

바이브 메이커 · 무료 자료

프롬프트 마스터 설치와 사용

거칠게 적은 요청을
붙여넣을 프롬프트로 바꿔주는 스킬.

설치 4단계 · 부르는 법 · 여섯 칸 틀 · 실제 예시 · 한계

목차

01 · 돈은 첫 답이 아니라 그다...	모자란 건 길이가 아니라 결과를 바꾸는 정보입니다.	03
02 · 설치 · 브라우저	브라우저에서 네 단계면 끝납니다.	04
03 · 설치 · 클로드코드	터미널이 익숙하다면 파일로 둘 수도 있습니다.	05
04 · 부르는 법	엉망인 문단으로 쏟아내도 됩니다.	06
05 · 돌려받는 것	길게 만드는 도구가 아닙니다.	07
06 · 설치 안 해도 쓰는 틀	여섯 칸만 채우면 추측 여섯 개가 사라집니다.	08
07 · 빈칸이 생기는 자리	보내기 전에 여섯 갈래만 확인합니다.	09
08 · 실제로 이렇게 바뀝니다	좋은 요청이 긴 건 설명을 붙여서가 아닙니다.	10
09 · 알고 쓰기	모델이 똑똑해지는 게 아닙니다.	11

모자란 건 길이가 아니라 결과를 바꾸는 정보입니다.

흔한 흐름은 **네 번**입니다. 짧게 요청하고, 다른 결과를 받고, 빠진 의도를 두세 번 더 설명하고, 네 번째를 씁니다. 한 번의 작업에 **답변 네 개와 확인 세 번**이 들어간 셈인데 구독형에서는 매번 돈이 빠지는 게 안 보여서 알아차리기 어렵습니다.

- **1 짧게 요청** — 정확한 조건을 쓰기 번거로워 핵심만 적는다
- **2 다른 결과** — AI 는 적힌 대로 했지만 기대한 모습이 아니다
- **3 의도 보충** — 빠진 범위와 조건을 두세 번 더 설명한다
- **4 네 번째 채택** — 그제서야 쓸 만한 답이 나온다

> ⚡ **모델이 엉뚱하게 일한 게 아닙니다**
>
> 「로그인 화면 예쁘게 만들어 주세요」에는 **성공한 화면의 모습, 쓰는 기술, 고쳐도 되는 파일, 멈출 시점**이 없습니다. AI 는 묻지 않고도 시작할 수 있으니 익숙한 구성으로 빈칸을 채웁니다. 그 추측을 보고 색·파일·기능·예외를 차례로 바로잡는 게 재작업입니다. 모델보다 **요청의 빈칸**을 먼저 보세요.

긴 배경 설명 대신 채울 네 가지

01
완료 조건
어떤 결과가 나오면 끝인지 형식·동작·통과 기준으로.

02
범위 경계
건드릴 곳과 건드리지 않을 곳을 **함께**.

03
중단 조건
언제 멈추고 확인을 받아야 하는지.

04
이미 정한 것
기술 구성·인증 방식·이름 규칙처럼 유지할 선택.

브라우저에서
네 단계면 끝납니다.

1 **ZIP 받기**
깃허브 저장소 `github.com/nidhinjs/prompt-master` 의 **Code** 에서 **Download ZIP**.

2 **스킬 메뉴 열기**
claude.ai 왼쪽 사이드바에서 **Customize** → **Skills**. 한국어 화면이면 **사용자 정의** → **스킬**.

3 **업로드 선택**
+ → **+ Create skill** → **Upload a skill**. 한국어 화면은 **+ 스킬 만들기** → **스킬 업로드**.

4 **활성화 확인**
받은 ZIP 을 그대로 올리고, 스킬 목록에서 **켜져 있는지** 확인합니다.

💡 사용자 정의 스킬은 **코드 실행·파일 생성 기능을 켜야** 동작합니다. 메뉴 이름은 제품 업데이트로 바뀔 수 있으니 **Skills / 스킬** 이라는 단어를 기준으로 찾으세요.

✅ **무엇을 받는 건가** — 저장소에 있는 건 파일 다섯 개뿐입니다. `SKILL.md` · `references/patterns.md` · `references/templates.md` · `README.md` · `LICENSE(MIT)`. **실행 스크립트도 API 키 요구도 없습니다.**

터미널이 익숙하다면 **파일로** 둘 수도 있습니다.

저장소는 이 방법에 **Not Suggested** 라고 적어 두었습니다. 터미널과 깃에 익숙할 때만 고르세요.

