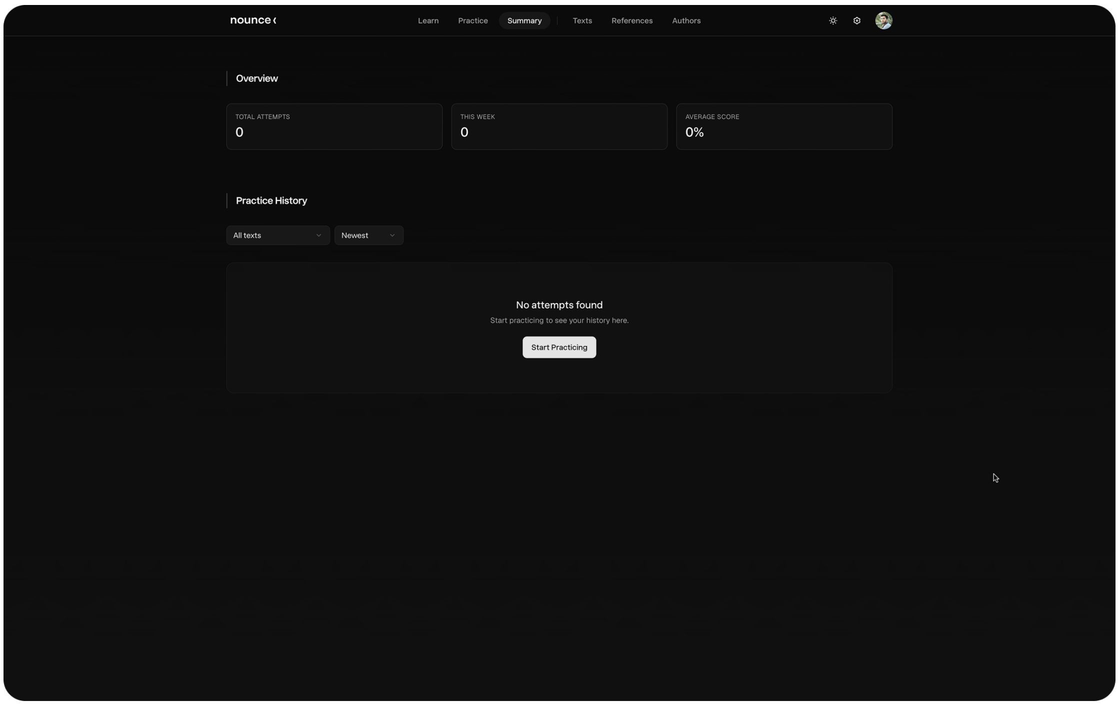
Phoneme Comparison — 2 differences

TARGET: a ɪ n i d m ɪ l k ə n d p ɹ ɛ d tʰ ə d e ɪ

DETECTED: a ɪ n i d m ɪ l k ə n d p ɹ e f tʰ ə d e ɪ

Figure 6.10: Analysis Results - Detailed pronunciation feedback with phoneme and word-level error visualization

