Quantum Jump Club

특이점 빌더스 클래스

특이점 빌더스 6주차

처음부터 끝까지 — 실전 프로젝트 빌드 & 배포

제작	퀀텀점프클럽
난이도	중급
소요 시간	3h30분
작성일	2026년 03월 12일

학생 핸드아웃

목차

표지

6주차 오리엔테이션

> ★ HIGHLIGHT
>
> 오늘은 연습경기다. 7-8주차 본경기 전에 한 번 뛰어보자.

개요

커리큘럼 전체 맵

특이점 빌더스 8주 커리큘럼 맵

개념

세션 1: 기획 — 나쁜 기획 vs 좋은 기획

> Claude Code에게 막연한 지시는 막연한 결과를 낳는다. 좋은 기획서 = 좋은 코드.

> ★ HIGHLIGHT
> 건축가에게 '예쁜 집 지어줘'라고 하면 어떻게 될까? 설계도 없이 집을 지을 수 없다. Claude Code도 마찬가지다.

핵심 포인트

- 나쁜 기획: '예쁜 앱 만들어줘' — 결과 예측 불가, 수정 반복, 시간 낭비
- 좋은 기획: 구체적 요구사항 + 기술 스택 + 화면 목록 + DB 스키마
- 기획의 품질이 곧 AI 협업의 효율을 결정한다

Claude Code는 주어진 맥락에서 최선을 다하지만, 맥락이 빈약하면 틀린 방향으로 최선을 다한다. 기획의 핵심은 '모호함 제거'다. 누가 쓰는지, 무엇을 할 수 있는지, 어떤 데이터가 필요한지를 구체적으로 정의하면 Claude는 놀라운 속도로 만들어준다.

교육 자료: 기획의 중요성 비교

나쁜 기획 vs 좋은 기획 비교

개념

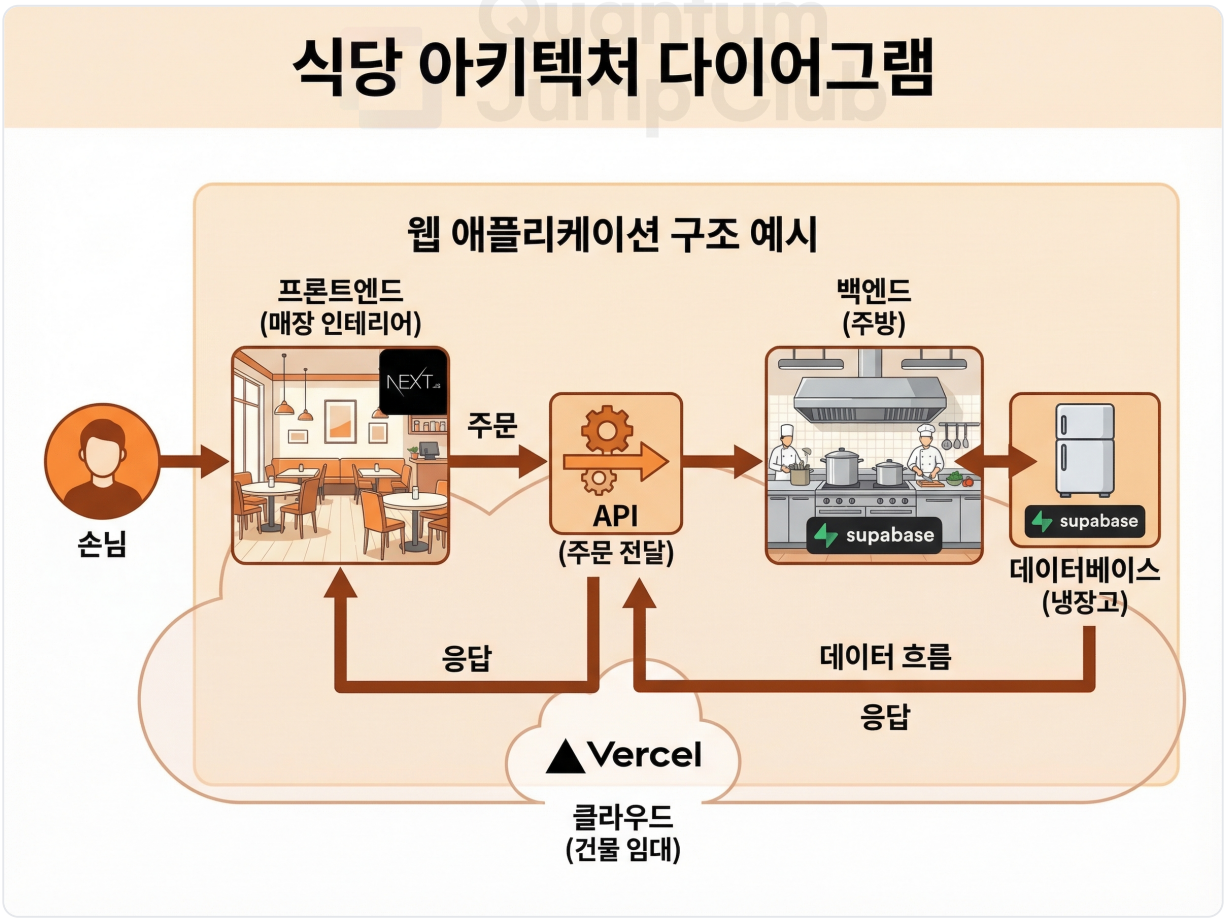
5주차 복습: 소프트웨어 5대 구성요소

> 모든 웹 서비스는 5가지 요소로 구성된다. 이것을 이해해야 기술 스택을 선택할 수 있다.

> ★ HIGHLIGHT
> 레스토랑 비유: 프론트엔드는 매장 인테리어, API는 주문 전달, 백엔드는 주방, 데이터베이스는 냉장고, 클라우드는 건물 임대.

핵심 포인트

- 오늘 사용할 스택: Next.js + Supabase + Vercel + Tailwind CSS
- 비개발자도 이 4가지 도구만으로 실제 서비스를 만들 수 있다
- 모든 결정은 기획서에서 나온다



소프트웨어 5대 구성요소 — 레스토랑 비유

튜토리얼

CLAUDE.md 작성법

> CLAUDE.md는 Claude Code에게 주는 '프로젝트 헌법'이다. 한 번 잘 써두면 매번 설명할 필요가 없다.

> ★ HIGHLIGHT
> 신입 직원에게 주는 업무 매뉴얼과 같다. 회사 규칙, 사용 도구, 하면 안 되는 것들을 미리 정리해두는 것.

팁

- 짧고 명확하게: 500줄 이상이면 오히려 역효과
- DB 스키마는 반드시 포함: Claude의 데이터 설계 실수를 막아줌
- 나중에 언제든 수정 가능: 프로젝트 진행하면서 계속 업데이트

Template

CLAUDE.md

프로젝트 루트에 위치. Claude Code가 자동으로 읽는 파일.

