ChatGPT

AI 기반 TTS 방송 서비스 – 제품 요구사항서 (PRD) 및 저수준 설계서 (LLD)

1. 제품 개요 (Overview)

제품명: (가칭) **스마트 TTS 방송 시스템** – 웹 기반의 AI 음성 합성(Text-to-Speech) 및 오디오 방송 관리 서비스. 회원 가입을 통해 사용자는 텍스트를 자연스러운 음성으로 변환하고, 생성된 음원을 다운로드하거나 **전관방송** 등으로 송출할 수 있습니다. 주요 활용 예로는 관공서에서 일정 시간마다 자동 안내방송 송출, 기업/시설의 정기 방송, 일반 사용자의 개인 방송/공지 등이 있습니다. 이 서비스는 최신 AI 음성합성 기술과 음성 클로닝(Voice Cloning)을 접목하여, 사용자 맞춤형 음성을 손쉽게 생성하고 **스케줄에 따라 자동 방송**하는 것을 목표로 합니다.

주요 기능 요약: 회원 관리, 텍스트→음성 생성(TTS), 보이스 클로닝(사용자 음성 업로드로 AI 음성 생성), 생성된 음원의 다운로드, **실시간 음원 송출**, **예약 송출**(시간대별/요일별/기간별 자동 방송), 복수 음원 **믹싱**, 이력 관리(생성/방송 내역 저장 및 재사용), 외부 API 연동(수퍼톤 Supertone API를 통한 음성합성 및 보이스 클로닝, Lemon Squeezy를 통한 과금/결제).

기술 스택 개요: 프론트엔드는 **React** 기반의 웹 애플리케이션으로 직관적이고 고급스러운 오디오 인터페이스를 제공합니다. 백엔드는 **RESTful API**와 데이터베이스로 구성되며, 수퍼톤 API 및 결제 API와 연동됩니다. 필요 시 로컬 PC에서의 원활한 방송 송출을 위해 **로컬 에이전트** 또는 OS 스케줄러 활용을 고려합니다. 서비스 전반에서 **클라우드 기반 아키텍처**를 사용하여 다수 사용자 및 관공서 고객을 동시에 지원할 수 있도록 설계합니다.

2. 주요 목표 및 목적 (Goals)

- **자동화된 고품질 방송:** 사용자가 사람이 직접 녹음하지 않아도 AI TTS를 활용해 **높은 품질의 음성 안내방송**을 제작하고 송출할 수 있게 합니다. 다양한 상황(정시 알림, 공지 방송, 홍보 멘트 등)에 맞는 음성 합성을 지원하고, **정해진 일정에 자동으로 방송 송출**하여 업무 효율을 높입니다.
- **개인화된 음성 및 콘텐츠:** 보이스 클로닝을 통해 기관장, 시장, 도지사 등 **특정 인물의 목소리**로 환영사나 안내 말씀을 생성할 수 있습니다. 이를 통해 청취자에게 더욱 친근하고 **개인화된 콘텐츠**를 제공하는 것이 가능합니다.
- **사용자 친화적 경험:** 비개발자나 일반 직원도 쉽게 사용할 수 있도록 **웹 기반의 직관적 UI**를 제공합니다. 오디오 플레이어 및 편집기와 유사한 인터페이스로 **음성 생성부터 스케줄 설정까지 원스톱으로 처리**할 수 있게 하고, 복잡한 기술적 요소는 숨겨져 있습니다.
- **안정성과 신뢰성:** 중요한 안내방송에 사용되는 서비스이므로 **예약된 방송의 정확한 송출**이 최우선입니다. PC가 절전 모드일 경우에도 **자동으로 깨워 방송을 재생**하며, 만약 일시적인 오류나 실패가 발생하면 **재시도 및 백업** 메커니즘을 통해 누락 없이 방송됩니다. 모든 방송 이력과 로그를 저장하여 문제 발생 시 검증할 수 있게 합니다.
- **유연한 과금 및 운영:** 다양한 **요금제**(월 정액, 사용량 기반 등)를 지원하고, 관공서/기업 vs 일반 사용자 등 **고객 유형별 맞춤 요금**을 적용합니다. Lemon Squeezy 결제 연동으로 **구독 결제**와 **사용량 기반 청구**(분당 사용 등)를 자동화하고, 관리자에게는 각 고객별 요금 설정과 사용 현황을 관리할 수 있는 도구를 제공합니다 [1]. 또한 서비스 운영자는 수퍼톤 API 이용에 필요한 **크레딧 관리**를 효율화하여 비용을 통제합니다 (예: 수퍼톤 API의 분당 과금 약 $0.1/분 기준을 고려한 가격 책정 [2]).

3. 대상 사용자 및 주요 사용 사례 (Target Users & Use Cases)

- **관공서 및 공공기관:** 도청, 시청, 구청 등의 행정기관에서 청사 내 **전관 방송**(예: 정시 타종, 민방위 훈련 안내, 공지사항 방송 등)에 활용합니다. 예를 들어 시청에서는 매일 오전 개청 안내 멘트를 시장님의 목소리로 송출하거나, 정해진 시간마다 자동으로 민방위 훈련 안내방송을 내보낼 수 있습니다. 또한 도서관, 박물관, 우체국 등 공공시설의 정기 안내방송에도 활용이 가능합니다.
- **기업 및 상업 시설:** 대형 마트, 쇼핑몰, 사옥 등에서 **정기적인 안내방송**(영업 시작/종료 안내, 이벤트 홍보 방송)을 자동화합니다. 사전에 등록된 멘트를 고품질 AI 음성으로 생성해 두고, 매일 정해진 시간에 자동 재생하여 직원의 반복 업무를 줄입니다. 긴급 재난 방송이나 공지사항도 신속히 입력하여 즉시 TTS 변환 후 방송할 수 있습니다.
- **교육 기관:** 학교나 캠퍼스에서 **종례 방송**, **시간 알림 방송**, **행사 안내** 등을 TTS로 대체할 수 있습니다. 특히 여러 언어로 안내가 필요할 경우 (예: 국제 학생 대상 영어 안내) 다국어 TTS 기능을 활용해 한국어뿐 아니라 영어/일본어 등으로도 방송을 내보낼 수 있습니다 [3].
- **일반 개인 사용자:** 1인 미디어나 콘텐츠 크리에이터가 **오디오 콘텐츠 제작**에 사용할 수 있습니다. 예를 들어 유튜버나 팟캐스터가 손쉽게 AI 보이스로 나레이션을 만들어 영상에 삽입하거나, 가정에서 스마트 스피커를 통한 일정 안내 방송을 예약해두는 용도로도 활용 가능합니다. 또한 시각장애인 등 **접근성 보조 도구**로서 텍스트 알림을 음성으로 자동 출력하는 기능 등도 응용될 수 있습니다.
- **특수 사례:** 해당 서비스는 **멀티테넌트 SaaS**로 설계되어 여러 기관/고객이 각각 전용으로 쓸 수 있으므로, 예를 들어 방송국이나 교통 안내 시스템 등에서도 API 연계를 통해 실시간 TTS 안내 방송을 송출하는 등 확장이 가능합니다. (초기 버전에서는 주로 웹 UI 사용에 초점을 두지만, 장기적으로 API 이용도 지원 가능)

4. 기능 요구사항 (Functional Requirements)

4.1 회원가입 및 사용자 관리

- **회원 가입 & 로그인:** 사용자는 이메일 및 비밀번호 또는 OAuth를 통해 계정을 생성하고 로그인할 수 있어야 합니다. 필요 시 관리자 승인이 있는 **기업/기관용 계정**과 자유 가입 가능한 **일반 계정**을 구분합니다. 회원 정보에는 이름, 소속(기관명), 연락처, **요금제/권한 정보** 등이 포함됩니다.
- **권한 수준:** 일반 사용자(Role: User)는 자신의 음성 생성/방송/결제 내역만 관리할 수 있으며, 관리자(Role: Admin)는 전체 사용자와 시스템 설정(음원DB, 요금정책 등)을 관리할 수 있습니다. 관공서 등 **조직별로 여러 계정**이 있을 경우 조직 관리자(Organization Admin) 개념을 두어 해당 조직(예: OO시청)에 속한 방송 콘텐츠를 공동 관리하도록 할 수 있습니다 (이 기능은 추후 확장 요소).
- **프로필 및 결제 정보 관리:** 사용자는 서비스 내에서 자신의 **구독 상태**(요금제 종류, 남은 사용량 등)를 조회할 수 있고, 결제 수단 관리(카드 정보 갱신 등)나 영수증 확인도 가능해야 합니다. Lemon Squeezy와 연동하여 구독 갱신, 취소, 업그레이드 등이 가능하며, **결제 포털**을 제공하여 사용자가 직접 요금제를 변경하거나 구독을 관리할 수 있습니다.
- **사용자별 환경설정:** UI 언어(초기에는 한국어, 장기적으로 다국어 UI 가능), TTS 음성 기본값, 예약방송 타임존 설정, 슬립 모드 해제 옵션 등 사용 환경을 개별 설정할 수 있습니다. 예를 들어 관공서 계정은 PC 슬립모드 해제를 허용하고, 일반 사용자는 해당 옵션을 숨기는 등 차별화합니다.

4.2 TTS 음성 생성 (Text-to-Speech Generation)

- **텍스트 입력 및 언어 지원:** 사용자는 방송하고자 하는 **문구(Text)**를 입력하여 해당 텍스트의 음성합성을 요청할 수 있습니다. 기본 언어는 **한국어**이며, 영어 및 일본어 텍스트도 지원합니다 [4]. 실제 Supertone API의 제한에 따라 **한글, 영어, 일본어만 지원**되므로 그 외 언어(예: 중국어 등)를 입력할 경우 경고 또는 결과 품질 저하 안내를 합니다 [4]. 한국어 문장 내에 **숫자나 외래어**가 포함될 수 있으므로, 서비스는 숫자의 적절한 읽기(예: "125" → "백이십오")나 외래어 발음 향상을 위한 처리를 할 수 있습니다 (필요 시 사용자에게 발음 수정 옵션 또는 SSML 태그 입력 기능 제공 검토). 입력 텍스트 길이는 엔진 제한에 따라 **최대 300자**로 제한하며 [3], 너무 긴 문장은 여러 개로 분할하도록 유도합니다.
- **음성 합성 엔진 선택:** 수퍼톤(Supertone)사의 AI Voice 모델을 기본 엔진으로 사용합니다. 사용자는 **목소리(Voice ID)**를 선택할 수 있어야 합니다. 기본 제공되는 다수의 **프리셋 보이스(Preset Voice)** 라이브러리에서

원하는 음색을 고를 수 있고, 사용자가 생성한 **커스텀 보이스(Custom Voice)**도 자신의 계정에서 선택 가능합니다. 예를 들어 남성 중저음 보이스, 여성 밝은 톤 보이스 등 카테고리별로 필터링하여 찾을 수 있습니다 (Supertone API의 **GET /v1/voices** 및 **검색 API**를 사용하여 언어=ko, 성별=male 등의 조건으로 목록 제공 [5] [6]). 각 보이스마다 지원하는 **언어와 스타일**이 다를 수 있으므로, UI에서 보이스 선택 시 해당 보이스가 한국어를 지원하는지, 감정 스타일(예: neutral, happy 등) 선택이 가능한지 보여줍니다 [7] [8].

