

Khairon 사용 매뉴얼

Khairon은 SpiderGen 기반 Electron 앱으로, 로컬/원격 LLM과 대화하며 프로젝트 코드를 읽고, 검색하고, 수정하고, 빌드/테스트까지 실행할 수 있는 AI 개발 어시스턴트입니다.

이 문서는 코드 기준(2026-10-14)으로 작성되었습니다. 실제 UI 라벨/동작은 빌드 버전에 따라 다를 수 있습니다.

1. 화면 구성

앱을 실행하면 다음과 같은 구조로 보입니다.

- **Session List (좌측 도킹 패널)** — 지금까지 만든 세션들이 프로젝트 경로 단위로 트리 형태로 표시됩니다.
`View > Session List` 메뉴로 표시/숨김을 토글할 수 있습니다.
- **Chat Window (중앙, 탭 형태)** — 세션 하나당 탭 하나가 열립니다.
 - 상단: 세션 이름, 토큰 사용량, AI Provider(모델) 선택 드롭다운
 - 중앙: 사용자/AI/시스템 메시지가 쌓이는 대화창. AI가 파일을 수정하면 "N개 파일 수정 완료" 표시와 함께 **Undo** 버튼이 나타나고, 그 표시를 누르면 바뀐 파일 목록이 펼쳐집니다(3절 참고).
 - 하단: 작업 폴더(cwd) 표시줄과 메시지 입력창(첨부, 전송/중단 버튼 포함)
- **코드 뷰어/에디터 탭 (MDI)** — 채팅 중 파일 경로(툴 카드/편집 카드)를 클릭하면 Monaco 기반 뷰어가 별도 탭으로 열립니다(5절 참고).
- **PreChat List (우측 도킹 패널)** — 채팅으로 보내기 전에 메시지를 미리 적어 두는 목록입니다. `View > PreChat List` 메뉴로 표시/숨김을 토글합니다(3-1절 참고).

메뉴바

메뉴	항목	설명
File	New Session	새 세션(대화) 생성
	Open File... / Save File / Close File	파일 열기·저장·닫기 (5절의 코드 뷰어/에디터에서 사용)
	Exit	앱 종료
View	Session List	세션 목록 패널 표시/숨김
	PreChat List	예비 채팅 리스트 패널 표시/숨김 (3-1절 참고)

메뉴	항목	설명
Build	Build Project / Simulate Project	SpiderGen 프로젝트 빌드 및 시뮬레이터 실행
Option	Settings...	General / Lint / Indexing / Logging 탭의 앱·서버 설정 (화면 테마 포함 — 아래 참고)
	AI Model...	LLM Provider(모델) 등록/편집, OAuth 로그인
	MCP Servers...	외부 MCP 서버 등록/편집 — AI 가 쓸 외부 도구 연결 (14절 참고)
Help	About	버전 확인 — Version(설치본 버전)과 Release : AS-빌드날짜-패치번호 표시. 문의·버그 신고 때 이 Release 줄을 알려 주세요 (15절 참고)

화면 테마

Option > Settings... 의 General 탭 테마 에서 화면 색을 고를 수 있습니다 — black (기본 · 어두운 배경) · green (밝은 배경의 초록 계열). OK 를 누르면 재시작 없이 바로 바뀌고, 코드 뷰어/에디터(5절) 색도 함께 바뀝니다. 테마를 건드리지 않고 OK 를 누르면 기존 선택이 그대로 유지됩니다.

2. 세션과 작업 폴더(cwd)

한 세션 = 한 탭 = 하나의 작업 폴더(cwd) + 하나의 대화 이력입니다. Claude Code의 "한 세션당 하나의 cwd" 모델과 동일합니다.

- 1. File > New Session 으로 새 세션을 만듭니다.
- 2. 채팅 탭 하단의 폴더 선택 버튼을 눌러 작업할 프로젝트/코드 폴더를 지정합니다.
- 3. 처음 선택하는 폴더는 신뢰 확인 다이얼로그가 뜹니다. "Yes"를 선택하면 해당 폴더가 trusted-folders.json 에 기록되어, 다음부터는 같은 폴더를 다시 신뢰 확인 없이 사용할 수 있습니다.
- 4. 폴더를 신뢰하고 나면, Option > Settings... > Indexing 에서 인덱싱을 켜 둔 경우 폴더 안 .aiassist/samples 자료의 인덱싱이 시작됩니다 (4절 참고). 프로젝트 종류와 무관하게 동작합니다. 프로젝트 소스 코드는 인덱싱하지 않으며 AI가 검색 도구로 직접 찾습니다.
- 5. 주의: 대화가 한 번 시작되면(cwd가 잠기면) 같은 세션 안에서는 작업 폴더를 바꿀 수 없습니다. 다른 폴더로 작업하려면 새 세션을 만드세요.
- 6. 폴더 안에 *.prj 파일이 있으면 SpiderGen 프로젝트로 자동 인식되어 폴더 표시줄에 [SpiderGen] 마크가 붙고, Build / Simulate 메뉴가 활성화됩니다.
- 7. 코드가 없는 폴더도 됩니다. 회의록·기획서·정의서 같은 문서(.md / .txt / .csv 와 .xlsx / .pptx / .docx)만 든 폴더를 작업 폴더로 잡고 "이 폴더 문서 전부 읽고 요약해줘"처럼 요청할 수 있습니다. 문서 안의 텍스트 검색도 됩니다(5-1절 참고).

세션 목록(좌측 패널)에서 세션 항목을 더블클릭하면 해당 세션의 대화 이력을 불러와 이어서 대화할 수 있습니다. 모든 메시지는 보낼 때마다 자동으로 로컬에 저장되므로 별도의 "저장" 조작이 필요 없습니다.

세션 이름 바꾸기 — 세션 목록의 항목을 **우클릭** → **Rename**(또는 채팅 탭을 **우클릭** → **Rename Session**)으로 세션에 원하는 이름을 붙일 수 있습니다. 이름을 지정하면 세션 목록·채팅 탭·세션 라벨에 모두 그 이름이 표시됩니다. 입력력을 **비운 채 저장하면 지정 이름이 지워지고** 원래의 자동 이름(첫 메시지 미리보기)으로 돌아갑니다.

2-1. 세션 보관(Archive)

다 쓴 세션을 지우지 않고 목록에서 접어 두는 기능입니다. 보관해도 대화 내용은 그대로 남아 있어 언제든지 다시 열어 볼 수 있습니다.

- **보관하기** — 세션 목록에서 세션을 **우클릭** → **Archive**. 폴더(프로젝트) 노드를 **우클릭** → **Archive All Sessions** 하면 그 폴더의 세션을 한 번에 보관합니다. 확인 창 없이 바로 처리되며 언제든지 되돌릴 수 있습니다.
- **보관한 세션 보기** — 트리 맨 아래의 **Archived** 노드 아래에 프로젝트별로 모입니다. 평소에는 접혀 있고, 펼치면 평소처럼 더블클릭해서 열 수 있습니다.
- **검색** — 트리 위의 **Search sessions** 입력창에 글자를 넣으면 제목(이름을 지정하지 않았다면 첫 메시지 미리보기)에 그 글자가 든 세션만 남습니다. 보관된 세션이 걸리면 **Archived** 쪽이 자동으로 펼쳐집니다. 입력창을 비우면 원래대로 돌아옵니다.
- **보관 풀기** — 보관된 세션을 **우클릭** → **Unarchive**. 또는 보관된 세션을 열어 메시지를 한 번 보내면 **자동으로 보관이 풀립니다**.
- **응답이 진행 중인 세션은 보관할 수 없습니다** — 그런 세션이 있으면 "응답이 진행 중인 세션은 보관할 수 없습니다" 안내가 뜹니다. 응답이 끝난 뒤(또는 Stop 을 누른 뒤) 다시 시도하세요.
- 폴더 노드의 **Delete All Sessions** 는 **보관된 세션을 지우지 않습니다**(그 폴더에 보이는 것만 지웁니다). 보관분까지 전부 지우려면 맨 위 **Sessions** 루트를 **우클릭**해 **Clear All Sessions** 를 쓰세요.
- 저절로 보관되거나 저절로 지워지는 세션은 없습니다 — 보관은 사용자가 걸고 푸는 것뿐입니다.
- **여러 세션을 한 번에** — 세션 목록에서 **Ctrl+클릭**(하나씩 추가/해제)이나 **Shift+클릭**(범위)으로 여러 세션을 고른 뒤 그중 하나를 **우클릭**하면 **Archive (N)** · **Unarchive (N)** · **Delete N Sessions** 메뉴가 선택 전체에 적용됩니다. 접혀 있거나 검색으로 가려진 세션은 포함되지 않습니다.

3. 기본 대화 사용법

- 하단 입력창에 메시지를 입력하고 **Send**를 누르면 서버로 전송되고, AI 응답이 스트리밍으로 표시됩니다.
- **Shift+Enter** (또는 **Ctrl+Enter** · **Alt+Enter**): 입력창에서 줄바꿈 (Enter는 전송)
- 입력창은 **내용에 맞춰 높이가 자동으로 늘어납니다**. 줄이 많아지면 기본 높이의 약 3배까지 커지고, 그 뒤로는 입력창 안에서 스크롤됩니다. 전송하거나 내용을 지우면 원래 높이로 돌아옵니다.
- **입력 히스토리** (**↑** / **↓**): 입력창에서 **↑** 를 누르면 이 세션에서 **직전에 보낸 메시지**가 채워지고, 계속 누르면 더 이전 것으로 올라갑니다(명령 프롬프트와 같은 감각). **↓** 로 다시 내려오며, 끝을 지나면 원래 쓰던 내용으로 돌아옵니다. 여러 줄을 입력 중일 때는 커서가 첫 줄에 있어야 **↑** 가, 마지막 줄에 있어야 **↓** 가 히스

토리로 동작하고 그 외에는 평소처럼 줄 이동입니다. 되불러온 내용을 고치면 그때부터는 새 입력으로 취급됩니다. 이전 세션을 다시 열어도 그 세션의 메시지가 히스토리에 채워집니다.

