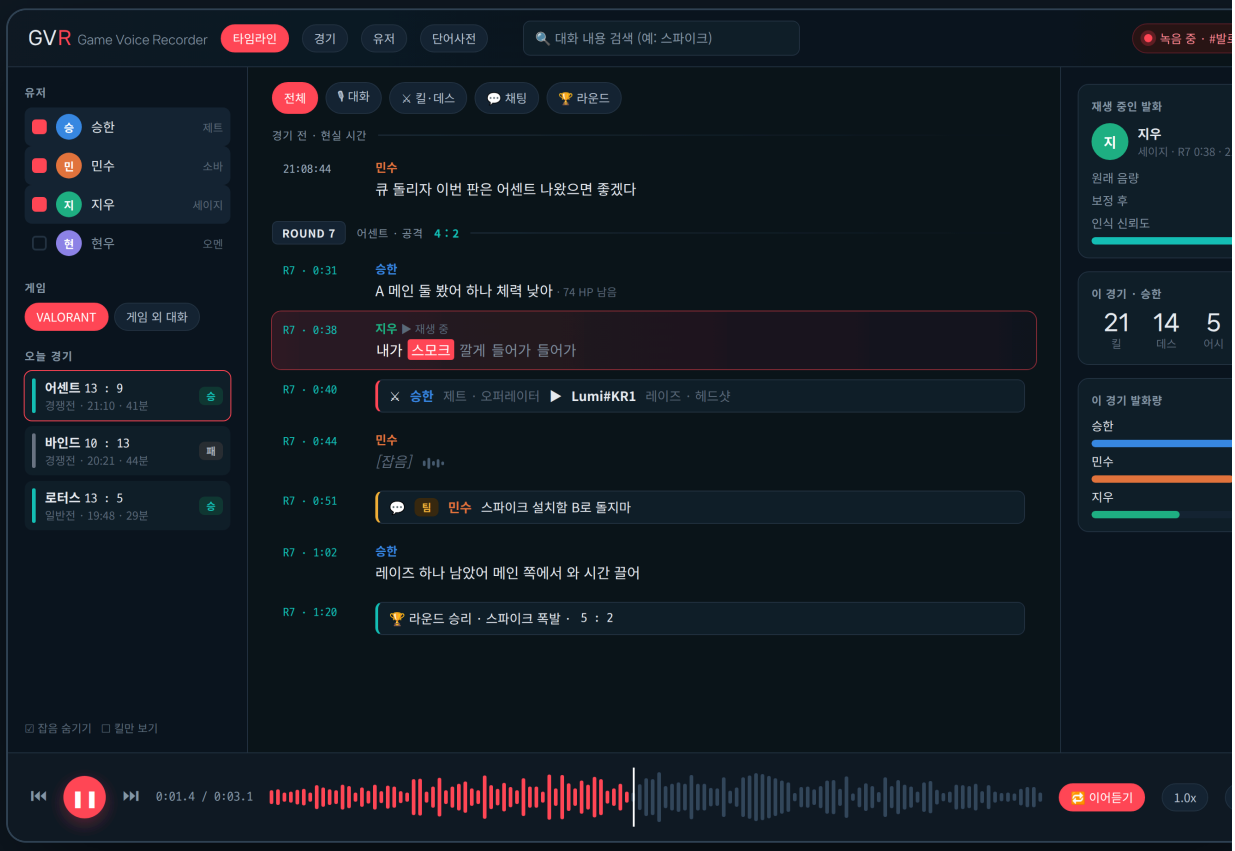
Game Voice Recorder

게임 연동 디스코드 음성 기록 봇 + 기록 열람 사이트
프로젝트 보고서 (구현 전 기획안)

문서 버전	v0.2 (화면 예시 · 인포그래픽 추가, 확인 질문 포함)
작성일	2026-10-14
저장소	git.tkrmagid.kr/tkrmagid/game_voice_recorder
1차 대상 게임	VALORANT
실행 형태	Windows 포터블 exe (게임 PC에서 실행)

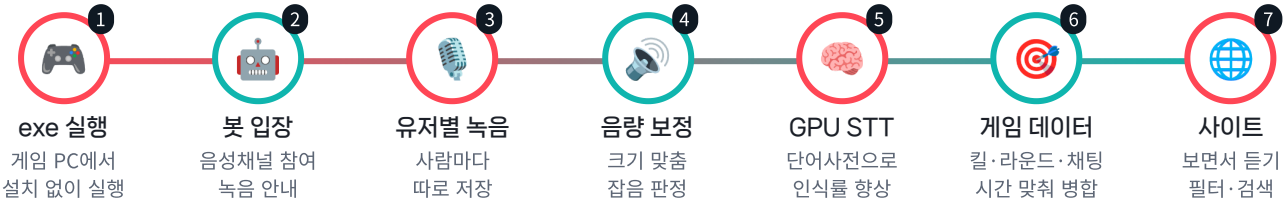


이 문서는 구현 착수 전 요구사항 정리·설계·확인 질문을 담은 기획 문서입니다. 수치(VRAM, 용량 등)는 추정치이며 프로토타입 단계에서 실측 후 확정합니다.

01 요약

디스코드 음성채널에 봇이 참여해 참가자 각자의 음성을 따로 녹음하고, 게임 PC의 GPU로 음성을 텍스트로 바꾼 뒤(STT), 발로란트 경기 데이터(킬·데스·라운드·인게임 채팅 등)와 같은 타임라인에 묶어 웹사이트에서 보고 들을 수 있게 하는 프로그램입니다. 설치 없이 실행하는 포터블 exe 하나로 봇, STT 엔진, 웹사이트 서버, 설정 화면이 함께 돌아갑니다.

한눈에 보는 흐름



100%
화자 구분 정확도
디스코드가 유저별로 음성을 따로 보냄

10명
경기 데이터 범위
같은 경기 10명 전원의 킬·데스

≈14MB
말한 시간 1시간당 음성 용량
(Opus 32kbps 기준)

0단계
설치 과정
포터블 exe, 모델은 첫 실행 때 다운로드

핵심 기능

🎙️ 음성 기록

디스코드 봇이 음성채널에 들어가 유저별로 음성을 따로 받아 발화 단위로 저장합니다. 화자는 디스코드가 유저별로 스트림을 나눠 보내 주기 때문에 추정하지 않고 정확히 구분됩니다.

📝 GPU STT

faster-whisper(Whisper 계열) 모델을 NVIDIA GPU로 구동합니다. GPU와 게임을 인식해 모델 크기를 자동으로 추천하고, 고른 폴더로 내려받습니다.

🔊 음량 보정 · 잡음 처리

작은 목소리와 큰 목소리를 같은 크기로 맞춘 뒤 인식하고, 말이 아닌 소리는 [잡음]으로 표시합니다.

🎯 발로란트 연동

경기 시작·종료, 라운드, 킬·데스·어시스트(시각·위치·무기), 점수, 인게임 채팅을 음성 기록과 같은 시간축에 배치합니다.

🕒 이중 시간 표시

경기 중 발화는 인게임 시간(예: `R7 · 0:42`)으로, 경기 밖 발화는 현실 시간으로 표시합니다.

🌐 기록 열람 사이트

시간·화자·텍스트 목록에서 문장을 누르면 그 위치부터 재생되고, 재생 중인 단어가 가라오케처럼 강조됩니다. 유저별·게임별·경기별 필터와 검색을 지원합니다.

구현 전에 꼭 정해야 할 3가지

① 발로란트 "공식 API"는 개인용 프로젝트로는 사실상 발급이 어렵습니다

Riot 정책상 VALORANT API는 개인 키(Personal Key)를 받지 않고, 프로덕션 키만 사용할 수 있습니다. 또 "공개되지 않은 개인용 앱"은 승인 불가 항목으로 명시돼 있고, RSO(Riot 로그인) 연동과 사용자 동의(opt-in) 절차가 필수입니다. 공식 API에는 인게임 채팅과 실시간 경기 상태도 없습니다. 대안을 **6장**에 정리했습니다. (추천: 게임 PC의 Riot 클라이언트 로컬 API를 주 경로로 쓰고, 공식 API는 나중에 붙일 수 있게 구조만 분리)

② 사이트를 누가, 어디서 보나요?
exe가 게임 PC에서 돌기 때문에 기본 구성에서는 게임 PC가 꺼지면 사이트도 함께 꺼집니다. 친구들이 밖에서도 보려면 터널(예: Cloudflare Tunnel)을 쓰거나, 기록을 별도 서버로 올려 사이트를 서버에서 띄워야 합니다.

③ 게임 PC의 GPU 사양
GPU 가속 STT(faster-whisper)는 NVIDIA(CUDA) 전용입니다. AMD·Intel GPU까지 지원하려면 whisper.cpp(Vulkan) 엔진을 추가로 붙여야 합니다. 게임과 STT가 같은 GPU를 나눠 쓰므로 VRAM 용량에 따라 모델 크기가 정해집니다.

