FSD

1부: 백엔드 시스템 명세 (Backend System Specification)

CleanBoost App의 백엔드 시스템은 프론트엔드 앱(Electron)의 요청을 처리하고, 시스템의 실시간 성능 데이터를 수집하며, 디스크 정리 로직을 수행하는 핵심 엔진 역할을 담당합니다.

1.1. 백엔드 아키텍처 및 기술 스택

구분	기술 스택	역할
핵심 언어	Python	FastAPI와 Python-SocketIO를 사용한 API 서버 구축 및 비즈니스 로직 처리.
웹 프레임워크	FastAPI, Uvicorn/Gunicorn	고성능 REST API 엔드포인트 제공 및 Uvicorn을 통한 비동기 처리.
실시간 통신	Python-SocketIO / Flask-SocketIO	FR-1 (실시간 모니터링)을 위한 WebSocket 기반 양방향 통신 구현.
데이터 수집	psutil 및 OS 레벨 API (필요시 C/C++ 확장 모듈)	CPU, RAM, GPU, Storage 사용량 등 시스템 자원 데이터를 안전하게 수집.

1.2. 기능 명세 (Functional Specifications)

FR-1: 실시간 시스템 모니터링 지원

항목 ID	명세 (Specification)	상세 내용
BE-FR-1.1	SocketIO 연결 관리	클라이언트 앱(Electron)과의 지속적인 WebSocket (Socket.IO) 연결을 관리하고 인증(IPC 채널 인증)을 확인합니다.
BE-FR-1.2	시스템 데이터 수집	**100ms 이내**의 지연 목표를 달성하기 위해 최적화된 저수준 API (예: psutil 또는 C 확장)를 사용하여 CPU, RAM, GPU, Storage 데이터를 수집합니다.
BE-FR-1.3	실시간 데이터 푸시	BE-FR-1.2에서 수집된 데이터를 1초 단위로 SocketIO 채널을 통해 Electron 클라이언트에 **푸시(Push)** 전송합니다.
BE-FR-1.4	임계치 경고 트리거	CPU 90% 이상 등 임계치 초과 상황을 탐지할 경우, 즉시 별도의 SocketIO 이벤트(warning_trigger)를 통해 경고문을 클라이언트에 전송합니다.

FR-2: 잔여물 스캔 및 삭제 지원

항목 ID	명세 (Specification)	상세 내용
BE-FR-2.1	스캔 요청 API	REST API 엔드포인트 (/scan/start)를 통해 클라이언트로부터 잔여물 스캔 요청을 수신합니다.
BE-FR-2.2	스캔 로직 수행	**FR-2에 명시된 주요 경로** (%TEMP%, 브라우저 캐시, 휴지통, 로그 파일 등)에 대해 **30초 이내**로 파일 목록과 크기를 계산합니다.
BE-FR-2.3	용량 계산 및 응답	스캔 완료 후, 각 항목별 **확보 가능 용량** (XX GB 확보 가능)을 정확하게 계산하여 클라이언트에 JSON 형태로 응답합니다.
BE-FR-2.4	삭제 요청 API	REST API 엔드포인트 (/clean/execute)를 통해 클라이언트가 선택한 항목 목록을 수신합니다.
BE-FR-2.5	안전 삭제 실행	삭제 전 **안전성 검증**을 다시 수행하고, 사용자 데이터 및 시스템 핵심 파일을 건드리지 않도록 경로 유효성을 최종 확인한 후, **관리자 권한으로 파일 삭제**를 실행합니다.
BE-FR-2.6	삭제 결과 반환	삭제 완료 후, **실제 삭제된 파일 개수**와 **확보된 최종 용량**을 클라이언트에 반환합니다.

1.3. 비기능 요구사항 명세 (Non-Functional Specifications)

성능 (Performance)

항목 ID	명세 (Specification)	상세 목표
BE-NFR-1.1	저부하 설계	FastAPI와 Python 데이터 수집 모듈은 평균 **CPU 점유율 1% 미만**을 유지하도록 설계합니다. (이를 위해 I/O 바운드 작업에 최적화된 비동기 처리 및 저수준 라이브러리 활용 필수)
BE-NFR-1.2	데이터 지연	시스템 데이터 수집 및 SocketIO 푸시 지연 시간은 **100ms 이하**를 보장합니다.
BE-NFR-1.3	파일 스캔 속도	모든 잔여물 유형에 대한 전체 스캔 작업은 **30초 이내**에 완료되어야 합니다.