- 프로젝트 개요 — 앱이 무엇을 하는지 1-2문장으로 — # 오늘 뭐 먹지? 팀원들이 점심 메뉴를 제안하고 투표할 수 있는 웹앱.
- 기술 스택 — 사용하는 기술과 버전 — ## Tech Stack - Next.js 15 (App Router) - Supabase (PostgreSQL) - Tailwind CSS - Vercel (배포)
- DB 스키마 — 테이블 구조를 SQL 또는 마크다운으로 — ## Database Schema ### suggestions - id: uuid (PK) - menu_name: text (NOT NULL) - created_at: timestamptz ### votes - id: uuid (PK) - suggestion_id: uuid (FK → suggestions) - created_at: timestamptz
- 화면 목록 — 구현할 페이지들 — ## Pages - `/` 메인: 투표 목록 + 투표 버튼 - `/suggest` 제안 폼 - `/history` 투표 기록
- 금지사항 — Claude에게 절대 하지 말라고 할 것들 — ## Rules - service_role 키를 프론트엔드에 넣지 말 것 - 한 번에 너무 많은 파일을 수정하지 말 것 - 삭제 기능은 확인 없이 구현하지 말 것

튜토리얼

plan.md 작성법

> plan.md는 구현 계획서다. CLAUDE.md가 '무엇을 만드는지'라면, plan.md는 '어떤 순서로 만드는지'다.

> ★ HIGHLIGHT
> 공사 현장의 공정표와 같다. 기초공사 → 골조 → 인테리어 순서처럼, 코딩도 순서가 있다.

팁

- Phase는 4개 이하로: 너무 많으면 관리 불가
- 하나씩 DONE으로 바꿔가며 진행: 작은 성취가 동기부여
- BLOCKED 표시가 중요: 어디서 막혔는지 Claude에게 알려줌

Template

plan.md

- Phase 1: 프로젝트 세팅 — PENDING — ['Next.js 15 프로젝트 생성 (`npx create-next-app@latest`)', 'Supabase 프로젝트 생성 + 테이블 생성', '.env.local 환경변수 설정', 'GitHub 리포 생성 + 초기 commit']
- Phase 2: 핵심 기능 구현 — PENDING — ['메인 페이지: 투표 목록 표시', '제안 폼: 메뉴 이름 입력 + 저장', '투표 버튼: 클릭 시 votes 테이블에 추가', '실시간 갱신: Supabase Realtime 연동']
- Phase 3: UI 개선 — PENDING — ['Tailwind CSS로 반응형 레이아웃', '모바일 최적화', '로딩 상태 표시']
- Phase 4: 배포 — PENDING — ['GitHub push', 'Vercel 연동', '환경변수 Vercel에 설정', '배포 URL 확인']

개요

라이브 코딩 데모: 오늘 뭐 먹지? 앱

오늘 뭐 먹지? 앱 화면 미리보기

가이드

세션 2: Supabase 설정 가이드

> ★ HIGHLIGHT
>
> 데이터베이스 설정은 식당의 냉장고 설치와 같다. 한 번만 제대로 설치하면 이후엔 꺼내 쓰기만 하면 된다.

STEP 1 Supabase 프로젝트 생성

STEP 2 테이블 생성

```
-- SQL로 만들 때
CREATE TABLE suggestions (
  id uuid DEFAULT gen_random_uuid() PRIMARY KEY,
  menu_name text NOT NULL,
  created_at timestamptz DEFAULT now()
);

CREATE TABLE votes (
  id uuid DEFAULT gen_random_uuid() PRIMARY KEY,
  suggestion_id uuid REFERENCES suggestions(id) ON DELETE CASCADE,
  created_at timestamptz DEFAULT now()
);
```


STEP 3 API 키 복사

STEP 4 .env.local 파일 생성

```
NEXT_PUBLIC_SUPABASE_URL=https://xxxxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJxxx...
```

STEP 5 RLS 정책 설정

```
-- 모든 사람이 읽기/쓰기 가능 (개발용)
CREATE POLICY "Enable all operations"
ON public.suggestions FOR ALL
USING (true)
WITH CHECK (true);

-- votes 테이블도 동일하게 적용
```

STEP 6 supabase-js 설치 및 클라이언트 생성

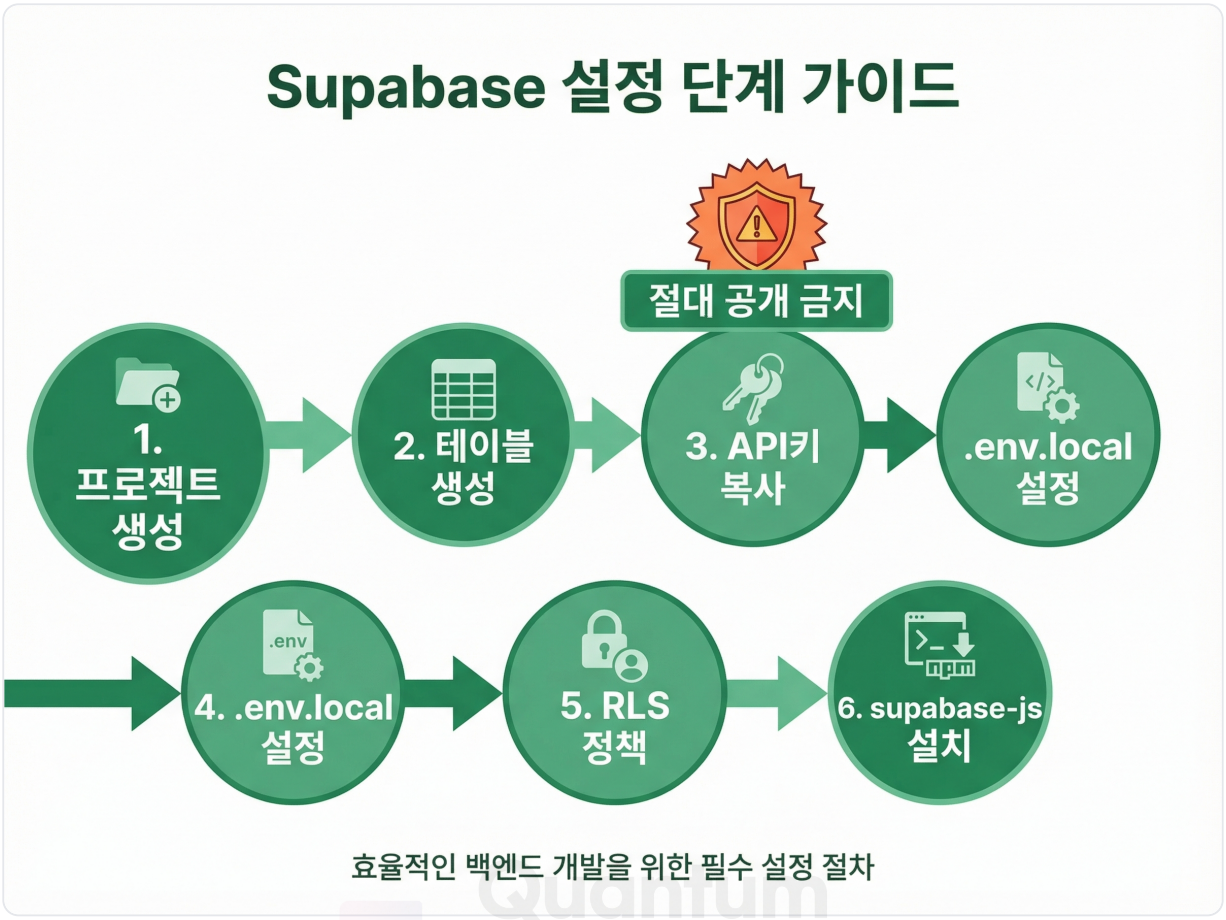
```
# 터미널에서
npm install @supabase/supabase-js

# lib/supabase.ts
import { createClient } from '@supabase/supabase-js'

export const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
)
```

npm install이란?