- **Stop 버튼:** 응답 생성 중 중단
- **작업 중 메시지 보내기:** AI가 작업하는 중에도 Enter(또는 **Send**)로 메시지를 보낼 수 있습니다.
 - 메시지는 **대기열**에 들어가고 입력창은 비워집니다. 대기 중인 메시지는 대화에 사용자 버블로 표시되고, 버블에 **지금 보내기**와 **X** 버튼이 붙습니다. 진행 줄에는 **· 대기 메시지 N**이 함께 표시됩니다.
 - 대기 메시지는 AI가 **다음 단계로 넘어갈 때** 전달되고(버블 표시가 **전달됨**으로 바뀝니다), AI는 그 내용을 반영해 작업을 이어 갑니다. 작업이 끝날 때까지 전달되지 않은 메시지는 **끝난 직후 자동으로 보냅니다**(여러 건이면 하나로 합쳐 보냅니다).
 - **지금 보내기:** 지금 작업을 멈추고 대기 메시지를 바로 다음 요청으로 보냅니다.
 - **X:** 대기를 취소합니다 — 본문이 입력창으로 돌아오므로 고쳐서 다시 보낼 수 있습니다. 이미 전달된 뒤라면 **이미 전달했습니다** 안내가 뜹니다.
 - **Stop**을 누르면 아직 전달되지 않은 메시지는 입력창으로 돌아옵니다.
 - 작업 중에는 **슬래시 명령과 이미지 첨부은 보낼 수 없습니다** — 안내가 뜨고 입력은 그대로 남으니 작업이 끝난 뒤 보내세요. **/open**만 작업 중에도 바로 실행됩니다. 예비 채팅 리스트(3-1절)에서 보내는 것도 같은 규칙을 따릅니다.
- 응답이 비정상적으로 길어지면 서버가 **생성 상한**에서 스스로 끊고 "응답 생성이 서버의 생성 상한에 도달하여 중단되었습니다" 안내를 표시합니다. 요청을 나누거나 새 대화로 다시 시도하세요.
- **파일 첨부 (📎 버튼 · 드래그 앤 드롭 · **Ctrl+V**):** 파일 종류에 따라 동작이 다릅니다.
 - **이미지:** 썸네일로 첨부되어 메시지와 함께 전송됩니다. 사용 중인 모델이 vision을 지원해야 합니다(아래 9절 Provider 설정의 **vision** 옵션). vision이 없는 모델을 쓸 때는 **비전 브로커**(9절)를 지정해 두면 다른 모델이 이미지를 대신 읽어 설명을 전달합니다 — 지정이 없으면 이미지는 빠지고 AI에게 그 사실만 안내됩니다. 최대 개수 제한이 있습니다. **Ctrl+V** 붙여넣기는 이미지만 받습니다.
 - **텍스트 파일·문서(.xlsx / .docx / .pptx / .pdf):** 파일 내용을 보내는 대신 **파일 경로가 입력창에 채워집니다**(작업 폴더 안이면 상대 경로). 그 뒤에 할 일을 적어 보내면 AI가 **read_file**로 읽습니다 — 여러 파일을 한 번에 끌어다 놓아도 됩니다.
 - **그 밖의 바이너리**(구형 .xls / .doc / .ppt, 압축·실행 파일 등): AI가 읽을 수 없으므로 첨부되지 않고 안내가 표시됩니다.
- 상단 **AI Provider 드롭다운**에서 현재 세션이 사용할 LLM을 즉시 전환할 수 있습니다. (전환 시 대화를 초기화할지 여부는 **Option > Settings... > General**의 옵션으로 결정됩니다.)
- **추론 단계 버튼:** 폴더 줄에 있는 **추론 단계 버튼**(현재 단계 이름 + ▲)을 누르면 버튼 위로 작은 팝업이 뜹니다 — 머리말 **추론 단계 · <현재 단계>**, ? 도움말, 그리고 단계를 고르는 슬라이더입니다. 슬라이더를 끌면 머리말이 따라오고, 손을 떼는 순간 적용됩니다. 팝업 바깥을 클릭하거나 **Esc**, 버튼을 다시 누르면 닫힙니다.
 - 단계가 높을수록 더 깊게 추론하지만 응답이 느려집니다.
 - **단계 이름:** 로컬 모델은 **Light · Medium · High · Extra High · Max** (단계가 오를수록 추론에 쓰는 예산이 커집니다), Claude·ChatGPT·Gemini 같은 클라우드 모델은 그 모델의 용어 그대로입니다 (**effort: high · reasoning: medium** 등).
 - **선택한 모델에 따라 칸 수가 달라집니다.** 로컬 vLLM·Ollama와 Claude·Gemini는 5칸, 로컬 OpenAI 호환 서버와 OpenAI·Codex는 4칸입니다. 모델을 바꾸면 슬라이더가 그에 맞춰 다시 그려집니다.

- **단계는 모델별로 기억됩니다.** 로컬 모델은 High, Claude는 `effort: medium` 처럼 따로 정해 두면 모델을 오갈 때마다 각각 마지막에 고른 단계로 돌아옵니다. 그 모델을 처음 쓸 때는 기본 단계(로컬은 예산 1K 단계, 클라우드는 3번째 단계)로 시작합니다. 칸 수가 더 적은 모델로 옮기면 표시만 마지막 칸으로 접히고, 원래 모델로 돌아오면 저장해 둔 단계가 그대로 살아납니다.
- 등록된 모델이 하나도 없으면 버튼은 비활성(**추론 단계**)이고 팝업도 열리지 않습니다 — `Option > AI Model...` 에서 모델을 먼저 등록하세요.
- 로컬 모델에서는 **어느 단계에서나** AI가 작업을 마친 뒤 **요청 내용과 결과물을 스스로 한 번 더 대조**하고, SpiderGen 프로젝트라면 **빌드와 화면 로드까지 확인**한 뒤 보고합니다.
- **큰 작업 위임:** 오래 걸리는 작업은 AI가 백그라운드 담당자에게 맡기고 대화를 이어갑니다 — "이 작업은 위임해서 진행해줘"처럼 직접 시킬 수도 있습니다(13절).
- **응답 표시 (대화 화면):** 요청 하나와 그 처리 과정·답이 **한 묶음**으로 표시됩니다.
 - **작업 중**에는 묶음 맨 아래의 **진행 줄** 한 개(`작업 중 · 경과 시간 · 지금 하는 일`)와 AI의 중간 설명만 보이고, 과정(도구 실행·명령 카드)은 숨겨집니다. 진행 줄(▶)을 누르면 과정을 펼쳐 볼 수 있습니다.
 - **작업이 끝나면** 과정은 접히고, **요약 줄**(예: ▶ `작업 1분 32초 · 스텝 7 · 도구 12회 · 파일 3개`) 아래에 최종 답이 남습니다. 요약 줄을 누르면 과정이 펼쳐집니다(▼). 접혀도 밖에 남는 것은 AI의 중간 설명 · 최종 답 · 파일 수정 카드와 Undo 바 · 아직 실행 중인 백그라운드 작업 카드(13절) · 경과와 오류입니다. 과정 없이 바로 답한 요청에는 요약 줄이 없습니다.
 - **명령 실행**은 이어진 명령끼리 **카드 하나로** 묶입니다 — 머리줄 `AI Command · 명령 N · 실패 M`, 기본은 접힌 상태입니다. 머리줄(`[show]`)을 누르면 명령마다 한 줄씩 보이고, 그 줄을 누르면 해당 명령의 출력이 펼쳐집니다.
 - 도구가 실패해 AI가 다시 시도한 경우 진행 줄과 요약 줄에 `재시도 N` 으로 표시됩니다.
 - 이전 대화를 다시 열어도 같은 모양(전부 접힌 채)으로 표시됩니다.
 - 코드 블록(접기/복사)·표(마크다운 표는 표로 렌더링)·파일 경로 링크(클릭하면 코드 뷰어) 등이 채팅에 그대로 표시됩니다.
- **Undo:** AI가 파일을 수정하면 메시지 하단에 `N개 파일 수정 완료` 표시와 **Undo** 버튼이 나타납니다.
 - 표시를 클릭하면 **이번에 바뀐 파일 목록**이 펼쳐집니다. 파일명을 클릭하면 코드 뷰어로 열리므로, AI의 설명에 의존하지 않고 실제로 무엇이 바뀌었는지 직접 확인할 수 있습니다.
 - **Undo** 를 누르면 방금 수정을 원상태로 1단계 되돌립니다("직전 상태" 기준). 이 되돌림 사실은 다음 사용자 메시지 전송 시 AI에게도 함께 전달되어 맥락이 어긋나지 않습니다.
 - 새로 **생성된** 파일은 되돌리기 대상이 아닙니다(수정된 기존 파일만 복원됩니다) — 목록에는 보이므로 필요하면 직접 지우시면 됩니다.
- **채팅 내 검색 (`Ctrl+F`):** 채팅 화면에서 `Ctrl+F` 를 누르면 우측 상단에 검색 창이 열립니다.
 - 입력하는 즉시 대화 내용에서 일치하는 부분이 강조 표시되고(대소문자 무시), 현재 위치가 `1/3` 형식으로 표시됩니다.
 - `Enter` 또는 ↓ 로 다음, `Shift+Enter` 또는 ↑ 로 이전 결과로 이동합니다(끝에서 처음으로 순환). 이동 시 해당 위치로 자동 스크롤됩니다.
 - `Esc` 또는 × 로 닫으면 강조 표시가 모두 사라집니다. 채팅에서 텍스트를 드래그해 둔 상태로 `Ctrl+F` 를 누르면 그 텍스트가 검색어로 채워집니다.