개인 스킬 경로 — 모든 프로젝트에서 쓰인다

```
mkdir -p ~/.claude/skills
```

```
git clone https://github.com/nidhinjs/prompt-master.git ~/.claude/skills/prompt-master
```

1. 한 프로젝트에서만 시험하려면 저장소를 읽어본 뒤 `.claude/skills/prompt-master/` 에 둘 수 있습니다.

⚡ **두 설치 경로는 따로 봅니다**

브라우저 업로드와 파일 설치가 **자동으로 동기화된다는 근거는 없습니다.** 한쪽에서 지웠다고 다른 쪽이 사라지지 않고, 한쪽만 업데이트해도 다른 쪽은 옛 버전 그대로입니다. 클로드코드의 스킬 기능은 공식 문서에서 **베타**로 표시돼 있습니다.

엉망인 문단으로 쏟아내도 됩니다.

정리는 내가 아니라 스킬이 합니다. **평소 말투 그대로** 부르면 됩니다.

그냥 말로 부르기

이름을 넣으면 더 잘 잡힙니다

커서에서 로그인 모듈 정리할 때 쓸 프롬프트를 만들어 주세요.

리액트랑 슈파베이스로 할 일 앱 만들라고 클로드코드에 보낼 프롬프트를 써 주세요.

제가 쓴 나쁜 프롬프트에요. 고쳐 주세요. [여기에 붙여넣기]

참고 이미지가 있는데 머리 각도만 바꾸는 프롬프트를 도와주세요.

이름으로 직접 부르기

기본 내장 명령이 아니라 설치한 스킬 이름

```
/prompt-master
```

1. 슬래시 목록에 안 보이면 **「프롬프트 마스터 스킬을 써서 … 프롬프트를 작성해 주세요」** 처럼 이름을 문장에 넣어 부르세요.

길게 만드는 도구가 아닙니다.

한 덩어리로 복사해 쓰는 프롬프트 하나를 돌려줍니다. **결과를 바꾸지 않는 문장은 오히려 덜어냅니다.**

아홉 항목	작업 · 대상 도구 · 출력 형식 · 제약 · 입력 · 맥락 · 독자 · 완료 조건 · 예시를 확인합니다.
최대 세 질문	결정적인 정보가 빠졌을 때만 **최대 세 가지**를 묻고, 그 이상은 묻지 않습니다.
도구별 번역	클로드는 XML 태그로 구역을 나누고, 지피티 계열은 출력 계약과 분량을 분명히 하고, 커서·윈드서프는 파일 경로와 수정 금지 목록을 넣고, 미드저니는 쉼표 묘사와 비율·제외 항목을 씁니다.
에이전트	클로드코드·커서·데빈처럼 **파일을 바꾸고 명령을 실행하는** 도구에는 시작 상태 · 목표 상태 · 허용 작업 · 금지 작업 · 중단 조건 · 진행 확인 지점을 넣습니다.

💡 네 가지가 **모든 요청에 기계적으로 붙지는 않습니다.** 짧은 글쓰기 요청에는 파일 경계가 필요 없고, 대화 이력이 없으면 「이미 정한 것」도 생기지 않습니다.

여섯 칸만 채우면
추측 여섯 개가 사라집니다.

파일을 다루는 **에이전트형 도구**(클로드코드·커서)용 틀입니다. 대괄호 안만 채워 복사하세요.

복사해서 쓰는 틀 대괄호만 바꾸면 됩니다

작업: [정확히 무엇을 만들거나 고칠지 한 문장으로.]

완료 조건: [결과 형식, 화면 크기, 동작, 검사 명령처럼 통과 여부를 확인할 기준.]

건드릴 파일: [수정해도 되는 파일과 폴더.]

건드리지 말 파일: [설정, 환경 변수, 결제, 데이터베이스처럼 수정하면 안 되는 곳.]

중단 조건: [삭제, 새 패키지 설치, 스키마 변경, 범위 밖 수정 전에는 멈추고 물어볼 것.]

이미 정한 것: [기술 구성, 인증 방식, 이름 규칙, 디자인 원칙, 시도했다 제외한 방법.]

💡 빈칸을 **억지로 채우지 마세요.** 글 요약처럼 파일을 안 건드리는 요청이라면 「건드릴 파일」과 「중단 조건」은 빼도 됩니다.

보내기 전에
여섯 갈래만 확인합니다.

스킬을 안 깔아도, 요청을 보내기 전에 이 여섯 가지만 보면 재작업이 줄어듭니다.