> **● INFO**
>
> 레고 세트에서 필요한 부품을 가져오는 것과 같다. 우리가 직접 부품을 깎아 만들 필요 없이, 이미 누군가 만들어 둔 부품을 가져다 쓰는 것.

- package.json — 부품 목록표. '이 프로젝트에는 이런 부품들이 필요해'라고 적어둔 파일 — 레고 설명서의 '필요한 부품 목록'
- node_modules — 실제 부품이 보관되는 폴더. npm install을 실행하면 여기에 부품이 다운로드됨 — 레고 부품 보관함 — 꺼내 쓰기만 하면 된다
- npm install — package.json에 적힌 부품들을 인터넷에서 다운로드하는 명령어 — 부품 목록을 보고 택배로 주문하는 행위

node_modules 폴더는 매우 크므로 절대 GitHub에 올리지 않는다 (.gitignore에 자동 포함). 다른 컴퓨터에서는 npm install만 실행하면 부품이 다시 배달된다.



Supabase 6단계 설정 가이드

모범 사례

Claude Code로 앱 빌드하는 법

> Claude Code에게 효과적으로 지시하는 방법. 마법 주문이 아니라, 명확한 소통이 핵심이다.

> ★ HIGHLIGHT
> Claude Code는 천재 신입 개발자와 같다. 실력은 뛰어나지만, 맥락을 모르면 방향을 잃는다. 당신이 좋은 팀장이 되어야 한다.

권장 사항

- DO CLAUDE.md를 항상 최신으로 유지
- DO 한 번에 한 가지 기능만 요청
- DO 에러는 전체 메시지를 복사해서 공유
- DO 결과를 직접 확인하고 구체적 피드백
- DO plan.md로 진행 상황 추적

피해야 할 것

- DON'T 여러 기능을 한 번에 요청
- DON'T 에러 없이 '작동 안 해'라고만 말하기
- DON'T Claude가 만든 코드를 이해 없이 복붙
- DON'T CLAUDE.md 없이 시작
- DON'T 에러를 무시하고 계속 진행

효과적인 패턴

단계별 지시

BAD 투표 앱 다 만들어줘

GOOD 메인 페이지(/page.tsx)에 Supabase에서 suggestions를 불러와서 카드 형태로 보여주는 컴포넌트를 만들어줘. 각 카드에는 메뉴 이름과 투표 수가 표시되어야 해.

구체적일수록 한 번에 원하는 결과를 얻음

에러 공유

BAD 에러 났어, 고쳐줘

GOOD 이런 에러가 났어: [에러 메시지 전체 복사]. /app/page.tsx 파일 수정해줘.

에러 메시지가 정확한 진단 정보

피드백 루프

BAD (결과 보고 다시 처음부터 지시)

GOOD 투표 버튼이 모바일에서 너무 작아. 최소 48px 높이로 만들어줘.

작은 수정 요청이 빠르고 정확함

파일 명시

BAD 그 파일 수정해줘

GOOD /components/VoteCard.tsx 파일에서 투표 버튼 부분만 수정해줘

어떤 파일을 건드릴지 명확히 해야 실수가 없음

AI 출력은 반드시 확인하라

> **INFO**
>
> Claude는 유능한 신입사원과 같다. 실력은 뛰어나지만, 가끔 모르는 것을 자신 있게 말한다. 보고서를 받으면 팩트체크를 하듯, AI 코드도 반드시 검증해야 한다.

- 존재하지 않는 API나 라이브러리를 만들어낼 수 있음 (할루시네이션)
- 오래된 버전의 코드를 최신이라고 제시할 수 있음
- 작동하는 것처럼 보이지만 미묘한 버그가 숨어있을 수 있음
- 보안 취약점이 있는 코드를 생성할 수 있음 (예: API 키 노출)
- 코드를 실행해서 실제로 동작하는지 확인
- 에러 없이 돌아가더라도 예상대로 작동하는지 테스트
- API나 라이브러리 이름이 실제로 존재하는지 공식 문서 확인
- '이해 안 되는 코드'가 있으면 반드시 질문: '이 부분 설명해줘'

AI가 만든 코드를 배포하기 전에 꼭 한 번 돌려보고, 이해하고, 확인하라.

코드 예제

Supabase CRUD 패턴

> Supabase JavaScript SDK로 데이터를 읽고, 쓰고, 수정하고, 삭제하는 패턴

app/page.tsx

```
import { supabase } from '@/lib/supabase'

// 모든 제안 가져오기 (투표 수 포함)
const { data: suggestions, error } = await supabase
  .from('suggestions')
  .select(`
  *,
    votes(count)
 `)
  .order('created_at', { ascending: false })

if (error) console.error(error)
console.log(suggestions)
```


supabase.from('테이블명').select()로 데이터를 가져온다. 관계형 쿼리(votes count)도 한 번에 가능.

app/suggest/page.tsx

```
// 새 메뉴 제안 추가
const { data, error } = await supabase
  .from('suggestions')
  .insert([
    { menu_name: '짜장면' }
 ])
  .select()

if (error) {
  console.error('제안 저장 실패:', error)
  return
}
console.log('저장 완료:', data)
```

insert()로 새 데이터를 추가한다. 에러 처리를 항상 추가할 것.

components/VoteButton.tsx

```
const handleVote = async (suggestionId: string) => {
  const { error } = await supabase
    .from('votes')
    .insert([{ suggestion_id: suggestionId }])

  if (error) {
    alert('투표 중 오류가 발생했습니다: ' + error.message)
    return
 }

 // 화면 새로고침
  router.refresh()
}
```

투표는 단순 INSERT다. suggestion_id로 어떤 메뉴에 투표했는지 연결.

app/page.tsx

```
useEffect(() => {
  const channel = supabase
    .channel('votes-changes')
    .on(
      'postgres_changes',
      { event: 'INSERT', schema: 'public', table: 'votes' },
      (payload) => {
        console.log('새 투표!', payload)
        fetchSuggestions() // 목록 다시 불러오기
   }
  )
    .subscribe()

  return () => {
    supabase.removeChannel(channel)
 }
}, [])
```

Supabase Realtime으로 다른 사람이 투표하면 자동으로 화면이 업데이트된다.

문제 해결

에러 대처 가이드

> 에러는 적이 아니다. 에러 메시지는 Claude에게 주는 '수리 설명서'다.

★ HIGHLIGHT

의사에게 증상을 말할 때 '몸이 아파요'보다 '어제부터 오른쪽 옆구리가 찌르는 것처럼 아파요'가 훨씬 낫다. 에러 메시지가 바로 그 '구체적 증상'이다.