- 검색 창은 채팅 탭마다 따로 동작합니다 — 다른 탭으로 전환하면 함께 숨겨지고, 돌아오면 다시 쓸 수 있습니다.

3-1. PreChat List (예비 채팅 리스트)

자주 쓰는 지시나 아직 보낼 준비가 안 된 메시지를 **미리 적어 두었다가** 필요할 때 채팅으로 보내는 패널입니다. [View > PreChat List](#) 로 엽니다.

- **등록:** 패널 하단 입력창에 메시지를 쓰고 **Enter** (줄바꿈은 **Shift+Enter**). 목록에 말풍선으로 쌓입니다.
- **보내기 (더블클릭):** 항목을 더블클릭하면 활성 채팅 세션으로 **바로 전송**됩니다. 항목은 목록에 남으므로 같은 지시를 반복해서 쓸 수 있습니다. 채팅 입력창에 쓰던 초안이 있으면 전송 후 그대로 되돌려 놓습니다. AI 가 작업 중이면 메시지는 대기열에 들어갑니다(3절 "작업 중 메시지 보내기").
- **복사 ():** 메시지를 채팅 입력창에 넣고(입력창에 내용이 있으면 줄바꿈 뒤에 이어 붙음) 시스템 클립보드에도 복사합니다. 항목은 목록에 남습니다.
- **삭제 ():** 해당 항목만 지웁니다. **모두삭제** 버튼은 확인 후 현재 폴더의 목록 전체를 비웁니다.
- **수정:** 항목을 클릭해 선택한 뒤 **F2** 또는 **수정 (F2)** 버튼 — 말풍선이 그 자리에서 편집 상태가 됩니다. **Enter** 저장, **Shift+Enter** 줄바꿈, **Esc** 취소.
- **우클릭 메뉴:** 항목을 우클릭하면 **복사 / 삭제 / 채팅으로 전송** 메뉴가 뜹니다 — 위 버튼들과 같은 동작이라, 패널 폭을 좁혀 버튼을 숨겨 둔 상태에서도 쓸 수 있습니다(항목 편집 중에는 뜨지 않습니다).
- **폴더별 목록:** 목록은 **작업 폴더(cwd)** 단위로 따로 저장됩니다. 채팅 탭을 전환하거나 작업 폴더를 바꾸면 그 폴더의 목록으로 자동 전환되고, 패널 상단에 현재 폴더 경로(뒤 2단)가 표시됩니다. 작업 폴더가 없는 세션(No Project)은 전용 목록을 씁니다. 앱을 재시작해도 유지됩니다.
- **대상 채팅:** 활성 채팅 탭 → 없으면 열린 첫 채팅 탭 → 그것도 없으면 새 세션을 만들어 보냅니다.

슬래시(/) 명령

AI의 판단을 거치지 않고 **사용자가 직접 실행**하는 명령입니다. 채팅 입력창에 **/명령** 형태로 입력하면 즉시 실행됩니다. 전체 목록은 언제나 [/help](#) 로 확인할 수 있습니다.

명령	설명
/find <파일명 일부> [-n 개수]	프로젝트 전체에서 파일명을 검색합니다(find_file 직접 실행, 대소문자 무시·부분 일치). 기본 50건까지 보여주며 더 있으면 limit reached 로 알립니다 — -n 으로 개수를 바꿀 수 있습니다(최대 500). — 예: /find manager , /find .lay -n 200
/grep <검색어> [-n 개수]	프로젝트 전체에서 텍스트를 검색합니다(search_project 직접 실행). 기본 20건까지 표시하며 -n 으로 상향할 수 있습니다(최대 500). 한도에 걸리면 N+ matches (limit reached) 로 표시됩니다. — 예: /grep loadConfig , /grep sendResult -n 100
/outline <파일 경로>	파일의 함수/클래스(마크다운은 헤딩) 목록을 라인 번호와 함께 보여줍니다(file_outline 직접 실행). — 예: /outline Source/mdi/AIChatView.js
/tree [경로] [덱스]	프로젝트 폴더 구조를 들여쓰기 트리로 보여줍니다(project_tree 직접 실행). 경로·덱스 모두 생략 가능(기본: 루트, 덱스 3·최대 5). — 예: /tree , /tree Source 2

명령	설명
<code>/open <파일 경로></code>	파일을 코드 뷰어(Monaco, 읽기전용)로 즉시 엽니다 — 분할이 없으면 오른쪽 새 분할에 열립니다. — 예: <code>/open Source/MenuInfo.js</code>
<code>/simulate</code>	빌드된 SpiderGen 프로젝트를 시뮬레이터 창으로 실행합니다(<code>spidergen_simulate</code> 직접 실행). <code>bin/index.html</code> 이 없으면 먼저 빌드하라는 안내가 표시됩니다.
<code>/skill <스킬 이름> [요청]</code>	지정한 스킬(6절)을 강제 발동 합니다. 다른 슬래시 명령과 달리 AI를 우회하지 않습니다 — 스킬 본문(지침)을 이번 요청에 주입한 뒤 AI가 그 지침에 따라 작업을 계속합니다. AI가 스킬을 자동 발동하지 못할 때 유용합니다. 이름 없이 <code>/skill</code> 만 입력하면 사용 가능한 스킬 목록이 표시됩니다. 주입은 해당 턴에만 적용됩니다. — 예: <code>/skill my_skill 로그인 화면에 적용해줘</code>
<code>/<스킬 이름> [요청]</code>	스킬 이름을 곧바로 슬래시 명령으로 쓸 수 있습니다 — <code>/my_skill 로그인 화면에 적용해줘</code> 는 <code>/skill my_skill 로그인 화면에 적용해줘</code> 와 동일합니다. 위 기본 명령과 이름이 겹치지 않는 모든 스킬(내장·프로젝트)에 적용되며, 기본 명령이 항상 우선합니다. 스킬에도 없는 이름이면 일반 대화로 처리됩니다.
<code>/help</code>	사용 가능한 슬래시 명령 전체와 스킬 목록(출처 <code>[내장]</code> / <code>[프로젝트]</code> 표기)을 즉시 보여줍니다.

4. 인덱싱 (Indexing · RAG)

- **기본값은 꺼짐** — 임베딩 서버가 필요한 기능이라 `Option > Settings... > Indexing` 에서 **인덱싱을 켜고 임베딩 서버를 입력**해야 동작합니다. 꺼져 있어도 대화·도구는 모두 정상입니다.
- **참고 자료 자동 참고**: 켜 두면 `{프로젝트폴더}/.aiassist/samples` 폴더의 파일을 인덱싱해 두었다가, 요청과 비슷한 내용을 자동으로 찾아 참고합니다. **프로젝트 종류를 가리지 않습니다** — SpiderGen 프로젝트든 일반 프로젝트든 이 폴더에 파일을 넣으면 동작합니다. 폴더가 없으면 이 기능만 꺼진 채 대화는 정상 동작합니다.
- **내용도 UI 샘플에 한정되지 않습니다** — 코드 예제, 규칙 문서, 설계 메모, 자주 쓰는 패턴 등 AI가 참고했으면 하는 자료를 자유롭게 넣을 수 있습니다.
- **프로젝트 소스 코드는 인덱싱하지 않습니다** — AI가 검색 도구(`search_project` 등)로 필요할 때 직접 찾습니다.

샘플 추가하기

AI가 참고할 자료를 늘리려면 `.aiassist/samples/` 아래에 폴더를 만들어 파일을 넣습니다. 아래는 SpiderGen UI 패턴을 넣는 예시이며, 파일 종류나 폴더 구조에 정해진 형식은 없습니다.

ButtonSample/

_desc.md	(선택) 샘플 설명
ButtonSample.lay	레이아웃
ButtonSample.js	스크립트

- 폴더가 없으면 먼저 만들어야 인덱싱이 시작되고, 요청과 충분히 비슷한 자료만 자동 채택됩니다.
- 인덱싱 결과는 캐시되어, 파일이 바뀌지 않았다면 다시 인덱싱하지 않습니다.
- `.aiassist` 폴더 자체는 `search_project` / `find_file` 같은 일반 검색 도구에서는 제외됩니다.

5. 코드 뷰어 / 에디터 (Monaco)

채팅 중 AI의 툴 호출 카드나 편집 카드에 표시된 **파일 경로를 클릭**하면 해당 파일이 Monaco 기반 뷰어로 MDI 탭에 열립니다. 세션을 새로 열어 이전 대화를 불러온 뒤에도 동일하게 클릭해서 열 수 있습니다. 채팅 본문에 **인라인 코드**로 표시된 파일 경로(예: AI가 보고에서 언급한 파일 목록)도 같은 방식으로 클릭해서 열 수 있습니다.