갈래	자주 빠지는 것
작업	「예쁘게」「고쳐 줘」 같은 모호한 동사만 적는다
맥락	쓰는 기술과 **이미 시도했다 실패한 방법**을 빼놓는다
형식	화면·표·코드·분량 같은 결과 형태를 안 정한다
범위	수정할 파일만 말하고 **건드리지 말 곳**은 안 적는다
추론	추론 모델에도 습관적으로 「단계별로 생각해」를 붙인다
에이전트	삭제·패키지 추가·DB 변경 앞에 확인 지점을 안 둔다

⚡ **긴 대화에서는 「이미 정한 것」을 다시 줍니다**

기술 구성·인증 방식·이름 규칙·제외한 선택을 짧게 앞에 붙이는 장치입니다. 대화 안의 정보를 추려 전달할 뿐 **새 대화의 과거를 자동으로 찾아오는 기억 기능이 아닙니다.** 세션을 새로 열면 다시 적어줘야 합니다.

좋은 요청이 긴 건
설명을 붙여서가 아닙니다.

나쁜 프롬프트

「로그인 화면 예쁘게 만들어 주세요」

· 완료 기준이 없다
· 수정 파일과 금지 영역이 없다
· 중단할 시점이 없다

좋은 프롬프트

· 화면 크기와 오류 상태를 정한다
· 기존 로그인 함수와 이동 경로를 적는다
· 수정할 파일 세 개와 금지 파일을 정한다
· 패키지·인증 API 변경 전에 멈추게 한다

고쳐 쓴 프롬프트 경로와 검사 명령은 프로젝트에 맞게

작업: Next.js 로그인 화면에 이메일·비밀번호 입력칸, 로그인 버튼, 비밀번호 찾기 링크를 구현해 주세요.

완료 조건: 375px·1440px 가로 스크롤 없음 · 빈 값과 잘못된 이메일에 한국어 오류 문구 · 성공 시 /dashboard 이동 · typecheck·lint·테스트 통과

건드릴 파일: app/login/page.tsx · components/auth/LoginForm.tsx · 기존 로그인 폼 테스트

건드리지 말 파일: .env 와 환경 변수 · 인증 API 와 DB 스키마 · package.json 과 잠금 파일

중단 조건: 새 패키지가 필요하거나 인증 API 변경이 필요하면 수정하지 말고 이유와 선택지를 먼저 알려 주세요.

이미 정한 것: Next.js 와 TypeScript · 기존 디자인 토큰과 공통 입력 컴포넌트 재사용 · 소셜 로그인과 회원가입은 이번 범위 제외

모델이 똑똑해지는 게 아닙니다.

빈칸을 추측해 **다른 일을 하는 문제**는 줄여줍니다. 최신 사실을 알게 되거나 어려운 과제의 실력이 올라가지는 않습니다.

1 **공식 스킬이 아닙니다**
Nidhin Joseph Nelson 이 공개한 **커뮤니티 스킬**입니다. 앤트로픽이 만들거나 검수한 게 아니에요. 설치 전에 파일을 읽고, 업데이트 뒤에 **권한·지시·외부 파일이 추가됐는지** 다시 확인하세요.

2 **「무료」의 범위**
스킬 코드와 설치에 **별도 요금이 없다**는 뜻입니다. 클로드 각 플랜의 사용량 제한은 그대로 적용됩니다. 재작업을 줄여주는 것이지 **사용량이 0 이 되는 게 아닙니다.**

3 **첫 실행이 느리게 느껴질 수 있습니다**
결정적인 정보가 빠지면 바로 답을 내지 않고 **최대 세 가지**를 먼저 묻습니다.

굳이 안 쓰는 게 나은 때

대상과 형식이 이미 분명한 한 번짜리	거치면 더 느리다
이미 검증된 틀에 입력값만 바꿀 때	저장한 틀이 낫다
비밀번호·API 키·고객 개인정보가 든 요청	**넣지 말 것**
삭제·결제·배포·DB 변경처럼 되돌리기 어려운 일	**사람이 먼저**

✅ **가장 작은 실험** — 평소 재작업이 많았던 요청 하나를 넣어보고, 나온 프롬프트에서 완료 조건 · 건드릴 곳 · 건드리지 않을 곳 · 중단 조건 · 앞선 결정이 실제 프로젝트와 맞는지 확인하기. **기준 2026-09-15 · 저장소 README·SKILL.md 와 앤트로픽 공식 도움말.**