- **발음 스타일 및 속성:** 기본적으로 **감정 스타일**은 "차분함/중립(neutral)"로 합성하지만, 지원되는 보이스의 경우 기쁨(happy), 격앙됨(excited) 등 **스타일**을 적용할 수 있습니다. 또한 **고급 설정**으로 음성의 **톤/피치**와 **속도**를 조정하는 옵션을 제공하여 사용자 요구에 맞게 음색을 튜닝할 수 있습니다. (예: pitch_shift ±12半音, 말하기 속도 0.5배~2배 등 [9]). 이러한 voice settings 옵션은 기본값(원음 톤, 보통 말하기 속도)을 제공하고, 고급 사용자만 조정하도록 UI에서 기본은 숨겨 두되 "고급 설정 펼치기" 등을 통해 접근할 수 있게 합니다 [10].
- **TTS 요청 및 응답 처리:** 사용자가 텍스트와 설정을 확정하면 **"음성 생성"** 버튼을 눌러 TTS를 요청합니다. 백엔드는 Supertone API의 **POST /v1/text-to-speech/{voice_id}** 엔드포인트를 호출하여 음성 합성을 수행하며, 요청에는 필요한 헤더와 파라미터(text, language, style 등)를 포함합니다 [11] [12]. 합성 성공 시 **오디오 파일(wav 또는 mp3)**이 응답 스트림으로 반환되며 [13], 백엔드는 이를 수신하여 파일로 저장하고, DB에 생성 기록을 남깁니다. 또한 API 응답 헤더의 `X-Audio-Length` 정보를 통해 합성된 음원의 **재생 길이**(초)를 확인하여 DB에 저장해둡니다 [14]. 만약 합성 요청이 실패하면(예: 잘못된 voice_id, 잔여 크레딧 부족 등) 오류를 사용자에게 알려주고 재시도 또는 설정 변경을 안내합니다 (이 때 Supertone API의 오류코드 400/401/402 등을 매핑하여 친숙한 에러메시지 출력 [15]).
- **생성 결과 관리:** 음성 생성이 완료되면 사용자에게 **오디오 미리듣기 플레이어**와 **다운로드 링크**를 제공합니다. 사용자는 브라우저 상에서 즉시 합성된 음성을 들어보고 만족 여부를 확인할 수 있습니다. 필요에 따라 "재합성" 기능으로 문구 수정이나 설정 변경 후 다시 생성할 수 있습니다 (이전 버전 음원은 히스토리에 남기되, 최신 버전만 사용할 수 있도록 UI 표기). 사용자가 만족하면 해당 음원을 **저장**하여 추후 방송에 사용할 수 있는데, 저장 시 제목/태그를 붙여 관리할 수 있습니다 (예: "월요일 오전 조회 방송멘트"). 생성된 음원 파일은 사용자의 전용 **라이브러리**에 목록화되어, 차후 재방송하거나 다운로드할 수 있습니다.

4.3 음성 클로닝 (Voice Cloning)

- **보이스 클로닝 개요:** 본 서비스는 **AI 보이스 클로닝** 기능을 통해 사용자가 업로드한 실제 음성 녹음 파일을 학습하여, 그 화자의 목소리를 닮은 **맞춤형 TTS 보이스**를 생성해줍니다. 예를 들어 도지사의 육성 녹음 파일을 업로드하면, 이를 바탕으로 도지사님의 음색으로 안내멘트를 합성할 수 있는 AI 보이스를 만들 수 있습니다. 이 기능은 Supertone API의 **Custom Voice** 생성 기능을 활용합니다 (2025년 7월 업데이트로 API를 통한 voice cloning 등록 지원 [16]).
- **음성 샘플 업로드:** 사용자는 클로닝할 **목표 화자의 음성 샘플 파일**(WAV 또는 MP3)을 업로드합니다. UI에서 업로드 시 파일 용량과 포맷 제한을 안내합니다. **제한:** 수퍼톤 API 정책에 따라 파일당 **최대 3MB 이하**의 녹음만 허용되며 [17], 형식은 WAV 또는 MP3여야 합니다 [18]. 3MB는 대략 수십 초~1분 분량의 음성에 해당하므로, 가능하면 **한 화자의 명료한 음성으로 30초 이상** 길이를 권장합니다. 파일 업로드 시 해당 화자의 **이름과 설명**을 입력하게 합니다 (예: 이름: "시장님 인사말", 설명: "OO시장 취임사 음성 클론"). 이름은 100자까지 입력 가능하며 초과 시 잘라냅니다 [17].
- **클로닝 생성 요청:** 사용자가 파일과 이름을 제출하면 백엔드는 Supertone API의 **POST /v1/custom-voices/cloned-voice** 엔드포인트를 호출해 **보이스 클론 생성**을 요청합니다. 이 때 요청은 **멀티파트 폼 데이터**로 전송되며, 업로드된 음성 파일(binary)과 지정한 name/description이 포함됩니다 [19] [20]. API 호출 전후처리로, 파일 크기 확인 및 형식 검증을 수행하고, 업로드 시간을 단축하기 위해 파일을 백엔드 서버에 잠시 저장하거나 스트림으로 바로 전송합니다.
- **처리 시간과 결과:** 보이스 클로닝은 합성보다 시간이 더 걸릴 수 있으므로, 요청 후 **수초~수분의 지연**이 예상됩니다. API가 즉시 voice_id를 응답하지 않고 비동기로 처리할 가능성이 있다면(예: 작업 큐에 들어가는 형태), 백엔드는 폴링 또는 웹훅 등을 통해 결과를 확인해야 합니다. 수퍼톤 API가 요청 성공 시 바로 **voice_id**를 응답한다면 [21] [22] 이를 받아서 DB에 해당 사용자의 새로운 Custom Voice로 저장합니다. 이 voice_id는 이후 TTS 생성 시 `voice_id`로 활용 가능하며, UI 상의 보이스 목록에 **사용자 보이스**로 표시됩니다. (API 응답 예시: `{"voice_id": "voice_123456789"}` 형태 [21])
- **클로닝 사용 및 제한:** 새로 등록된 **커스텀 보이스**는 해당 사용자(또는 조직)의 계정 내에서만 사용 가능하도록 설정합니다. 사용자는 TTS 생성 화면에서 보이스 선택 시 자기 계정의 Custom Voices를 볼 수 있고, 이를 선택

해 텍스트를 합성하면 해당 목소리로 TTS 결과를 얻게 됩니다. 주의: 수퍼톤 정책상 **무료 플랜 사용자에게는 API를 통한 보이스 클로닝 등록 기능이 제공되지 않으므로**, 본 서비스에서도 **유료 플랜 가입자만 이 기능을 활성화**합니다 [23] . 또한 클로닝 음성의 품질은 업로드한 샘플 음성의 음질과 발음에 영향을 받으므로, 사용자가 양질의 데이터를 제공하도록 가이드합니다. 만약 클로닝 생성에 실패하거나 품질이 낮을 경우, 추가 녹음 업로드를 요청하거나 포기할 수 있도록 안내합니다.

- **윤리적/법적 고려:** 타인의 목소리를 무단으로 클로닝하는 것은 **법적 문제**가 될 수 있으므로, 서비스 이용 약관에 사용자 본인 혹은 적법한 권리를 가진 음성만 업로드하도록 명시합니다. 관공서의 경우 공무원이나 기관장의 음성을 사용할 때 **당사자 동의**를 받도록 하고, 업로드 시 이에 대한 확인 절차(체크박스 동의 등)를 거칩니다. 생성된 클론 음성은 오용되지 않도록 해당 사용자 계정에만 국한되고, 서비스 제공자는 음성 데이터 보호에 유의합니다.

4.4 음원 저장 및 다운로드

- **오디오 파일 관리:** TTS 생성이나 업로드/믹싱 등을 통해 얻은 모든 **오디오 파일(음원)**은 사용자별 라이브러리에 저장됩니다. 파일 정보에는 제목, 생성일시, 길이, 사용된 보이스/설정(메타데이터), 소유자 ID 등이 기록됩니다. 사용자는 웹 UI에서 자신의 음원 리스트를 확인하고, 정렬/검색(제목, 생성일 등) 기능을 통해 원하는 음원을 찾을 수 있습니다.
- **파일 형식 및 품질:** 기본 파일 형식은 고음질 **WAV**(PCM 16-bit, 22kHz 이상)로 생성하며, 사용자 요청 시 **MP3**로 변환하여 제공할 수 있습니다 (Supertone API의 output_format 파라미터 지원 [24]). WAV는 방송 시스템에 적합하도록 무손실 고품질을 보장합니다. 단, 다운로드나 저장 용량을 고려해 기본적으로 MP3 192kbps 등으로 변환 저장할 수도 있으나, 초기 버전에서는 WAV 우선으로 하고 필요 시 변환 옵션을 제공합니다.
- **다운로드 기능:** 사용자는 자신의 음원을 개별적으로 **다운로드**할 수 있습니다. 다운로드 링크를 클릭하면 해당 오디오 파일을 브라우저로 내려받습니다. 관공서 등의 폐쇄망 환경에서는 다운로드한 파일을 별도 장치에 옮겨 사용하는 경우도 있을 수 있으므로, 다운로드 권한은 사용자에게 주되 **외부 유출**에 대한 책임은 사용자에게 있음을 알립니다. 또한 한꺼번에 여러 파일을 받고자 하는 경우를 위해 **ZIP 묶음 다운로드**(여러 음원 선택 후 다운로드) 기능도 고려합니다.
- **용량 관리:** 각 사용자별로 저장 가능한 총 용량 또는 파일 수에 제한을 둘 수 있습니다 (요금제에 따라 다르게 설정). 예를 들어 일반 무료 사용자는 100MB까지만 저장 가능, 유료는 1GB 등. 용량이 초과될 경우 오래된 파일을 지우거나 상위 플랜으로 업그레이드하도록 안내합니다. 관리자 화면에서 사용자별 **스토리지 사용량**을 모니터링하며 필요시 조정할 수 있습니다.
- **저장 및 보안:** 저장된 음원 파일은 서버의 파일 스토리지 또는 클라우드 스토리지(S3 등)에 보관되며, **파일명/경로는 접근 제어**가 된 형태로 관리합니다. 사용자의 다운로드 요청 시 백엔드가 인증을 거쳐 파일을 제공하거나, 일시 유효한 URL을 발행하여 접근하도록 합니다. 또한 중요한 안내방송 음원의 무결성을 위해 해시 검증 또는 백업 정책을 둘 수 있습니다. (예: 관공서 VIP 음성 등은 별도 백업)

4.5 즉시 음원 송출 (실시간 방송)

- **수동 방송 송출:** 사용자가 생성한 음원을 즉시 해당 컴퓨터의 **스피커 또는 연결된 방송 장비**로 재생하여 송출할 수 있습니다. 웹 UI 상에서 각 음원 항목에 **"방송하기"** 버튼을 제공하여, 클릭 시 해당 음원을 선택 출력 장치로 재생합니다. 예를 들어 긴급 안내멘트를 작성하고 바로 방송하기를 누르면 곧바로 건물 내 스피커로 소리가 나도록 합니다. 이 기능은 주로 관제PC(방송 장비에 연결된 PC)에서 사용될 것으로 예상합니다.
- **오디오 출력 경로 설정:** 관공서/기업의 방송 장비는 보통 PC의 **오디오 출력(jack 또는 mixer 연결)**과 연동됩니다. 사용자는 서비스 설정에서 **출력 장치**를 선택할 수 있어야 합니다 (예: PC의 사운드카드 출력, USB 오디오 인터페이스 등). 브라우저만으로는 장치 제어에 한계가 있으므로, 로컬 클라이언트 애플리케이션 또는 브라우저 확장 기능을 통해 특정 출력으로 강제 재생하는 방안을 고려합니다. 1차적으로는 PC 기본 오디오 출력으로 소리가 나오도록 하고, 시스템 설정에서 해당 출력이 방송망에 연결되도록 환경 세팅을 안내합니다.
- **즉시 방송 피드백:** "방송하기"를 수행하면 UI에 현재 **방송 중**임을 표시하고, 재생 진행바를 보여줍니다. 방송 완료 후에는 "방송 완료" 메시지를 띄워 사용자에게 확인시킵니다. 또한 이 **방송 실행 이벤트를 로그**로 기록하여, 나중에 언제 어떤 메시지가 수동 송출되었는지 조회 가능하게 합니다.
- **긴급 방송 모드:** 혹시 긴급 상황(화재 경보 등) 시 즉시 방송이 필요할 경우를 대비해, **UI 어디서나 접근 가능한 빠른 방송 패널**을 제공할 수 있습니다. 예를 들어 상단바에 "★긴급 방송★" 버튼을 두고, 누르면 바로 텍스트

입력→TTS→즉시 방송까지 한 번에 수행하는 모드를 띄웁니다. 이때는 기본 보이스 (또는 지정된 경보음 톤)로 신속히 합성합니다. 이 기능은 추후 추가 고려 사항으로 둡니다.