- `.html`, `.lay` 등 HTML 계열 파일은 보기 좋게 자동 정렬(`beautify`)되어 표시됩니다.
- `.md` (마크다운) 파일은 **미리보기로 열립니다** — 제목·표·목록·코드블록이 채팅처럼 렌더된 문서 형태로 표시되고, 우상단 "원문 보기" 버튼으로 소스와 전환할 수 있습니다. 분석 보고서 같은 문서 산출물을 읽기 좋게 보는 용도입니다. ("소스 오픈 시 수정모드" 설정이 켜져 있으면 원문(편집)으로 먼저 열립니다)
- 기본적으로 **읽기 전용**으로 열립니다. `Option > Settings... > General` 의 "소스 오픈 시 수정모드" 체크박스를 켜두면 다음부터 **편집 가능한 상태**로 열립니다.
- 편집 모드에서 내용을 고치면 탭 제목에 수정 마크가 표시되고, `Ctrl+S` 또는 `File > Save File` 로 저장하면 마크가 사라집니다.
- **Git 변경분 거터 마커**: Electron 환경에서 로컬 git이 사용 가능하면, 파일을 열 때 좌측 거터에 변경 라인이 표시됩니다 — 추가(녹색 막대) / 수정(파랑 막대) / 삭제(빨강 삼각형). 비교 기준은 미커밋 변경이 있으면 작업트리 vs HEAD, 없으면 마지막 커밋(HEAD vs HEAD~1)입니다. 거터 마커를 클릭하면 뷰 전체를 덮는 오버레이로 원본(base)과 현재 내용을 비교해서 볼 수 있습니다(peek).

5-1. 문서 파일 (Excel / PowerPoint / Word / PDF)

AI가 코드 파일처럼 **오피스 문서와 PDF**를 읽고, 오피스 문서는 **고칠** 수도 있습니다.

- **읽기**: `.xlsx` (시트별 표), `.pptx` (슬라이드별 텍스트·노트), `.docx` (문단·표)를 텍스트로 읽습니다. 예: "spec/기획서.pptx 읽고 화면 만들어줘", "전문정의.xlsx 의 W1805A01 시트만 봐줘" — 엑셀은 시트를 지정해 일부만 읽을 수 있고, 차트·이미지는 개수만 표시됩니다.
- **PDF 읽기**: `.pdf` 는 페이지별 텍스트로 읽습니다(`## Page N`). 긴 PDF는 "3~5쪽만 봐줘"처럼 페이지 범위를 지정할 수 있습니다. 글자가 들어 있는 PDF(오피스·한글·브라우저 인쇄 등으로 만든 것)가 대상이며, **스캔 이미지 PDF**는 글자를 뽑을 수 없어 그 사실을 알려 줍니다. 암호가 걸린 PDF도 읽지 못합니다.

- **고치기:** 엑셀 셀 값, 파워포인트 슬라이드의 텍스트, 워드 문단 텍스트를 제자리에서 바꿉니다. 바뀐 부분만 패치하므로 서식·수식·차트·이미지는 그대로입니다. 행/열/시트 추가나 서식 변경은 지원하지 않습니다.
- **되돌리기:** 코드 파일과 똑같이 **Undo** 로 원본 바이트 그대로 복원됩니다.
- **주의:** 구형 형식(`.xls` / `.ppt` / `.doc`)은 읽지 못합니다 — 최신 형식(`.xlsx` / `.pptx` / `.docx`)으로 저장해 두세요. 압축·실행 파일 등 텍스트가 아닌 파일은 AI가 열지 못하며 그 사실을 알려 줍니다.
- 문서를 채팅창에 끌어다 놓거나 🖱️ 버튼으로 고르면 경로가 입력창에 채워집니다(3절 참고). 작업 폴더 밖의 파일도 절대 경로로 읽을 수 있습니다.

6. Skills (프로젝트별 커스텀 가이드)

Skill은 "이런 요청이 오면 이렇게 처리해라"는 작업 가이드(마크다운 문서)입니다. AI가 매 요청마다 스킬 목록(이름 + 설명)을 인지하고 있다가, 사용자 요청이 특정 스킬 설명과 맞아떨어지면 `use_skill` 도구로 본문 전체를 불러와 그 지침을 따릅니다.

내장 스킬

앱이 기본 제공하는 내장 스킬은 프로젝트 스킬과 한 목록으로 합쳐져 노출됩니다(`/help` 에서 `[내장]` 표기). 프로젝트에 같은 `name` 의 스킬을 만들면 **프로젝트 것이 우선합니다**(내장 스킬을 프로젝트 사정에 맞게 덮어쓰는 방법).

현재 내장 스킬은 `create_agent` 하나입니다 — "코드 조사만 하는 에이전트 만들어줘"처럼 요청하면 AI가 이 스킬을 읽고 13절의 에이전트 정의 파일을 올바른 위치·형식으로 만들어 줍니다.

저장 위치

작업 폴더를 처음 선택하면 아래 폴더가 자동 생성됩니다.

```
<프로젝트폴더>/aiassist/skills/  
my_skill.md          (사용자가 만든 스킬 - 이 프로젝트에서만)
```

모든 프로젝트에서 같이 쓸 스킬은 사용자 데이터 폴더(8절의 `<userData>`)의 `skills/` 에 두면 됩니다 (`<userData>/skills/my_skill.md` — 이 폴더는 자동으로 만들어지지 않으니 필요할 때 직접 만드세요). 이름이 겹치면 **프로젝트 > 전역(`<userData>/skills`) > 내장** 순으로 앞의 것이 쓰이고, `/help` 에 `[프로젝트]` · `[전역]` · `[내장]` 으로 출처가 표시됩니다. 전역 스킬은 작업 폴더를 선택(또는 다시 선택)할 때 읽힙니다.

스킬 파일 형식

```
---
name: my_skill
description: 언제 사용하는지 한 줄로 (최대 300자)
---
# 본문

여기에 단계별 지침, 예시, 주의사항 등을 자유롭게 작성합니다.
```

- `name`, `description` 은 **권장**입니다(없으면 파일명과 본문 첫 문단으로 자동 채워집니다).
- `description` 은 AI가 이 스킬을 쓸지 판단하는 기준이 되니, 한 줄로 직접 적어 주시면 더 정확하게 발동합니다.

사용 방법

1. **AI에게 요청해서 만들기**: 채팅창에 "이 작업 방식을 스킬로 저장해줘"처럼 요청하면 AI가 알아서 올바른 위치/형식으로 파일을 생성해줍니다.
2. **새로 만든 스킬은 그 자리에서 바로 쓸 수 있습니다** — 재인덱싱이 필요 없습니다. 기존 스킬의 `name` / `description` 을 직접 고친 경우에만 **작업 폴더를 다시 선택**하면 반영됩니다(본문 수정은 즉시 반영됩니다). 새로 만든 스킬의 `name` 이 이미 있는 스킬(전역·내장 포함)과 같으면 AI가 그 사실을 알려 줍니다 — 바로 쓰려면 이름을 바꾸세요(전역·내장과 겹친 경우에는 작업 폴더를 다시 선택한 뒤부터 새 프로젝트 스킬이 우선합니다).
3. 이후 대화에서 스킬 설명과 맞는 요청을 하면 AI가 자동으로 해당 스킬을 불러와 그 지침에 따라 작업합니다. AI가 자동 발동하지 못하는 경우에는 `/skill <스킬이름> [요청]` 으로 직접 발동할 수 있습니다(3절 슬래시 명령 참고).

7. 메모리(Memory) 기능

프로젝트별로 지속되는 메모리 기능이 구현되어 있습니다. 같은 프로젝트 폴더로 새 세션을 열어도 이전에 저장해둔 사실/선호가 계속 유지됩니다.

- 저장 위치: `{프로젝트폴더}/.aiassist/memory.md` (프로젝트마다 별도 파일 — 전역/모든 프로젝트 공용 메모리는 아닙니다.)
- 저장 방법: 사용자가 새로운 지속적 선호를 말하거나 "기억해줘"라고 명시적으로 요청하면, AI가 `save_memory` 도구로 한 줄씩 기록합니다(별도 승인 없이 바로 저장).
- 활용: 저장된 내용은 이후 모든 대화에 자동 반영되어, AI가 지속적인 지시사항처럼 참고합니다. 파일을 열어 직접 수정/삭제해도 되며, 다음 채팅부터 바로 반영됩니다.

- 프로젝트를 넘나드는 공통 규칙(코딩 컨벤션 등)을 여러 프로젝트에서 재사용하고 싶다면, 위 6절의 **Skills**로 관리하는 편이 낫습니다 — 특히 `<userData>/skills/` 의 전역 스킬은 모든 프로젝트에서 보입니다(Skills는 별도 파일을 두고 필요할 때만 전체 본문을 불러오는 방식이라 더 긴 절차/규칙에 적합합니다). Memory는 프로젝트 하나에 계속 누적되는 짧은 사실/선호 bullet에 적합합니다.

8. 세션 데이터 저장 위치

모든 대화는 자동으로 로컬(Electron `userData`)에 저장됩니다.

```
<userData>/aiassist/  
sessions.json      (세션 인덱스: sessionId, projectPath, 미리보기, 생성/수정 시각, 메시지  
data/  
  <sessionId>.jsonl  (세션별 대화 내용, append-only)  
archive/data/  
  <sessionId>.jsonl.gz (보관한 지 7일이 지난 세션의 대화 내용 - 압축본)
```

보관(2-1절)한 세션은 7일이 지나면 대화 내용이 `archive/data/<sessionId>.jsonl.gz` 로 압축되어 옮겨집니다. 여는 방법은 똑같고(자동으로 풀어서 읽습니다), 보관을 풀면 원래 자리로 되돌아오므로 사용자가 이 파일들을 직접 손댈 일은 없습니다.