Progress Dashboard



TOTAL ATTEMPTS 0

THIS WEEK 0

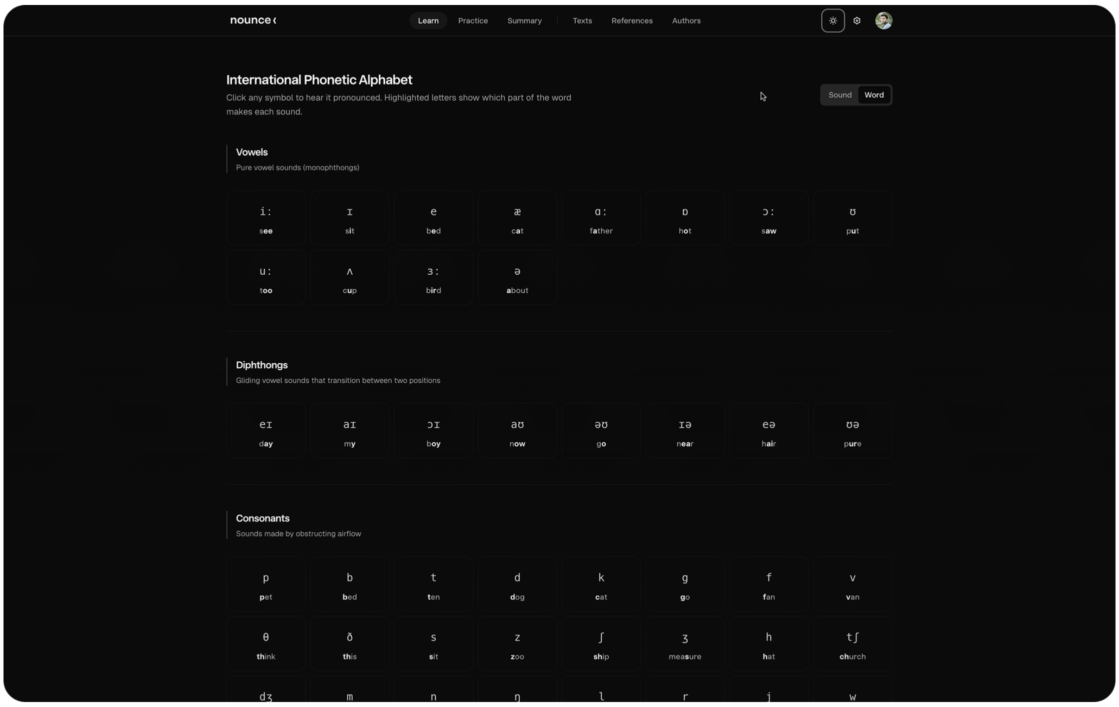
AVERAGE SCORE 0%

No attempts found

Start practicing to see your history here.

Figure 6.11: Progress Dashboard - User statistics and error analysis over time

Educational Content



International Phonetic Alphabet

Click any symbol to hear it pronounced. Highlighted letters show which part of the word makes each sound.

Vowels

Pure vowel sounds (monophthongs)

iː	ɪ	e	æ	ɑː	ɒ	ɔː	ʊ
see	sit	bed	cat	father	hot	saw	put
uː	ʌ	ɜː	ə				
too	cup	bird	about				