4.6 예약 방송 (Scheduled Broadcasting)

- **일정 기반 자동 방송:** 본 서비스의 핵심 기능으로, **특정 시간에 미리 지정한 음원을 자동으로 재생**하는 **예약 스케줄링**을 제공합니다. 사용자는 캘린더 UI 또는 리스트 UI에서 새 **예약 이벤트**를 만들 수 있습니다. 예약 생성 시 필요한 정보:
- 방송할 **음원 선택:** 이미 생성/저장된 TTS 음원이나 업로드된 파일 중에서 선택. (또는 새 텍스트 입력 즉시 TTS 생성 후 그 결과를 사용할 수도 있음)
- **방송 시간** 설정: 정확한 날짜와 시간을 지정하거나 **반복 규칙**을 설정. 예: 2025-11-01 09:00 1회성, 매주 월~금 09:00 반복, 매월 1일 12:00, 특정 기간 매일 15:00 등. UI에서 단일 날짜/시간 선택 뿐 아니라 **반복 (Recurring)** 옵션을 제공하여 일간/주간/월간 반복 스케줄을 간편히 설정하도록 합니다. (반복 규칙은 cron 표현식 또는 라이브러리를 활용하여 구현)
- **유효 기간:** (선택 사항) 이 일정이 실행될 기간 범위. 예: "12월 한달간만 실행" – 시작일/종료일 설정. 없으면 무기한 반복.
- **방송 제목/설명:** 해당 예약의 라벨 (예: "점심 방송 음악", "주간 안내사항 방송")을 붙여 관리.
- **스케줄 UI/UX:** 예약된 항목들은 **리스트**와 **캘린더** 두 가지 형태로 볼 수 있습니다. 리스트에서는 시간순으로 다음 방송 예정들을 보여주고, 캘린더/타임라인 뷰에서는 하루/주간 단위로 예약이 배치된 모습을 시각적으로 표시합니다. (React용 스케줄러 컴포넌트 활용 검토 [25]). 사용자는 예약 항목을 클릭하여 편집 또는 삭제할 수 있습니다.
- **예약 방송 실행:** 예약 시각이 되면 시스템은 해당 음원을 **자동으로 재생/송출**합니다. **핵심 요구사항:** PC가 **절전 모드(Sleep)**나 대기 상태에 있어도 방송 시각 전에 **자동으로 깨워(Wake up)** 음원을 출력해야 합니다. 이를 위해 두 가지 방안을 고려합니다:
- **OS 스케줄러 활용:** 윈도우 작업 스케줄러 또는 BIOS 알람 등 **시스템 예약 기능**을 이용해 PC를 깨우고, 백그라운드에서 재생 프로그램을 실행하도록 설정. 예를 들어 예약 시각 1분 전에 PC 깨우기 이벤트를 등록하고, 시간 도래 시 지정된 음원을 플레이하는 스크립트를 OS 레벨에서 실행합니다. 이 경우 본 서비스는 예약 설정 시 OS 작업 스케줄에 해당 예약 정보를 기록해두고, 적절한 권한으로 Wake-on timers를 설정해야 합니다.
- **상주 애플리케이션(에이전트):** PC에 **백그라운드 에이전트 프로그램**을 설치/실행하여, 이 프로그램이 클라우드 서버로부터 스케줄을 미리 가져와 모니터링하다가 시간에 맞춰 PC 깨우기(Wake on LAN 등) 및 오디오 재생을 수행합니다. 에이전트는 시스템 트레이에 상주하면서 슬립 모드를 제어할 권한을 가질 수 있습니다. 클라우드와 통신하여 최신 일정을 동기화하고, 이벤트 시 OS 깨우기 API를 호출합니다.
- **재생 신뢰성 및 모니터링:** 예약된 시간에 방송 재생을 시도했으나 만약 **오류**가 발생한 경우 (예: PC 음향 장치 문제, 일시적 파일 접근 에러 등), **재시도 메커니즘**이 필요합니다. 첫 시도 실패 시 수 초 간격으로 여러 번 재시도하거나, 에러를 로그하고 **대체 경로**를 취합니다. 예를 들어 3번 시도 모두 실패하면 관리자에게 경고를 표시하거나 이메일/문자 알림을 보내도록 합니다. 또한 예약 방송의 성공/실패 여부와 실제 재생된 시간을 기록하여, 이력이 남도록 합니다.
- **예약 관리:** 사용자는 예약된 항목을 일시적으로 **비활성화/일시중지**할 수 있습니다 (enable/disable 토글). 이를 통해 당일에만 방송을 건너뛰거나 할 수 있습니다. 또한 이미 지난 예약이라도 해당 설정을 보존하여 다시 활성화하거나, 과거 실행 내용을 확인할 수 있게 합니다.
- **겹치는 스케줄 처리:** 동일한 시각에 두 개 이상의 방송이 예약된 경우 우선순위에 따라 처리합니다. 기본적으로는 중복을 막기 위해 UI 단계에서 경고/방지하지만, 만일 발생 시 시작 시간이 동일한 두 예약이 있다면 우선 만들어진 순서나 우선순위 필드에 따라 하나를 먼저 틀고 다른 하나는 대기 또는 즉시 다음 순서에 틀도록 합니다. (관공서 사용시 일반적으로 겹치지 않게 설정할 것으로 예상)
- **슬립 모드 복귀:** 방송 후 PC를 다시 절전 상태로 둘지 여부를 옵션으로 제공합니다. 예를 들어 야간에만 잠깐 깨워 방송했다가 다시 슬립하도록 할 수 있습니다. 이는 에이전트/OS 설정으로 구현 가능하며, 관리자 설정으로 전체 시스템에 적용하거나 개별 스케줄에 설정하도록 합니다.
- **인터넷 연결 요구:** 예약 방송 시에도 TTS 음원은 **사전에 생성**되어 로컬에 존재하므로, 실제 재생 시에는 인터넷 연결이 불안정해도 이미 저장된 파일을 출력할 수 있습니다. 단, 클라우드 서버와 동기화가 필요한 부분(예: 새로운 일정 변경 전파 등)은 오프라인 시 동기화가 안될 수 있으므로, 가능한 **로컬 캐싱**을 활용하여 네트워크가 없어도 직전의 스케줄을 수행하도록 합니다.

4.7 음원 믹싱 기능 (Audio Mixing)

- **믹싱 개요:** 보다 풍부한 방송 콘텐츠를 위해 두 개 이상의 오디오를 **합성/믹싱**하여 하나의 음원으로 출력하는 기능을 제공합니다. 예를 들어 **배경음악 BGM과 TTS 음성**을 섞어서 보다 전달력 있는 안내방송을 만들 수 있습니다. 또는 여러 개의 안내멘트를 순차적으로 이어붙이는 **플레이리스트** 개념으로 활용할 수도 있습니다.
- **믹싱 사용 방법:** UI에서 믹싱 작업을 위한 전용 **에디터 화면**을 제공합니다. 사용자는 자신의 라이브러리에서 **여러 음원 선택** 후 "믹스/편집" 기능을 클릭하면 편집 UI가 열립니다. 이 UI에서는 **타임라인 형식**으로 각 음원의 파형을 시각화하고 Drag & Drop으로 순서나 겹치는 부분을 조절할 수 있습니다. 예를 들어 10초짜리 배경음악을 깔고, 2초 시점부터 TTS 음성이 나오도록 겹쳐 배치할 수 있습니다. 또한 개별 트랙의 **볼륨 조절**(BGM은 살짝 낮추고, 음성은 크게)을 설정할 수 있습니다.
- **자동 믹싱 옵션:** 사용자가 직접 편집하지 않더라도 간단히 **BGM 추가** 기능을 통해 TTS 음성에 자동으로 배경음악을 깔 수 있습니다. 몇 가지 제공되는 배경음악(저작권 프리 음악) 중 선택하거나, 사용자가 업로드한 음악 파일을 배경으로 지정하면, 시스템이 미리 정해진 볼륨 비율로 두 트랙을 합성합니다. (예: BGM -15dB, Voice -0dB로 믹스)
- **믹싱 처리:** 편집 완료 후 "믹싱 저장"을 누르면 백엔드에서 **오디오 프로세싱**을 수행합니다. 다중 트랙을 합성해야 하므로, 서버 측에서는 **FFmpeg** 등의 라이브러리를 사용해 지정된 타임라인대로 오디오를 믹스 다운합니다. 이 과정에서 모든 입력 음원의 샘플레이트를 통일하고, 볼륨/팬 조절 등을 적용하여 하나의 WAV/MP3 파일을 생성합니다. 생성된 믹스 음원은 새로운 파일로서 사용자 라이브러리에 저장되고 이후 일반 음원처럼 다운로드, 방송 예약 등에 사용될 수 있습니다.
- **성능 고려:** 오디오 믹싱은 긴 파일의 경우 시간이 걸릴 수 있으므로, 작업 진행 중 **로딩 인디케이터** 또는 "서버에서 처리 중" 메시지를 보여줍니다. 믹싱 완료 시까지 UI에서 대기하거나, 완료 후 알림(이메일 또는 앱 내 알림)을 주는 방식을 고려합니다. 대기 시간이 길 경우 백엔드 작업 큐로 처리하고, 완료 시 결과를 DB에 기록해두는 방식으로 구현합니다.
- **사용 시나리오:** 이 기능으로 사용자는 예를 들어 **차임벨 소리**와 안내 멘트를 결합한 파일을 만들어, 정각 방송시 맨 앞에 차임음을 넣을 수 있습니다. 혹은 여러 언어 안내를 하나로 묶어 순차 방송되게 파일을 편집할 수도 있습니다. 이렇게 편집된 결과물은 하나의 예약 이벤트로 송출되므로, 복잡한 방송도 자동화할 수 있습니다.
- **UI 단순화 모드:** 일반 사용자에게는 전문 오디오 편집까지 필요하지 않을 수 있으므로, 기본 인터페이스에서는 단순히 "배경음 넣기" 정도만 노출하고, 상세 편집 모드는 "고급 편집" 버튼을 통해서만 접근하도록 UX를 설계합니다.

4.8 이력 관리 및 분석 (History & Analytics)

- **생성 이력:** 사용자가 합성한 모든 TTS 음성 및 클로닝/믹싱 결과에 대한 **이력(Logs)**을 저장합니다. 이력 항목에는 생성일시, 텍스트 내용(또는 작업 종류), 사용한 목소리 및 언어, 생성된 파일명, 소요 크레딧 등이 포함됩니다. 사용자는 자신의 생성 이력을 리스트로 볼 수 있으며, 필요한 경우 특정 이력의 **다시 생성**(동일한 파라미터로 재합성) 기능을 제공할 수 있습니다. 또한 생성 이력은 과금과도 연관되므로, **분당 생성시간 합계** 등을 집계하여 요금 산정의 근거로 삼습니다.
- **방송 이력:** 예약된 방송 및 수동 방송의 **실행 이력**도 관리합니다. 이력에는 방송한 음원, 실제 재생 일시 (예약 대비 지연이 있었는지), 성공 여부, 실행한 장치/사용자 정보 등이 포함됩니다. 예를 들어 "2025-10-29 09:00 시장님_인사말.mp3 예약방송 – 성공 (09:00:02 재생 시작, 00:00:30 길이)" 식으로 기록합니다. 만약 실패나 취소가 있었으면 그 사유도 기록합니다. 이러한 방송 로그는 나중에 **감사 또는 보고용**으로 출력할 수 있게 해야 합니다. (관공서의 경우 방송 내역을 월간 보고할 필요가 있을 수 있습니다)
- **로그 조회 및 필터:** 사용자 인터페이스에서 생성/방송 이력을 날짜별, 유형별로 필터링하고 검색할 수 있습니다. 예를 들어 10월 한달간 몇 건의 방송이 있었는지, 어떤 내용이 주로 방송되었는지 통계를 볼 수 있습니다. 관리자라면 전체 사용자들의 이력을 종합해볼 수도 있습니다.
- **데이터 재사용:** 저장된 **이력 데이터로부터 편리 기능**을 제공합니다. 예를 들어 과거에 사용한 방송 멘트를 쉽게 **재사용(복사)**하여 새로운 예약을 만들 수 있게 합니다. 사용자 히스토리에서 특정 항목을 선택해 "새 예약에 사용" 버튼을 누르면 해당 음원과 내용을 불러와 새로운 일정에 할당할 수 있습니다. 이를 통해 반복 행사를 매년 생성할 때 등 편의성을 높입니다.
- **분석 및 대시보드:** 향후 고도화 단계로, **방송 활용 분석**을 제공할 수 있습니다. 예를 들어 시간대별 방송 횟수, 가장 많이 쓰인 음성 유형, 월별 TTS 사용량 등의 지표를 시각화한 **대시보드**를 관리자나 사용자에게 제공합니

다. Lemon Squeezy를 통한 **과금 분석** (예: 이달의 사용요금 추산)도 UI에 표시하여, 사용자들이 자신의 사용 패턴을 알고 효율적으로 요금제를 선택하도록 유도합니다.