증상 1 Error: NEXT_PUBLIC_SUPABASE_URL is not defined

원인 .env.local 파일이 없거나 변수명 오타

해결 .env.local 파일 확인. NEXT_PUBLIC_ 접두사 필수. 서버 재시작 필요 (npm run dev 재실행)

증상 2 Error: new row violates row-level security policy

원인 Supabase RLS가 활성화되어 있지만 정책이 없음

해결 Supabase Dashboard → Authentication → Policies → 해당 테이블에 INSERT 정책 추가

증상 3 TypeError: Cannot read properties of undefined

원인 데이터가 로딩되기 전에 화면에 표시하려고 함

해결 로딩 상태 처리 추가. if (!data) return <Loading />

증상 4 Vercel 배포 후 환경변수 오류

원인 Vercel에 환경변수를 설정하지 않음

해결 Vercel Dashboard → Settings → Environment Variables에 NEXT_PUBLIC_SUPABASE_URL 등 추가 후 재배포

증상 5 Supabase 프로젝트가 일시정지됨

원인 무료 플랜: 7일 미사용 시 자동 일시정지

해결 Supabase Dashboard → Restore 클릭. 개발 중엔 매주 한 번 접속해두기

가이드

세션 3: GitHub + Vercel 배포 가이드

★ HIGHLIGHT

집에서 요리를 잘 했다면, 이제 식당을 열어서 손님을 받는 것과 같다. GitHub은 요리법을 보관하는 곳, Vercel은 실제 식당이다.

STEP 1 GitHub 리포 생성

```
# 터미널에서
git init
git add .
git commit -m "init: 점심 투표 앱 초기 커밋"
git branch -M main
git remote add origin https://github.com/[유저명]/lunch-vote.git
git push -u origin main
```

STEP 2 Vercel 가입 및 연동

STEP 3 환경변수 설정 (중요!)

STEP 4 Deploy 클릭

STEP 5 자동 재배포 확인


```
# 수정 후 push만 하면 자동 배포
git add .
git commit -m "fix: 투표 버튼 크기 개선"
git push
```

좋은 커밋 메시지 쓰기

● INFO

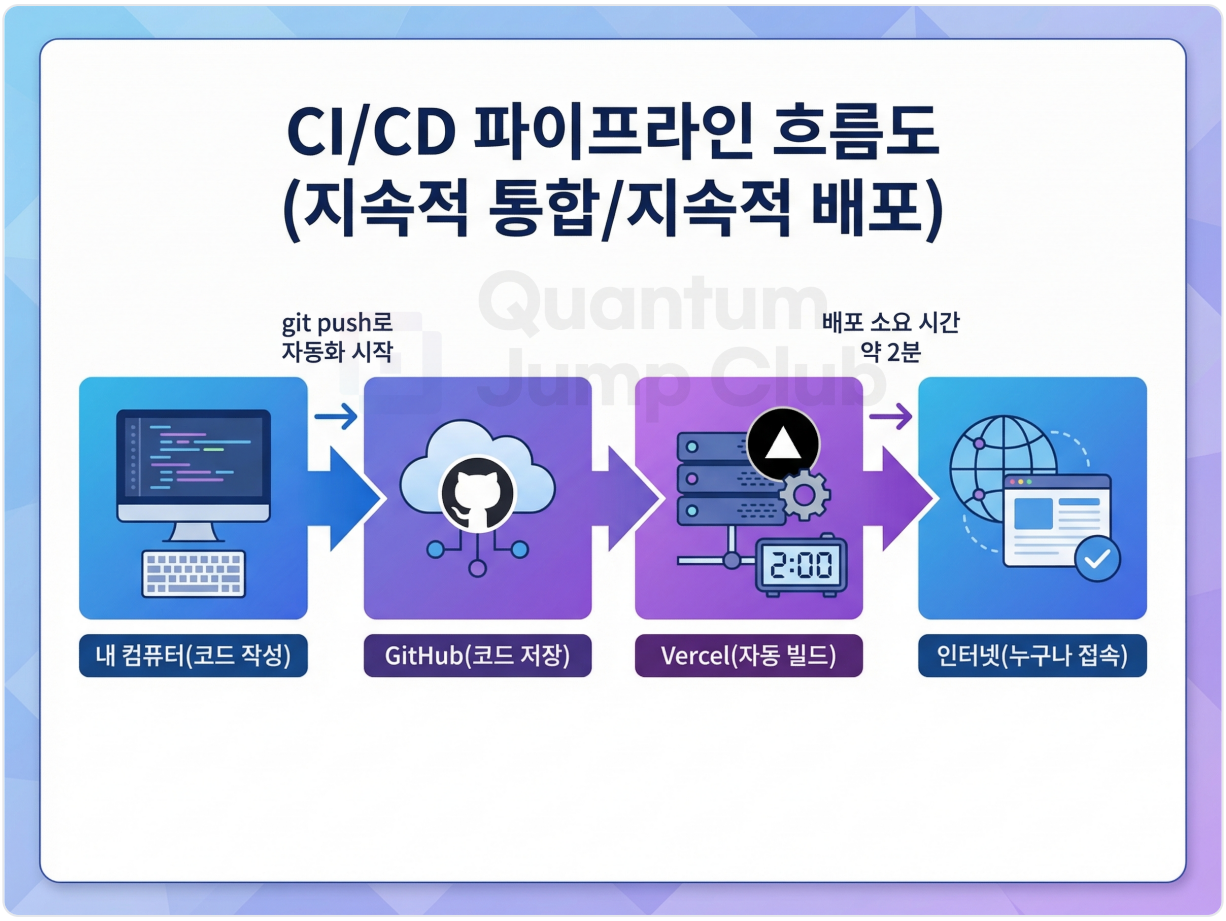
커밋 메시지는 일기장과 같다. 6개월 뒤 내가 봤을 때 '그때 왜 이걸 바꿨더라?'에 답할 수 있어야 한다. What(무엇)이 아닌 Why(왜)를 쓰라.

타입: 설명 (예: fix: 투표 버튼이 모바일에서 안 눌리는 문제 수정)

- feat — 새 기능 추가 — feat: 투표 히스토리 페이지 추가
- fix — 버그 수정 — fix: 투표 수가 음수로 표시되는 오류 수정

- docs — 문서 수정 — docs: README에 설치 방법 추가
- style — 디자인/UI 변경 — style: 버튼 색상 파란색으로 변경
- git commit -m "수정"
- git commit -m "update"
- git commit -m "ㅇㅇ"
- git commit -m "asdf"
- git commit -m "feat: 메뉴 제안 폼 추가"
- git commit -m "fix: 모바일에서 투표 버튼 터치 안 되는 문제 해결"
- git commit -m "style: 메인 페이지 카드 레이아웃 그리드로 변경"

Claude Code에게 'git commit 해줘'라고 하면 자동으로 좋은 커밋 메시지를 만들어준다.



GitHub → Vercel 자동 배포 파이프라인

치트시트

배포 전 체크리스트

개요

7-8주차 프로젝트 가이드

비교

프로젝트 아이디어 예시 (30개 중 추천 5개)

> 7-8주차 프로젝트로 만들 수 있는 아이디어들. 핵심은 '실제로 쓰고 싶은 것'을 만드는 것.