Diphthongs

Gliding vowel sounds that transition between two positions

eɪ	aɪ	ɔɪ	aʊ	əʊ	ɪə	eə	ʊə
day	my	boy	now	go	near	hair	pure

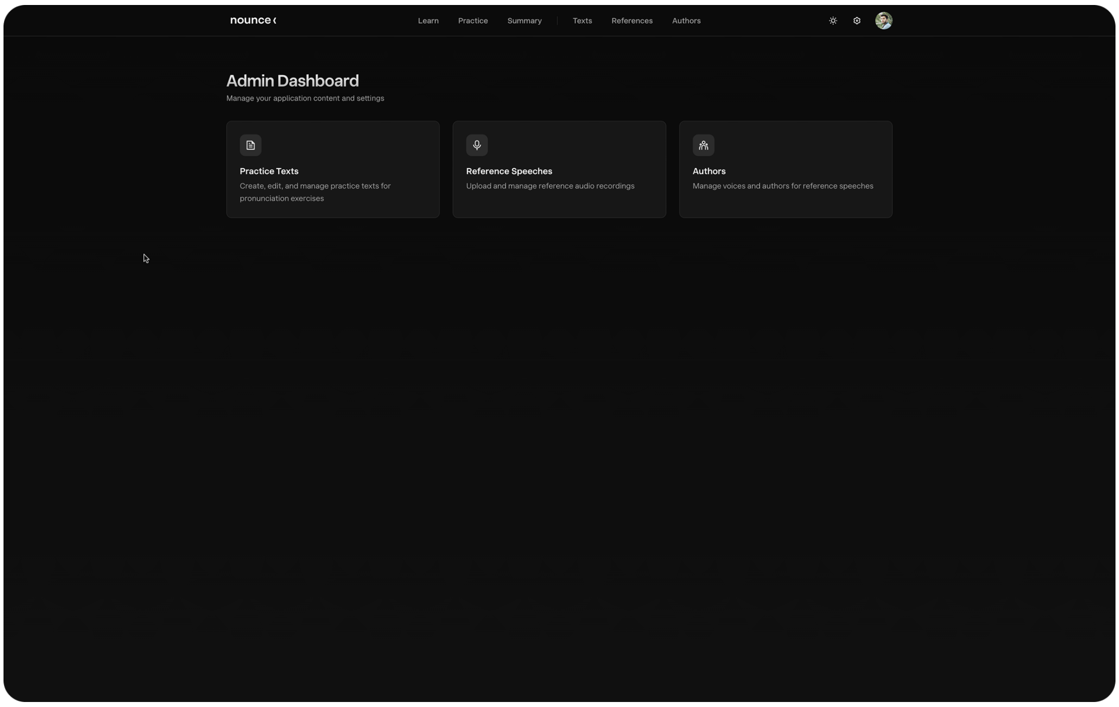
Consonants

Sounds made by obstructing airflow

p	b	t	d	k	g	f	v
pet	bed	ten	dog	cat	go	fan	van
θ	ð	s	z	ʃ	ʒ	h	tʃ
think	this	sit	zoo	ship	measure	hat	church
dʒ	m	n	ŋ	l	r	j	w

Figure 6.12: Educational Content - IPA and English phonology learning materials

Admin Interface



Admin Dashboard

Manage your application content and settings

Practice Texts

Create, edit, and manage practice texts for pronunciation exercises

Reference Speeches

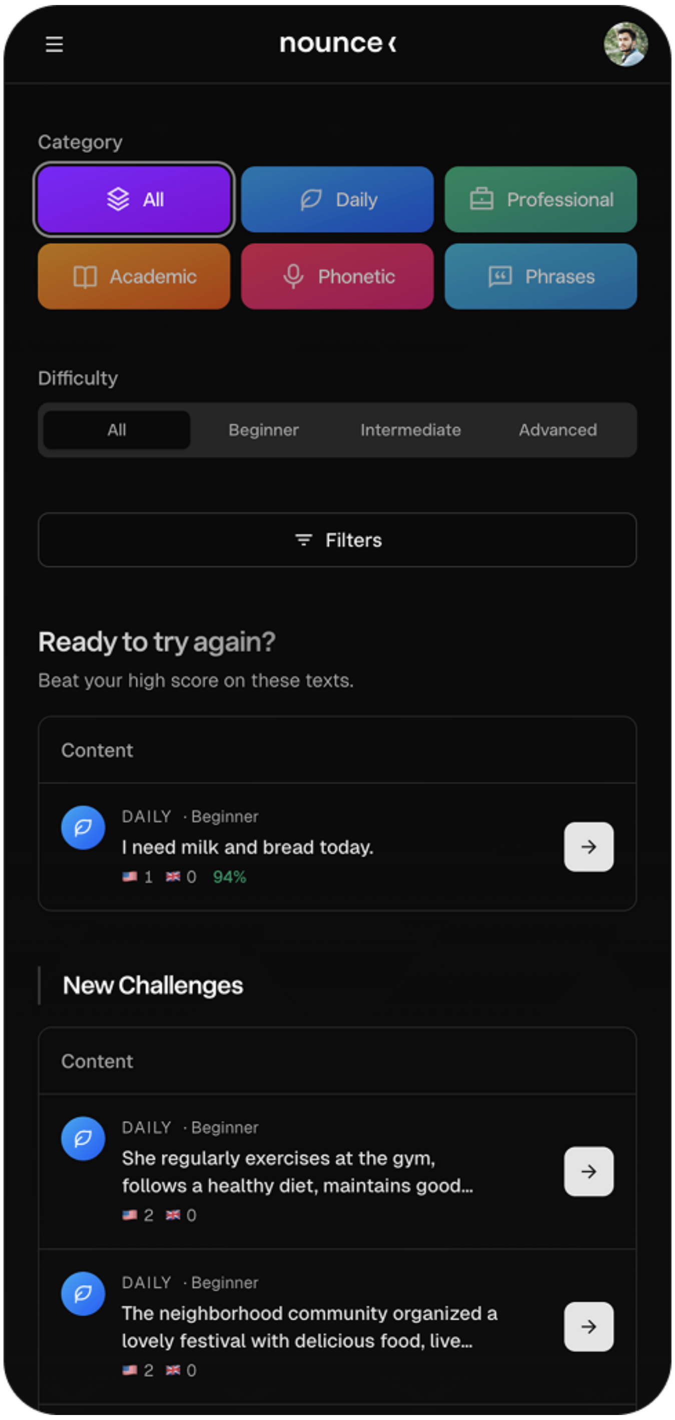
Upload and manage reference audio recordings

Authors

Manage voices and authors for reference speeches

Figure 6.13: Admin Interface - Content management for practice texts and reference speeches

Mobile View



Ready to try again?