4.9 관리자 기능 (Admin & Billing Management)

- **요금제 및 과금 관리:** 관리자(Admin)는 서비스의 **요금 정책**을 설정하고 각 사용자에게 플랜을 할당/조정할 수 있어야 합니다. UI에서 다양한 요금 옵션을 입력 가능하도록 설계합니다:
- 플랜 이름, 유형(월 구독 vs 종량제 등), 기본 제공내역(예: 월 100분 TTS 포함), 초과 단가(예: 분당 ₩X), 저장공간 한도, 보이스 클로닝 가능 여부, 가격 등. 복수의 플랜을 만들고 수정/삭제할 수 있습니다.
- 특정 고객에게 **맞춤 플랜** 적용: 관공서 고객 A에게만 월 정액 ₩x만원에 음원 무제한 등의 **커스텀 계약**이 가능하도록, 개별 사용자/조직 프로필에 개별 요금 설정 override 기능을 둡니다.
- 과금 방식: Lemon Squeezy의 **구독(Subscription)** 상품과 연동하여, 정액제의 경우 월 단위 결제가 자동 청구되고, **사용량 기반(Usage-based)** 요금의 경우 매 결제주기마다 사용량에 따라 금액이 계산되도록 합니다 [26] [27]. Lemon Squeezy의 Usage-Based Billing을 활용하면 고객이 **사용한 만큼만 정확히 결제**하도록 할 수 있습니다 [1]. 예를 들어 일반 사용자의 경우 "기본 요금 + 초과 사용량" 모델로 설정하고, 매월 말 초과분 분당 과금액을 Lemon Squeezy API로 업데이트하여 다음 달 청구에 반영합니다.
- **결제 및 구독 관리:** Lemon Squeezy와의 연동으로, 관리자는 **고객의 구독 현황**(활성/취소 여부, 다음 결제일 등)을 조회할 수 있습니다. 또한 특정 고객의 구독을 강제로 업그레이드/다운그레이드하거나 해지 처리할 수 있는 권한도 고려합니다 (이 경우 LS API로 해당 subscription을 PATCH 또는 DELETE 호출 [28] [29]). 새로운 상품을 출시하거나 가격을 변경하면 LS 대시보드에서 설정한 뒤, 서비스 DB에도 해당 정보를 업데이트/동기화하여 일관성 유지합니다. (필요 시 LS Webhook을 받아 우리 DB의 플랜 상태 업데이트)
- **사용량 모니터링:** 관리자 화면에서 **전체 시스템 사용량**을 모니터링합니다. 여기에는 일별 총 TTS 호출 수, 총 음성 합성 시간(분), 보이스 클로닝 시도 횟수, 저장공간 사용량 등이 포함됩니다. 이 수치는 운영상 중요한 **수퍼톤 API 크레딧** 소비와 직결되므로, 관리자는 수퍼톤 콘솔의 크레딧 잔액이나 사용 추이를 볼 수 있는 링크 또는 정보도 확인할 수 있습니다. (예: 수퍼톤 **GET /v1/credits** API를 주기적으로 호출하여 크레딧 잔여량을 가져와 표시 [30]).
- **콘텐츠 관리:** 서비스에서 제공하는 **기본 음성 라이브러리**, **배경음악 목록** 등을 관리할 수 있습니다. 예를 들어 금지어 필터(방송 부적절한 단어) 목록을 설정하여 해당 단어가 포함된 TTS 요청을 차단하거나, 기본 제공 BGM을 추가/교체하는 등의 콘텐츠 업데이트 작업이 필요합니다. 이러한 관리 기능도 웹 UI로 제공합니다.
- **로그 및 감사:** 시스템 로그(오류 로그, 사용자 활동 로그 등)를 조회하는 관리자 기능도 둡니다. 특히 보이스 클로닝이나 음원 생성 같은 민감 기능의 사용 내역을 추적하여 악용을 방지합니다. 관리자용으로 **특정 사용자의 모든 활동 열람**이나 **로그 내보내기** 기능을 두어, 필요시 감사 자료로 활용할 수 있게 합니다.

5. UI/UX 요구사항 (User Interface & Experience)

- **직관성과 전문성의 균형:** UI는 **오디오 장비** 또는 **DAW(Digital Audio Workstation)**를 연상시키는 깔끔하고 세련된 디자인을 채택합니다. 다만 전문 툴처럼 복잡해 보이지 않도록 **필수 기능만 노출**하는 간결함을 유지합니다. 예를 들어 배경은 어두운 계열 테마에 파형(waveform) 시각화 등을 활용하여 **고급스러운 오디오 비주얼**을 주고, 버튼이나 아이콘은 직관적으로 이해되도록 디자인합니다 (재생 ▶, 정지 ■, 마이크 아이콘 등 친숙한 오디오 관련 아이콘 사용).
- **반응형 웹과 접근성:** React 기반 SPA(Single Page Application)로 구현하며, **반응형 디자인**을 통해 다양한 화면 (방송실 PC의 큰 모니터, 노트북, 태블릿 등)에서 레이아웃이 최적화됩니다. 주요 기능들은 사이드바 내비게이션으로 묶고, 작업 영역은 넓게 확보하여 파형이나 일정표를 보기 쉽게 합니다. 폰트는 가독성이 높고 한국어 최적화된 것을 사용합니다. 또한 시각 장애인 등의 접근성을 위해 스크린리더 호환도 염두에 둡니다 (적절한 ARIA 레이블 등).
- **오디오 미리보기 및 시각화:** 음원 생성 후 **미리듣기 플레이어**에는 **파형 시각화**나 **재생 진행바**를 제공하여 사용자에게 청각적인 결과를 시각적으로도 확인할 수 있게 합니다. 재생 중 일시정지/재생, 시크바 이동 등의 조작을 지원하고, 볼륨 조절도 가능하게 합니다. 이러한 **오디오 플레이어 컴포넌트**는 서비스 전반 (미리듣기, 믹싱 미리보기 등)에서 일관되게 사용됩니다.
- **일정/캘린더 UI:** 예약 방송 관리 화면에서는 **캘린더**와 **타임라인** UI를 강조합니다. 예를 들어 주간 뷰에서 매일 9시에 이벤트가 있으면 해당 칸에 아이콘과 제목이 표시되고, 드래그하여 시간 변경도 가능하도록 할 수 있습니

다. 마우스 오버 시 세부 정보 팝업(어떤 음원, 길이 등)을 보여주는 등 사용자 친숙성을 높입니다. 반복 예약 설정은 다소 복잡할 수 있으므로, “매주 월~금”, “매월 첫째주 월요일” 등 자연어 스타일의 옵션 선택지를 제공하여 쉽게 선택하게 합니다.

- **모던한 비주얼 요소:** 전체적인 색상과 그래픽은 **전문 오디오 장비**나 **미디 인터페이스** 느낌을 살립니다. 가능하다면 서비스 로고나 로딩 애니메이션에 **사운드파형 모티프**를 활용해 브랜드 아이덴티티를 부여합니다. 버튼 hover 효과나 전환 애니메이션은 부드럽게 처리하여 고급스럽게 보이도록 하고, 특히 중요한 작업 (예: “방송 송출 중”)에서는 눈에 띄는 강조색(예: 빨간 불빛 같은 효과)을 사용합니다.
- **언어 및 용어:** UI 언어는 기본 **한국어**로 제공되며, TTS 도메인에 익숙하지 않은 사용자도 이해할 수 있게 **쉬운 용어**를 사용합니다. 예: Text-to-Speech → “문자음성 변환”, Voice Cloning → “보이스 클로닝(목소리 복제)”, Schedule → “예약 방송”, Output Device → “출력 장치” 등으로 안내합니다. 전문용어에는 툴팁을 달아 자세한 설명을 제공하고, 사용자 가이드 문서나 FAQ에 UI 사용법을 정리합니다.
- **오류/예외 상황 UX:** 만약 TTS 합성 실패, 네트워크 오류, 예약 충돌 등의 예외 상황이 발생하면 **사용자 친화적인 메시지**로 안내합니다. 예를 들어 “음성 생성에 실패했습니다 - 인터넷 연결을 확인해주세요” 또는 “남은 크레딧이 부족하여 음성을 생성할 수 없습니다 [31] ” 등의 메시지를 표시하고, 대응 행동(재시도, 결제 페이지 이동 등)을 제안합니다. 치명적 오류 시 고객지원에 연락하도록 정보도 함께 제공합니다.
- **튜토리얼 및 도움말:** 초심자들을 위해 첫 사용 시 **온보딩 튜토리얼**(슬라이드 또는 안내 메시지)을 제공하여, 주요 기능 (TTS 생성, 예약 설정 등)을 차례로 체험하도록 유도합니다. 또한 화면 내 **도움말 아이콘(?)**을 눌러 수퍼톤 API 기반 기능 설명이나 용어 정의를 볼 수 있게 합니다. (예: 클로닝 기능 옆에 “i” 아이콘 -> “보이스 클로닝이란? 업로드한 음성으로 AI가 학습해 닮은 목소리를 생성하는 기능입니다...” 설명 표시)

6. 비기능 요구사항 (Non-Functional Requirements)

- **성능 및 확장성:**
- TTS 합성 요청 시 **대기 시간**이 중요합니다. 일반적인 한 문장(1~2초 음성) 합성에 1~3초 이내 응답을 목표로 최적화합니다. 수퍼톤 API는 클라우드 기반으로 비교적 빠른 응답을 제공하지만, 동시에 다수 요청이 들어올 경우 지연될 수 있으므로, 서버는 요청을 **큐**에 넣고 순차 처리하거나, 수퍼톤 API의 **요청 수 제한** (기본 권장 1분당 20건)을 준수하도록 합니다 [32] . 향후 사용자가 증가하면 수퍼톤과 협의해 API 한도를 높이거나, 서버가 요청을 분산 호출하는 로드밸런싱을 고려합니다.
- 다수 관공서/기업 고객을 수용할 수 있도록 **멀티테넌시** 구조로 DB를 설계하고, 각종 쿼리나 배치작업이 고객별 격리되어 동작하도록 합니다. 또한 AWS 등의 클라우드 인프라 상에서 서버 인스턴스를 쉽게 수평 확장(autoscaling)할 수 있게 해 두어, 사용자 급증 시 성능을 유지합니다.
- 오디오 파일 저장소는 CDN 연계를 통해 다운로드/스트리밍 시 **대역폭**을 충분히 확보합니다. 특히 동시에 여러 스케줄 방송이 다른 고객사에서 진행될 경우 서버 부하 없이 각자의 로컬에서 재생되도록, 필요하면 **음원 미리 캐싱**을 권장합니다 (예: 예약시간 5분 전에 미리 음원을 로컬로 가져다 놓기).
- **신뢰성 및 가용성:**
- 이 서비스는 **24/7 지속 동작**이 필수적입니다. 관공서 방송의 특성상 새벽이나 주말에도 일정대로 방송이 나가야 하므로, 서버 측은 **고가용성(HA)**으로 구성하고 장애 시 자동전환(failover)이 가능해야 합니다. 데이터베이스도 이중화하여 장애에 대비합니다.
- 로컬 에이전트 또는 OS 스케줄링 기능은 인터넷 장애 시에도 로컬에 캐싱된 일정과 음원으로 **오프라인 상태에서도 예약 방송을 실행**할 수 있어야 합니다. 네트워크 단절 시에도 최소 N일치 예약을 수행하도록 디자인합니다.
- 방송 송출의 신뢰성을 높이기 위해 **이중화 방안**을 고려합니다. 예를 들어 동일 음원을 2회 재생(티거 음성 두 번)하거나, TTS 음성과 별도로 준비된 비상 대비 멘트를 두어 예기치 않은 상황에 대비합니다. (이 부분은 운영 가이드 차원)
- **보안:**
- 사용자의 음원 데이터와 개인정보를 안전하게 보호하기 위해 **전송 구간 암호화(HTTPS)**와 **저장 데이터 암호화**(적용 가능한 부분에)를 수행합니다.
- 중요한 API 키(수퍼톤, Lemon Squeezy 등)는 백엔드 서버에 안전하게 보관하고 노출되지 않도록 합니다.
- 사용자 인증 세션은 적절한 토큰 방식(JWT 혹은 세션 쿠키)으로 관리하고, **다중인증(MFA)** 옵션을 추후 제공할 수 있습니다 (관공서 보안 요구사항 대응).

