

spanish-voice-trainer

Project Summary:

Custom Spanish Voice TrainerA high-performance, completely private desktop application built on macOS to accelerate Spanish language training through automated flashcard creation and intelligent pronunciation analysis.The application utilizes a local-first architecture to ensure complete data privacy, storing all configurations, historical student analytics, and multimedia binary files strictly on the user's local drive.

The Core Technical Stack

- Environment & Package Management: uv (Rust-based Python package manager) for ultra-fast, isolated virtual environments and dependency locking.
- Database & Persistence: SQLite to map phrase metadata, local media file paths, and chronological user practice scores without external database infrastructure.
- Audio & Signal Processing: sounddevice for hands-free, voice-activated microphone capture; librosa and fastdtw (Dynamic Time Warping) to extract phoneme features (MFCCs) and score user pronunciation accuracy against reference files.
- Asset Generation: edge-tts to stream high-quality, neural text-to-speech Spanish audio clips, and Pillow (PIL) to auto-render widescreen flashcard JPEGs matching the phrases.
- User Interface: Developed in two phases—starting as a clean, text-based Command Line Interface (CLI) before migrating to a dual-mode desktop GUI built with PyQt6.

How the System Works

The application operates across two distinct, integrated operational frameworks managed via a unified QStackedWidget interface:

1. Content Creator Mode The user inputs a Spanish phrase and its English translation. The system automatically triggers the asset generator to output a custom high-quality flashcard image and a native-sounding neural audio file. The localized text strings and file paths are instantly committed to the SQLite database.
2. Student Training ModeThe system pulls cards from the database, displays the visual flashcard image, and plays the target Spanish pronunciation. The student speaks into their MacBook microphone. The system detects when the student begins and stops talking via a voice-activation threshold, records the sample, and runs a Dynamic Time Warping alignment algorithm to provide an objective pronunciation match score (e.g., 87% accuracy).
3.  Long-Term Capability: Custom Video Compilations Because all image assets share standard HD video dimensions (1280 X 720) and all audio samples are tracked deterministically in the database, the core engine can be commanded to interface with ffmpeg-python. It can seamlessly compile entire batches of database assets into standalone, continuous .mp4 video lessons complete with timed visual pauses and silent audio gaps, providing an additional passive learning medium for language immersion.

Create Basic Structure as suggested

```

spanish-voice-trainer/
├── .gitignore
├── pyproject.toml
├── uv.lock
├── README.md
├── main.py                # Entry point for the application
├── database/              # Storage & Data Layer
│   ├── __init__.py
│   └── connection.py      # SQLite schema initialization and CRUD
├── queries
│   ├── core/              # Pure Python Business Logic (The Engine)
│   │   ├── __init__.py
│   │   └── audio_engine.py # Recording (sounddevice) and scoring
│   │   (librosa/DTW)
│   └── asset_generator.py # TTS generation (edge-tts) and flashcard
│       creation (Pillow)
├── ui/                    # Presentation Layer (Views)
│   ├── __init__.py
│   ├── cli/
│   │   └── interface.py   # Simple command-line menus for Phase 1
│   └── gui/
│       ├── interface.py   # Main QMainWindow shell for Phase 2
│       ├── creator_mode.py # QWidget for content creation panel
│       └── trainer_mode.py # QWidget for student practice panel + QThread
├── workers
│   └── media/              # Local storage for physical binary files
│       └── .gitkeep        # Keeps directory alive in Forgejo repo