보안 및 안정성 ($\text{Security & Stability}$)

항목 ID	명세 (Specification)	상세 내용
BE-NFR-2.1	권한 최소화 (POLP)	파일 스캔 및 삭제는 **관리자 권한**이 필요한 작업이며, 이 권한은 FastAPI 백엔드 프로세스 또는 BE가 호출하는 C 확장 모듈에서 **최소한의 작업**에만 사용되도록 설계합니다.
BE-NFR-2.2	IPC 채널 보안	TLS/SSL 대신, 로컬 SocketIO 채널은 UUID 또는 **프로세스 ID 기반 인증 메커니즘**을 사용하여 외부 접근을 차단합니다.

| **BE-NFR-2.3** | 파일 접근 제한 | FR-2에 명시된 **안전 경로(Temp, Cache, Log)** 외의 시스템 핵심 경로 또는 사용자 개인 폴더에 대한 접근을 엄격히 제한하고 로그를 기록합니다. |
| **BE-NFR-2.4** | 입력 검증 | 모든 REST API 입력값 및 SocketIO 메시지에 대해 **파일 경로 유효성 검사** 및 Pydantic을 이용한 데이터 형식 검증을 철저히 수행하여 보안 취약점을 방지합니다. |

2부: 프론트엔드 시스템 명세 (Frontend System Specification)

이어서 프론트엔드(Electron/React.js) 시스템 명세를 작성하겠습니다.

2.1. 프론트엔드 아키텍처 및 기술 스택

구분	기술 스택	역할
앱 플랫폼	Electron	Windows, macOS 등 OS 독립적인 데스크톱 환경 제공. Node.js 환경을 통해 백엔드와 통신.
프론트엔드	React.js	FR 구현을 위한 UI/UX 컴포넌트 개발 및 상태 관리.
스타일/디자인	TailwindCSS	'직관적 UI' 요구사항 충족을 위한 빠르고 반응성이 뛰어난 스타일링.
데이터 시각화	Chart.js, D3.js	FR-1 시스템 모니터링을 위한 원형/막대 그래프 및 실시간 데이터 업데이트 처리.
통신	SocketIO Client	BE에서 푸시되는 실시간 모니터링 데이터를 수신 및 처리.

2.2. 기능 명세 (Functional Specifications)

FR-1: 실시간 시스템 모니터링 표시

항목 ID	명세 (Specification)	상세 내용
FE-FR-1.1	SocketIO 연결	앱 실행 시 BE 서버에 SocketIO 연결을 초기화하고, 연결 상태를 관리합니다.
FE-FR-1.2	실시간 그래프 표시	BE로부터 1초마다 수신되는 CPU, RAM, GPU, Storage 데이터를 Chart.js 또는 D3.js를 사용하여 **원형 및 막대 그래프** 형태로 시각화합니다.
FE-FR-1.3	수치 동시 제공	그래프와 함께 해당 자원의 **현재 수치** (% 또는

GB 단위)를 동시에 표시하여 정확도를 높입니다. |
| **FE-FR-1.4** | 임계치 경고 UI | BE로부터 $\text{CPU 90\%}$ 초과 경고(warning_trigger) 메시지를 수신하면, UI 상단에 **경고문**을 명확하게 표시합니다. (예: 빨간색 바/배너) |

FR-2: 잔여물 스캔 및 삭제 사용자 경험

항목 ID	명세 (Specification)	상세 내용
FE-FR-2.1	스캔 시작 UI	'스캔 시작' 버튼을 중앙에 배치하고, 스캔 중에는 **30초 이내**로 완료될 수 있도록 진행률 표시줄(Progress Bar)을 제공합니다.
FE-FR-2.2	스캔 결과 시각화	스캔 완료 후 BE로부터 수신된 데이터를 바탕으로 FR-2의 두 가지 유형(대용량 임시 파일, 프로그램 잔여물)을 분류하여 표시합니다.
FE-FR-2.3	용량 확보 UI	'대용량 임시 파일' 섹션에는 BE가 계산한 총 **"XX GB 확보 가능"** 문구를 강조하여 표시합니다.
FE-FR-2.4	선택적 삭제 제어	'프로그램 데이터 잔여물'에 대해 사용자가 항목별로 삭제 여부를 **체크박스로 선택**할 수 있도록 UI를 제공합니다.
FE-FR-2.5	삭제 전 경고	'정리하기' 버튼 클릭 시, **"삭제 전 사용자 승인 및 되돌릴 수 없음 경고"**에 대한 맞춤형 모달 창을 띄웁니다. (alert() 사용 금지)
FE-FR-2.6	삭제 완료 피드백	삭제 완료 후 BE로부터 받은 결과 데이터(삭제 항목, 확보 용량)를 바탕으로 **안심할 수 있는 피드백 메시지**를 제공합니다.