Beat your high score on these texts.

Content

DAILY · Beginner
I need milk and bread today.
1 0 94%

New Challenges

Content

DAILY · Beginner
She regularly exercises at the gym, follows a healthy diet, maintains good...
2 0

DAILY · Beginner
The neighborhood community organized a lovely festival with delicious food, live...
2 0

Figure 6.14: Mobile View - Practice text selection on mobile devices

Chapter 7

IMPLEMENTATION AND TESTING

7.1 Implementation

The implementation of Nounce follows the architecture and methodology outlined in previous chapters. The source code is organized in a structured repository [8] with clear separation between the full-stack application, machine learning services, documentation, and experimental work.

7.1.1 Full-Stack Application

The main application codebase is located in the `app/` directory [4], built with modern web technologies:

Frontend: The React-based user interface is implemented using TanStack Start for server-side rendering and routing. The UI follows modern design principles with Tailwind CSS, Shadcn UI, and ElevenLabs UI components, providing a sleek, responsive, and accessible experience across desktop and mobile devices. Key user-facing features include:

- Practice text selection with filtering by difficulty (beginner, intermediate, advanced) and category (daily, professional, academic, phonetic challenge, common phrase)
- Browser-based audio recording with real-time waveform visualization
- Audio upload functionality with format conversion
- Pronunciation analysis results visualization with phoneme-level error highlighting

- Progress tracking and summary dashboard with statistics and error analysis
- Educational content pages for IPA and English phonology

Backend: The TanStack Start backend provides server functions for API endpoints and business logic. Authentication is handled through Clerk (free tier), supporting email/password and social sign-in options (Google, GitHub). Authorization is implemented using user metadata stored in Clerk, enabling role-based access control. The application uses TanStack Query for efficient data fetching, caching, and state management.

Database: The complete database schema is implemented using Drizzle ORM with Neon serverless PostgreSQL. All planned tables are operational, including:

- `practice_texts` - Practice materials with difficulty and category classification
- `authors` - Reference voice/author management with ElevenLabs integration
- `reference_speeches` - Target pronunciation audio files with IPA transcriptions
- `user_recordings` - User-submitted audio recordings with metadata
- `audio_quality_metrics` - Quality assessment results (SNR, silence ratio, clipping ratio)
- `analyses` - Pronunciation assessment results with phoneme and word-level scores
- `alignments` - Phone-level time alignment data (TextGrid files)
- `phoneme_errors` and `word_errors` - Normalized error data for aggregation
- `jobs` and `ipa_generation_jobs` - Asynchronous job tracking for RunPod endpoints
- `user_preferences` - User settings and preferences

Storage: Audio files are stored in Cloudflare R2 object storage, with processed audio files (16-bit, 16kHz, mono WAV) maintained for long-term access. Metadata is stored in the PostgreSQL database with proper indexing for efficient queries.

Admin Features: Administrative interfaces are implemented for content management:

- Practice text CRUD operations with bulk import capabilities

- Reference speech management with audio upload and IPA generation
- Author/voice management with ElevenLabs API integration for TTS generation
- IPA generation job monitoring and management

Reference speech generation currently supports US English with two different speaker configurations via ElevenLabs API, providing variety in target pronunciations for practice materials.