- 음성 합성 결과나 클로닝 보이스에 대해 **접근 제어**를 철저히 하여, 타인이 임의로 URL을 알아도 다운로드할 수 없게 만듭니다.
- 또한 클로닝 기능 등 악용 소지가 있는 부분에 대해 모니터링을 실시하고, **비정상적 사용 패턴 탐지** 시 관리자가 조치할 수 있도록 합니다. (예: 한 사용자가 하루 수백 건의 클로닝 시도 -> 계정 일시 정지 등)
- **호환성:**
- 웹앱은 **크로스브라우저** 호환성을 가져야 합니다. 최신 크롬/엣지/파이어폭스는 기본, 가능하면 관공서에서 아직 사용하는 IE모드도 대응(폴리필)하거나, 최소한 크롬 기반으로 사용하도록 안내합니다.
- 로컬 에이전트는 Windows 10/11을 우선 지원하고, 필요시 Linux (우분투 등) 혹은 macOS용도 개발하여 다양한 환경에서 동작할 수 있게 합니다. 특히 학교 등에서 라즈베리파이 등의 소형 PC로 방송 제어를 할 수 있는 가능성도 염두에 둡니다.
- **유지보수 및 로깅:**
- 체계적인 **로깅과 모니터링** 시스템을 구축하여, TTS API 응답 시간, 에러 발생률, 예약 실행 성공률 등을 지속적으로 수집합니다. 이를 통해 성능 저하나 문제 발생 시 신속히 인지하고 조치합니다. 예를 들어 Sentry 등을 붙여 클라이언트 오류도 수집, 서버측은 Kibana 등으로 로그 분석.
- 코드베이스는 지속적인 **CI/CD** 파이프라인을 통해 배포되어, 작은 수정도 안전하게 롤아웃하며 문제 시 쉽게 롤백할 수 있게 합니다.
- 수퍼톤 API나 Lemon Squeezy API의 버전 변경, 가격 정책 변경 등에 대비하여, 해당 업체의 **릴리즈 노트**를 주시하고 서비스에 필요한 업데이트를 선제적으로 적용합니다 (예: 수퍼톤 API 스트리밍 TTS 정식 출시 시 이를 추가 기능으로 도입 검토 [33]).
- **법규 준수:**
- 방송 음원 콘텐츠에 대해 저작권 문제가 없도록, 예시로 BGM 라이브러리는 **저작권 클리어된 음원**만 제공하고, 사용자가 자체 업로드하는 경우에도 불법 음원 업로드를 금지합니다.
- 개인 정보 처리방침 및 서비스 이용약관을 마련하여, 특히 음성 데이터(녹음 파일 등)의 수집/처리에 대한 동의를 명시적으로 받습니다.
- 관공서 등 공공기관 납품의 경우 필요한 보안인증(예: KMS 인증, 망 분리 환경 설치 지원 등)에 대응할 수 있도록 설계상 옵션을 열어둡니다.

---

7. 제한 사항 및 가정 (Assumptions & Constraints)

- **수퍼톤 API 의존:** TTS 합성과 보이스 클로닝의 품질과 성능은 **Supertone API**에 크게 의존합니다. 현재 수퍼톤은 한국어 등 3개 언어를 지원하며 [4] , 300자 이내 텍스트 처리 등의 제한이 있습니다. 본 서비스는 이러한 제약 내에서 기획되었으며, 수퍼톤 API의 **크레딧 체계**(별도 과금)를 사용합니다. 수퍼톤 API 비용(분당 $0.1 내외 [2])이 변동될 경우 서비스의 가격 정책도 그에 맞춰 조정해야 합니다. 또한 수퍼톤 API 장애 시 서비스 핵심 기능에 영향이 있으므로, **API 응답 실패에 대비한 안내** (예: “현재 음성 합성 서비스가 지연되고 있습니다”) 등을 준비합니다. 장기적으로 여러 TTS 엔진을 선택적으로 사용할 수 있게 확장도 고려합니다 (예: 클라우드 대비 온프레미스 TTS 엔진 등).
- **Lemon Squeezy 결제 제한:** Lemon Squeezy를 통해 한국 원화 결제와 세금 계산 등이 가능한지 확인해야 합니다. 만약 제한이 있다면 국내 PG와 연동하거나, 관공서의 경우 **후불 청구**(수기 계약) 등 별도 프로세스를 둘 수 있습니다. 현재 기획에서는 Lemon Squeezy의 **구독/결제 기능**을 이상적으로 활용하는 것으로 가정합니다.
- **로컬 장비 가정:** 관공서 방송용 PC 등 **Windows PC 1대를 방송 서버로 사용**하는 시나리오를 상정합니다. 이 PC에는 인터넷 연결이 되어 있고, 수면모드에서 깨울 수 있으며, 사운드 출력이 방송 앰프에 연결되어 있다고 가정합니다. 만약 네트워크 분리 등으로 클라우드 연결이 어려운 환경이라면, **온프레미스 설치판**(사내 서버에 설치되는 버전)의 제공도 고려해야 하나 본 설계 범위는 클라우드 모델입니다.
- **사용자 콘텐츠 품질:** TTS 특성상 합성 음성이 완벽히 자연인 음성과 동일하지 않을 수 있습니다. 안내방송 분야에서는 약간 로봇틱하더라도 명료도가 중요하므로, 현재 수퍼톤 음성 품질이 충분히 명료하다고 판단합니다. 숫자나 영어 발음 등 일부 한계는 있으나 사용자 교육(필요시 숫자는 한글로 적기 등)이나 엔진 개선으로 해결 가능하다고 가정합니다.
- **스케줄 정확도:** Windows의 작업 스케줄러나 JS 타이머의 정확도가 수 초 단위 오차는 있을 수 있으나, 관공서 방송에서는 수 초 정도의 편차는 허용 가능하다고 봅니다. (예: 9:00 정각 방송이 9:00:05에 나오더라도 큰 문제 없음) 아주 정밀한 예약(초 단위)은 요구되지 않는 것으로 가정합니다.

- **동시 방송 수요:** 한 기관 내에서 **동시에 여러 다른 방송을 동시에** 틀 가능성은 낮다고 가정 (일반적으로 한 번에 하나의 안내만 출력). 따라서 동시 다중 스트림 출력을 기본 제공하진 않습니다. 다만 여러 기관 고객이 동시간대에 각각 방송하는 것은 가능하므로, 서버나 API 측 부하는 수평적으로 분산 처리될 것입니다.

저수준 설계서 (Low-Level Design, LLD)

본 장에서는 상기의 요구사항을 실현하기 위한 구체적인 시스템 아키텍처, 구성 요소, 데이터 모델, 외부 연동, 알고리즘 등에 대해 상세 설계를 기술합니다.

8. 시스템 아키텍처 개요 (System Architecture Overview)

본 서비스의 전체 구조는 **클라이언트-서버 아키텍처**를 기본으로, 방송 송출을 위한 **로컬 컴포넌트**를 보조적으로 갖습니다. 주요 구성은 아래와 같습니다:

- **웹 클라이언트 (React SPA):** 사용자 인터페이스를 담당하는 React 기반 싱글 페이지 어플리케이션입니다. 브라우저 환경에서 동작하며, 백엔드 API를 호출해 데이터를 주고받습니다. 로그인, 대시보드, TTS 생성 UI, 스케줄 편집 UI, 플레이어 등이 모두 이 클라이언트에서 구현됩니다.
- **백엔드 서버 (Application Server):** Node.js (Express 또는 NestJS 등)나 Python (Django, Flask 등) 기반의 웹 서버로 구현되며, **RESTful API**를 제공합니다. 이 서버는 클라이언트 요청을 받아 **비즈니스 로직**을 처리하고 DB를 조작하며, 외부 API(Supertone, Lemon Squeezy 등)를 호출합니다. 또한 예약 스케줄 관리, 알림, 로그 기록 등의 기능을 수행합니다.
- **데이터베이스 (DB):** 사용자, 음원, 일정, 결제 등 애플리케이션 데이터를 저장하는 **영구 저장소**입니다. 관계형 DB(예: PostgreSQL, MySQL)를 사용하여 정규화된 데이터 (회원, 주문 등)를 저장하고, 필요에 따라 Redis와 같은 인메모리 DB를 캐싱/큐 용도로 병행 사용할 수 있습니다. 오디오 파일 자체는 DB에 저장하지 않고 파일스토리지에 경로만 저장합니다.
- **파일 스토리지:** 생성된 음성 파일 및 업로드된 음성 데이터를 저장하는 공간입니다. 초기에는 **서버의 로컬 디렉토리**나 NAS에 저장 가능하며, 확장성이나 외부 공개 링크를 고려해 **클라우드 스토리지**(AWS S3 등)로 전환할 수도 있습니다. 스토리지 URL은 접근 제어를 거쳐 제공됩니다.
- **로컬 방송 에이전트 (선택적):** 관공서 PC 등에 설치되는 경량의 **클라이언트 프로그램**입니다. 이 프로그램은 백엔드와 **양방향 통신**(웹소켓 또는 폴링)을 유지하며 해당 PC에서 예약된 시간에 음원을 재생하고, PC를 깨우는 역할을 합니다. 구현 방식은 Windows 서비스 또는 Electron 기반 앱 등 다양하게 고려될 수 있습니다. (초기 버전에서는 이 컴포넌트 없이도 동작 가능하나, 신뢰성 향상을 위해 옵션으로 설계)
- **외부 서비스 연동:**
- **Supertone API 서비스:** 수퍼톤에서 제공하는 클라우드 TTS/Voice Cloning 서비스로, 백엔드 서버가 HTTP REST 호출을 통해 사용합니다. 백엔드에서는 수퍼톤 **개발자 콘솔**에서 발급된 API 키를 저장하고, 합성 요청, 보이스 목록 조회, 클론 생성 등에 활용합니다.
- **Lemon Squeezy 결제 플랫폼:** 결제 처리를 위해 사용됩니다. 백엔드는 Lemon Squeezy의 API 및 웹훅을 사용하여 결제 창 생성, 구독 상태 확인, 사용량 보고 등을 수행합니다. 프런트엔드는 구매/결제 단계에서 Lemon Squeezy의 Checkout 페이지로 리디렉션하거나 임베드하는 방식을 쓸 수 있습니다.
- **이메일/SMS 발신 서비스:** (별도 언급 없지만) 계정 확인이나 알림을 위해 이메일 또는 문자 발송 서비스(SendGrid, Twilio 등)와 연동할 수 있습니다. 예: 예약 방송 실패시 관리자에게 경고 이메일.

아래는 서비스 아키텍처의 논리적 다이어그램입니다:

【이미지: System Architecture Diagram – 클라이언트, 서버, 외부 API, 로컬 에이전트의 상호작용을 나타낸 그림】
(다이어그램은 개략적으로: React Web → API Server → {DB, FileStorage, Supertone API, Payment API}, 그리고 API Server ↔ Local Agent, Local Agent → Audio System)

설명: 웹 클라이언트는 백엔드 API와 통신하여 데이터 CRUD와 작업 요청을 수행하고, 백엔드는 DB와 파일스토리지에 접근하며 필요 시 외부 API를 호출합니다. 예약 관련 이벤트는 백엔드에서 트리거 되거나, 로컬 에이전트가 서버로부터 명령을 받아 로컬 오디오 출력으로 실행합니다.