세션 목록 패널에서 세션을 지우거나 관리할 수 있습니다(우클릭 메뉴).

9. AI 모델(Provider) 설정

`Option > AI Model...` 메뉴에서 사용할 LLM Provider를 등록/편집합니다.

필드	설명
이름(name)	드롭다운에 표시될 식별자
API Type	모델 서버 종류 — vLLM / Ollama / Claude / OpenAI / Gemini / OpenAI 호환 서버
model	모델 이름 (예: <code>qwen3-coder-next</code>)
host / port	API 엔드포인트
chatPath	채팅 API 경로
Max Context	모델 컨텍스트 한도(토큰, <code>max_model_len</code>) — 미입력 시 자동 조회 (아래 참고)

필드	설명
(고급) <code>max_output_tokens</code>	응답 한 번의 최대 출력 토큰. 보통은 컨텍스트 크기에 따라 자동(8K~32K)이므로 비워 둡니다. 설정 파일(<code>ai-client-config.json</code> / <code>ai-config.json</code>)에 직접 넣는 항목입니다
https	엔드포인트를 HTTPS로 호출할지 여부
인증 방식	API Key 또는 OAuth(계정 로그인) 중 선택 (아래 참고)
apiKey	인증 키 (API Key 방식일 때만 표시)
headers	요청에 추가할 커스텀 헤더 (한 줄에 하나, <code>"이름": "값"</code> 형식)
vision	이미지 입력(첨부) 지원 여부
비전 브로커	이 모델이 이미지를 못 받을 때(vision 해제) 대신 이미지를 읽어 줄 모델 — 아래 참고
스트리밍	응답을 받는 대로 보여줄지 여부. 해제하면 응답이 끝난 뒤 한 번에 표시됩니다. Claude 와 ChatGPT 구독(OAuth)은 항상 켜집니다

기본 제공 Provider(서버 관리, `ai-config.json`)와 사용자가 추가한 Provider(`ai-client-config.json`)가 함께 드롭다운에 표시됩니다. 임베딩(샘플 인덱싱용) 모델도 별도로 설정되어 있습니다.

비전 브로커 — 이미지를 못 받는 모델에 "대신 읽어 줄 모델" 지정

vision 을 지원하지 않는 모델(예: 텍스트 전용 GLM)을 쓰면서도 이미지를 활용하고 싶을 때 씁니다. 지정해 두면 첨부 이미지나 자동 UI 테스트의 화면 캡처가 도착할 때마다 브로커 모델이 먼저 그 이미지를 상세히 서술하고, 현재 모델은 그 설명 텍스트를 받아 "본 것처럼" 작업을 이어갑니다.

1. `Option > AI Model...` 에서 이미지를 못 받는 모델을 선택하고 **vision 체크를 해제**합니다 — 해제해야 아래 셀렉트가 활성화됩니다.
2. **비전 브로커** 셀렉트에서 대신 읽어 줄 모델을 고릅니다. 목록에는 **vision 이 체크된 사용자 추가 모델**만 나오고 (자기 자신·기본 제공 모델은 제외), "선택 안함" 이 끄기입니다. 클라우드 모델을 고르면 이미지가 외부 서비스로 전송되고 과금될 수 있습니다.
3. 확인을 누르면 바로 적용됩니다(앱 재시작 불필요).

사용 중 보게 되는 안내(채팅의 System 줄):

- **"* 현재 모델은 ... 이 이미지를 대리 해석합니다."** — 브로커가 동작한 대화에서 처음 한 번 표시됩니다(로컬·클라우드 구분 없음). 같은 이미지는 다시 읽지 않습니다.
- **"▲ 이 모델(...)은 이미지 입력을 지원하지 않습니다(백엔드가 거절)"** — vision 을 체크해 두었지만 실제 모델이 이미지를 거절한 경우입니다. 이때는 브로커가 자동으로 켜지지 **않습니다** — 그 모델의 vision 체크를 끄고 브로커를 지정하세요.
- **"▲ 비전 브로커(...)가 이미지 ... 를 해석하지 못했습니다"** — 브로커 모델 쪽 오류입니다. 해당 이미지는 빠진 채 진행되니 브로커 모델 설정을 확인하세요.

알아둘 점: 전달되는 것은 설명 텍스트라 "왼쪽 위 버튼의 정확한 색" 같은 세부는 놓칠 수 있습니다. 브로커로 고른 모델이 나중에 삭제되거나 vision 이 해제되면 설정은 "선택 안함" 으로 돌아갑니다.

컨텍스트 한도 수동 지정 (`max_model_len`)

모델의 컨텍스트 한도(전체 토큰 수)는 보통 자동 인식됩니다. 자동 인식이 안 되는 환경(사내 게이트웨이, 프록시 등)에서는 긴 대화에서 출력이 일찍 잘리거나 컨텍스트 초과 오류가 날 수 있으니, 이 경우 한도를 직접 지정하세요.

- **사용자 추가 Provider:** `Option > AI Model...` 설정 창의 **Max Context** 필드에 입력 (`ai-client-config.json` 의 `max_model_len` 으로 저장됨)
- **기본 제공 Provider:** `ai-config.json` 의 provider 항목에 `max_model_len` 을 직접 기입:

```
"my-gateway": {
  "host": "gateway.example.com",
  "port": 443,
  "chatPath": "/v1/chat/completions",
  "model": "OpenAPI",
  "max_model_len": 65536
}
```

`ai-config.json` , `ai-client-config.json` 모두에서 사용 가능하며, 파일을 저장하면 바로 반영됩니다.

OAuth(계정) 로그인

API 키 대신 **Claude** 또는 **ChatGPT** 계정으로 로그인해서 해당 구독으로 요청을 보낼 수 있습니다.

1. Provider 편집 화면에서 인증 방식을 **OAuth**로 전환합니다.
2. Provider 종류(`claude` | `chatgpt`)를 선택합니다.
3. **로그인** 버튼을 누르면 브라우저가 열려 해당 서비스의 로그인/권한 승인 화면으로 이동합니다.
 - Claude: 로그인 후 표시되는 코드를 복사해 입력창에 붙여넣고 "코드 적용" 버튼을 눌러 교환합니다.
 - ChatGPT: 로컬(loopback)로 인증 코드를 자동 수신해 별도 붙여넣기 없이 교환됩니다.
4. 연결 성공 시 상태 표시가 "연결됨"으로 바뀝니다.
5. 발급된 토큰은 **암호화되어** 저장되며, 앱을 재설치해도 유지됩니다.

`Option > Settings...` 에서는 일반 옵션(모델 전환 시 대화 초기화 여부, 소스 오픈 시 수정 모드 등)과 Lint 관련 설정을 관리합니다. Lint 의 **Auto Lint**(`edit_file` 성공 후 해당 파일만 ESLint 자동 실행)는 프로젝트 `package.json` 에 `eslint` 의존성이 있어야 동작하며, 켜 두었는데 `eslint` 가 없으면 편집 결과에 스킵 안내가 세션당 1회 표시됩니다. AI 가 직접 호출하는 `lint` 도구도 마찬가지로 프로젝트에 `eslint/prettier` 가 설치되어 있어야 하고, 없으면 설치를 제안하는 안내를 반환합니다.

10. 명령 실행 승인 및 안전장치

- AI가 셸 명령을 실행하려 하면 **승인 요청 팝업**이 뜨고, **Deny**(거부) 또는 **Allow**(허용) 로 결정합니다.
- 승인 팝업은 **요청한 대화 탭 안에** 뜹니다. 한 탭에서 승인 요청이 여러 건 겹치면(대화과 위임 담당자가 동시에 요청하는 경우 등) **한 번에 하나씩 차례로** 뜨고, 다른 탭에서 요청이 오면 그 탭 버튼이 **배경색으로 강조**됩니다 (나타나는 순간 한 번 깜빡이고, 그 탭을 열거나 요청이 끝나면 원래대로 돌아옵니다). **시간 제한은 없습니다** — 답할 때까지 기다립니다.
- 승인 팝업이 떠 있는 동안에는 그 탭의 Stop 버튼이 가려집니다. 작업을 멈추려면 **Deny** 를 누른 뒤 **Stop** 을 누르세요. Stop 을 누르면 대화가 기다리던 승인 요청은 함께 취소됩니다(백그라운드 위임 담당자의 요청은 위임이 계속 도는 동안 남습니다 — 13절).
- 승인 팝업 하단에는 **"이번 세션에서 자동승인"** 체크박스가 있습니다. 체크한 뒤 **Allow** 를 누르면, 이후 같은 세션(연결)에서는 `run_command` / `process_op` (백그라운드 프로세스 시작) 요청이 팝업 없이 자동 승인됩니다. 단, 아래 차단 목록 검사는 이 경우에도 항상 먼저 수행됩니다. (Deny를 선택하면 체크 여부와 관계없이 자동 승인은 적용되지 않습니다.)
- AI가 **외부 주소로 HTTP 요청**을 보내려 할 때도 승인 팝업이 뜹니다(로컬 주소 `localhost` 는 바로 실행). 외부 전송은 **매번** 확인하며, 이 팝업에는 자동승인 체크박스가 없습니다.
- **프로젝트 명령**(npm·npx·pytest)도 같은 셸로 실행됩니다. `package.json` 에 있는 스크립트(`npm run <이름>` · `npm test`)와 패키지 이름 없는 `npm install` / `npm ci` , `npx jest|vitest|mocha|eslint|prettier|tsc` , `pytest` 는 승인 없이 바로 실행되고, **의존성을 추가하는** `npm install <패키지>` 와 그 밖의 명령은 승인 팝업이 뜹니다. 개발 서버처럼 스스로 끝나지 않는 명령은 `process_op` 로 백그라운드에서 띄우며 시작할 때 한 번 승인합니다.
- **git 명령**은 별도 도구 없이 같은 셸(`run_command`)로 실행됩니다. `git status` · `diff` · `log` 같은 조회와 `add` · `commit` · 브랜치 전환 같은 로컬 작업은 승인 없이 바로 실행되고, `git push` 와 되돌리기·삭제류(`reset --hard` , `checkout -- <경로>` , `clean` , `branch -D` , `stash drop` , `rebase` , `commit -- amend`)는 **자동승인 여부와 관계없이 매번** 승인 팝업이 뜹니다(이 팝업에는 자동승인 체크박스가 없습니다). `git diff` / `git show` 결과는 채팅에 색이 입혀진 diff 로 표시됩니다.
- **프로세스 종료 명령**(`taskkill` · `kill` · `pkill` · `Stop-Process` 등)도 **자동승인 여부와 관계없이 매번** 승인 팝업이 뜹니다(체크박스 없음). AI 가 실수로 Khairon 자신이나 다른 작업을 끝내지 못하게 하기 위한 것입니다.
- Windows 에서 AI 가 쓰는 셸은 **Git Bash** 입니다(Git for Windows 가 없으면 PowerShell). 사용자 터미널 탭은 종전대로 PowerShell 입니다.
- `rm -rf /` , `git push --force` , `sudo` , `curl | bash` 등 파괴적이거나 위험한 명령은 자동승인 여부와 관계없이 서버에서 원천 차단됩니다.
- 명령 실행 결과(표준출력)는 실시간으로 스트리밍되어 화면에서 확인할 수 있습니다.