7.1.2 Machine Learning Services

The machine learning pipeline is implemented as separate RunPod serverless endpoints in the `mod/` directory:

Assessment Endpoint: The pronunciation assessment service (`mod/assessment/`) integrates POWSM [11] models for phonetic transcription:

- Phone Recognition (PR): Extracts IPA phonemes from user audio using POWSM PR model (average execution time: 3 seconds)
- Grapheme-to-Phoneme (G2P): Generates target IPA from text using POWSM G2P model with audio-guided transcription (average execution time: 2 seconds)
- Edit Distance Calculation: Compares actual vs. target phonemes to identify substitution, insertion, and deletion errors
- MFA Alignment: Integrates Montreal Forced Aligner for phone-level timestamp generation
- Score Calculation: Computes pronunciation accuracy scores based on error rates

The endpoint is deployed on RunPod serverless GPU infrastructure, utilizing network volumes for model caching to minimize cold-start delays. Maximum observed cold-start time is 30 seconds, with subsequent requests benefiting from cached models.

IPA Generation Endpoint: The IPA generation service (`mod/ipa_generation/`) provides phonetic transcription for reference speeches, supporting both text-only and audio-guided G2P modes.

Both endpoints are containerized with Docker and deployed on RunPod, utilizing GPU resources for efficient model inference. The services handle audio download, preprocessing, model inference, and result formatting.

7.1.3 Signal Processing and Experiments

Signal processing experiments and deep learning model evaluations are conducted in the `sig/exp/` directory [9], which contains Jupyter notebooks for:

- Audio preprocessing and quality assessment (SNR, silence detection, clipping detection)
- POWSM [11] model evaluation and fine-tuning experiments
- MFA alignment testing and TextGrid parsing
- G2P accuracy evaluation and comparison
- Visualization of phonetic features and alignment results

These notebooks document the experimentation process and serve as reference implementations for the production pipeline.

7.1.4 Documentation and Deployment

The repository includes comprehensive documentation in the `doc/` directory, with learning materials, technical notes, database schema documentation, and project reports. The main README file [7] provides an overview of the project structure, setup instructions, and development guidelines. Environment variables and configuration are documented to ensure reproducible deployments across different environments [5].

The report build process is automated through `doc/report/build.sh`, which handles LaTeX compilation, bibliography processing, and diagram generation from PlantUML source files. The script integrates with `generate-diagrams.sh` to automatically generate all architecture and design diagrams before PDF compilation, ensuring documentation stays synchronized with code changes.

7.2 Deployment

Nounce is containerized using Docker, ensuring consistency across development, staging, and production environments. The deployment process is automated through GitHub Actions CI/CD pipelines [6], which handle building Docker images, running linting, running tests, and generating PlantUML diagrams from source files.

7.2.1 Production Deployment

The production application is deployed on Fly.io and served at `https://nounce.pro`. DNS management is handled through Cloudflare, which also provides DDoS protection via its global network infrastructure. The database is hosted on Neon serverless PostgreSQL, offering automatic scaling, point-in-time recovery, and branching capabilities. Audio files are stored in Cloudflare R2 object storage.

Machine learning services are deployed as RunPod serverless GPU endpoints, utilizing network volumes for model caching to reduce cold-start times. The deployment architecture is illustrated in Figure 6.4.

7.2.2 Local Development

For local development, a custom RunPod serverless simulator replicates the production API behavior without requiring cloud GPU resources. The simulator architecture, illustrated in Figure 6.5, consists of a FastAPI proxy server that mimics RunPod's job queue API, worker containers running locally via Docker Compose, and background async loops for job processing. The simulator is orchestrated using `docker-compose.dev.yml` and can be started using the `scripts/runpod.py` utility script.

The Docker configuration includes environment variable management and health checks for container monitoring. Building instructions, environment setup, and deployment procedures are documented in the repository README [7], with the build process automated through `doc/report/build.sh` script that handles LaTeX compilation and diagram generation.

Chapter 8

RESULTS

This chapter presents the results achieved in the development of Nounce, including technical achievements, system capabilities, and implementation status.

