AURA 기술 스택 합의서 V2.0 (Project Trinity)

작성일: 2025년 12월 25일

합의 참여: Sir Marlon (CEO) + Gemini (CTO) + Claude (CPO)

1. 아키텍처 방향 (전면 수정)

항목	기존 (V1 - 폐기)	변경 (V2 - 신규 확정)
플랫폼	웹 브라우저 (Chrome 등)	**Electron** 데스크탑 앱 **(Win/Mac)**
핵심 가치	접근성 (설치 없음)	성능 **&** 호환성 **(VST** 사용**)**
오디오 엔진	Tone.js (Web Audio API)	**Python + Pedalboard (JUCE** 기반**)**
VST 지원	❌ 포기 (Smart Knob 대체)	✅ 완벽 지원 **(VST3 / AU)**
타겟 유저	아마추어, 취미생	기존 장비**(VST)**를 가진 프로

2. 기술 스택 상세 (Tech Stack)

2.1 Application Shell (껍데기)

- 프레임워크**: Electron** (크로스 플랫폼 앱 패키징)
- **UI: React 18 + TypeScript** (기존 웹 기술 재사용)
- 디자인**: Tailwind CSS** ("Neon Noir" 테마 적용)
- 상태 관리**: Zustand** (가볍고 빠름)

2.2 Audio Engine (핵심 엔진) - 가장 큰 변화

- 언어**: Python 3.10+** (C++ 대신 사용)
- 코어 라이브러리**: Pedalboard (by Spotify)**
    - 선정 이유*:* 내부적으로 JUCE(C++)를 사용하여 VST 호스팅과 오디오 처리를 담당하되, 코딩은 파이썬으로 쉽게 가능함.

- 오디오 **I/O:** sounddevice (ASIO/CoreAudio 드라이버 접근용)
- **MIDI I/O:** mido, python-rtmidi (마스터 건반 연동)

2.3 AI Engine (On-Device)

- 구동 방식**:** 로컬 **GPU** 가속 (서버비 절감 + 보안)
- 프레임워크**: PyTorch / ONNX Runtime**
- 주요 모델**:**
    - RVC: 내 목소리 학습/변환 (Voice Cloning)
    - Demucs: 스템 분리 (MR 제거)
    - Generative MIDI: 오디오가 아닌 'MIDI 노트' 생성 모델

2.4 통신 **(Frontend ↔ Backend)**

- 프로토콜**: Socket.IO** (localhost 내부 통신)
- 구조**:**
    - React(UI)가 버튼을 누르면 → Socket.IO로 신호 전송 → Python(엔진)이 소리 재생

3. 데이터 및 저장소

- 프로젝트 저장**: SQLite** (로컬 파일 DB, .aura 파일 포맷)
- 사용자 인증**: Firebase / Supabase** (구독 관리용)

4. 개발 우선순위 **(Roadmap)**

- **Phase 1 (MVP): "**소리가 나는가**?"**
    - Electron + React UI 띄우기
    - Python으로 VST(.dll) 로딩해서 소리 내보기
- **Phase 2 (Hybrid UI):** 3-Way(비트/편곡/세션) 뷰 구현
- **Phase 3 (AI):** 내 목소리 학습 기능 탑재