11. SpiderGen 프로젝트 빌드/시뮬레이션

이 장은 SpiderGen 프로젝트로 작업할 때만 해당합니다. 작업 폴더가 SpiderGen 프로젝트(`*.prj` 포함)로 인식되면 빌드·시뮬레이션을 사용할 수 있습니다.

- **Build > Build Project** : 프로젝트를 빌드합니다.
- **Build > Simulate Project** : 빌드 결과를 시뮬레이터 창으로 실행합니다.

화면 동작을 자동으로 검증하려면 12절의 **자동 UI 테스트**를 사용하세요.

12. 자동 UI 테스트 (run_test)

이 장도 SpiderGen 프로젝트에서만 해당합니다.

채팅에서 "**~ 테스트해줘**" 라고 요청하면 AI가 테스트 시나리오를 세우고 시뮬레이터를 직접 조작해 단계별로 확인한 뒤 결과를 보고합니다. 모드 전환이나 사전 준비가 필요 없습니다.

요청하기

- 예: "로그인 화면에서 아이디 찾기 눌러서 다이얼로그 뜨는지 테스트해줘", "회원가입 입력 검증 테스트해줘"
- 계정·인증번호처럼 **테스트에 쓸 값은 요청에 그대로 적어 주세요**: "아이디 asoosoft, 비밀번호 winway1, 인증번호 123456 으로 로그인 테스트해줘". 지정한 값은 화면에 이미 채워져 있어도 AI가 다시 입력합니다(기억된 값으로 엉뚱하게 통과하는 것을 막기 위함입니다).
- 동작: AI가 코드를 읽어 진입 경로를 파악 → 시나리오 파일을 작성 → (필요 시 빌드 후) 시뮬레이터 실행 → 화면을 단계별로 클릭·입력하며 각 단계의 성공 조건을 확인 → 단계별 성공/실패와 근거를 채팅으로 보고합니다. 과정은 채팅창의 도구 실행 표시로 볼 수 있고, 시뮬레이터 창에서 직접 눈으로 볼 수도 있습니다(테스트가 끝나도 창은 닫지 않습니다).
- 테스트 중 결함을 발견해도 AI는 **보고만 하고 프로젝트 코드를 고치지 않습니다**. 수정이 필요하면 결과를 보고 따로 요청하세요.

시나리오만 만들기 (실행하지 않음)

테스트를 바로 돌리지 않고 **계획만 먼저 받아 검토**하고 싶을 때 씁니다.

- 예: "**order_test** 라는 이름으로 로그인 테스트 시나리오 만들어줘", "회원가입 테스트 시나리오만 작성해줘"
- AI가 코드를 확인해 단계와 각 단계의 성공 조건을 적은 **파일만 만들고 멈춥니다** — 빌드도 시뮬레이터 실행도 하지 않습니다.
- 같은 이름의 시나리오가 이미 있으면 새로 만들지 않고 **그 파일을 갱신**합니다.
- 내용을 읽어 보고 필요하면 직접 고친 뒤, "그 시나리오로 테스트해줘"(또는 "order_test 돌려줘")로 실행하면 됩니다.

시나리오 파일

- 시나리오는 작업 폴더의 **.aiassist/tests/** 아래 **.md** 파일로 남습니다. 단계 목록과 각 단계의 성공 조건이 적혀 있어 사람이 읽고 직접 고칠 수 있습니다.
- 테스트가 끝나면 결과(pass/fail·요약)가 **같은 파일 끝에 날짜와 함께 덧붙입니다** — 돌릴 때마다 이력이 쌓입니다.

저장한 테스트 다시 돌리기

- "order_test 다시 돌려줘", "order_test.md 로 테스트해줘" 처럼 **파일 이름만 말해도** 됩니다.
 .aiassist/tests/ 에서 찾아 그대로 실행하며, 확장자 유무·대소문자·- 와 _ 차이는 무시합니다.
- 재실행할 때는 코드 조사와 시나리오 작성을 건너뛰고 바로 실행하므로 처음보다 훨씬 빠릅니다.
- 이름이 겹치는 파일이 여러이면 AI가 후보를 알려주고 어느 것인지 되묻습니다. 정확한 경로
 (.aiassist/tests/order_test.md)로 다시 말해 주면 됩니다.
- 이름이 맞는 시나리오가 없으면 **새 테스트로 보고** 시나리오를 새로 만들어 그 이름으로 저장합니다. 재실행하려던 것이었다면 파일 이름을 확인하세요.

직접 조작을 녹화해서 테스트 만들기 (● Rec)

시뮬레이터를 손으로 조작한 기록을 남겨 두었다가, 그 기록대로 테스트를 만들 수 있습니다.

- 상단 바의 ● **Rec** 토글 버튼으로 녹화를 시작합니다(시뮬레이터 창이 떠 있어야 합니다). 앱을 평소처럼 쓰면서 클릭·입력이 기록됩니다.
- 버튼을 다시 눌러 정지하면 **이름 입력창**이 뜨고, 저장하면 작업 폴더의 test-scenario/records/<이름>.json 으로 보관됩니다. 입력창에서 취소하면 녹화 내용은 사라지지 않고 유지됩니다(앱을 새로 고치면 사라집니다).
- 이후 채팅에서 "**records/<이름> 기록대로 테스트해줘**" 라고 요청하면 AI가 기록을 읽고 같은 이름의 시나리오(.md)로 변환해 실행합니다 — 행위 단계는 기록에서, 성공 조건(검증)은 AI가 화면 맥락에서 도출합니다. 기록 원본(.json)은 그대로 보존되어 재변환 재료로 남습니다.
- 기존 시나리오에 기록을 반영하고 싶으면 "**<시나리오 이름> 을 records/<이름> 참조해서 갱신해줘**" 처럼 요청합니다.

자주 쓰는 테스트는 스킬로 — 반복해서 돌리는 테스트는 스킬(6절)로 만들어 두면 /이름 한 번으로 실행할 수 있습니다. 스킬 본문에는 ".aiassist/tests/<파일> 시나리오로 테스트를 실행한다"와 계정 같은 파라미터만 적고, 단계와 기대값은 시나리오 파일에 두세요 — 그래야 실행 결과가 스킬 파일이 아니라 시나리오 파일에 쌓입니다.

한계: 시뮬레이터는 앱 시작 화면부터 시작하므로 AI가 화면을 이동해 가며 테스트합니다. 프로젝트가 별도 백엔드 서버를 필요로 하는 화면은 그 서버가 켜져 있지 않으면 테스트할 수 없습니다(AI가 도달 불가 사실을 보고합니다).

13. 백그라운드 위임 (서브에이전트)

큰 작업을 **다른 AI 담당자에게 맡기고, 나는 대화를 계속하는** 기능입니다. 별도 버튼이나 메뉴는 없습니다 — AI가 필요하다고 판단하면 스스로 사용하며, "이 작업은 위임해서 진행해줘"처럼 직접 시킬 수도 있습니다. 모든 모델(로컬·Claude·ChatGPT)에서 쓸 수 있습니다.

동작 흐름

1. AI가 작업을 넘기면 채팅에 **DELEGATE: d1g1** 카드가 생기고, 담당자가 무슨 파일을 읽고 고치는지 실시간으로 흐름니다.

2. AI는 기다리지 않고 턴을 끝냅니다. 카드는 대화가 끝난 뒤에도 계속 진행되며, **Stop 버튼으로 대화를 멈춰도 위임은 살아 있습니다.**
3. 담당자가 끝나면 **말을 걸지 않아도 AI가 알아서 이어받습니다.** 채팅에 **✓ 백그라운드 위임이 끝났습니다 - 완료 보고를 받아 이어갑니다.** 한 줄이 뜨고, AI가 완료 보고를 읽어 원래 요청에 맞게 다음 일을 진행합니다 — 산출물을 확인해 실제 프로젝트에 반영하고 무엇을 반영했는지 알려 주거나, 이어지는 작업을 다시 담당자에게 맡깁니다.
4. SpiderGen 프로젝트에서 코드·화면 파일을 반영한 경우에는 **이어서 빌드를 확인하고, 화면(.lay)이면 실제 로드까지 확인**한 뒤 결과를 함께 알려줍니다.
5. 산출물에 문제가 있다고 판단하면 반영하지 않고 그 사실을 알립니다.
6. 위임이 도는 동안 다른 대화를 하고 있었다면, 진행 중인 답변이 끝난 뒤에 이어받습니다. 그 대화 중에 완료 보고가 섞여 들어가면 화면에 **백그라운드 위임 완료 보고를 이번 메시지에 첨부했습니다.** 한 줄이 표시됩니다.