목차

1. 요약
2. 요구사항 정리
3. 시스템 구성
4. 음성 처리 파이프라인 (음량 보정 · 잡음 · STT)
5. STT 모델 · GPU 자동 선택 · 단어사전
6. 발로란트 연동 (데이터 경로 · 시간 동기화)
7. 사이트 기능 · UI/UX 콘셉트
8. exe 설정 페이지
9. 데이터 저장 구조 · 용량
10. 기술 스택 · 개발 일정
11. 위험 요소 · 대응
12. 확인이 필요한 질문

02 요구사항 정리

요청 내용을 기능 단위로 다시 묶고 번호를 붙였습니다. 우선순위: 필수 1차 버전에 반드시 포함 · 권장 1차에 포함하되 범위 조정 가능 · 확장 2차 이후

ID	영역	요구사항	우선
F-01	디스코드 봇	설정한 봇 토큰으로 접속해 음성채널에 참여하고, 참여자별 음성을 따로 수신	필수
F-02	음성 저장	발화 단위로 실제 음성 파일을 저장하고 사이트에서 재생	필수
F-03	STT	GPU 기반 음성→텍스트 변환, 단어별 시각(word timestamp) 포함	필수
F-04	음량 보정	유저별 음량 차이를 보정해 같은 크기로 맞춘 뒤 STT	필수
F-05	잡음 표시	말이 아닌 소리(키보드, 숨소리, 게임 소리 등)는 [잡음]으로 표시	필수
F-06	단어사전	게임·일상·대학생·20대·발로란트 등 단어사전으로 인식률 향상	필수 (방식은 5장 확인)
F-07	시간 표시	경기 중에는 인게임 시간, 경기 밖에서는 현실 시간	필수
F-08	기록 목록	시간 · 화자 · 텍스트 표시, 클릭하면 해당 음성 재생	필수

F-09	따라 읽기 재생	재생 위치에 맞춰 단어 단위 강조, 단어 클릭 시 그 지점으로 이동, 다음 발화 자동 이어듣기	필수
F-10	필터	유저별, 게임별(+경기별, 날짜, 키워드 검색) 필터	필수
F-11	발로란트 연동	경기 시간, 킬·데스·어시, 라운드 결과, 점수, 인게임 채팅 등을 STT와 함께 표시	필수 (경로는 6장 확인)
F-12	모델 관리	모델 다운로드 위치·크기 선택, GPU와 게임 사양에 맞춘 자동 추천·다운로드	필수
F-13	설정 페이지	exe 안에서 봇 토큰 등 각종 설정 입력·저장	필수
F-14	UI/UX	창의적인 디자인, 애니메이션, 효과음 등 부가 효과	권장
F-15	멀티 게임	이후 다른 게임을 붙일 수 있도록 게임 연동부를 플러그인 구조로 분리	확장
F-16	사전 개선 루프	사이트에서 잘못 인식된 텍스트를 고치면 교정 사전에 자동 반영	권장

비기능 요구사항

ID	항목	내용
N-01	배포 형태	Windows 10/11 x64 포터블 실행 파일. 설치 과정이나 관리자 권한 없이 실행
N-02	게임 성능	STT가 게임 프레임을 떨어뜨리지 않도록 VRAM 예산 제한, 낮은 프로세스 우선순위, "경기 중 경량 / 경기 후 정밀" 모드 제공
N-03	보안	봇 토큰 등 비밀값은 Windows DPAPI로 암호화해 저장. 사이트 외부 공개 시 로그인 필수
N-04	개인정보	녹음 사실 안내(봇 입장 시 메시지·효과음), 녹음 제외 유저 설정, 보관 기간 자동 삭제
N-05	안정성	STT 처리가 밀려도 녹음은 계속되도록 녹음과 STT를 분리(대기열), 프로그램 재시작 시 미처리분 이어서 처리

03 시스템 구성

게임 PC에서 실행되는 하나의 포터블 앱 안에 네 부분이 들어갑니다. 녹음·게임 연동·웹 서버는 Node.js 코어가 맡고, GPU STT는 별도 Python 워커 프로세스가 맡습니다. STT 워커가 멈추거나 느려져도 녹음은 끊기지 않습니다.



왜 이렇게 나눴나

- **디스코드 음성 수신은 Node.js**: 2026년 3월부터 디스코드 음성은 종단간 암호화(DAVE)를 지원하지 않는 클라이언트·봇의 참여를 막습니다. 수신(receive)과 DAVE를 함께 지원하는 라이브러리 중 가장 검증된 것이 @discordjs/voice입니다.
- **STT는 Python 워커**: GPU STT 생태계(faster-whisper, CTranslate2, Silero VAD)가 Python 쪽이 가장 성숙합니다. 별도 프로세스라 GPU 메모리를 따로 관리하고 문제가 생기면 워커만 재시작할 수 있습니다.
- **CUDA 런타임과 모델은 첫 실행 때 내려받기**: CUDA 라이브러리(cuBLAS·cuDNN)만 수백 MB~1GB 이상이라 exe에 넣지 않습니다. exe 크기를 줄이고, NVIDIA가 아닌 PC에서는 내려받지 않습니다.
- **웹사이트는 앱이 직접 띄움**: 별도 서버 없이 `http://게임PC:포트`로 바로 열립니다. 외부 공개 방식은 12장 질문 Q2에 따라 정합니다.

> **포터블 exe 참고**
> Electron의 단일 파일 포터블 exe는 실행할 때마다 임시 폴더에 압축을 풀기 때문에 첫 화면이 몇 초 늦게 뜰 수 있습니다. 대안으로 "폴더째 들고 다니는 포터블(zip 압축 해제 후 exe 실행)" 형태도 가능합니다. 이 경우 실행이 빠르고 모델·기록을 같은 폴더에 둘 수 있습니다.

04 음성 처리 파이프라인

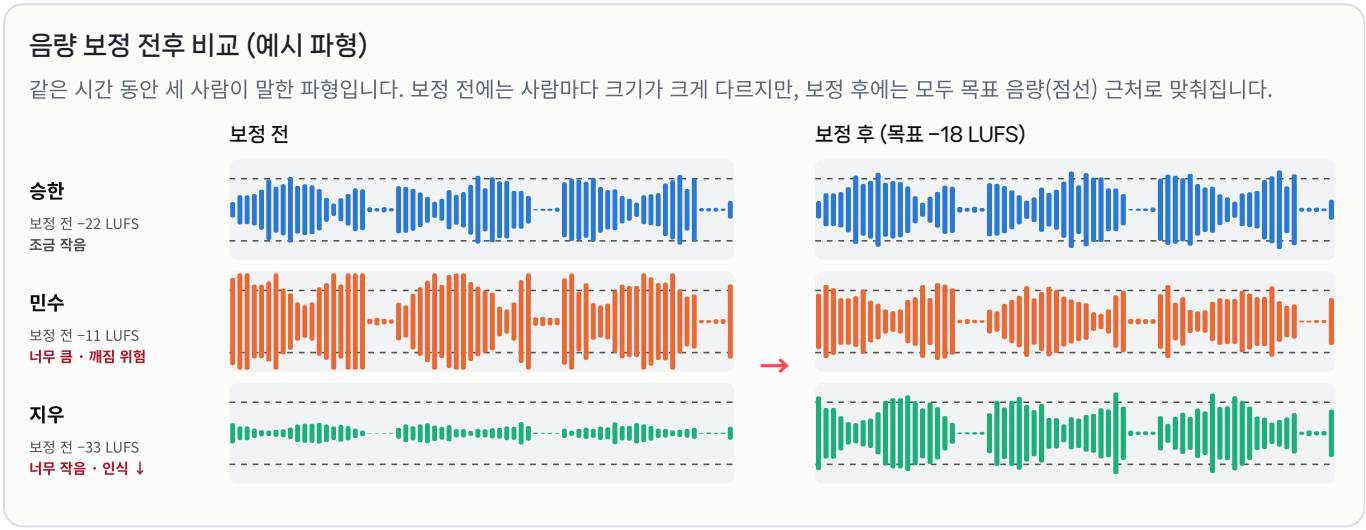


4-1. 유저별 수신과 화자 구분

디스코드는 음성채널 참가자마다 별도 스트림을 보내므로 화자 분리(diarization) 모델이 필요 없고, 누가 말했는지 100% 정확합니다. 여러 명이 동시에 말해도 서로 섞이지 않습니다. 발화 시작 시각은 PC 시계 기준 밀리초 단위로 기록합니다.