2.3. 비기능 요구사항 명세 (Non-Functional Specifications)

항목 ID	명세 (Specification)	상세 목표
FE-NFR-1.1	UI 응답성	모든 UI 상호작용은 **100ms 이내**의 반응 속도를 보장합니다.
FE-NFR-1.2	직관적 UI	TailwindCSS와 Lucide/Font-Awesome 등의 아이콘을 활용하여 '직관적 UI'를 구현하며, **기술 용어 사용을 최소화**합니다.
FE-NFR-1.3	오프라인 모드	네트워크 연결이 없어도 FR-2 잔여물 스캔 및 삭제 기능은 정상적으로 동작해야 하며, FR-1 모니터링은 로컬 데이터 수집 결과만 표시해야 합니다.

API 명세서

1. REST API 명세 (스캔 및 삭제 명령)

REST API는 클라이언트의 **요청(스캔 시작, 삭제 실행)**과 서버의 **응답**이 명확한 비실시간 작업에 사용됩니다.

기능	메소드	URL	REQUEST (요청 본문)	RESPONSE (성공 200 OK)	에러문 (실패 400/500 에러)
잔여물 스캔 시작	POST	/api/v1/scan/start	Headers: Authorization (인증 UUID)	{ "message": "스캔 요청이 시작되었습니다.", "scanId": "1a2b3c4d-..." }	403 Forbidden: {"error": "인증 토큰이 유효하지 않습니다."}
					500 Internal Error: {"error": "스캔 엔진 초기화에 실패했습니다."}
스캔 결과 조회	GET	/api/v1/scan/results	Headers: Authorization	{ "status": "SUCCESS", "total_scannable_size": 5368709120, "scan_results": [{ "type": "TEMP_FILES", "name": "브라우저 캐시", "size": 1073741824 }] }	404 Not Found: {"error": "진행 중인 스캔 결과를 찾을 수 없습니다."}
선택 항목 삭제 실행	POST	/api/v1/clean/execute	{ "items_to_delete": [{ "type": "TEMP_FILES", "path": "C:tempcache..." }] }	{ "message": "삭제 작업이 완료되었습니다.", "total_cleaned_size": 5368709120, "deleted_count": 150, "error_count": 0 }	400 Bad Request: {"error": "유효하지 않은 파일 경로가 포함되어 있습니다."}

---

2. Socket.IO 명세 (실시간 모니터링 및 경고)

Socket.IO는 FR-1 (실시간 시스템 모니터링) 데이터를 서버(BE)에서 클라이언트(FE)로 푸시하는 **양방향 통신**에 사용됩니다.

A. 클라이언트 → 서버 (제어 및 Ping)

클라이언트가 서버로 명령을 보내는 이벤트입니다.

기능	이벤트명	REQUEST (전송 데이터)	용도
모니터링 시작/중지	monitor_toggle	{"active": true/false}	BE 엔진의 데이터 수집 루프 시작/중지 제어
연결 상태 확인	ping	{"timestamp": 1678886400}	연결 활성 상태 유지 및 지연 시간 측정을 위한 Ping 요청

B. 서버 → 클라이언트 (데이터 푸시 및 경고)

서버가 클라이언트에게 데이터를 푸시하는 이벤트입니다. (지연 시간 100ms 이하 목표)

기능	이벤트명	RESPONSE (수신 데이터)	푸시 주기/조건
실시간 시스템 상태	system_status	{"cpu_percent": 35.5, "ram_used_gb": 8.2, "gpu_percent": 12.0, "storage_usage": 78.5}	1초마다 주기적 푸시
임계치 경고 알림	warning_trigger	{"type": "CPU_OVERLOAD", "message": "CPU 사용량이 90%를 초과했습니다."}	CPU 90% 초과 등 임계치 탐지 시 즉시
연결 확인 응답	pong	{"timestamp": 1678886400}	