9. 주요 컴포넌트 설계 (Major Components Design)

본 장에서는 핵심 컴포넌트들을 모듈 단위로 상세 설명합니다.

9.1 프론트엔드 (React SPA)

- **기술 스택:** React + TypeScript를 사용하여 개발합니다. 라우팅에는 React Router, 상태관리는 Redux or Context, UI 컴포넌트 라이브러리는 Material-UI or Ant Design 등을 고려합니다. 오디오 재생은 HTML5 `<audio>` 태그와 Web Audio API (시각화) 등을 활용하고, WaveSurfer.js 같은 라이브러리를 검토합니다 [34].
- **페이지 구성:**
- 로그인/회원가입 페이지
- 대시보드 홈 (요약 정보: 다음 방송 일정, 이번달 사용량, 공지 등)
- TTS 생성 페이지 (텍스트 입력 및 보이스 선택 UI)
- 음원 라이브러리 페이지 (생성된 음원 목록, 플레이/다운로드/편집)
- 일정 관리 페이지 (캘린더/목록 뷰, 예약 추가/편집 다이얼로그)
- 결제/플랜 페이지 (현재 플랜 정보, 업그레이드 옵션, 결제수단 관리)
- 관리자 페이지 (전체 사용자/요금제 관리 UI) - 관리자에게만 보임
- **상태 및 데이터 흐름:** React Query 또는 Redux를 통해 서버 상태를 관리합니다. 예를 들어 음원 리스트, 스케줄 리스트는 서버에서 fetch하여 캐싱하고, 추가/수정 시 mutation 후 리스트를 갱신합니다.
- **특징 UI 컴포넌트:**
- **Voice 선택기:** 수퍼톤 제공 보이스가 많으므로, 검색/필터 UI를 제작합니다. 드롭다운 또는 모달 팝업으로 구현하되, 목록에 보이스 이름과 특징(성별, 감정) 표시, 샘플 소리 🔊를 들어볼 수 있는 버튼도 넣습니다 (수퍼톤 API의 `samples.url` 제공 활용 [6] [35]).
- **Audio Player:** 재생, 일시정지, 정지, 타임시크, 볼륨 등의 기능을 가진 공통 컴포넌트로서, wave visualization 포함. WaveSurfer.js를 React로 래핑하거나, `canvas`로 간단히 파형 그려도 됩니다 [34].
- **Scheduler Calendar:** React용 달력/스케줄러 컴포넌트(예: FullCalendar, DevExtreme Scheduler 등)를 활용해 UI 구축합니다 [36]. 반복 이벤트 설정 UI는 자체 구현 (모달 내 여러 옵션).
- **Modals/Dialogues:** TTS 생성 진행 중 (“생성 중...”), 클로닝 중 (“학습 중... 완료되면 알림”), 믹싱 편집기 등은 별도 모달창으로 띄워 작업을 완료하게 합니다.
- **Forms & Validation:** 회원가입, 텍스트 입력 등에는 form validation을 적용해 필수값 누락, 글자 제한 등을 즉시 검출합니다. TTS 텍스트 300자 제한은 남은 글자수 카운트 표시.
- **오프라인/백그라운드 고려:** 만약 브라우저를 항상 켜두고 예약 방송을 돌리는 모드일 경우, React 앱이 브라우저 탭에서 백그라운드로 있어도 이벤트를 캐치하여 음원을 재생할 수 있어야 합니다. 이를 위해 Web API의 `Notification`이나 `Wake Lock` 등을 검토했으나, 브라우저만으로 PC sleep 해제는 불가하므로, PC 깨우기는 에이전트로 처리하는게 나음. 그래도 브라우저가 열려있다면 **웹소켓 연결**을 통해 서버->클라이언트로 예약 실행 신호를 보내고, 클라이언트는 자동 재생을 수행하게 할 수 있습니다. (브라우저 정책으로 사용자 상호작용 없는 자동재생이 제한될 수 있으므로, 이를 우회하기 위해 사전에 사용자 동작으로 권한을 획득하거나, Chrome 정책에 사이트를 신뢰추가 안내)
- **에러 처리:** AJAX 실패나 서버 에러 시 공통 에러 핸들러를 두어 사용자에게 토스트/다이얼로그로 알려줍니다. 예: “세션이 만료되었습니다. 다시 로그인해주세요.”, “네트워크 오류 - 연결을 확인하세요.” 등.

9.2 백엔드 (Server-side Application)

- **기술 스택:** Node.js + Express 또는 NestJS 프레임워크 (TypeScript)로 구현을 가정합니다. 실시간 통신이 필요한 경우 Socket.io 등을 사용할 수 있습니다 (예약 알림 등). 데이터베이스 ORM은 Prisma or Sequelize 등을 이용하고, REST API는 OpenAPI 스펙을 정의하여 문서화합니다.

- **모듈 구조:**
- Auth Module: JWT 기반 인증, 비밀번호 해시 저장, OAuth 등. 로그인 시 토큰 발급, 미들웨어로 보호된 엔드포인트 접근 제어.
- User Module: 사용자 CRUD, 프로필 조회/수정, 권한 관리.
- TTS Module: 수퍼톤 TTS API 연동 기능. 텍스트→음성 요청, 음원 저장, voice 목록 캐싱/제공.
- VoiceClone Module: 수퍼톤 클로닝 API 연동. 파일 업로드 처리(Multer 등 사용), 클론 요청, 완료 polling 또는 webhook 처리.
- Audio Library Module: 음원 파일 DB관리, 다운로드 링크 발행, FFmpeg 툴 연계(믹싱).
- Schedule Module: 예약 일정 저장 및 실행 관리. Cron 스케줄러 or Node Schedule 사용.
- Broadcast Module: (로컬 에이전트 연계) 웹소켓 서비스, 에이전트와 메시지 교환.
- Billing Module: Lemon Squeezy API 연동, 웹훅 수신 (구독 생성/취소 이벤트 등), 사용량 집계 및 주기적 LS API 호출 (usage update).
- Admin Module: 관리자용 통계, 로그조회, 설정변경 API.
- **API 설계:** 주요 API 엔드포인트 예시는 다음과 같습니다:
- `POST /api/login`, `POST /api/register` – 인증 관련
- `GET /api/voices` – 사용 가능한 Voice 목록 조회 (캐시 활용하여 수퍼톤 **GET /voices** 결과 반환)
- `POST /api/tts` – 텍스트로 음성합성 요청 (본문: text, voice_id, style 등) -> 생성 후 음원 ID 반환
- `POST /api/voice-clone` – 보이스 클로닝 요청 (멀티파트: file, name) -> 처리 시작 및 결과 polling or async
- `GET /api/audios` – 음원 리스트 조회, `GET /api/audios/{id}` – 상세(다운로드 링크)
- `POST /api/mix` – 믹싱 작업 요청 (입력: 여러 음원 ID + 타임라인 설정) -> 결과 음원 생성
- `GET /api/schedules` – 예약 목록 조회, `POST /api/schedules` – 새 예약 생성, `PUT /api/schedules/{id}` – 편집, `DELETE /api/schedules/{id}` – 삭제
- `POST /api/schedules/{id}/toggle` – 활성/비활성 토글
- `GET /api/logs` – (사용자별) 생성/방송 이력 조회
- `GET /api/admin/users`, `GET /api/admin/usage` 등 – 관리자 전용
- `POST /api/billing/create-checkout-session` – (일반) 결제창 생성 (LS와 통신)
- ... 등
- **스케줄러 구현:** 백엔드 서버는 Node Schedule(node-cron) 등을 이용해 초단위 정밀하지 않은 예약도 처리 가능하지만, 수백개 이상의 작업 동시 스케줄링에는 부적절할 수 있습니다. 따라서 설계는 **하이브리드**로:
- 서버는 **가까운 시점(예: 다음 1시간)** 내의 예약만 스케줄러에 등록해두고 실행, 그 이후 것은 시간이 다가오면 추가 등록. (이렇게 하면 메모리 낭비 방지)
- 또는 각 예약시간에 DB 쿼리해서 확인하는 방식. 하지만 초단위 polling은 비효율.
- **추천:** Agenda.js 같은 job scheduler를 써서 DB 기반 job 관리 + worker로 실행. Reservation job이 트리거되면 해당 음원 play 명령을 에이전트에 전달.
- **로컬 에이전트 연계:** 백엔드는 **웹소켓 서버**를 열어 에이전트(클라이언트)와 실시간 통신합니다. 에이전트가 로그인(토큰)하여 소켓 연결을 맺으면, 서버는 해당 소켓을 그 PC 사용자/장치에 매핑해둡니다. 예약 실행 시각이 오면, 서버는 해당 예약의 소켓에 "play audio {audio_id}" 메시지를 보냅니다. 에이전트는 이를 수신해 로컬에서 재생 처리 후 서버에 ACK/완료 통보. 만약 정해진 시간에 소켓이 연결 안돼있으면 (PC 꺼짐), 서버는 그 정보를 모니터링(미연결)하여 실패 처리하거나, WOL packet 등을 시도할 수 있습니다 (WOL은 서버가 MAC 주소 알아야 가능). WOL까지 지원하려면 백엔드에서 각 에이전트 PC의 MAC/IP 정보 관리와 같은 네트워크 저수준 작업 필요, 우선범위 밖이지만 가능성 열어둠.
- **수퍼톤 API 연동 상세:**
- API Key는 서버 환경변수에 저장. 모든 Supertone API 호출 시 `x-sup-api-key` 헤더에 키 포함 [37].
- **음성 합성:** `POST /v1/text-to-speech/{voice_id}` 호출 [38]. 요청 바디로 JSON 전달 [11], 응답은 스트림(바이너리)로 오므로 axios 등의 HTTP 라이브러리에서 스트림 모드로 받아 파일로 저장. 시간 측정하여 로그 남김. 한 번에 여러 문장 요청시 루프 처리 및 async/await 제어.
- **보이스 목록:** 서비스 초기 가동 시 또는 관리자가 명시 요청 시 `GET /v1/voices`를 호출하여 전체 voice 목록을 캐싱합니다 [39] [40]. 이 목록을 DB나 메모리에 저장해두고, `/api/voices` 요청시 필터 적용 후 반

환. 또는 실시간 API proxy로 `GET /voices/search?language=ko` 등을 호출 가능 [41]. 캐싱으로 성능 향상하되, 수퍼톤 voice 업데이트 반영 위해 일정 주기(예: 하루1회)로 새로 받아 업데이트.