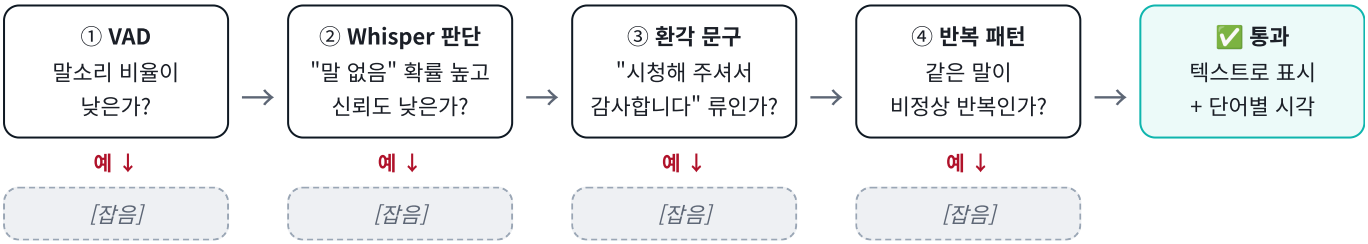
4-2. 음량 보정 (작은 목소리 · 큰 목소리)

- **발화 단위 라우드니스 정규화**: 각 발화의 체감 음량(LUFS)을 측정해 목표값(약 −18 LUFS)으로 끌어올리거나 낮춥니다. 단순 최대값(peak) 기준이 아니라 사람이 듣는 크기 기준이라 작은 목소리도 충분히 커집니다.
- **유저별 게인 프로필**: 유저마다 평균 음량을 누적 학습해, 매우 짧은 발화("어", "ㅇㅇ")처럼 측정이 불안정한 경우에도 그 유저의 평소 보정값을 씁니다.
- **리미터 · 클리핑 방지**: 크게 키울 때 소리가 깨지지 않도록 마지막에 리미터를 겁니다.
- **선택: 잡음 억제**: 마이크 잡음이 심한 유저를 위해 노이즈 억제(RNNoise 계열)를 STT 입력에만 선택적으로 적용합니다.
- **저장 정책**: 원본은 그대로 보관하고, 사이트 재생 시에는 정규화한 음량으로 들리게 합니다(재생 시 게인 적용 → 파일을 두 벌 저장할 필요 없음).



4-3. 잡음 판정 → [잡음]

발화 하나가 들어오면 아래 순서대로 검사합니다. 어느 단계에서든 "예"가 나오면 [잡음]으로 표시하고, 네 단계를 모두 통과해야 텍스트로 보여 줍니다.



②는 모델이 계산한 무음 확률(no_speech_prob)과 평균 로그 확률(avg_logprob)을 씁니다. ③은 한국어 Whisper가 키보드 소리·숨소리 같은 잡음에서 자주 지어내는 문장 목록입니다. 잡음으로 판정된 발화도 음성 파일은 남기고 목록에는 흐리게 표시하며, 사이트에서 "잡음 숨기기"로 감출 수 있습니다. 판정 기준값은 실제 녹음으로 조정합니다.

4-4. 실시간 처리와 게임 성능

녹음은 항상 실시간으로 하고, STT는 대기열로 처리합니다. 기본 모드는 **"경기 중 실시간 변환(경량 설정) → 경기 종료 후 같은 구간을 큰 모델로 다시 변환해 결과 교체"**입니다. 사용자가 "경기 후 일괄 처리만" 모드를 고르면 경기 중에는 GPU를 전혀 쓰지 않습니다.

05 STT 모델 · GPU 자동 선택 · 단어사전

5-1. 엔진

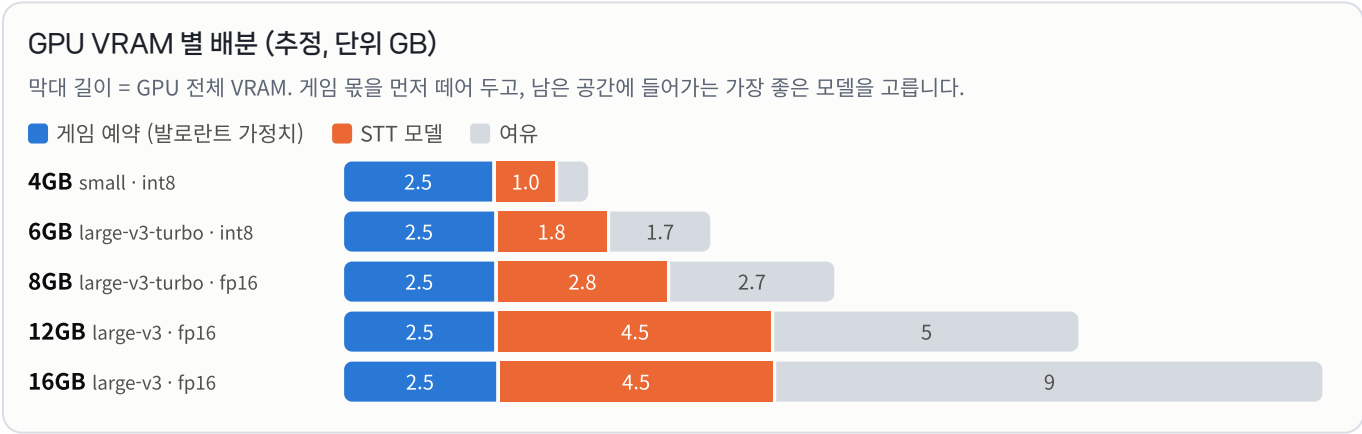
faster-whisper(CTranslate2 기반 Whisper 구현)를 기본 엔진으로 씁니다. 한국어 인식이 좋고, GPU에서 원본 Whisper 보다 빠르고 메모리를 적게 쓰며, 단어별 시각과 핫워드(hotwords) 기능을 지원합니다. 같은 크기의 한국어 특화 파인튜닝 모델이 있으면 선택지로 함께 제공합니다.

5-2. 자동 추천 규칙 (초안)

NVML로 GPU 이름과 VRAM을 읽고, 설정한 게임이 쓰는 VRAM을 미리 뺀 **"STT 예산"** 안에서 가장 좋은 모델을 고릅니다. 사용자는 추천을 무시하고 직접 고를 수 있습니다.

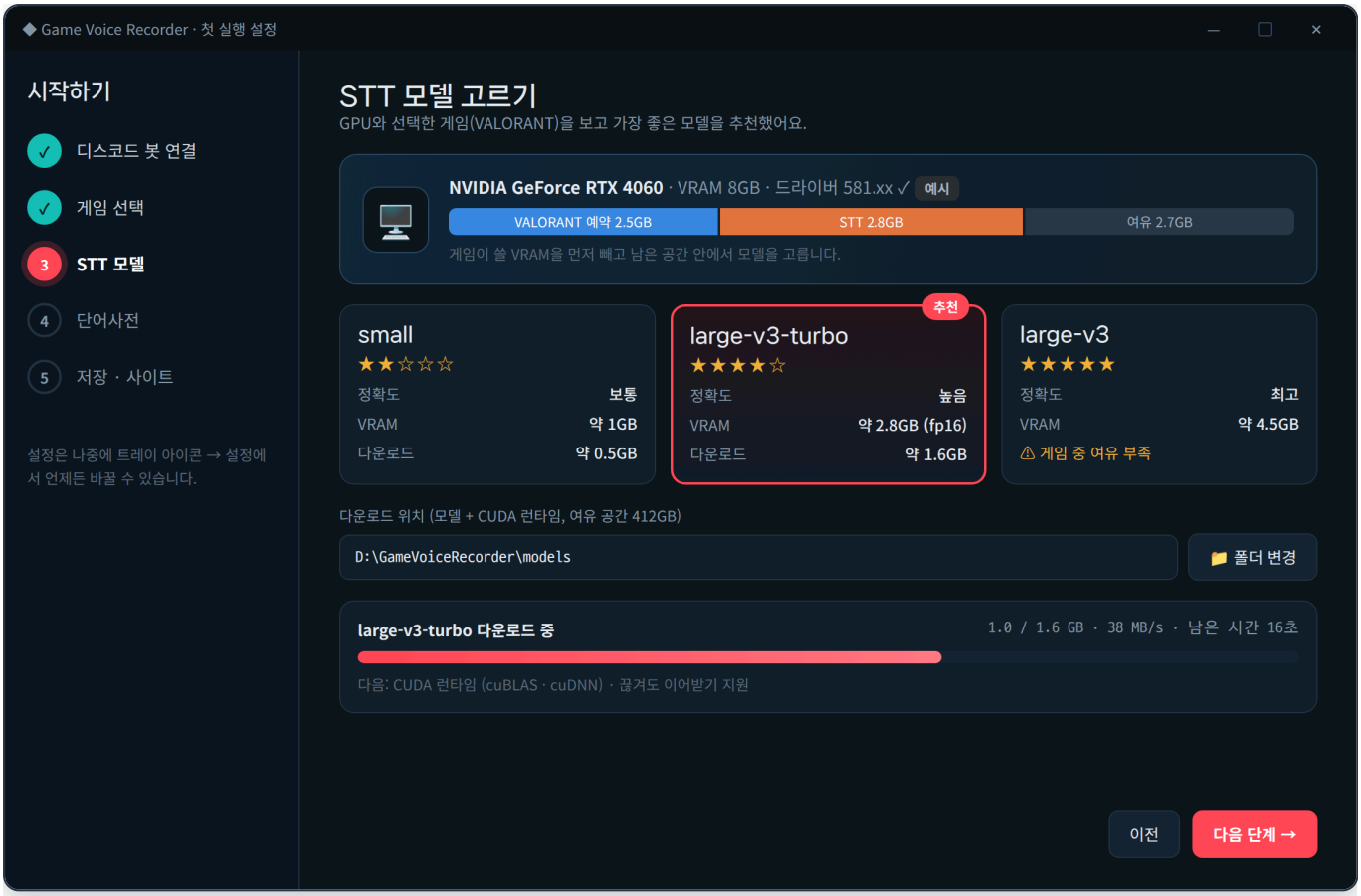
GPU VRAM (게임 PC)	추천 모델	연산 정밀도	STT 사용 VRAM (추정)	비고
NVIDIA 아님 / GPU 없음	small	int8 (CPU)	–	느림. 경기 후 일괄 처리 권장 (AMD·Intel GPU는 Q3 답변에 따라 whisper.cpp Vulkan 검토)
4GB	small	int8_float16	약 1GB	경기 후 large-v3-turbo로 재변환 가능
6GB	large-v3-turbo	int8_float16	약 1.5~2GB	대부분의 보급형 GPU
8~10GB	large-v3-turbo	float16	약 2.5~3GB	실시간 품질 · 속도 균형
12GB 이상	large-v3	float16	약 4.5GB	최고 정확도