프로젝트	설명	핵심 기능	난이도
오늘의 할일	할일 체크 + 습관 트래커	할일 추가/완료, 연속 달성일 계산	★ 쉬움
독서 기록장	책 등록 + 메모 + 통계	책 등록, 읽기 상태, 월별 통계	★★ 보통
점심 투표	메뉴 투표 + 실시간 결과	제안, 투표, 실시간 순위	★★ 보통
포트폴리오	프로젝트 쇼케이스 + 좋아요	프로젝트 카드, 좋아요, 방문자 수	★★★ 도전
AI 면접 코치	AI 질문 생성 + 답변 가이드	직무별 질문 생성, 답변 힌트	★★★ 도전

치트시트

Claude Code 프롬프트 패턴 8가지

모범 사례

자주 하는 실수 8가지

> 비개발자가 Claude Code로 프로젝트를 진행할 때 가장 자주 하는 실수들

권장 사항

- DO Phase 1 → Phase 2 → Phase 3 순서대로 진행
- DO 에러가 나면 즉시 Claude에게 공유
- DO 매일 git commit으로 진행 상황 저장
- DO 완성된 기능 먼저, 예쁜 디자인은 나중
- DO Vercel 배포는 빨리 (Phase 1 완료 직후도 OK)
- DO 막히면 10분 뒤 다시 시도 (Claude 세션 리셋)
- DO CLAUDE.md를 업데이트하며 프로젝트 관리
- DO 작동하는 MVP 먼저, 기능 추가는 그 다음

피해야 할 것

- DON'T 기획 없이 바로 '만들어줘' (가장 흔한 실수)
- DON'T 에러를 무시하고 계속 진행 (나중에 더 큰 문제)
- DON'T git 없이 작업 (실수 복구 불가)
- DON'T 처음부터 완벽한 UI를 만들려 함 (시간 낭비)
- DON'T 배포를 마지막에만 하려고 함 (환경 차이 문제)
- DON'T 막혀서 몇 시간을 혼자 고민 (Claude에게 물어보기)
- DON'T 코드 한 줄도 이해 없이 진행 (발표 때 문제)
- DON'T 기능을 너무 많이 기획 (완성도 > 기능 수)

모범 사례

Claude Code 실전 팁

Claude Code를 더 효과적으로 사용하기 위한 실전 노하우. 도구를 잘 쓰는 것도 실력이다.

★ HIGHLIGHT

자동차를 사면 매뉴얼을 한 번은 읽어야 한다. 에어컨 켜는 법, 기름 넣는 곳, 계기판 경고등 의미를 모르면 좋은 차도 제대로 못 쓴다. Claude Code도 마찬가지다.

컨텍스트 윈도우 관리

● INFO

Claude의 기억력에는 '책상 크기' 제한이 있다. 대화가 길어지면 책상 위에 서류가 쌓여서 중요한 것을 못 찾는다.

- /clear 명령어로 대화 초기화 — 막힐 때 새로 시작하면 오히려 빨라짐
- CLAUDE.md에 핵심 맥락을 적어두면 /clear 후에도 Claude가 프로젝트를 바로 이해함
- 30분 이상 같은 주제로 대화했으면 /clear 고려
- 에러가 계속 반복되면 대화가 꼬인 것 — /clear가 답일 수 있음

코드 리뷰 with Claude

● INFO

요리를 다 만들고 나서 셰프에게 '이거 맛 좀 봐주세요'라고 하는 것과 같다. Claude에게 리뷰를 요청하면 놓친 문제를 찾아준다.

- '내 코드 리뷰해줘. 특히 보안 문제가 있는지 확인해줘'
- '이 컴포넌트가 비효율적인 부분이 있는지 알려줘'
- '초보자 관점에서 이 코드가 이해하기 어려운 부분이 있으면 알려줘'
- 작성한 코드를 Claude에게 설명해달라고 하면 이해도도 높아짐

비용 관리

> **INFO**
>
> Claude Code는 전기처럼 쓴 만큼 비용이 나간다. 불 끄듯 안 쓸 때는 절약하고, 필요할 때 집중해서 쓰는 게 현명하다.

- /cost 명령어로 현재 사용량 확인
- CLAUDE.md를 잘 써두면 반복 설명이 줄어 토큰 절약
- 큰 파일은 전체 대신 특정 부분만 지정: '이 파일의 handleVote 함수만 수정해줘'
- 불필요한 대화 줄이기: 감사합니다/네 같은 짧은 응답 대신 다음 지시 바로 주기

과제

6주차 → 7주차 과제

> **INFO**
> 제출 방법: {'how': '7주차 수업 시작 시 GitHub URL + Vercel URL 공유', 'format': 'Slack 또는 수업 중 발표'}

AI 에티켓 — AI 사용을 떳떳하게 공개하기

> **INFO**
> 좋은 요리사는 재료 출처를 숨기지 않는다. '제주산 흑돼지'라고 당당히 쓰듯, AI를 잘 활용하는 것도 역량이다.

- README에 AI 사용 공개 — 프로젝트 README.md에 'AI 도구 활용' 섹션을 추가하라 — ## Built with - Claude Code (AI 코딩 어시스턴트) - Next.js 15 - Supabase
- AI 활용은 역량이다 — AI를 쓴다고 부끄러워할 필요 없다. 검색 잘하는 것이 실력이듯, AI를 잘 활용하는 것도 실력이다. — 발표 때 'Claude Code로 어떻게 만들었는지' 과정을 공유하면 오히려 높은 평가를 받는다
- AI가 만든 코드도 내 책임 — AI가 생성한 코드의 버그, 보안 문제는 사용자의 책임이다. 반드시 이해하고 검증한 후 사용하라. — 발표에서 '이 부분은 Claude가 만들었지만, 왜 이렇게 작동하는지 이해하고 있습니다'라고 말할 수 있어야 한다

숨기지 말고 당당하게 — AI를 도구로 잘 쓰는 능력이 미래 경쟁력이다.

개념

Ralph Loop(랄프 루프) — 반복 검증 자동화

> Ralph Loop은 Claude Code에게 '조건이 충족될 때까지 반복 실행'을 시키는 자동 재시도 도구다. 에러 해결·린트 통과 등 객관적 완료 조건이 있는 작업에 최적화되어 있다.

> ★ HIGHLIGHT
>
> Ralph Loop은 AI에게 '해결될 때까지 반복해'라고 시키는 자동 재시도 버튼이다. 처음 개발에 쓰는 것보다, 에러가 안 풀릴 때 투입하는 '소방차'다.

핵심 포인트

- 이름의 유래: 심슨 캐릭터 '랄프 위검(Ralph Wiggum)' — 실패해도 될 때까지 밀어붙이는 캐릭터
- Anthropic이 공식 채택한 오픈소스 플러그인 (Claude Code에 설치하여 사용)
- 기존 문제: 명령 → 결과 확인 → 수정 요청 → 또 확인... 이 반복을 자동화
- 작동 원리: Claude Code가 결과를 반환하면 Ralph가 자동으로 '다시 해' 명령을 주입 → 조건 충족까지 반복
- 메인 세션에서 직접 반복 실행 (서브 에이전트가 아님)

Ralph Loop은 Claude Code의 '순수한 반복 실행'이다. 스킬, 커맨드, MCP 등 고급 도구 없이 기본 Claude Code만 계속 돌리는 방식이므로, 디자인 등 세밀한 작업보다는 객관적 기준이 있는 문제 해결에 강하다. 프롬프트의 completion-promise에는 반드시 수치적·객관적 조건을 넣어야 효과적이다.