- **보이스 클로닝:** `POST /v1/custom-voices/cloned-voice` 사용 [42]. Node에서는 FormData를 구성해 파일 스트림과 name을 첨부하여 보내고, 요청 후 곧바로 응답 voice_id 받으면 DB에 기록. 만약 수퍼톤이 바로 안주고 async 처리한다면, `GET /v1/custom-voices/{voice_id}`로 상태 조회를 해야 할 수도. Release note에 "API를 통해 등록" 추가된 것이 바로 voice_id 주는 것으로 보임 [43]. 생성된 voice_id는 사용자 계정 scope이므로, TTS 호출 시 동일 API Key 조건에서 이용 가능 [44] [45]. 클론 voice도 voice 목록 조회 API에서 검색되므로 (GET voices/search?name=XX) 확인 가능 [46].
- **에러 처리:** 수퍼톤 API가 에러시 status code와 메시지를 줌. 401(인증실패) 등은 서버 설정 문제, 402(잔액부족) [15] 은 사용자 크레딧 부족인데, 크레딧은 공유라서 이 경우 서비스 운영측 크레딧 모자란 것이므로 **긴급 알림** 필요. 400(매개변수 오류)은 텍스트 길이 초과 등일 수 있어 사용자에 알려야. 429(호출제한) 발생시 약간 딜레이 후 재시도 정책 (exponential backoff) 넣습니다.
- **Lemon Squeezy 연동 상세:**
- **제품/구독 설정:** 우선 Lemon Squeezy 대시보드에서 제품(Product)과 플랜(Variants)을 구성하고, 그 ID를 서버 설정에 기록합니다. 예: Free, Personal, Enterprise 등.
- **Checkout:** 사용자가 플랜 업그레이드 요청 시 백엔드가 LS API를 호출하거나, 간단히는 프론트에서 Lemon Squeezy의 **Hosted Checkout** 페이지 링크로 보내 결제 진행합니다. (LS는 결제페이지 UI 제공). 결제 완료되면 LS가 우리 서버의 webhook URL에 통지하므로, 백엔드에서 **/webhook/lemonsqueezy** 엔드포인트 구현하여 이벤트 수신 (구독 생성/결제 등 이벤트 타입 확인).
- **사용량 보고:** 만약 usage-based pricing을 쓰는 플랜의 경우, 예를 들어 1분당 과금 모델은 LS에서 subscription의 `quantity` 필드를 사용해 사용량을 반영하도록 합니다. 백엔드는 매일 혹은 매월 말 `PATCH /v1/subscriptions/{id}`로 `{"quantity": 누적분수}`를 업데이트 하여 LS가 다음 청구서에 반영토록 합니다 [26]. 또한 LS의 consumption 모델(Volume/Ggraduated)을 활용하여 일정량까지 무료, 초과시 단가 적용 등의 설정을 할 수 있습니다 [47]. 이러한 로직을 Billing Module에서 플랜별로 다르게 처리합니다.
- **API 통신:** LS API는 JSON:API 규격이므로, 백엔드에서 적절한 헤더(Authorization Bearer 등)와 함께 GET/POST/PATCH 등을 호출합니다 [48] [49]. 예를 들어 구독 변경, 취소 등을 API로 처리할 수 있습니다.
- **보안:** 결제 관련 webhook은 신뢰할 수 있도록 LS 제공 signature를 검증하거나, secret token을 써서 위변조를 막습니다. 사용자 결제정보(카드 등)는 LS에서 관리하므로 우리 DB에는 최소한의 식별자만 둡니다 (customer_id, subscription_id 등).
- **로깅 및 모니터링:**
- 각 합성/클로닝 요청, 예약 실행, 결제 이벤트 등을 로그 파일 및 DB(Log table)에 남깁니다. Winston 등 로깅 라이브러리를 사용해 파일로 기록하고, 심각 오류는 관리자에게 알림.
- DB의 Log table은 Admin이 조회하게도 하지만, 너무 누적되면 rotation/archiving 필요.
- 또, 모니터링 툴(NewRelic 등)을 붙여 API latency나 서버자원 사용을 시각화합니다.

9.3 데이터베이스 설계 (Database Schema)

주요 엔티티 및 스키마를 정의합니다.

- **User (사용자 계정)** – `User(id, name, email, password_hash, role, organization, plan_id, credit_balance, created_at, ...)`
- `role`: {USER, ADMIN, ORG_ADMIN}
- `organization`: 소속 단체 (관공서명 등), null 가능 (개인 사용자)
- `plan_id`: 현재 요금제 (참조: Plan 테이블)
- `credit_balance`: (옵션) 수퍼톤 크레딧 또는 자체 크레딧 잔량
- 기타: 마지막 로그인, 설정(JSON: e.g. default_voice)
- **Plan (요금제)** – `Plan(id, name, type, price, billing_cycle, tts_included_minutes, storage_limit_mb, allow_cloning, ...)`

- `type` : {FREE, SUBSCRIPTION, USAGE} 등
- `billing_cycle` : monthly, yearly, etc.
- `tts_included_minutes` : 기본 제공 TTS 분 수
- `extra_minute_price` : 초과 분당 가격 (Usage형일 때)
- `allow_cloning` : 보이스 클로닝 허용 여부
- `ls_product_id` , `ls_variant_id` : Lemon Squeezy 상 대응 상품 ID (결제 연계)
- **Audio (음원)** – `Audio(id, user_id, title, filename, duration_sec, voice_id, text, created_at, type, ...)`
- `type` : {TTS, MIX, UPLOAD, CLONE_SAMPLE} 등 생성 방식 구분
- `voice_id` : 사용된 voice (TTS의 경우), 없으면 NULL
- `text` : 원본 텍스트 (TTS의 경우 저장하여 나중에 수정 가능성, 클론샘플의 경우 메모 등)
- `filename` : 저장된 경로 또는 파일 식별자 (ex: `user_{id}/audio_{id}.wav`)
- `duration_sec` : 재생 길이(초)
- `filesize` : 용량
- `meta` : JSON (pitch, speed 등의 설정 값)
- **Schedule (예약)** – `Schedule(id, user_id, audio_id, title, start_time, cron_expr, enabled, repeat_until, last_executed_at, ...)`
- `start_time` : 단일 실행 시간 (datetime) – 반복 없을 때 사용
- `cron_expr` : 반복 일정 표현 (있으면 start_time 무시)
- `repeat_until` : (optional) 종료 일시
- `enabled` : bool
- `last_executed_at` : 마지막 실제 실행시각 (성공한 경우 업데이트)
- `next_occurrence` : 다음 예정시각 (스케줄러 편의 캐싱)
- **History / Log** (세분하여 두 개 테이블로 분리 가능)
- **TTSHistory** – `TTSHistory(id, user_id, audio_id, text, voice_id, created_at, result, error_msg, cost_credits)` : 각 합성/클로닝 시도의 기록. 성공시 audio_id, 실패시 error_msg 저장.
- **BroadcastHistory** – `BroadcastHistory(id, schedule_id, user_id, audio_id, planned_time, executed_time, status, error_msg)` : 예약 또는 수동 방송 실행의 기록.
- **VoiceClone** – `VoiceClone(id, user_id, name, status, source_filename, voice_id, created_at, completed_at)`
- 사용자가 보이스 클로닝 시도한 내역 관리. voice_id 생성 완료전까지 status=PENDING, 완료되면 voice_id 필드 채우고 status=READY, 실패시 status=FAILED+오류원인.
- **UsageAccumulation** – (과금용) `UsageAccumulation(id, user_id, period_start, period_end, tts_seconds, cloning_count, storage_used_mb, ... processed)`
- 매달 또는 매기간별 사용량을 합산 기록. `processed` 는 청구 반영 여부.
- `tts_seconds` 를 분으로 환산하거나, `tts_minutes` 등으로 저장하여 나중에 결제 반영.
- **Payment** – `Payment(id, user_id, plan_id, amount, currency, status, timestamp, ls_invoice_id, ls_subscription_id, next_billing_date, ...)`
- 결제 거래 기록. 구독이면 subscription_id로 구독 연결하고, 주기적 거래는 invoice로 기록.
- status: PAID, FAILED, REFUNDED, etc.

관계: - User–Audio: 1:N (사용자별 여러 음원) - User–Schedule: 1:N - User–History: 1:N - Audio–Schedule: 1:Many (한 음원 여러 일정에 재사용 가능) - Plan–User: 1:N (다수 사용자 같은 플랜) - etc.

9.4 로컬 에이전트 (Local Broadcast Agent)

- **역할:** Node.js 또는 Python으로 작성된 경량 서버로 PC에서 실행됩니다. 주요 기능:
- **WebSocket Client:** 백엔드 WS 서버에 연결, 인증(JWT 전달) 후 대기. 특정 메시지 (예: play, volume, stop 등) 수신 시 동작.
- **Audio Playback:** OS의 기본 오디오 출력을 통해 음원 재생. Node라면 `speaker` 모듈이나 `play-sound` 패키지, Python이면 `pydub` / `simpleaudio` 등 사용 가능. 또는 FFmpeg/OS명령 줄로 mpg321 같은 플레이어 호출.
- **Wake Lock:** Windows의 경우 `SetThreadExecutionState` API 또는 `Powercfg` 이용해 예약에 맞춰 깨우기. 이 부분은 관리자 권한 필요할 수 있어, 설치 시 권한 부여 안내.
- **Offline Scheduling:** 에이전트 자체적으로도 스케줄을 가질 수 있음. 서버로부터 향후 N시간의 일정을 받아 로컬 타이머 설정 → 시간이 되면 바로 재생. 이 경우 WS 없어도 실행 가능. 그러나 정확히 Sleep상태에서 깨우려면 OS-scheduler와 협업 필요.
- **Feedback:** 재생 시작/종료/오류 시 즉시 백엔드에 REST나 WS로 이벤트 보내 로그기록 및 UI 표시.
- **Auto Start:** PC 부팅 시 자동 시작 (Startup 등록)하여 항상 대기. UI는 CLI 또는 system tray 아이콘으로 최소화되어 존재 (상태표시: 연결됨/끊김, 다음 예약시각 등).
- **설치 & 배포:** 관공서 IT관리자가 손쉽게 설치하도록 **인스톨러** 제공 (Windows .exe installer). 필요한 설정(서버 URL, 인증토큰 등)을 묻고, OS Schedule 등록(예: 작업 스케줄러) 가능 옵션 "이 PC를 방송용으로 설정" - 클릭 시 PC 절전모드 해제 설정(On), 에이전트도 계속 실행되도록.
- **Security:** 에이전트와 서버 통신은 wss (secure websocket)로 암호화. JWT 토큰은 만료 시 재인증 로직(재로그인 필요) 혹은 장기 Refresh token 사용. 에이전트는 최소한의 권한만 가지며, PC 제어는 사용자 동의 하에 이뤄짐.

9.5 외부 서비스 및 API 상세 (External Services)

위에서 언급한 Supertone, Lemon Squeezy의 API 활용 이외에, 혹시 필요한 외부 도구: - **FFmpeg** (or GStreamer): 오디오 인코딩/디코딩, 믹싱에 활용. 서버에 라이브러리 설치 후 child_process로 명령 실행. 예를 들어 믹싱 시: `ffmpeg -i voice.wav -i bgm.mp3 -filter_complex "[1]volume=0.5[a];[0][a]amix=inputs=2:duration=first:dropout_transition=2" output.wav` 식으로 볼륨 50% BGM과 음성 합성 [50]. - **Email API:** Nodemailer (SMTP)나 외부 API (SendGrid)로 사용자 알림 (가입인사, 예약실패 통보 등) 발송. - **SMS API:** 긴급알림용으로 Twilio 등. 관공서는 문자 통보 요구할 수 있어 옵션으로 고려. - **Monitoring APM:** NewRelic, Prometheus+Grafana etc. (비기능)

10. 시퀀스 흐름 예시 (Sequence Flow Examples)

몇 가지 핵심 시나리오에 대한 시퀀스 다이어그램 형태의 설명입니다.

10.1 TTS 음성 생성 흐름

1. **사용자**가 웹에서 텍스트 입력, 보이스 선택 후 "생성" 클릭.
2. **React 클라이언트**는 `/api/tts` POST 요청을 백엔드에 전송 (JSON: text, voice_id 등).
3. **백엔드**는 요청 수신 → 수퍼톤 **POST /v1/text-to-speech/{voice_id}** 호출 [12]. 헤더에 API키 첨부, 바디에 text와 설정 전송.
4. **수퍼톤 서버**가 합성 처리 → 성공하면 **오디오 바이너리 스트림** 응답 [13].
5. **백엔드**는 스트림을 받아 `audio.wav` 파일 저장 (경로 결정, Audio DB 레코드 생성: 상태=완료, duration=헤더정보).
6. 백엔드 응답으로 **Audio ID**와 메타데이터(파일URL 또는 ID, 길이 등)를 클라이언트에 반환.
7. **클라이언트**는 응답 받으면 사용자에게 **플레이어 UI**에 파형과 플레이 버튼 표시. 사용자 “재생” 클릭 시 `<audio src=...>`로 재생.
8. 사용자가 "저장" 버튼 누르면 해당 음원 Audio ID가 라이브러리에 남음 (사실 이미 DB에 등록됨).

9. (만약 합성 실패 시) 백엔드가 에러코드/메시지 전송 → 클라이언트가 팝업으로 오류 알림.