게임별 VRAM 예약값은 게임 프로필로 관리합니다(발로란트는 비교적 가벼운 게임이라 2~3GB 정도를 우선 가정하고, 실측 후 조정). 수치는 모두 추정치입니다.



5-3. 모델 다운로드

- 첫 실행 마법사에서 **다운로드 폴더 선택 → GPU 감지 결과·추천 모델 표시 → 크기 선택 → 다운로드** 순서로 진행합니다.
- 이어받기, 해시 검사, 디스크 여유 공간 확인, 진행률·남은 시간 표시를 지원합니다.
- NVIDIA GPU가 있으면 CUDA 런타임(cuBLAS·cuDNN)도 같은 폴더에 함께 받습니다. 그래픽 드라이버 버전이 낮으면 업데이트 안내를 띄웁니다.



화면 예시 · 첫 실행 마법사 3단계 (STT 모델 고르기) GPU 감지 결과와 VRAM 배분, 추천 모델, 다운로드 위치와 진행률을 한 화면에서 보여 줍니다. 시안 · GPU 정보는 예시

5-4. 단어사전으로 인식률 높이기

"단어사전 학습"은 세 단계로 나눠 볼 수 있습니다. 1·2단계는 바로 적용할 수 있고 효과가 확실하며, 3단계(실제 모델 재학습)는 비용이 큰 별도 작업입니다.

단계	방식	비용 · 효과
1. 핫워드 주입	변환할 때마다 상황에 맞는 단어 목록을 모델에 힌트로 줍니다. 경기 중이면 발로란트 요원·맵·스킬·콜아웃, 경기 밖이면 일상·대학생 단어를 우선합니다.	즉시 적용 · 고유명사 인식 개선
2. 교정 사전	자주 틀리는 표현을 바로잡는 규칙입니다(예: "레이저" → "레이즈", "스파이크 설치" 표기 통일). 한글 자모 유사도로 사전 단어와 가까운 오인식을 찾아 고치고, 사이트에서 사용자가 고친 내용도 사전에 쌓입니다.	즉시 적용 · 사용할수록 개선
3. 파인튜닝 (선택)	단어사전 문장을 음성 합성(TTS)으로 만들고, 실제 녹음 중 사용자가 교정한 문장을 모아 LoRA 방식으로 모델을 추가 학습합니다. 학습은 게임 PC가 아닌 별도 환경에서 하고, 결과 모델을 다운로드 목록에 올립니다.	수일~수주 작업 · GPU 학습 필요

단어사전 적용 예시

같은 발화를 사전 없이 인식한 결과와 핫워드 + 교정 사전을 적용한 결과 비교 (예시)

사전 없음	"레이저 궁 빠졌어 비 싸이트 로태 하자 나 오퍼래이터 삼"
사전 적용	"**레이즈** 궁 빠졌어 **B 사이트** **로테** 하자 나 **오퍼레이터** 삼"

빨간 밑줄: 사전 없이 틀리기 쉬운 부분 · 초록 바탕: 핫워드와 교정 사전으로 바로잡힌 부분

기본 제공 사전 (예시)

발로란트

요원 제트, 레이즈, 세이지, 소바, 오멘, 킬조이, 사이퍼, 스카이, 요루, 네온, 페이드, 하버, 게코, 클로브 …
맵·콜아웃 어센트, 바인드, 헤이븐, 스플릿, 로터스, 선셋 … A사이트, 미드, 헤븐, 로비, 메인, 롱, 숏, 링크
용어 스파이크, 설치, 해체(디퓨즈), 에코, 풀바이, 세이브, 오퍼(오퍼레이터), 밴달, 팬텀, 셰리프, 원탭, 에이스, 클러치, 로테

게임 공통 · 일상 · 20대/대학생

게임 공통 킬, 데스, 어시, 힐, 쿨, 궁, 빽, 라인, 랭겜, 티어, 큐 돌리자, 트롤, 던짐, 캐리
일상·20대 레전드, 미쳤다, 개웃기네, 인정, 에바, 킹받네, 갓생, 존맛, 퇴근각
대학생 과제, 팀플, 중간고사, 기말, 휴강, 학식, 에타, 종강, 개강, 학점, 동아리, 엠티

기본 사전은 수천 단어 규모로 준비하고, 사이트·설정에서 직접 추가·수정할 수 있게 합니다. 실제 목록은 구현 단계에서 공식 게임 데이터(요원·맵·무기 이름)를 기반으로 자동 생성·갱신합니다.

06 발로란트 연동

6-1. 데이터를 얻는 세 가지 경로

	A. Riot 공식 API (VAL-MATCH-V1)	**B. Riot 클라이언트 로컬 API** (게임 PC에서 실행 중인 클라이언트)	**C. 서드파티 API** (예: HenrikDev)
발급 조건	어려움 프로덕션 키만 가능. 개인 키 신청 불가. 공개 서비스 + RSO 로그인 + 동의 절차 필수. 심사 1~3주 이상, 승인 불확실	불필요 게임 PC에서 로그인된 클라이언트의 인증 정보를 그대로 사용	보통 해당 서비스 키 발급 필요
정책 상태	공식 지원	비공식(문서화되지 않음). 읽기 전용 사용은 다수 트래커 앱이 쓰는 방식이지만 Riot이 보장하지 않음	비공식 제3자 서비스
경기 시작/종료 감지	× (종료 후 조회만)	○ 실시간	× (종료 후)
킬·데스·어시 (시각·위치·무기)	○ 경기 종료 후	○ 경기 종료 후 (경기 상세 조회)	○ 경기 종료 후
라운드 결과·경제·점수	○	○ (경기 중 점수는 실시간)	○
인게임 채팅	×	○ 이 PC 플레이어가 보는 채팅(팀·전체·파티)	×
키·인증 관리	앱 키 + 각 플레이어 RSO 로그인	없음 (클라이언트 실행 중일 때만 동작)	서비스 키

Riot 개발자 정책 요약 (developer.riotgames.com/docs/valorant)
· Personal Key 신청은 지원하지 않음 · 승인 불가 예시: "공개되지 않고 개인용으로만 설계된 앱", 경기 전 상대 전적 조회(스카우팅), "즉시 플레이에 영향을 주는 실시간 데이터 오버레이" · 모든 VALORANT 앱은 사용자 opt-in(RSO) 필수

> **추천안: B를 기본 경로로, A는 나중에 추가할 수 있게 구조만 분리**
> 요청하신 "인게임 채팅 · 실시간 경기 상태"는 공식 API에 아예 없는 데이터라 B 경로 없이는 구현할 수 없습니다. 게임 연동부를 "데이터 출처 어댑터"로 나눠, 공식 키를 받게 되면 A 어댑터로 바꿔 끼우는 구조로 만듭니다. 이 프로그램은 경기 중 플레이에 영향을 주는 정보를 띄우지 않고(오버레이 없음) 기록·복기용이므로, 데이터 사용 성격 자체는 Riot 정책의 "복기·학습" 범주에 해당합니다.