8.1 System Implementation Status

The Nounce pronunciation assessment system has been successfully implemented with the following components operational:

8.1.1 Full-Stack Web Application

The complete web application is functional and deployed, providing users with a comprehensive pronunciation assessment platform. The system successfully implements:

- **User Authentication:** Secure authentication through Clerk with support for email/password and social sign-in (Google, GitHub)
- **Practice Text Management:** Full CRUD operations for practice texts with filtering by difficulty and category
- **Audio Recording:** Browser-based recording with real-time waveform visualization and audio upload support
- **Reference Speech System:** Management of target pronunciations with TTS integration (ElevenLabs) and native speaker uploads
- **Pronunciation Assessment:** End-to-end workflow from recording to detailed analysis results

- **Progress Tracking:** User history, statistics, and error analysis with aggregation capabilities
- **Admin Interface:** Complete administrative tools for content and system management

8.1.2 Database Schema

The complete database schema is implemented and operational, with all planned tables, relationships, and indexes in place. The schema supports:

- User recordings with metadata and quality metrics
- Pronunciation analyses with phoneme and word-level scoring
- Error tracking with normalized data for aggregation
- Asynchronous job processing with status tracking
- Reference speech management with IPA transcriptions
- User preferences and progress data

8.1.3 Machine Learning Pipeline

The machine learning assessment pipeline is fully integrated:

- **POWSM Integration:** Successfully integrated POWSM [11] models for Phone Recognition (PR) and Grapheme-to-Phoneme (G2P) tasks
- **Phonetic Transcription:** Accurate IPA transcription from audio using state-of-the-art models
- **Error Detection:** Phoneme-level error classification (substitution, insertion, deletion) with edit distance algorithms
- **Time Alignment:** MFA integration for phone-level timestamp generation
- **Score Calculation:** Pronunciation accuracy scoring based on error rates

The assessment endpoint processes audio recordings and returns detailed analysis including:

- Actual IPA transcription from user speech

- Target IPA transcription from reference text
- Overall pronunciation score (0.0-1.0)
- Phoneme-level error list with timestamps
- Word-level error analysis

8.1.4 Audio Quality Assessment

Audio quality metrics are calculated and stored for each user recording:

- Signal-to-Noise Ratio (SNR) in decibels
- Silence ratio (percentage of quiet segments)
- Clipping ratio (percentage of clipped samples)
- Noise ratio (percentage of noise segments)
- Quality status classification (accept, warning, reject)

Quality thresholds are applied at the application layer, allowing flexible adjustment without database migrations.

8.1.5 Deployment Infrastructure

The system is deployed using modern cloud infrastructure:

- **Domain:** The application is served at `https://nounce.pro`, with the domain purchased and configured for production use
- **DNS and Security:** DNS management and DDoS protection are handled through Cloudflare's global network infrastructure
- **Application Hosting:** Docker containerized application deployed on Fly.io with global edge deployment
- **Database:** Neon serverless PostgreSQL with automatic scaling and backups
- **Object Storage:** Cloudflare R2 for audio file storage
- **ML Services:** RunPod serverless endpoints for GPU-powered inference
- **CI/CD:** GitHub Actions pipelines for automated testing and deployment

8.2 Technical Achievements

8.2.1 Architecture and Design

The system demonstrates a well-architected full-stack application with:

- Clear separation of concerns between frontend, backend, database, and ML services
- Type-safe database operations using Drizzle ORM with TypeScript
- Modern React patterns with server-side rendering and efficient data fetching
- Scalable serverless architecture supporting concurrent users
- Comprehensive error handling and user feedback mechanisms

8.2.2 User Experience

The application provides an intuitive and responsive user experience:

- Modern, accessible UI following web standards and best practices
- Real-time feedback during recording and processing
- Clear visualization of pronunciation errors with IPA highlighting
- Progress tracking with historical data and statistics
- Mobile-responsive design supporting multiple devices

8.2.3 Integration Challenges and Solutions

Several technical challenges were successfully addressed during implementation:

Model Integration: Integrating POWSM [11] models required extensive experimentation due to limited documentation. The solution involved custom implementation of model loading, inference pipelines, and output parsing.

Audio Processing: Handling various audio formats and quality levels required robust preprocessing pipelines. The system implements format conversion, quality assessment, and normalization to ensure consistent input for ML models.

Asynchronous Processing: Managing long-running ML inference tasks required a job queue system. The implementation uses RunPod's serverless architecture with status tracking and webhook callbacks for result delivery.

Time Alignment: Integrating MFA for phone-level timestamps required careful handling of transcription formats and TextGrid parsing. The system supports both MFA-based alignment and fallback timestamp estimation.