개념

Agent Teams — 에이전트 팀 병렬 개발

Agent Teams는 여러 에이전트를 팀으로 구성하여 병렬로 개발하는 방식이다. 가장 빠르지만 가장 비용이 높으므로, 대규모 기능 구현에 전략적으로 사용한다.

★ HIGHLIGHT

에이전트 팀은 '식당 직원 4명이 동시에 일하는 것'이다. 혼자 하면 1명 비용이지만, 4명이 동시에 일하면 4명 인건비가 든다. 대신 4배 빠르다.

핵심 포인트

- 에이전트 팀 = Leader(총괄) + Agent A + Agent B + Agent C 구조
- Leader가 계획을 세우고, 각 에이전트에게 작업을 분배 → 병렬 실행 → 결과 합치기
- 워크트리를 활용하여 각 에이전트가 독립 공간에서 작업 후 합치는 방식 권장
- 모델도 개별 지정 가능: '모든 에이전트는 오퍼스 모델로 진행하라'
- Plan Mode와 함께 사용하면 가장 정확하지만 가장 비용이 높음 (7배)

에이전트 팀은 Claude Code의 가장 강력한 병렬 개발 기능이다. Leader가 전체 계획을 세우고, 각 에이전트가 독립된 워크트리에서 동시에 작업한 후 결과를 합친다. 레스토랑에서 주방장·서빙·주문 담당이 동시에 일하는 것과 같다. 비용은 4~7배이지만, 시간은 1/4로 줄어든다. Plan Mode와 함께 사용하면 '계획의 정확성 + 실행의 병렬성'을 모두 얻을 수 있어 가장 품질이 높다.

▶ TIP

effort(이폴트)를 high 이상으로 설정해야 에이전트 팀이 제대로 작동한다

▶ TIP

강사는 동시에 8개 세션까지 병렬 실행 — 초기 구축 시 할 일이 많을수록 효과적

▶ TIP

Plan Mode 없이 바로 팀 에이전트를 쓸 수도 있지만, Plan Mode 조합이 가장 정확

▶ **TIP**

토큰 비용이 걱정되면 오퍼스 모델 지정을 빼고 기본 모델로 진행

가이드

Worktrees(워크트리) — 병렬 기능 개발

★ HIGHLIGHT

워크트리는 '조립 작업장'이다. 본체(메인)는 놔두고, 옆 테이블에서 팔을 만들고, 다른 테이블에서 다리를 만든다. 완성되면 본체에 붙인다. 몸통 + 팔 + 다리 + 머리가 모여야 사람이 되듯, 메인 + 각 기능 브랜치가 합쳐져야 완성된 소프트웨어가 된다.

STEP 1 워크트리란?

STEP 2 워크트리 생성

```
# Claude Code가 실행하는 명령어 (참고용)
git worktree add ../프로젝트명-기능명 -b f/기능명

# 생성 후 폴더 구조
프로젝트/          ← 메인 (본체)
프로젝트-기능A/    ← 워크트리 1 (팔)
프로젝트-기능B/    ← 워크트리 2 (다리)
```


STEP 3 워크트리에서 작업하기

```
# 워크트리 폴더로 이동
cd ../런치보트-슈퍼베이스-셋업

# Claude Code 실행
claude

# 브랜치 확인 (f/supabase-setup으로 표시됨)
git branch
```

STEP 4 병렬 작업 — 터미널 분할

```
# 터미널 1: 비주얼 개선
cd ../런치보트-비주얼
claude

# 터미널 2: 위치 추천 기능
cd ../런치보트-로케이션
claude
```

STEP 5 워크트리 머지 흐름

```
# 워크트리에서 작업 완료 후
git add .
git commit -m "feat: 슈퍼베이스 연동"
git push origin f/supabase-setup

# GitHub에서 PR 생성 → merge
# 또는 Claude Code에게 "PR 만들어줘" 지시
```

STEP 6 워크트리 정리

워크트리 전체 흐름 — 사람 조립 비유

- 메인(몸통) → 워크트리A(팔) 생성 → 팔 개발 → 몸통에 붙이기(merge) → 워크트리A 정리
- 메인(몸통) → 워크트리B(다리) 생성 → 다리 개발 → 몸통에 붙이기(merge) → 워크트리B 정리
- 메인(몸통) → 워크트리C(머리) 생성 → 머리 개발 → 몸통에 붙이기(merge) → 워크트리C 정리
- 결과: 완성된 사람(소프트웨어)

개념

Plan Mode & Model Effort — 계획하고 실행하기

Plan Mode는 코드를 바로 쓰지 않고 먼저 설계하는 모드이고, Model Effort는 Claude의 사고 깊이를 조절하는 설정이다. 여기에 Sync(싱크)를 결합하면 '계획 → 실행 → 정리'의 완전한 사이클이 만들어진다.

★ HIGHLIGHT

Plan Mode는 '설계도를 먼저 그리는 것'이고, Model Effort는 '장인의 집중도를 조절하는 다이얼'이며, Sync는 '작업일지를 정리하는 것'이다. 설계 없이 시작하면 방향을 잃고, 집중도가 낮으면 실수가 늘고, 정리 없이 계속하면 혼란스러워진다.

핵심 포인트

- Plan Mode: /plan 명령어로 구현 전에 계획을 먼저 세운다. 코드 수정 없이 탐색과 설계만 진행
- 계획의 계획: Plan Mode 안에서 다시 Plan Mode를 켜면 Agent Teams 설계까지 가능 (메타 플래닝)
- Plan Mode + Agent Teams = 가장 정확한 결과를 내지만, 비용이 약 7배 증가
- Model Effort: /model 명령어로 low/medium/high/max 중 선택. effort가 높을수록 추론과 사고를 더 많이 함
- 강사 권장: high 이상으로 설정. max effort는 복잡한 작업에서 정확도가 크게 올라감
- Sync는 기능 추가 후 반드시 실행. git pull + CLAUDE.md/rules 문서 동기화를 한 번에 처리
- CLAUDE.md가 길어지면(예: 140줄 → 80줄 초과) 자동으로 rules 폴더로 분리
- 완전한 사이클: 기능 추가 → 머지 → /sync → 다음 기능 개발 (반복)

프로젝트가 커질수록 '계획 → 실행 → 정리' 사이클의 중요성이 커진다. Plan Mode로 방향을 잡고, Model Effort를 높여 정확한 코드를 만들고, Sync로 문서를 정리하면 프로젝트가 일관성 있게 성장한다. 이 세 가지를 루틴으로 만들면 비개발자도 대규모 프로젝트를 안정적으로 운영할 수 있다.

가이드

Skills 설치 가이드 — Claude Code 능력 확장

> ★ HIGHLIGHT
> 스킬은 '직원에게 자격증을 따게 하는 것'이다. 기본 Claude Code도 잘하지만, Supabase 스킬을 설치하면 데이터베이스를 훨씬 더 잘 다루게 된다.