6-2. 수집 항목

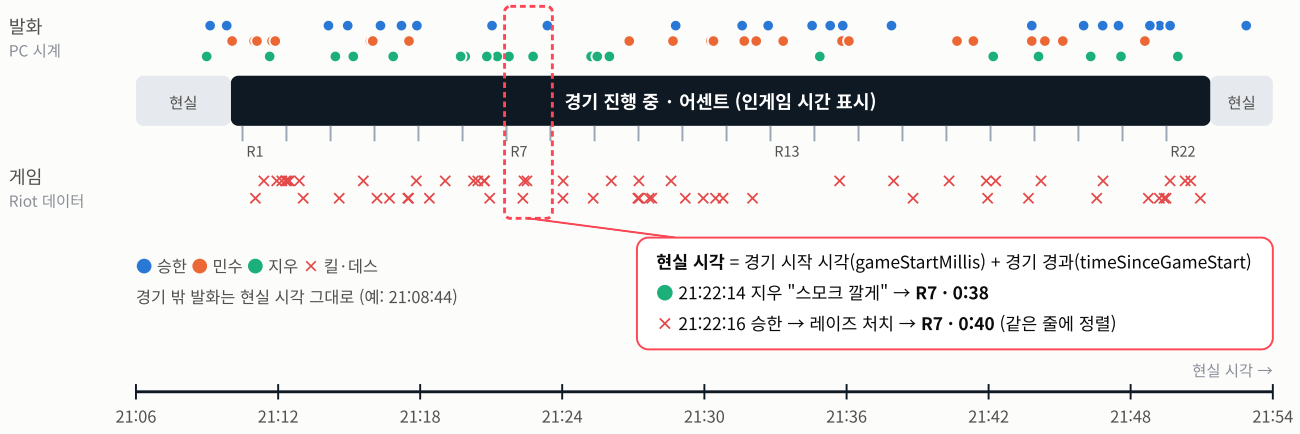
구분	항목
경기 정보	경기 ID, 맵, 모드(경쟁·일반·스파이크 돌격·사설 등), 시작 시각, 길이, 최종 점수, 승패
플레이어	10명의 Riot ID, 요원, 팀, 티어, K/D/A, 점수(ACS), 데미지, 헤드샷 비율
라운드	라운드 번호, 승리 팀·방식(처치/설치 폭발/해체/시간 초과), 스파이크 설치·해체 시각과 위치, 라운드별 경제(구매·잔액)
킬 이벤트	시각(경기 시작 기준·라운드 시작 기준), 킬러, 피해자, 어시스트, 무기·스킬, 피해자 위치와 당시 전원 위치
채팅	시각, 보낸 사람, 채널(팀/전체/파티), 내용

6-3. 시간 동기화 · 인게임 시간 표시

- 경기 데이터에는 **경기 시작 시각(Unix ms)**과 이벤트별 **경기 시작 후 경과 시간**이 있어, 둘을 더하면 현실 시각이 나옵니다. 음성 발화도 현실 시각으로 저장하므로 하나의 시간축으로 정렬할 수 있습니다.
- 로컬 API가 감지한 실제 경기 시작 시점과 비교해 PC 시계와 서버 시계의 차이를 자동 보정합니다.
- 경기 중 발화는 `R7 · 0:42`(7라운드, 라운드 시작 후 42초)처럼, 구매 단계는 `R7 · 구매`로 표시하는 안을 제안합니다. 표기 방식은 Q5에서 확인합니다.
- 킬·라운드 상세는 경기가 끝난 뒤 수 분 안에 조회되므로, **경기 중에는 발화와 채팅만 실시간으로 보이고, 경기 종료 후 킬 피드·라운드 정보가 채워지는 애니메이션**으로 표시합니다.

시간 동기화 도식

발화(위)와 경기 이벤트(아래)를 하나의 현실 시간축에 놓고, 경기 구간 안의 항목만 인게임 시간으로 바꿔 표시합니다. 점선 상자는 7라운드입니다.

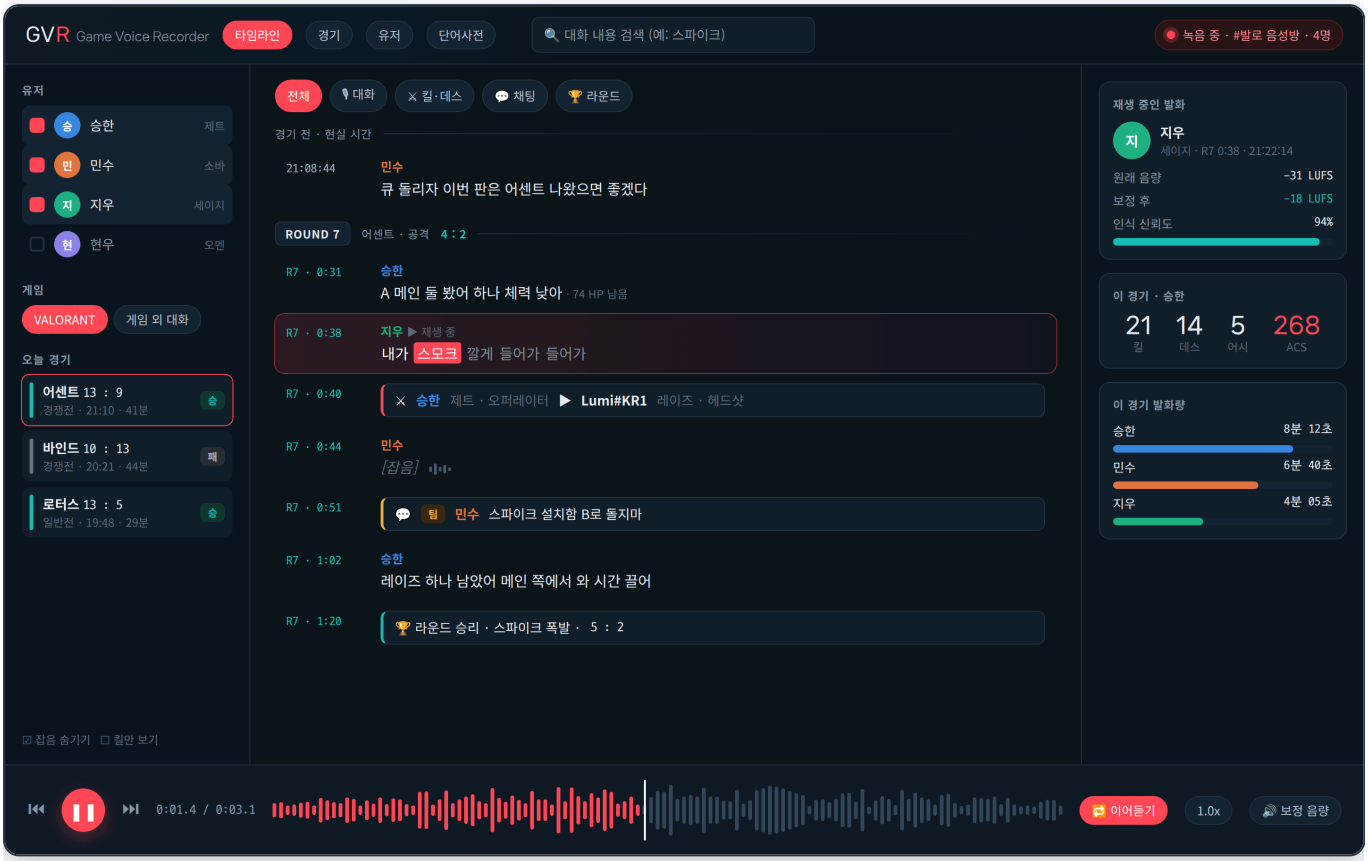


6-4. 디스코드 유저 ↔ Riot 계정 연결

설정에서 디스코드 유저와 Riot ID를 짝지어 두면, 그 사람의 발화 옆에 같은 순간의 킬·데스가 함께 표시되고 유저 페이지에서 "말한 내용 + 전적"을 같이 볼 수 있습니다. 경기 상세에는 같은 경기에 있던 10명 전원이 포함되므로, 게임 PC 한 대만으로 함께 플레이한 친구들의 기록까지 연결됩니다.