10.2 보이스 클로닝 생성 흐름

1. **사용자**가 보이스 클로닝 페이지에서 음성 파일 선택하고 이름 입력 후 "업로드" 클릭.
2. **클라이언트**는 파일을 멀티파트로 `/api/voice-clone` POST 요청.
3. **백엔드**는 파일 수신하여 임시 저장 → 곧바로 Supertone **POST /v1/custom-voices/cloned-voice** 호출 [19]. `x-sup-api-key` 헤더와 함께 FormData(files=@파일, name) 전송.
4. **수퍼톤 API**가 음성 분석/학습 수행 → 완료 시 `voice_id` 반환 [21] (이 작업이 수십초 걸릴 경우 API가 sync 응답인지 async인지 가정 필요; 일단 sync로 가정).
5. **백엔드**는 `voice_id` 수신하면 **Custom Voice DB**에 저장 (VoiceClone.status=READY, voice_id 필드 기입). 또한 사용자 Audio 라이브러리에 새로운 voice로 TTS 생성 가능함을 알리기 위해, 캐시된 voice 목록에 추가 (혹은 다음 GET /voices시 포함).
6. 백엔드는 클라이언트에 성공 응답 (e.g. {"status": "success", "voice_id": "..."}).
7. **클라이언트**는 사용자에게 "새 보이스 클론 생성 완료" 메시지 표시. 이제 TTS 생성 시 보이스 선택 목록에 해당 voice_id(사용자지정 이름으로) 나타남.
8. (실패 시) 백엔드 에러 응답, 클라이언트 안내.

10.3 예약 방송 등록 및 실행 흐름

- **등록 시:**
- 사용자 "예약 추가" -> 일정 설정 폼 입력 -> "확인".
- 클라이언트 `/api/schedules` POST (audio_id, 시간/반복 정보 등) 전송.
- 백엔드 DB에 Schedule 레코드 생성 (enabled=Y).
- 백엔드 **스케줄러**에 해당 작업 등록:
    - 예: Node-cron으로 cron_expr 해석하여 다음 실행 시간 계산, 또는 setTimeout으로 특정 datetime까지 남은 시간 후 실행 설정. 반복일 경우 cron 등록.
    - 또한 로컬 에이전트 연결상태 고려: 만약 해당 사용자에 에이전트 연결 정보가 있으면, 그쪽에도 일정을 전달(동기화)할 수 있음.
- 백엔드 성공 응답 -> 프론트 리스트/캘린더에 반영.
- **실행 시:**
- 정해진 시간이 가까워지면:
    - 서버 측 cron 트리거: “지금 X예약 실행” -> 서버는 해당 Audio 파일 경로를 알아내고
    - (전략 A: 서버가 직접 audio 데이터를 socket 통해 보내서 에이전트 재생? vs 전략 B: 에이전트가 미리 파일 받아 재생?)
    - 효율상 **전략 B**: 에이전트가 미리 파일을 다운받거나 접근가능하도록, Audio 저장소를 공유 or URL로 fetch.
    - 서버, 에이전트 소켓이 연결되어 있다면 -> `socket.emit('play', {audio_id})`.
    - 에이전트 없거나 연결 안됨 -> 서버가 OS 작업 스케줄에 등록해놨다면 그게 PC를 깨워 Media Player로 해당 파일 재생 (Backup).
- **에이전트** 수신 후:
    - 지정 audio를 로컬 경로에서 찾음 (캐시에 없으면 서버에서 다운로드).
    - PC 깨우기는 시점상 이미 깨어있어야 WS 연결이 유지됐으므로, 이 경우 Sleep이 아니었음. 만약 Sleep 상태면 WS도 끊겨있을테니, OS 스케줄러가 대신 깨워야함. (여기서 복잡; 둘 중 하나 적용)
    - 에이전트, audio.wav를 재생 시작 -> 완료 콜백 받음.
    - 에이전트, 서버로 `socket.emit('played', {schedule_id, time: actual_time})` 전송.
- **백엔드**는 played 이벤트 받아 BroadcastHistory 기록 (status=SUCCESS, executed_time).
- 프론트(만약 열려있는)에서 실시간 상태 업데이트 (예: "09:00 방송 실행됨").
- PC는 설정에 따라 일정 종료 후 다시 Sleep 진입 (에이전트가 Idle 일정 설정).
- **실패 시:**

- 만약 WS없고 OS Task로 했는데 Media Player 실패 -> OS Task Scheduler 재시도 (설정에서 3회 재시도 등).
- 에이전트가 소켓 받았지만 파일 재생 중 오류 -> 에이전트 `played` 이벤트에 status=FAILED, error=msg 보내기.
- 서버 BroadcastHistory에 status=FAILED 기록, 사용자에게 즉시 알림 (UI 또는 이메일).
- 관리자 대시보드에 미처리 실패 건 알림.

10.4 과금 및 결제 흐름

- **구독 구매:**
- 사용자 플랜 페이지에서 원하는 플랜 선택 -> "구매" 클릭.
- 클라이언트, 백엔드 `/api/billing/create-checkout-session` 호출 (또는 LS checkout direct link).
- 백엔드 LS API 호출하여 checkout session URL 획득 [51] , 클라이언트에 전달.
- 클라이언트, 새 탭으로 LS Checkout 페이지 열기 -> 사용자 결제 완료.
- LS 서버, 결제 완료 webhook -> 우리 백엔드 `/api/webhook/ls` POST.
- 백엔드 해당 user의 Plan 할당 변경, Payment 기록 생성, UsageAccumulation 초기화 등 처리.
- 사용자 브라우저도 LS checkout success URL 통해 리다이렉트 -> 우리 사이트 Thank You 페이지. 거기서 백엔드로부터 최신 plan 정보 확인하여 UI 갱신.
- **월말 사용량 처리 (Usage-based):**
- 매일 밤 12시 백엔드 cron -> UsageAccumulation에 모든 사용자 그날 tts_seconds 합 적재.
- 매월 말일 cron -> 각 유료 사용자 usage 계산 (이번달 tts_minutes - included_minutes = extra, if <0 then 0).
- LS API `PATCH subscription`으로 `quantity=extra_minutes` 업데이트 [1] (또는 tiered pricing이면 LS가 알아서 청구).
- LS가 자동으로 카드 청구 -> webhook -> Payment 기록 (Paid).
- 백엔드 UsageAccumulation 처리표시, 사용자 credit_balance (내부 추적용) 리셋 or 차감.
- **결제 실패:**
- LS webhook에서 failed 오면 user plan을 일시중지 상태로 바꾸고, 서비스 기능 일부 제한 (TTS 생성 막기 등) + 사용자에게 결제 문제 알림.

10.5 믹싱 편집 및 저장 흐름

1. 사용자, 라이브러리에서 음원 2개 선택 -> “믹싱” → 믹서 에디터 화면 오픈.
2. 클라이언트, 선택 음원들의 파일을 필요하면 미리 다운로드 (Waveform 표시 용 분석).
3. 사용자가 타임라인 편집 (시작/종료 시간 조절, 볼륨 설정).
4. “저장” 클릭 -> 클라이언트 편집 세팅 (각 트랙 offset, volume 등) `/api/mix` POST.
5. 백엔드, 입력받은 음원 ID 리스트로 파일 경로 확보 -> FFmpeg 명령어 생성 (필터graphs).
6. FFmpeg 실행, 출력 파일 생성. (또는 ffmpeg.wasm 사용해 서버 or 클라에서 처리 가능하나, 서버가 성능 좋으니 서버권장)
7. 새 Audio 레코드 생성 (type=MIX, composed of N tracks).
8. 백엔드 응답 (Audio ID).
9. 클라이언트 “믹스 완료” 표시, 새로운 음원 라이브러리에 등장 -> 사용자는 즉시 들어보고 (Player) 만족하면 예약에 활용.

11. 보완 고려사항 (Further Considerations)

- **스트리밍 TTS:** 향후 수퍼톤의 **Streaming TTS (Beta)** 기능 도입 시, 실시간으로 긴 문장도 지연 없이 재생 가능해집니다 [33] . 이를 활용하면 긴 안내방송을 쪼개어 생성하지 않고 바로 스트리밍 출력하는 기능을 넣을 수 있습니다. 현재는 Beta라 안정성 상 제한하지만, 차후 업그레이드 포인트입니다.

- **멀티 출력 채널:** 방송 시스템에서 구역별 다른 방송을 동시에 내보낼 요구가 생긴다면, 소프트웨어적으로 **멀티 채널 출력**을 관리해야 합니다. 예: 1층 스피커엔 A안내, 2층엔 B음악. 이 경우 에이전트가 사운드카드의 채널 혹은 여러 출력 디바이스를 제어할 수 있어야 하고, UI에서도 구역 개념 추가 필요. 현재 범위에서는 단일 채널로 가정합니다.
- **AI 음성 업그레이드:** 수퍼톤 API 외에 다른 AI 모델(예: 번역, STT for voice commands)을 추가해 기능 확장 가능. 특히 음성 합성 전 텍스트 교정(발음 어려운 단어 음철표기) 등을 위해 간단한 NLP 모듈 넣을 수 있습니다.
- **캐시 및 CDN:** 음원 파일이 많아지면 CDN 캐싱으로 다운로드 부하 분산. 특히 일반 사용자 대상이라면 음원 스트리밍용으로 CDN 도입 고려.
- **사용자 지원 도구:** UI에서 "문의하기" 또는 피드백 수집 기능, 그리고 상세한 도움말 문서 페이지(튜토리얼 영상 링크 포함) 제공하여 서비스 채택을 돕습니다.

본 PRD와 LLD에 따라 개발을 진행하면, **AI 기반 TTS 방송 서비스**의 기획 의도와 기술 요구사항을 충실히 구현할 수 있을 것으로 예상됩니다. 주요 기능과 설계 요소들을 종합적으로 다뤘으며, 실제 개발 시에는 우선순위에 따라 일부 기능을 단계별 론칭할 수 있습니다 (예: MVP에서는 핵심인 TTS 생성, 예약 방송, 기본 과금만 구현하고, 이후 보이스 클로닝 등 확장). 이 문서를 바탕으로 디자인, 개발 팀이 협업하여 서비스를 완성하고, 궁극적으로 관공서 및 일반 사용자에게 **안정적이고 혁신적인 음성 방송 경험**을 제공하길 기대합니다.

---

1 26 27 47 Tailored Pricing for Every User: Unveiling Consumption Billing at LemonSqueezy • Lemon Squeezy
https://www.lemonsqueezy.com/blog/consumption-billing

2 Release Notes - Supertone API Documentation
https://docs.supertoneapi.com/en/user-guide/release-notes

3 4 9 10 11 12 13 14 24 Text-to-Speech 가이드 - Supertone API Documentation
https://docs.supertoneapi.com/ko/user-guide/text-to-speech

5 6 7 8 35 39 40 41 Voice Selection Guide - Supertone API Documentation
https://docs.supertoneapi.com/en/user-guide/voice-selection

15 30 31 32 44 45 46 Frequently Asked Questions - Supertone API Documentation
https://docs.supertoneapi.com/en/user-guide/faq

16 33 43 환영합니다 - Supertone API Documentation
https://docs.supertoneapi.com/ko/user-guide/welcome

17 18 19 20 21 22 23 42 Create cloned voice - Supertone API Documentation
https://docs.supertoneapi.com/ko/api-reference/endpoints/create-cloned-voice

25 36 DHTMLX React Scheduler: Create Customizable ... - DEV Community
https://dev.to/plazarev/dhtmlx-react-scheduler-create-customizable-scheduling-interfaces-with-react-2dlg

28 29 48 49 51 Guides: Subscription Management using the Lemon Squeezy API • Lemon Squeezy
https://docs.lemonsqueezy.com/guides/developer-guide/managing-subscriptions

34 Implementing an Audio Waveform with React Hooks and Wavesurfer.js
https://andreidobrinski.com/blog/implementing-an-audio-waveform-with-react-hooks-and-wavesurferjs/

37 38 Text-to-Speech Guide - Supertone API Documentation
https://docs.supertoneapi.com/en/user-guide/text-to-speech

50 Audio Interface vs. Mixer—What's the Difference and Why ... - MACKIE
https://mackie.com/en/blog/all/Audio_Interface_vs._MixerWhats_the_Difference_and_Why_You_Should_Care.html?srsltid=AfmBOoqZRW_dvHOvOVT91egna_8OjT01N2DX4pu-CtKRfpK053EWvr11