STEP 1 Skills이란?

STEP 2 Supabase Agent Skill 설치

```
# Claude Code 터미널에서 (느낌표로 직접 입력 가능)
!mpx skill add supabase-agent-skill

# 또는 일반 터미널에서
mpx skill add supabase-agent-skill
```

STEP 3 Vercel Agent Skill 설치


```
mpx skill add
# → Vercel Labs 검색
# → 스페이스바로 원하는 스킬 다중 선택
#   (deploy-vercel, composition-pattern 등)
# → Enter로 확인
```

STEP 4 UI/UX Pro Max Skill 설치

```
# Claude Code에게 지시하는 방법
"UI UX 프로 스킬과 프론트 디자인 스킬로 더 고급스럽게 디자인하라"
```

STEP 5 스킬 설치 공통 패턴 정리

```
# 공통 설치 명령어
mpx skill add [스킬명]

# 설치 시 선택 순서
# 1. 플랫폼: Claude Code 선택
# 2. 범위: Global (전체 프로젝트) 또는 Project (이 프로젝트만)
# 3. 방식: Symlink 선택
# 4. 확인: Yes
```

가이드

외부 API 연동 가이드 — Kakao Map & Context7 & GWS

> ★ HIGHLIGHT
>
> 외부 API는 '다른 가게와 제휴하는 것'이다. 카카오맵 API는 지도 서비스와 제휴, Context7은 도서관과 제휴하여 최신 자료를 빌려올 수 있게 하는 것.

STEP 1 Kakao Map REST API — 1단계: 개발자 등록

STEP 2 Kakao Map REST API — 2단계: 앱 생성

```
# 앱 정보 예시
앱 이름: lunch-bot (프로젝트명)
회사명: 내 회사명 또는 개인 이름
카테고리: 여행/지역 정보
```

STEP 3 Kakao Map REST API — 3단계: REST API 키 복사

```
# .env.local 파일에 추가
KAKAO_REST_API_KEY=여기에_복사한_키_붙여넣기
```


STEP 4 Kakao Map REST API — 4단계: 카카오맵 사용 설정 ON

STEP 5 Context7 API 키 설정

```
# Claude Code에게 지시
"context7 API 키 MCP에 키 넣어줘"
# → Claude Code가 자동으로 MCP 설정에 키를 등록해 줌
```

STEP 6 GWS CLI (Google Workspace CLI) 소개

Comparison Note

MCP = 축구장에서 럭비공으로 축구 (무겁고 헤비)

CLI = 축구장에서 축구공 (가볍고 빠름, AI의 홈그라운드)

가이드

배포 파이프라인 심화 — PC → GitHub → Vercel 자동화

> ★ HIGHLIGHT
> 배포는 '요리를 식당에 올리는 것'이다. 내 컴퓨터(주방)에서 만든 요리를 GitHub(레시피 보관함)에 올리고, Vercel(식당)이 자동으로 손님에게 서빙한다.

STEP 1 배포의 개념 이해

STEP 2 3단계 파이프라인 이해: PC → GitHub → Vercel

STEP 3 GitHub에 코드 올리기: 커밋 → 푸시 → PR → 머지

```
/commit-push-pr --merge
```

STEP 4 Vercel 자동 배포 설정

STEP 5 Vercel 환경변수 설정

STEP 6 배포 상태 확인

STEP 7 코드 리뷰 & 핸드오프 베리파이

```
# 1. 코드 리뷰
/code-review

# 2. 핸드오프 베리파이 (별도 에이전트가 검증)
/handoff-verify

# 3. 검증 통과 후 커밋+푸시+머지
/commit-push-pr --merge
```

모범 사례

개발 후 검증과 기록 — 테스팅, 로그, AI 메모리

배포된 앱의 품질을 확인하는 디바이스 테스팅, 에러 디버깅, 그리고 AI가 더 잘 기억하도록 경험을 기록하는 로그 시스템.

★ HIGHLIGHT

로그는 '업무일지'다. 오늘 뭘 했고, 어떤 결정을 내렸고, 뭘 실패했는지 적어두면 다음에 같은 실수를 피할 수 있다. AI도 마찬가지로, 기록을 남겨야 다음에 더 잘 도와준다.

디바이스 테스팅 (Chrome 개발자도구)

● INFO

매장 오픈 전에 다양한 손님 입장에서 동선을 점검하는 것과 같다. PC 화면에서는 멀쩡해도 핸드폰에서는 깨질 수 있다.

- 웹페이지에서 우클릭 → '검사' 클릭 → 개발자도구 열기
- 상단의 디바이스 토글 버튼(핸드폰+태블릿 아이콘) 클릭
- 다양한 기종(iPhone, Galaxy 등) 선택하여 모바일 화면 미리보기
- 모바일에서 문제 발견 시 크롬 개발자도구에서 바로 수정 진행
- 강력 새로고침: Cmd+Shift+R (Mac) / Ctrl+Shift+R (Windows) — 캐시 무시하고 최신 버전 로드

브라우저 콘솔 에러 확인

● INFO

자동차 계기판의 경고등과 같다. 빨간 불이 들어오면 무시하지 말고 원인을 찾아야 한다.

- 우클릭 → '검사' → Console 탭 클릭
- 빨간색 에러 메시지가 있으면 문제가 있다는 신호
- 에러 메시지를 스크린샷 찍거나 텍스트로 복사
- 복사한 에러를 Claude Code에게 그대로 전달: '이런 에러가 났어: [붙여넣기]'
- 에러가 환경변수 관련이면 Vercel Settings에서 환경변수 확인

로그 기능 — AI 메모리 강화

> **INFO**
> 게임의 세이브 포인트와 같다. 경험치, 의사결정, 실패를 기록해두면 AI가 다음 플레이에서 더 똑똑해진다.

- Experience(경험) 로그: 성공한 것, 배운 것 기록 → 'log experience [내용]'
- Decision(의사결정) 로그: 주요 선택과 이유 기록 → 'log decision [내용]'
- Failure(실패) 로그: 실패한 것, 교훈 기록 → 'log failure [내용]'
- 로그 기록 후 반드시 sync 실행 — AI가 메모리에 반영하도록
- 기능 완료 시마다: 로그 기록 → sync → 다음 기능 진행

검증 습관 체크리스트

> **INFO**
> 비행기 조종사의 이륙 전 체크리스트와 같다. 매번 같은 순서로 확인해야 사고를 방지한다.

- ① 기능 구현 완료
- ② 디바이스 테스팅 (PC + 모바일 확인)
- ③ 콘솔 에러 확인 (빨간 에러 없는지)
- ④ 코드 리뷰: /code-review
- ⑤ 핸드오프 베리파이: /handoff-verify
- ⑥ 커밋+푸시+머지: /commit-push-pr --merge
- ⑦ 로그 기록 + sync
- 이 순서를 기능마다 반복하면 품질이 유지된다

실습 문제

문제 1: 나만의 CLAUDE.md 작성하기

EASY 10분

오늘 수업에서 배운 형식으로 7-8주차 프로젝트의 CLAUDE.md 초안을 작성하세요.