07 사이트 기능 · UI/UX 콘셉트

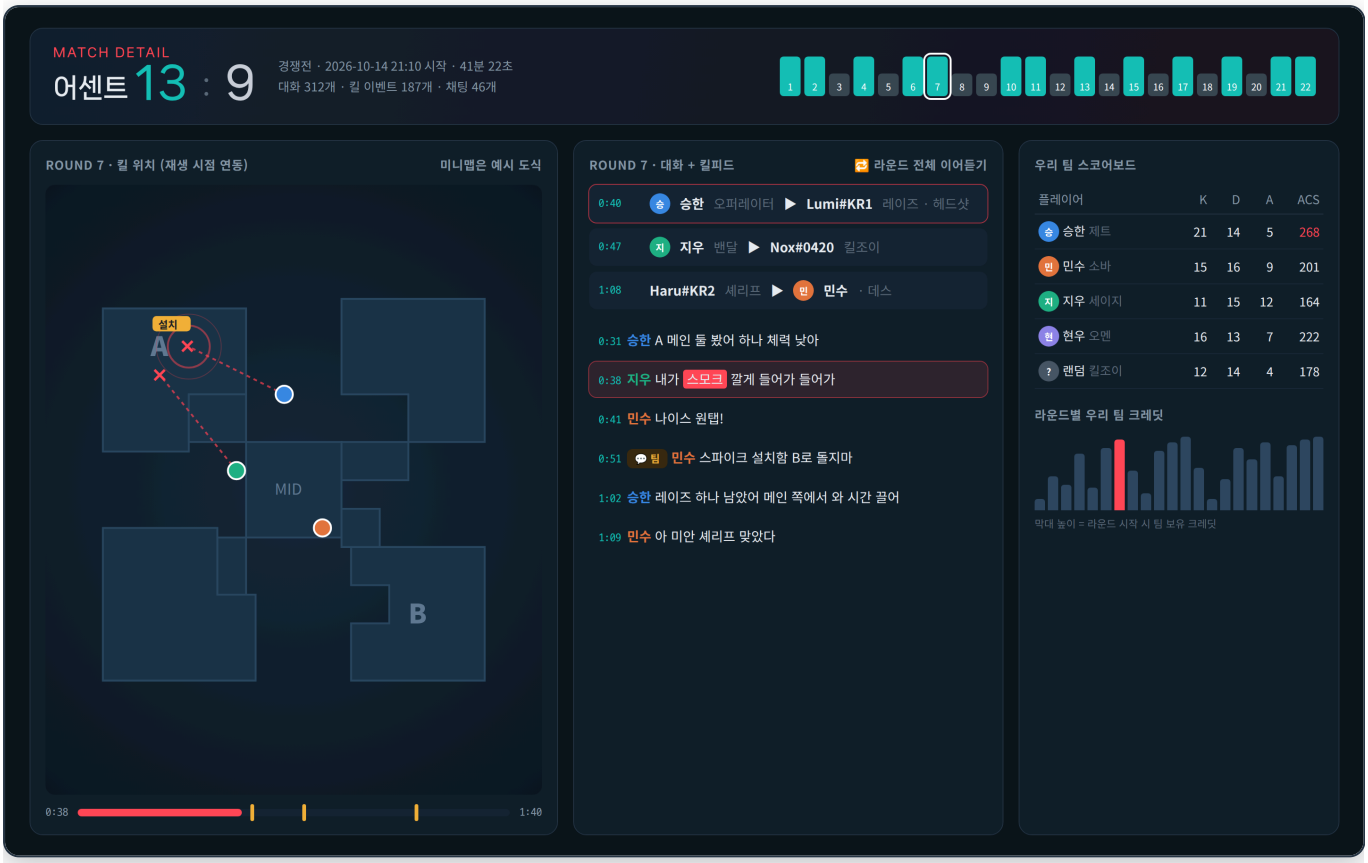
7-1. 메인 화면 (타임라인)



화면 예시 · 타임라인 왼쪽은 유저·게임·경기 필터, 가운데는 시간순 대화·킬·채팅·라운드 결과, 오른쪽은 재생 중인 발화 정보(보정 전후 음량, 인식 신뢰도)와 경기 요약입니다. 아래 플레이어에서 파형을 보며 들을 수 있고, 지금 재생 중인 단어("스모크")가 강조됩니다.

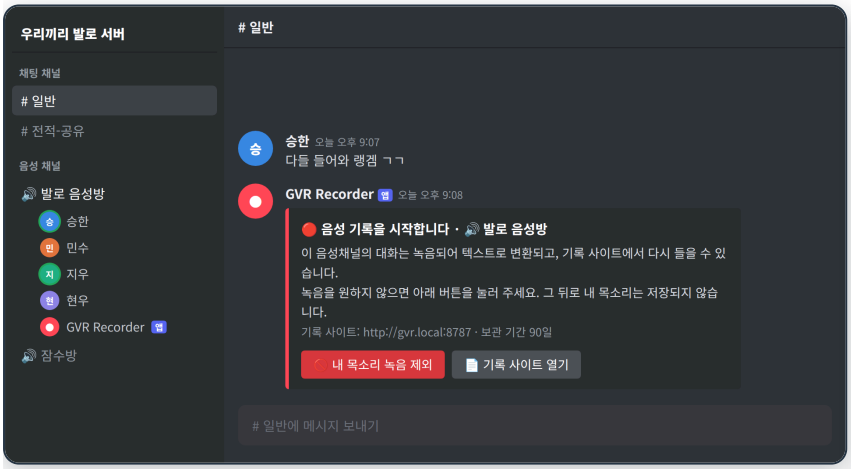
시안 · 이름과 데이터는 예시

7-2. 경기 상세 화면

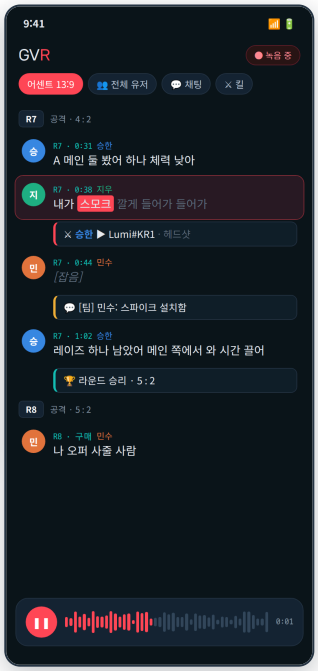


화면 예시 · 경기 상세 위쪽 라운드 띠(초록 = 승리)에서 라운드를 고르면 그 라운드의 킬 위치(미니맵), 대화와 킬피드, 스코어보드, 라운드별 크레딧이 함께 바뀝니다. 재생 위치에 맞춰 미니맵의 킬 표시가 움직입니다. 미니맵은 실제 맵이 아닌 도식

7-3. 모바일 화면 · 디스코드 녹음 안내



디스코드 안내 메시지 봇이 입장하면 녹음 사실을 알리고, 원하지 않는 사람은 버튼 하나로 녹음에서 빠질 수 있습니다.



휴대폰 화면 같은 사이트를 휴대폰에서도 볼 수 있습니다. 필터는 칩으로 접고, 플레이어는 아래에 고정합니다.

7-4. 주요 기능

- **따라 읽기 재생**: 문장을 누르면 그 발화부터 재생되고 재생 중인 단어가 강조됩니다. 단어를 누르면 그 단어 위치로 이동하고, "이어듣기"를 켜면 다음 발화로 자동으로 넘어갑니다. 여러 명이 겹쳐 말한 구간은 실제 순서대로 섞어서 재생하는 "현장 모드"도 제공합니다.

- **필터 · 검색**: 유저(복수 선택), 게임, 경기, 날짜 범위, 키워드(전문 검색), 이벤트 종류(발화/킬/채팅), 잡음 숨기기. 필터 상태가 URL에 남아 링크로 공유할 수 있습니다.
- **경기 상세**: 라운드별 점수 띠, 라운드를 누르면 그 라운드의 대화·킬·채팅만 보기. 미니맵에 킬 위치를 찍고 재생 시점에 맞춰 위치를 애니메이션합니다.
- **유저 페이지**: 말한 시간·횟수, 자주 쓰는 단어, 연결된 Riot 계정의 경기 전적.
- **텍스트 교정**: 잘못 인식된 문장을 바로 고치면 교정 사전에 반영됩니다(F-16).
- **실시간 보기**: 녹음 중이면 새 발화가 WebSocket으로 바로 올라옵니다(STT 처리 중에는 파형만 먼저 표시).

7-5. UI/UX 콘셉트: "택티컬 HUD 리플레이"

비주얼

짙은 남색 바탕, 레드·민트 포인트 색, 각진 카드와 얇은 HUD 선. 유저마다 고유 색과 요원 아이콘. 다크 기본, 라이트 테마 선택 가능. 게임마다 색 테마가 바뀌는 구조.

애니메이션

재생 중인 발화의 파형이 실시간으로 움직임 · 라운드 경계에서 화면을 가로지르는 와이프 전환 · 킬 이벤트 등장 시 짧은 플래시 · 경기 종료 후 킬피드가 시간 순으로 "채워지는" 연출 · 스크롤 시 타임라인 스냅.