위임한 파일은 직접 고치지 마세요. 위임이 도는 사이 같은 파일을 편집기에서 고치면, AI는 그 파일을 덮어쓰지 않고 **보류한 뒤 알려줍니다**(고친 내용이 사라지지 않도록). 그 다음 현재 파일을 확인해 합치거나, 사용자 판단을 기다립니다.

산출물과 반영

담당자가 만든 파일은 **작업 폴더(.aiassist/quality-work/<위임 id>/)**에만 저장됩니다. 실제 프로젝트 파일은 담당자가 직접 건드릴 수 없고, AI가 검토한 뒤 반영(**apply_files**)할 때만 바뀝니다. 반영 후에는 일반 편집과 똑같이 **Undo**로 되돌릴 수 있습니다.

반영을 일일이 승인할 필요는 없습니다. 결과가 마음에 안 들면 **Undo**로 되돌리거나 "다시 해줘"라고 하면 됩니다. Undo 버튼은 그 턴에만 표시되므로, 지나간 반영을 되돌리려면 "아까 반영한 파일 되돌려줘"라고 하면 됩니다.

담당자의 **마지막 답변(보고)** 전문은 언제나 작업 폴더의 **report.md**에 저장됩니다. 조사·분석처럼 파일이 아니라 답변이 결과인 위임도 AI가 그 내용을 그대로 이어받고, 답변이 아주 길면 AI가 **report.md**를 이어 읽습니다. 답변 길이를 줄이고 싶으면 위임을 시킬 때 "핵심만 2천 자 이내로"처럼 말하면 됩니다.

작업 폴더는 자동으로 지워지지 않으므로, 반영하지 않은 결과물도 나중에 직접 열어볼 수 있습니다.

에이전트 정의 파일 (담당자 지정하기)

담당자가 쓸 **모델·사고 단계·도구·추가 지시**를 파일로 정해 둘 수 있습니다. 작업 폴더의 **.aiassist/agents/<이름>.md**에 6절 Skills와 같은 형식(frontmatter + 본문)으로 두면, AI가 위임할 때 **description**을 보고 알맞은 담당자를 골라 씁니다. "grep_local한테 맡겨줘"처럼 사용자가 이름을 직접 말해도 됩니다.

모든 프로젝트에서 같이 쓸 담당자는 사용자 데이터 폴더의 **<userData>/agents/**에 두면 됩니다(스킬의 전역 폴더와 같은 규칙 — 직접 만들고, 이름이 겹치면 프로젝트 것이 우선). 전역 담당자의 **model**은 그 PC의 **AI Model...**에 등록된 이름이어야 합니다.

name: grep_local

description: 코드 조사 · 검색 전용 담당자. 파일을 고치지 않고 찾은 내용만 보고한다.

model: qwen-local

quality: 1

tools: read_file, search_project, list_files, find_file

maxRounds: 40

프로젝트 파일은 고치지 마라. 요청한 기능이 어디에 어떻게 구현돼 있는지 찾아 파일 경로와 함께 보고

필드	설명
<code>name</code> · <code>description</code>	필수. <code>description</code> 은 AI가 이 담당자를 고르는 기준이니 한 줄로 또렷하게 적으세요
<code>model</code>	<code>Option > AI Model...</code> 에 등록해 둔 모델 이름. 비워 두면 채팅과 같은 모델을 씁니다
<code>quality</code>	사고 단계 <code>1 ~ `5`</code> . 비워 두면 채팅과 같은 단계를 씁니다
<code>tools</code>	담당자에게 허용할 도구를 심표로 나열. 기본 도구 가운데서 고르거나, 14절에서 연결한 MCP 도구 (<code>mcp__<서버>__<도구></code>)를 적어 담당자에게 줄 수 있습니다. MCP 도구는 위임할 때 그 서버가 연결돼 있어야 합니다(끊겨 있으면 위임이 시작되지 않고 AI 가 알려 줍니다). 비워 두면 기본 도구 전부
<code>maxRounds</code>	담당자가 도는 작업 횟수 상한. 모델별 상한을 넘겨 적으면 그 상한으로 줄여서 씁니다
<code>wait</code>	<code>true</code> 면 기다리는 위임 — AI 가 턴을 끝내지 않고 담당자의 결과를 받아 같은 답변 안에서 이어 갑니다(아래)
<code>profile</code>	<code>plain</code> 이면 코딩이 아닌 역할 (검토·판단·요약 등)용 담당자 — 코딩 규칙·작업 폴더 안내가 빠지고, 본문에 적은 출력 형식을 그대로 따릅니다. 기본 도구는 읽기 전용이며, <code>tools</code> 에 파일을 고치는 도구를 적으면 정의가 무효입니다. 생략하면 종전 코딩 담당자
본문	담당자에게 주는 추가 지시. 기본 지시(작업 폴더 제한·보고 형식)는 그대로 유지됩니다(<code>profile: plain</code> 이면 본문의 형식이 우선)

기다리는 위임 (`wait: true`)

기본 위임은 백그라운드라 AI 가 턴을 끝내고, 끝나면 알아서 이어받습니다(위 동작 흐름). `wait: true` 인 담당자에게 맡기면 **AI 가 그 자리에서 결과를 기다렸다가 같은 답변 안에서 다음 단계로 넘어갑니다** — "설계 검토 → 구현 → 테스트"처럼 단계를 잇는 스킬(6절)에 씁니다. 한 번에 여러 담당자를 부르면 **동시에** 돌고, 기다리는 동안 진행은 `DELEGATE` 카드가 보여 줍니다. 이때 **Stop** 을 누르면 담당자도 **함께 멈춥니다**(백그라운드 위임은 Stop 과 무관하게 계속 돕니다). 멈춘 위임은 "dlg3 이어서 해줘"처럼 다시 이어 갈 수 있습니다.

- 에이전트를 지정해 위임하면 **DELEGATE** 카드 제목과 완료 보고에 `agent=<이름>` · `model=<모델>` 이 함께 표시됩니다.
- 채팅창에 "코드 조사만 하는 에이전트 파일 만들어줘"라고 하면 AI가 내장 스킬 `create_agent` (6절)를 읽고 올바른 위치·형식으로 만들어 주고(쓸 모델 이름을 모르면 먼저 물어봅니다), **새로 만든 정의는 그 세션에서 바로 쓸 수 있습니다**(Skills와 같은 규약 — 다만 **같은 이름의 정의를 고쳐 쓴 경우에는** 작업 폴더를 다시 선택해야 반영됩니다).
- 형식이 잘못된 정의(필수 항목 누락, `tools`에 쓸 수 없는 도구 이름 등)는 목록에서 빠지고 서버 로그에 그 사유가 한 줄 남습니다. AI에게 만들게 했다면 AI가 그 사유를, 이미 있는 이름과 겹치면 그 사실을 바로 알려 줍니다.
- **정의 파일이 하나도 없으면 종전과 똑같이 동작합니다** — 채팅과 같은 모델·사고 단계로, 도구도 전부 쓰면서 돌아갑니다.

알아둘 점

- 기본적으로 **동시에 2개까지** 돌고, 그보다 많이 맡기면 거절하지 않고 **차례를 기다렸다가**(기본 8건까지) 앞 작업이 끝나는 대로 시작합니다. 대기열까지 차면 그때 거절됩니다. 동시 개수와 대기 건수는 `ai-config.json`의 `options.delegate`에서 바꿀 수 있습니다(`maxConcurrent` 18 → `maxQueued` 64 — 로컬 LLM은 2, 클라우드는 4처럼 모델 처리량에 맞춥니다).
 - 한 작업이 **15분 동안 진척이 없거나**(명령 실행처럼 한 단계가 오래 걸리는 것만으로는 중단되지 않고, 승인 카드에 답을 기다리는 시간도 세지 않습니다) **시작한 지 2시간**이 지나면 자동으로 중단됩니다(대기열에서 기다린 시간은 세지 않습니다).
 - 담당자의 완료 보고를 받으면 AI가 다음 일을 **사용자 입력 없이 이어서 진행**합니다. 이렇게 자동으로 이어 가는 것은 **연속 20회까지**이고, 그 뒤에는 채팅에 안내가 뜨며 메시지를 보내야 계속합니다(남은 보고는 그 메시지에 함께 실립니다).
 - 완료 보고에는 **사실만** 실립니다 — 담당자가 확인한 내용(가정·미확인 사항은 따로 표시)과 산출 파일 위치. 반영할지, 다시 맡길지, 사용자에게 물어볼지는 AI가 원래 요청을 보고 정합니다.
 - 진행 상황이 궁금하면 "위임 상태 알려줘", 그만두려면 "위임 중단해줘"라고 하면 됩니다. 중단해도 그때까지 만든 파일은 작업 폴더에 남습니다.
 - 담당자가 셸 명령을 실행하려 하면 평소와 같은 **승인 카드**가 뜹니다. 여러 담당자가 동시에 요청하면 한 번에 하나씩 차례로 뜹니다(10절).
 - 담당자는 기본적으로 **세션과 같은 모델·사고 단계**를 씁니다(위의 에이전트 정의 파일로 바꿀 수 있습니다) — 클라우드 모델(Claude/ChatGPT)에서는 대화와 위임 양쪽에 요금이 나갑니다.
 - **앱이나 서버를 다시 시작하면** 진행 중이던 위임은 멈추고 완료 보고도 전달되지 않습니다. 그때까지의 산출물은 작업 폴더에 남아 있으니 직접 열어 쓰면 됩니다(이 경우 AI에게 반영을 시키면 거부됩니다 — 다시 위임하거나 파일을 직접 복사하세요).
-

14. MCP 서버 연결 (외부 도구 붙이기)

MCP(Model Context Protocol) 서버는 디자인 도구·사내 시스템·파일 서버처럼 **다른 프로그램이 제공하는 도구** 모음입니다. 여기에 연결해 두면 AI가 대화 도중 그 도구들을 자기 도구처럼 불러 씁니다.