요구사항

- 프로젝트 개요 1-2문장
- 기술 스택 명시
- DB 테이블 2개 이상 설계
- 화면 목록 2개 이상
- Rules 3개 이상

▶ HINT

- 자신이 실제로 쓰고 싶은 앱을 생각해보세요
- 처음엔 대략적으로 써도 OK — 나중에 수정 가능
- DB 스키마는 테이블당 3-5개 컬럼으로 시작

문제 2: Supabase RLS 정책 작성하기

MEDIUM 5분

점심 투표 앱의 suggestions 테이블에 적절한 RLS 정책을 작성하세요.

요구사항

- 모든 사용자가 suggestions를 읽을 수 있어야 함
- 모든 사용자가 suggestions를 추가할 수 있어야 함
- 삭제는 금지 (삭제 정책 없음)

시작 코드

```
-- SELECT 정책 (읽기)
CREATE POLICY "[정책 이름]"
ON public.suggestions FOR SELECT
USING (???);

-- INSERT 정책 (쓰기)
CREATE POLICY "[정책 이름]"
ON public.suggestions FOR INSERT
WITH CHECK (?????);
```

> ▶ **HINT**
>
> - USING (true)는 '모든 행에 적용'을 의미
> - FOR SELECT는 읽기, FOR INSERT는 쓰기 권한
> - 삭제 정책을 만들지 않으면 자동으로 차단됨

정답 (스스로 풀어본 후 확인하세요)


```
-- SELECT 정책 (읽기)
CREATE POLICY "Anyone can read suggestions"
ON public.suggestions FOR SELECT
USING (true);

-- INSERT 정책 (쓰기)
CREATE POLICY "Anyone can add suggestions"
ON public.suggestions FOR INSERT
WITH CHECK (true);
```

문제 3: 프로젝트 기획서 피어 리뷰

MEDIUM 15분

옆 사람의 CLAUDE.md를 읽고 피드백을 줘보세요. '좋은 기획'과 '나쁜 기획' 기준으로 평가합니다.

요구사항

- 파트너의 CLAUDE.md를 5분간 읽기
- 아래 체크리스트로 평가
- 3가지 개선점 제안

▶ **HINT**

- 좋은 피드백: '이 부분이 모호해서 Claude가 혼란스러울 것 같아요'
- 나쁜 피드백: '잘못됐어요' (이유 없이)
- 제안은 구체적으로: '~ 이렇게 바꾸면 어떨까요?'

6주차 학습 체크리스트

참고 자료

공식 문서

Next.js 15 공식 문서

https://nextjs.org/docs

App Router, Server Components, API Routes 등 공식 가이드

Supabase 공식 문서

https://supabase.com/docs

Database, Auth, Realtime, Storage 전체 가이드

Supabase JavaScript SDK

https://supabase.com/docs/reference/javascript/introduction

supabase-js API 레퍼런스 (select, insert, update, delete)

Vercel 배포 가이드

https://vercel.com/docs/deployments/overview

GitHub 연동, 환경변수, 도메인 설정 가이드

튜토리얼

Tailwind CSS 빠른 참조

https://tailwindcss.com/docs/utility-first

자주 쓰는 클래스: flex, grid, p, m, text, bg, rounded, shadow

GitHub

오늘 뭐 먹지? 라이브 코딩 코드

https://github.com/singularity-builders/week6-demo

오늘 라이브 코딩 전체 소스코드 (강사 GitHub)

template

CLAUDE.md 템플릿

https://github.com/singularity-builders/claude-md-template

수업에서 사용한 CLAUDE.md + plan.md 템플릿

영상

Supabase 30분 입문

https://youtube.com/watch?v=dU7GwCOgvNY

Supabase 공식 채널 시작하기 영상 (영어)

용어 사전

용어	정의
CLAUDE.md	Claude Code에게 프로젝트를 설명하는 파일. 기술 스택, DB 스키마, 규칙 등을 담는다.
plan.md	구현 계획서. Phase별로 할 일을 나누고 진행 상태(PENDING/IN_PROGRESS/DONE)를 관리한다.
Next.js	React 기반 풀스택 웹 프레임워크. 서버사이드 렌더링, API Routes, 자동 최적화를 지원한다.
Supabase	오픈소스 Firebase 대안. PostgreSQL 데이터베이스 + 자동 생성 API + 인증 + Realtime을 제공한다.
RLS (Row Level Security)	Supabase의 행 단위 보안 정책. 어떤 사람이 어떤 데이터를 읽고 쓸 수 있는지 제어한다.
Vercel	Next.js를 만든 회사의 클라우드 배포 플랫폼. GitHub 연동 시 push마다 자동으로 배포된다.
CI/CD	Continuous Integration/Continuous Deployment. 코드 변경 시 자동으로 테스트하고 배포하는 파이프라인.
환경변수	코드에 직접 넣으면 안 되는 비밀 값(API 키, 비밀번호). .env.local 파일에 저장하고 코드에서 process.env.변수명으로 읽는다.
API Route	Next.js에서 서버 사이드 로직을 처리하는 엔드포인트. /app/api/폴더에 route.ts 파일로 만든다.
Tailwind CSS	유틸리티 클래스 기반 CSS 프레임워크. className='flex p-4 bg-blue-500 rounded' 형태로 스타일을 적용한다.
피드백 루프	만들기 → 확인 → 피드백 → 수정의 반복 사이클. Claude Code와의 협업에서 핵심 개발 방식이다.
MVP (Minimum Viable Product)	최소 기능 제품. 핵심 기능만 구현하고 나머지는 나중에 추가하는 전략. 완성도 > 기능 수.
Ralph Loop	Claude Code에 '조건 충족까지 반복 실행'을 시키는 자동 재시도 도구. completion-promise, max-iterations, cancel-ralph 3요소로 구성된다.
Agent Teams	여러 에이전트를 팀으로 구성하여 병렬 개발하는 방식. Leader + Agent A/B/C 구조. 비용 4~7배이지만 속도 4배.
Worktree (워크트리)	하나의 Git 저장소에서 여러 브랜치를 별도 폴더로 체크아웃하여 동시 작업하는 기능.
Plan Mode	/plan 명령으로 진입. 코드 수정 없이 프로젝트를 탐색하고 구현 계획을 먼저 세우는 모드.

용어	정의
Model Effort	/model 명령으로 low/medium/high/max 중 선택. AI의 사고 깊이(추론량)를 조절하는 설정.
Sync (/sync)	git pull + CLAUDE.md/rules 문서 동기화를 한 번에 실행하는 명령어. 기능 머지 후 반드시 실행.
GWS CLI	Google Workspace CLI. Gmail, Drive, Sheets 등을 터미널에서 조작하는 도구. MCP보다 가볍고 빠름.
Context7	에이전트가 최신 공식 문서를 참조하여 정확한 코드를 생성하게 해주는 MCP 서비스. API 키 등록 권장.
Skills (에이전트 스킬)	mpx skill add로 설치하는 Claude Code 확장 기능. Supabase, Vercel, UI/UX 등 전문 능력을 추가한다.
Handoff Verify	/handoff-verify 명령으로 별도 에이전트가 독립 컨텍스트에서 코드를 검증하는 품질 확인 도구.