효과음

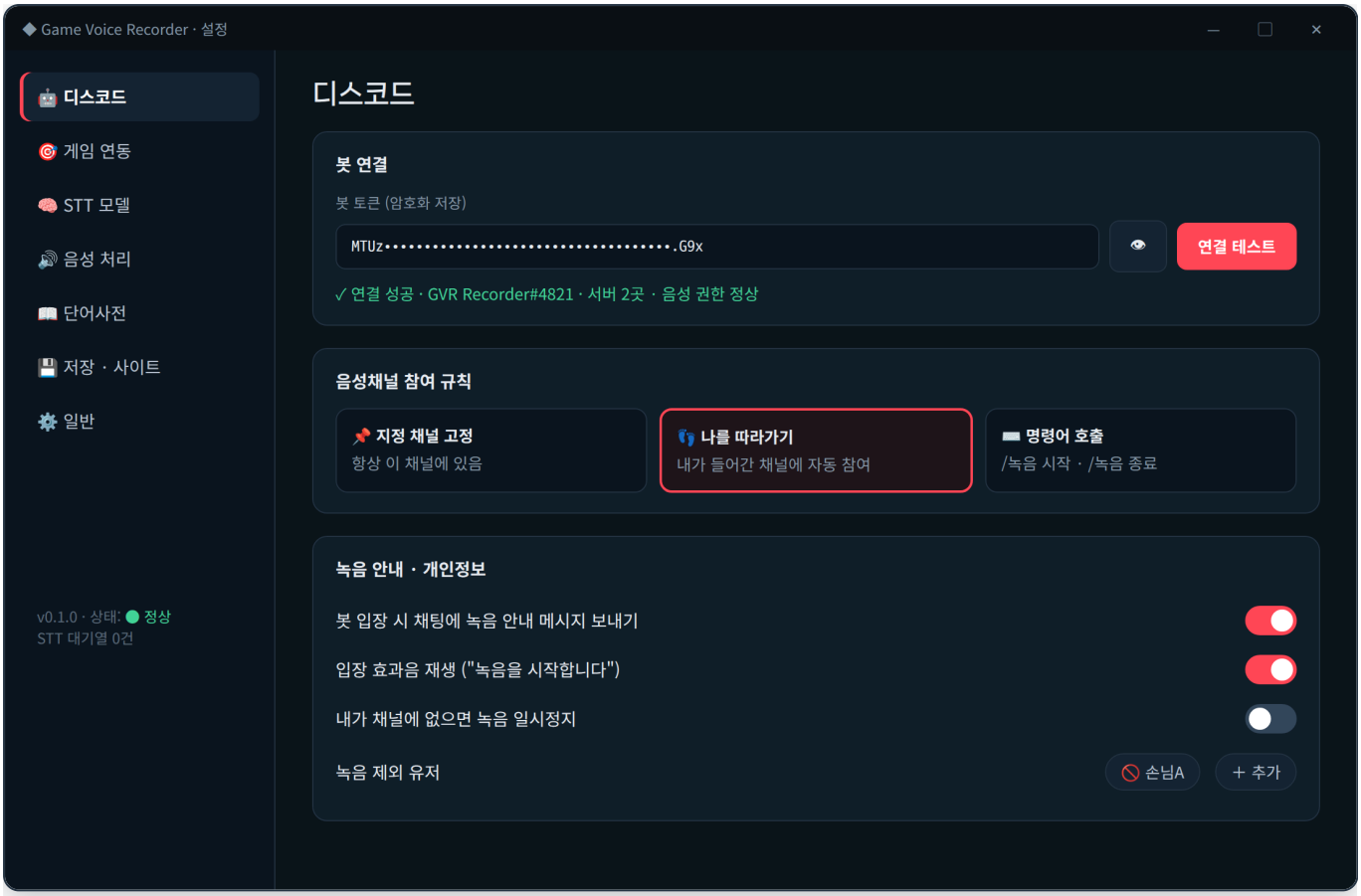
필터·버튼 클릭 틱, 재생 시작 효과음, 킬 이벤트 지나갈 때 짧은 효과음, 라운드 전환음, 새 발화 실시간 알림음. 전체 볼륨·종류별 켜기/끄기. 효과음은 직접 제작하거나 무료 라이선스 음원 사용(게임 원본 음원은 저작권상 사용하지 않음).

편의 · 접근성

단축키(스페이스 재생, ←/→ 이전·다음 발화, / 검색), "모션 줄이기" 설정 시 애니메이션 최소화, 모바일 화면 대응, 앱 셸에서는 녹음 상태를 트레이 아이콘 색으로 표시.

08 exe 설정 페이지

첫 실행 때는 마법사로 순서대로 안내하고, 이후에는 트레이 아이콘 → 설정 창에서 언제든 바꿀 수 있습니다. 비밀값은 Windows DPAPI로 암호화해 앱 폴더에 저장합니다.



화면 예시 · 설정 창 (디스코드 탭) 봇 토큰을 입력하고 연결을 테스트하며, 음성채널 참여 규칙과 녹음 안내·제외 유저를 정합니다. 시안

탭	항목
디스코드	봇 토큰(입력 후 연결 테스트), 대상 서버 선택, 참여 규칙(지정 채널 고정 / 내가 들어간 채널 따라가기 / 슬래시 명령으로 수동 호출), 녹음 안내 메시지·효과음, 녹음 제외 유저
게임 연동	사용할 게임(발로란트), 데이터 경로(로컬 클라이언트 / 공식 API 키), 연결 상태 표시, 디스코드 유저 ↔ Riot ID 매핑
STT 모델	감지된 GPU·VRAM·드라이버 표시, 추천 모델, 모델 크기 선택, 다운로드 폴더, 다운로드·삭제·교체, 처리 모드 (실시간 / 경기 후 정밀 재변환 / 경기 후 일괄)
음성 처리	목표 음량, 노이즈 억제 켜기/끄기, 잡음 판정 민감도, 발화 분할 무음 길이, 언어(한국어 / 자동)
단어사전	기본 사전 켜기/끄기(게임·일상·대학생·20대 등 묶음별), 사용자 단어 추가, 교정 규칙 편집, 사이트 교정 내역 반영 여부
저장 · 사이트	녹음 저장 폴더, 보관 기간(자동 삭제), 사용 용량 표시, 사이트 포트, 접속 비밀번호, 외부 접속 방식(Q2), 백업/내보내기
일반	윈도우 시작 시 자동 실행, 효과음·테마, 로그 보기, 업데이트 확인

09 데이터 저장 구조 · 용량

테이블	주요 컬럼
users	디스코드 ID, 표시 이름, 색, 아바타, 녹음 제외 여부, 게인 프로필
game_accounts	유저 ID, 게임, Riot PUUID, Riot ID

sessions	음성채널 세션 시작·종료, 서버·채널
utterances	화자, 시작·종료 시각(ms), 음성 파일 경로, 측정 음량, 텍스트, 단어별 시각(JSON), 잡음 여부, 신뢰도, 모델 이름, 경기 ID·라운드, 교정 여부
matches / rounds	경기 정보, 라운드 결과·경제·스파이크 이벤트
game_events	종류(킬·채팅 등), 현실 시각, 경기 시간, 관련 플레이어, 상세(JSON)
dictionary	단어, 묶음(발로란트·대학생 등), 교정 규칙, 출처(기본/사용자/사이트 교정)

≈14 MB
말한 시간 1시간
(Opus 32kbps)

≈60 MB
하루 사용량
4명 × 각자 1시간 발화

≈2 GB
한 달 사용량
매일 게임 기준

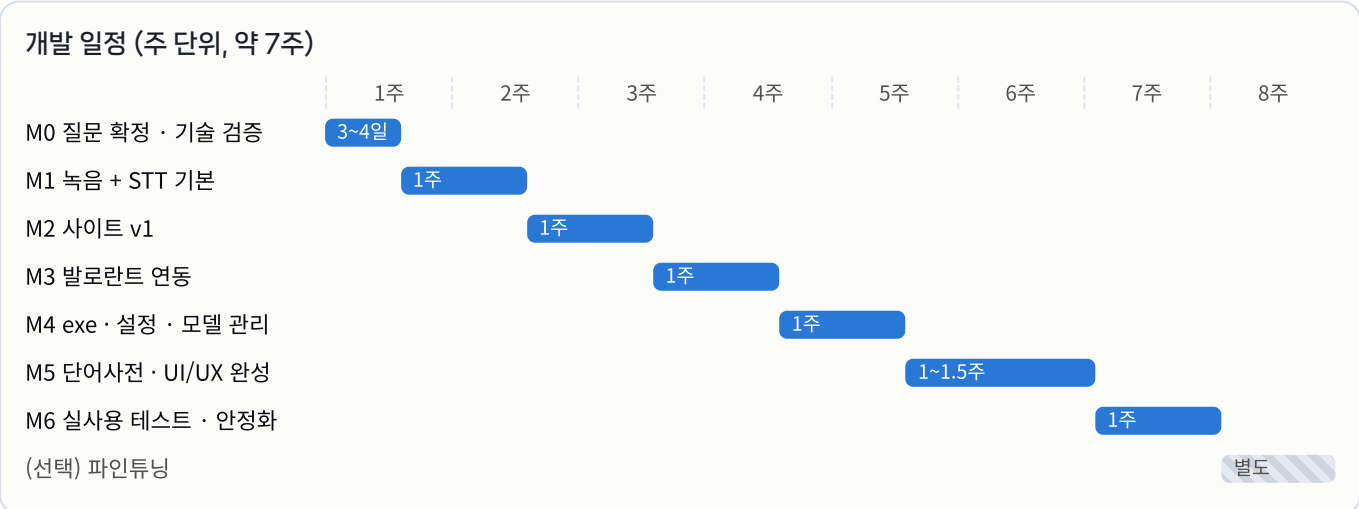
≈24 GB
1년 영구 보관 시
(텍스트·경기 데이터는 미미)

가정: 4명이 하루 3시간 함께 게임하고, 각자 실제로 말하는 시간이 하루 1시간 정도. 실제 사용량은 설정의 "사용 용량"에서 확인할 수 있습니다.