```

Commands

```

stephenlohning@Scotty 139_spanish-voice-trainer % mkdir doc
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir doc/images
stephenlohning@Scotty 139_spanish-voice-trainer % touch .gitignore
stephenlohning@Scotty 139_spanish-voice-trainer % git branch -M main
stephenlohning@Scotty 139_spanish-voice-trainer % git remote -v
origin  https://dev.oxnee.com/stephen/139_spanish-voice-trainer.git (fetch)
origin  https://dev.oxnee.com/stephen/139_spanish-voice-trainer.git (push)
stephenlohning@Scotty 139_spanish-voice-trainer % git config --global
push.followTags true
stephenlohning@Scotty 139_spanish-voice-trainer % touch main.py
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir database
stephenlohning@Scotty 139_spanish-voice-trainer % touch
database/__init__.py
stephenlohning@Scotty 139_spanish-voice-trainer % touch
database/connection.py
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir core
stephenlohning@Scotty 139_spanish-voice-trainer % touch core/__init__.py

```

```
stephenlohning@Scotty 139_spanish-voice-trainer % touch
core/audio_engine.py
stephenlohning@Scotty 139_spanish-voice-trainer % touch
core/asset_generation.py
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir ui
stephenlohning@Scotty 139_spanish-voice-trainer % touch ui/__init__.py
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir ui/cli
stephenlohning@Scotty 139_spanish-voice-trainer % touch ui/cli/interface.py
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir ui/gui
stephenlohning@Scotty 139_spanish-voice-trainer % touch ui/gui/interface.py
stephenlohning@Scotty 139_spanish-voice-trainer % touch
ui/gui/create_mode.py
stephenlohning@Scotty 139_spanish-voice-trainer % touch
ui/gui/trainer_mode.py
stephenlohning@Scotty 139_spanish-voice-trainer % mkdir media
stephenlohning@Scotty 139_spanish-voice-trainer % touch media/.gitkeep
```

use uv

```
stephenlohning@Scotty 139_spanish-voice-trainer % uv venv
Using CPython 3.13.5
Creating virtual environment at: .venv
Activate with: source .venv/bin/activate
stephenlohning@Scotty 139_spanish-voice-trainer %
```

Install Your Dependency Stack

Run the uv add commands to populate your environment lockfile:

```
uv add sounddevice numpy scipy librosa fastdtw pillow edge-tts pyqt6
```

checking the packages are installed

```
(139_spanish-voice-trainer) stephenlohning@Scotty 139_spanish-voice-trainer
% uv pip list
Package            Version
-----
aiohappyeyeballs   2.6.2
aiohttp            3.14.0
aiosignal          1.4.0
attrs              26.1.0
audioop-lts        0.2.2
audioread          3.1.0
```

certifi	2026.5.20
cffi	2.0.0
charset-normalizer	3.4.7
decorator	5.3.1
edge-tts	7.2.8
fastdtw	0.3.4
frozenlist	1.8.0
idna	3.18
joblib	1.5.3
lazy-loader	0.5
librosa	0.11.0
llvmlite	0.47.0
msgpack	1.1.2
multidict	6.7.1
narwhals	2.22.1
numba	0.65.1
numpy	2.4.6
packaging	26.2
pillow	12.2.0
platformdirs	4.10.0
pooch	1.9.0
propcache	0.5.2
pycparser	3.0
pyqt6	6.11.0
pyqt6-qt6	6.11.1
pyqt6-sip	13.11.1
requests	2.34.2
scikit-learn	1.9.0
scipy	1.17.1
sounddevice	0.5.5
soundfile	0.14.0
soxr	1.1.0
standard-aifc	3.13.0
standard-chunk	3.13.0
standard-sunau	3.13.0
tabulate	0.10.0
threadpoolctl	3.6.0
typing-extensions	4.15.0
urllib3	2.7.0
yaml	1.24.2

Running Scripts with uv Moving Forward

One of the cleanest features of uv is that you don't necessarily have to remember to run `source .venv/bin/activate` every single time you open a new terminal window to write code.

Instead, you can prefix your execution scripts using `uv run`. It automatically detects your local `.venv`, verifies that all dependencies in your lockfile are fully up to date, and runs the script seamlessly:

```
# Example: To execute your main script once you write it later today  
uv run main.py
```

this is the basic