8.3 Performance and Scalability

The system architecture supports scalability and demonstrates good performance characteristics:

- Serverless database (Neon) with automatic scaling
- Object storage (Cloudflare R2) for efficient audio file management
- RunPod GPU endpoints for parallel ML inference with network volume caching
- Job queue system for handling concurrent assessment requests
- Efficient database indexing for fast queries

Performance Metrics:

- Average Phone Recognition (PR) task execution time: 3 seconds
- Average Grapheme-to-Phoneme (G2P) task execution time: 2 seconds
- Maximum observed cold-start time: 30 seconds (first request after inactivity)
- Subsequent requests benefit from cached models in network volumes, eliminating cold-start delays

These performance characteristics enable responsive user experience while maintaining cost efficiency through serverless GPU infrastructure that scales to zero when not in use.

Chapter 9

CONCLUSION

This project successfully developed Nounce, a comprehensive web-based pronunciation assessment system for English language learners. The system integrates state-of-the-art machine learning models, modern web technologies, and pedagogical principles to provide detailed, actionable feedback on pronunciation accuracy.

9.1 Project Summary

Nounce addresses the critical need for accessible pronunciation learning tools by providing:

- **Phonetic-Level Analysis:** Detailed pronunciation assessment at the phoneme level using the International Phonetic Alphabet (IPA)
- **Automated Feedback:** Instant, judgment-free feedback on pronunciation errors with clear visualization
- **Educational Integration:** Guidance on English phonology concepts to help learners understand and improve
- **Progress Tracking:** Historical analysis and statistics to monitor improvement over time
- **Accessible Design:** Web-based platform accessible from any device without requiring native speaker tutors

9.2 Technical Contributions

The project demonstrates several technical achievements:

Model Integration: Successfully integrated POWSM [11] (Phonetic Open Whisper-Style Speech Model), a state-of-the-art phonetic transcription model, for accurate phone recognition and grapheme-to-phoneme conversion. The integration required custom implementation due to limited documentation, contributing to the open-source community's understanding of these models.

Full-Stack Architecture: Developed a production-ready full-stack application using modern technologies (TanStack Start, React, TypeScript, Drizzle ORM) with a focus on type safety, scalability, and maintainability. The architecture demonstrates best practices in separation of concerns, error handling, and user experience design.

ML Pipeline: Implemented a complete machine learning pipeline combining phonetic transcription, forced alignment (MFA), and error detection algorithms. The pipeline processes audio recordings asynchronously and provides detailed analysis results with phone-level timestamps.

Database Design: Designed and implemented a comprehensive database schema supporting user recordings, analyses, error tracking, and progress data. The schema enables efficient querying and aggregation for analytics and reporting.

9.3 Educational Impact

Nounce contributes to language learning by:

- Providing 24/7 access to pronunciation assessment without requiring human tutors
- Offering detailed, phonetic-level feedback that helps learners understand specific sound production issues
- Creating a safe, judgment-free environment for practice
- Supporting self-directed learning with progress tracking and historical analysis
- Introducing learners to IPA and phonetic concepts, building domain knowledge

The system acknowledges that it does not replace human-to-human interaction in language learning, but rather serves as a supplementary tool for learners who may lack access to native speakers or formal language education resources.

9.4 Conclusion and Future Work

9.4.1 State of the Platform

All core features are functional, and the application is deployed. The system provides a complete pronunciation assessment workflow from recording to detailed feedback, with administrative tools for content management. Additional content management and data curation is still needed for future expansion.

9.4.2 Limitations

All core features are functional, and the application is deployed. Additional content management and data curation is still needed for future expansion to support a broader range of practice materials and user populations.

The POWSM [11] model operates in a hybrid space between phonemic and phonetic representation, which can introduce occasional inaccuracies. The model's output may not always perfectly align with canonical IPA transcriptions, particularly for non-standard pronunciations or regional variations.

Nounce is designed exclusively for English to streamline initial development and ensure high-quality pronunciation assessment. Future development can expand support to additional languages, but this requires significant model adaptation, dictionary resources, and acoustic model training for each target language.

9.4.3 Future Directions

Future Directions for Turkish-Native Speakers: Fine-tuning the model to capture the nuances of people who use Turkish as their native language would improve accuracy for the target user population. This would involve collecting and curating training data from Turkish L1 speakers and adapting the model to better recognize common pronunciation patterns and errors specific to this population.