10 기술 스택 · 개발 일정

영역	선택	이유
앱 셸 · 패키징	Electron + electron-builder (portable)	설정 창·트레이·웹 UI 공유, 포터블 exe 지원
코어	Node.js + TypeScript	디스코드 음성 라이브러리 성숙도
디스코드	discord.js, @discordjs/voice (DAVE)	음성 수신 + 2026년 필수 E2EE 대응
웹 서버 · DB	Fastify, SQLite(better-sqlite3, FTS5)	설치 불필요, 단일 파일 DB, 한국어 전문 검색
웹 UI	React + Vite, Framer Motion, wavesurfer.js, Howler.js	애니메이션 · 파형 · 효과음
STT 워커	Python 3.11, faster-whisper, Silero VAD, pyloudnorm, PyInstaller	GPU STT · 음량 정규화 · 별도 프로세스 패키징
GPU 감지	NVML (pynvml)	GPU 이름·VRAM·드라이버 확인
빌드 · CI	Gitea Actions + Windows 빌드 환경	exe 자동 빌드 · 릴리스

일정 (초안, 답변 확정 후 조정)



단계	내용	완료 기준	예상
M0	질문 확정 · 기술 검증	DAVE 환경에서 유저별 음성 수신, 게임 PC GPU에서 STT 속도·VRAM 실측, 로컬 API로 경기 감지 확인	3~4일
M1	녹음 + STT 기본	봇 참여 → 발화 저장 → 음량 정규화 → STT → DB 저장 → 잡음 판정	1주
M2	사이트 v1	타임라인, 따라 읽기 재생, 유저·게임 필터, 검색, 실시간 갱신	1주
M3	발로란트 연동	경기 감지, 경기 상세·킬·라운드·채팅 수집, 시간 동기화, 인게임 시간 표시	1주
M4	exe · 설정 · 모델 관리	포터블 exe, 첫 실행 마법사, GPU 감지·모델 자동 다운로드, 설정 저장(암호화)	1주
M5	단어사전 · UI/UX 완성	핫워드·교정 사전, 사이트 교정 반영, 애니메이션·효과음·경기 상세·미니맵	1~1.5주
M6	실사용 테스트 · 안정화	실제 게임 세션으로 인식률·게임 프레임 영향 측정, 기준값 조정, 버그 수정	1주
(선택)	파인튜닝	TTS 합성 + 교정 데이터로 LoRA 학습, 모델 배포	별도

11 위험 요소 · 대응

위험	영향	대응
Riot 공식 API 키 미승인	"공식 API" 경로 불가	로컬 API 기본 경로 + 어댑터 구조로 나중에 교체 가능 (6장)
로컬 API 변경 · 정책	패치 후 일부 데이터 수집 실패, 비공식 경로의 정책 불확실성	읽기 전용만 사용, 메모리 접근·주입 없음, 실패해도 녹음·STT는 계속 동작, 엔드포인트를 한 곳에 모아 빠르게 수정
디스코드 음성 수신 변경	음성 수신은 디스코드가 공식 문서화하지 않은 기능	라이브러리 버전 고정 · 업데이트 추적, 수신 실패 감지 시 알림
게임 중 GPU 경합	게임 프레임 저하	VRAM 예산, 낮은 우선순위, 경기 중 경량 모드 / 경기 후 처리
Whisper 환각 · 짧은 발화 오인식	없는 문장 생성	VAD 선처리, 환각 문구 필터, 신뢰도 기준 [잡음] 처리
녹음 동의 · 개인정보	참가자 동의 없는 녹음 문제	입장 안내 메시지·효과음, 제외 유저, 보관 기간, 사이트 접근 제한

12 확인이 필요한 질문

Q1~Q4는 설계 자체가 달라지는 질문이라 구현 전에 꼭 답이 필요합니다. 나머지는 답이 없으면 추천안대로 진행합니다.

Q1 발로란트 데이터 경로를 어떻게 할까요?

공식 API는 개인용 앱에 키가 나오지 않고 인게임 채팅·실시간 상태도 제공하지 않습니다. (A) 공식 API 신청: 사이트를 공개 서비스로 만들고 RSO 로그인 구현, 승인 불확실 · (B) 게임 PC의 Riot 클라이언트 로컬 API (비공식, 읽기 전용, 채팅·실시간 가능) · (C) 서드파티 API

추천: B로 만들고 A로 교체할 수 있는 구조 유지

Q2 사이트는 누가 어디서 보나요?

(a) 게임 PC에서만 · (b) 같은 공유기 안의 기기(휴대폰 등) · (c) 친구들이 외부에서도 접속 → 터널(Cloudflare Tunnel 등)로 게임 PC를 공개하거나, 기록을 별도 서버로 올려 서버에서 사이트 운영. 게임 PC가 꺼져 있을 때도 사이트가 보여야 하나요?

추천: (c)가 필요하면 별도 서버 동기화 방식이 가장 안정적

Q3 게임 PC의 GPU 모델과 VRAM은 무엇인가요? AMD·Intel GPU 지원도 필요한가요?

NVIDIA면 faster-whisper(CUDA) 하나로 충분합니다. 다른 사람 PC에서도 쓸 예정이고 AMD·Intel GPU까지 지원해야 하면 whisper.cpp(Vulkan) 엔진을 추가로 붙여야 합니다.

Q4 경기 중에 사이트에서 바로 텍스트가 보여야 하나요?

(a) 경기 중 실시간 변환(GPU 일부 사용) + 경기 후 정밀 재변환 · (b) 경기 후 일괄 변환만(게임 중 GPU 사용 0)

추천: (a) 기본, (b)는 설정에서 선택

Q5 "인게임 시간"을 어떤 형식으로 보여 줄까요?

(a) 라운드 + 라운드 경과 시간 `R7 · 0:42` · (b) 경기 전체 경과 시간 `23:15` · (c) 둘 다

추천: (a) 기본 표시, 마우스를 올리면 (b)와 현실 시간 표시

Q6 "단어사전 학습"은 어디까지 원하시나요?

(a) 핫워드 주입 + 교정 사전(즉시 효과, 사용할수록 개선) · (b) 여기에 실제 모델 파인튜닝까지(학습용 GPU 작업 별도, 수일~수주)

추천: 1차는 (a), 실사용 데이터가 쌓이면 (b)

Q7 봇은 언제 음성채널에 들어가나요? 봇 토큰은 이미 있나요?

(a) 지정한 채널에 항상 · (b) 게임 PC 주인(본인)이 들어간 채널을 따라감 · (c) 슬래시 명령으로 호출. 사용할 디스코드 서버는 몇 개인가요? (봇 하나는 서버당 음성채널 하나만 참여 가능)

추천: (b) + (c)

Q8 녹음 동의 · 안내는 어떻게 할까요?

봇 입장 시 채팅 안내 + 짧은 효과음, 녹음을 원치 않는 유저 제외 기능을 넣을 계획입니다. 추가로 "본인이 채널에 없으면 녹음 중지" 같은 규칙이 필요한가요?

Q9 보관 기간과 저장 위치

녹음을 영구 보관하나요, 기간(예: 90일)이 지나면 자동 삭제하나요?

추천: 기본 영구 보관, 설정에서 기간 선택

Q10 사이트 로그인 방식

(a) 로그인 없음(게임 PC 전용일 때) · (b) 공용 비밀번호 · (c) 디스코드 로그인(그 서버 멤버만 접속, 본인 발화만 삭제 가능 등)

추천: 외부 공개 시 (c)

Q11 그 밖의 선호 사항

· 인식 언어: 한국어 고정인가요, 영어가 섞인 대화도 많나요? · 디자인: 발로란트 분위기를 살릴까요, 게임과 무관한 독자 디자인이 좋을까요? 참고하고 싶은 사이트·앱이 있나요? · 다음에 붙일 게임 후보(리그 오브 레전드 등)가 있나요?