`Option > MCP Servers...` 로 등록 창을 엽니다. 왼쪽이 등록된 서버 목록, 오른쪽이 선택한 서버의 설정입니다.

서버 등록

1. **추가** 버튼으로 항목을 만들고 **Name** 에 서버 이름을 적습니다. 영문·숫자·`_`·`-` 만 쓸 수 있으며, 이 이름이 도구 이름 앞에 붙습니다.
2. **방식** 을 고릅니다. 고른 방식에 필요한 칸만 표시됩니다.
 - **stdio (로컬 프로세스)** — 이 PC에서 서버 프로그램을 직접 실행합니다.
Command(`npx` · `node` · `python` 또는 실행 파일 경로), **Args**(공백으로 구분하고 공백이 든 인자는 `"따옴표"`), **Cwd**(작업 폴더, 선택), **Env**(환경 변수, 한 줄에 하나 `"이름": "값"`).
 - **http (Streamable HTTP) · sse (구형 HTTP+SSE)** — 이미 떠 있는 서버에 주소로 붙습니다.
URL(`http://` 또는 `https://` 로 시작), **Headers**(요청 헤더, 한 줄에 하나 `"Authorization": "Bearer ..."`).
3. **사용** 체크를 끄면 그 서버에는 접속하지 않습니다(설정은 남습니다).
4. **호출마다 승인** 을 켜면 AI가 그 서버의 도구를 쓸 때마다 승인 카드가 뜹니다(아래 참고).
5. **Timeout** 은 도구 한 번 호출의 제한 시간(ms)입니다. 비워 두면 120000(2분)입니다.
6. **삭제** 는 선택한 항목을 목록에서 지웁니다.
7. **확인** 을 누르면 저장과 동시에 접속합니다 — 앱이나 서버를 다시 시작할 필요가 없습니다.

연결 상태 확인

- **상태** 줄에 선택한 서버의 현재 상태가 표시됩니다 — ● 연결됨 - 도구 N개 · ... 연결 중 · ● 오류: ... · ○ 사용 안 함 · ○ 미저장 - **확인을 누르면 서버가 접속합니다** .
- 연결되면 그 아래 칸에 서버가 제공하는 **도구 목록**(이름과 설명)이 표시됩니다.
- **연결 테스트** 버튼은 저장하지 않고 지금 입력한 값으로 한 번 붙어 봅니다. 성공하면 ● 연결 성공 - 도구 N개 와 도구 목록이 표시됩니다.

계정 로그인 필요한 서버 (OAuth)

http/sse 서버 중에는 토큰 헤더 대신 **계정 로그인**을 요구하는 곳이 있습니다.

1. **OAuth 로그인 사용** 을 체크합니다 — 체크해야 로그인·로그아웃 버튼이 활성화됩니다.
2. **로그인** 을 누르면 브라우저가 열려 그 서비스의 승인 화면으로 갑니다. 승인이 끝나면 앱이 이어받아 자동으로 접속하고, 상태 문구가 "로그인 완료" 로 바뀝니다.
3. 발급된 토큰은 **암호화되어** 저장되고 만료되면 자동으로 갱신됩니다. **로그아웃** 은 저장된 토큰을 지웁니다.
4. 연결 테스트에서 ● 로그인 필요 가 나오면 이 로그인을 먼저 하면 됩니다.

채팅에서 보이는 모습

- 연결된 서버의 도구는 채팅에서 `mcp__<서버이름>__<도구이름>` 이라는 이름으로 보입니다 — 예를 들어 `files` 서버의 `read_file` 도구는 `mcp__files__read_file` 입니다. AI 가 부르면 다른 도구와 똑같이 도구 실행 카드로 표시됩니다.
- **호출마다 승인** 을 켜 서버에서는 실행 전에 `MCP <서버> / <도구>` 와 넘길 값이 적힌 승인 카드가 뜹니다. 읽기 전용이라고 서버가 알려 온 도구는 승인 없이 실행됩니다. 이 승인 카드에서 **"이번 세션에서 자동승인"** 을 체크하고 Allow 를 누르면, 그 세션(연결) 동안은 MCP 도구 호출을 더 묻지 않습니다(새 세션에서는 다시 묻습니다). 이 체크는 MCP 호출에만 적용되며, 10절의 셸 명령 자동승인과는 서로 별개입니다.
- 도구는 **연결에 성공한 서버 것만** AI 에게 알려집니다. 사용을 켜거나 접속에 실패한 서버의 도구는 아예 없는 것으로 취급됩니다.

알아둘 점

- 등록 내용은 세션 데이터와 같은 사용자 데이터 폴더의 `<userData>/ai-mcp-config.json` (8절 참고)에 저장됩니다. 형식은 Claude Code-Cursor 와 같은 `mcpServers` 맵이라 다른 도구의 안내에 있는 예시를 그대로 붙여 넣어도 되고, 파일을 직접 고쳐 저장하면 바로 반영됩니다.
- Headers 에 적는 토큰은 모델 설정의 API 키와 같은 수준으로 보관됩니다.
- 도구 설명이 긴 서버를 여러 개 붙이면 매 요청에 실리는 내용이 늘어 응답이 느려지고 비용이 늘 수 있습니다. 쓰지 않는 서버는 **사용** 체크를 꺼 두세요.
- 13절의 백그라운드 담당자에게는 MCP 도구가 기본으로 주어지지 않습니다. 담당자에게 쓰게 하려면 에이전트 정의 파일의 `tools` 에 `mcp__<서버>__<도구>` 이름을 적어 주세요(13절 참고).

15. 자동 업데이트

Khairon 은 켜져 있는 동안 새 판이 있는지 스스로 확인하고, 있으면 화면 안에서 알려 줍니다.

- 앱이 **먼저 열리고 12초 뒤** 새 판을 확인하며, 그 뒤로는 **4시간마다** 다시 확인합니다. 시작할 때 확인이 끝나기를 기다리는 창은 없습니다.
- 새 판이 있으면 화면 **오른쪽 아래에 알림 카드**가 뜹니다.
 - `새 버전 x.y.z 이 있습니다` — 설치 파일이 바뀐 판입니다.
 - `화면 업데이트가 있습니다 (콘텐츠 N)` — 화면만 바뀐 판입니다.
- **[업데이트]** 를 누르면 내려받기가 시작되고 카드에 진행률이 표시됩니다. 설치 파일 판은 받은 뒤 설치하고 다시 시작하며, 화면 판은 다시 시작 한 번으로 적용됩니다.
- **[나중에]** 를 누르면 카드가 닫히고, 다음 확인 때 다시 안내합니다.
- 내려받지 못하면 카드에 `업데이트를 받지 못했습니다 · 네트워크를 확인한 뒤 다시 시도하세요` 가 표시됩니다. **[다시 시도]** 로 다시 받거나 **[닫기]** 로 닫습니다.
- 폐쇄망용 설치본(업데이트 확인을 끄고 만든 판)은 확인 자체를 하지 않습니다. 사내 서버에서 업데이트를 받으려면 관리자가 안내한 설정 파일을 두면 됩니다.

- **지금 쓰는 판 확인:** `Help > About` 에서 `Version : 1.5.2` · `Release : AS-20261014-54` 처럼 보이며, 마지막 숫자가 패치 번호입니다. 판마다 무엇이 바뀌었는지는 Khairon 홈페이지의 **업데이트 내역** 메뉴에서 볼 수 있습니다.
-

요약: 첫 사용 흐름

1. `File > New Session` 으로 세션 생성
2. 채팅창 하단에서 작업 폴더(cwd) 선택 → 신뢰 확인
3. 필요하면 `Option > AI Model...` 에서 사용할 모델 등록/선택 (API 키 또는 계정 OAuth 로그인)
4. 채팅창에 요청 입력 → AI가 필요 시 코드/샘플을 자동 참고하고, 파일 수정이나 명령 실행이 필요하면 도구를 호출(위험한 명령은 승인 요청). 결과가 마음에 안 들면 **Undo**로 되돌리기.
5. 파일 경로를 클릭하면 Monaco 뷰어로 열람/편집 가능(Git 변경분은 거터에 표시)
6. 반복되는 작업 규칙은 "스킬로 저장해줘"라고 요청해 `.aiassist/skills/` 에 누적 → 다음부터 자동 활용. 짧은 사실/선호는 "기억해줘"라고 요청하면 `.aiassist/memory.md` 에 누적되어 이후 세션에도 자동 반영.
7. 세션 목록에서 언제든지 이전 대화로 돌아와 이어가기