Language Expansion: Nounce is designed exclusively for English to streamline initial development and ensure high-quality pronunciation assessment. Future development can expand support to additional languages, but this requires significant model adaptation, dictionary resources, and acoustic model training for each target language.

Voice Cloning: Leverage voice cloning technology to generate target pronunciations with the user's voice for personalized feedback. This would allow learners to hear correct pronunciations in their own voice, potentially improving comprehension and retention.

LLM Analysis: Integrate a Large Language Model (LLM) to analyze performance data and provide user-specific, targeted feedback on each attempt, aggregated over time. The LLM could identify patterns in errors, suggest personalized practice strategies, and provide contextual explanations for pronunciation challenges.

9.4.4 Ethical Considerations

Accent Variations & Linguistic Diversity: All dialects, accents, and language variants are linguistically valid. The system focuses on US English pronunciation as a widely recognized standard, but this choice does not minimize or devalue other accents. The primary goal is comprehensibility and effective communication rather than accent elimination. Users are encouraged to understand that accent reduction is optional and that clear communication is the objective.

User Autonomy & Privacy: Users control their learning process. Recordings are processed securely, and the platform prioritizes privacy in data handling and storage. Users must explicitly opt-in for any potential use of their data in model fine-tuning, voice cloning, or pronunciation generation features. Clear consent mechanisms and transparent data usage policies protect user privacy and maintain trust.

9.5 Conclusion

Nounce successfully demonstrates the practical application of state-of-the-art speech recognition models in an educational context. The system provides a functional, scalable platform for pronunciation assessment that combines technical excellence with pedagogical considerations. The project contributes to both the academic community through its reference implementation and to language learners through its accessible, detailed feedback system.

The development process highlighted the importance of iterative experimentation, careful architecture design, and user-centered development. The resulting system serves as a foundation for future enhancements and research in automated pronunciation assessment and language learning technology.

Bibliography

[1] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[2] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

[3] ELSA Corp. Elsa speak: Ai-powered english pronunciation coach. `https://elsaspeak.com/en/`, 2024. Accessed: November 2025.

[4] Evleksiz, Ümit Can. Application source code directory. `https://github.com/umitcan07/senior/tree/main/app`, 2024. Full-stack application codebase. Accessed: November 2025.

[5] Evleksiz, Ümit Can. Environment configuration. `https://github.com/umitcan07/senior/tree/main/app`, 2024. Environment variables and configuration files. Accessed: November 2025.

[6] Evleksiz, Ümit Can. Github actions ci/cd workflows. `https://github.com/umitcan07/senior/tree/main/.github/workflows`, 2024. Continuous integration and deployment pipelines. Accessed: November 2025.

[7] Evleksiz, Ümit Can. Project readme. `https://github.com/umitcan07/senior/blob/main/README.md`, 2024. Project documentation and setup instructions. Accessed: November 2025.

[8] Evleksiz, Ümit Can. Senior project repository. `https://github.com/umitcan07/senior/`, 2024. Boğaziçi University Computer Engineering Senior Project. Accessed: November 2025.

[9] Evleksiz, Ümit Can. Signal processing experiments directory. `https://github.com/umitcan07/senior/tree/main/sig/exp`, 2024. Jupyter notebooks for DSP and deep learning experiments. Accessed: November 2025.

[10] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the International Conference on Machine Learning*, pages 369–376, 2006.

[11] Chin-Jou Li, Kalvin Chang, Shikhar Bharadwaj, Eunjung Yeo, Kwanghee Choi, Jian Zhu, David Mortensen, and Shinji Watanabe. Powsm: A phonetic open whisper-style speech foundation model. 2025.

[12] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. *Proceedings of Interspeech*, pages 498–502, 2017.

[13] Pronounce Inc. Pronounce ai: Ai-powered speech feedback and pronunciation practice. `https://www.getpronounce.com/`, 2024. Accessed: November 2025.

[14] Silke M. Witt and Steve J. Young. Phone-level pronunciation scoring and assessment for interactive language learning. *Speech Communication*, 30(2-3):95–108, 2000.