

Claude Max \$2000の燃やし方

Claude Codeで作る「自分専用開発チーム」

発表者: @noricha-vr

所要時間: 10 min | 対象: 個人開発者

前提：なぜ \$200 払うのか

Profile



@noricha-vr

Role: 個人開発者
(VR/Web)

// 誤字脱字が多く、
言葉足らず

Motivation

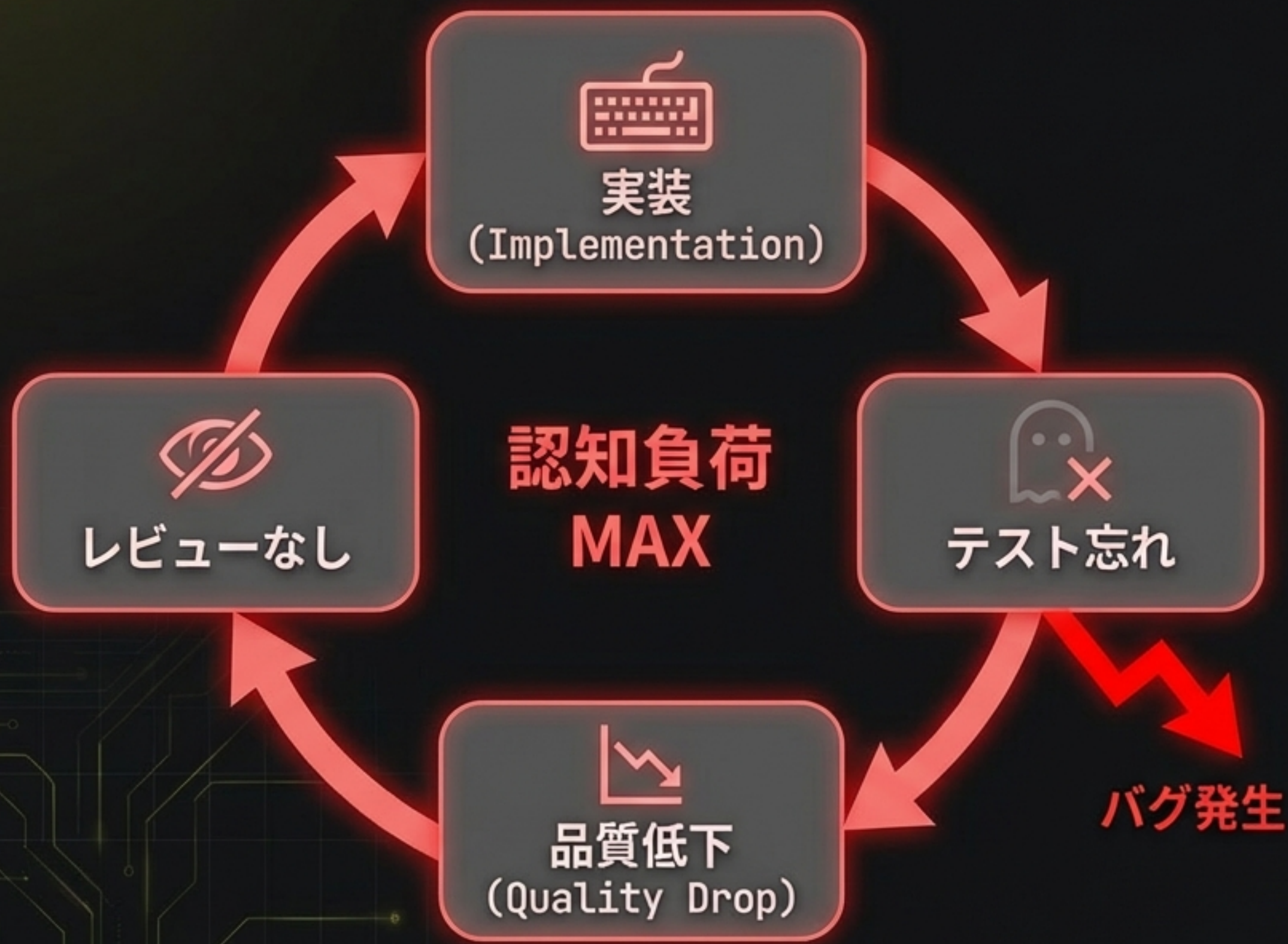


Reason: 無制限の
Opus / Sonnet 3.7
JetBrains Mono

Goal: トークン残量
を気にせず、開発に
集中する

\$ > AIを単なるチャット相手ではなく、チームとして雇う

課題：1人開発の限界 (The Solo Loop of Doom)



- 同じ指示の繰り返し
= 非効率 (DRY違反)
- コンテキスト喪失
- 解決策:
AIに「役割」を
与えて任せる

Inter : チームを作る3つの仕組み (Architecture)

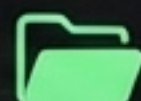


AGENTS

Inter
担当者・役割

AGENTS

Inter
担当者・役割

 `.claude/agents/`

JetBrains Mono
implementer, reviewer

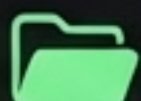


COMMANDS

Inter
ショートカット

COMMANDS

Inter
ショートカット

 `.claude/commands/`

JetBrains Mono
/orc, /commit

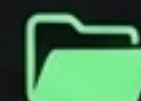


SKILLS

Inter
知識・手順書

SKILLS

Inter
知識・手順書

 `.claude/skills/`

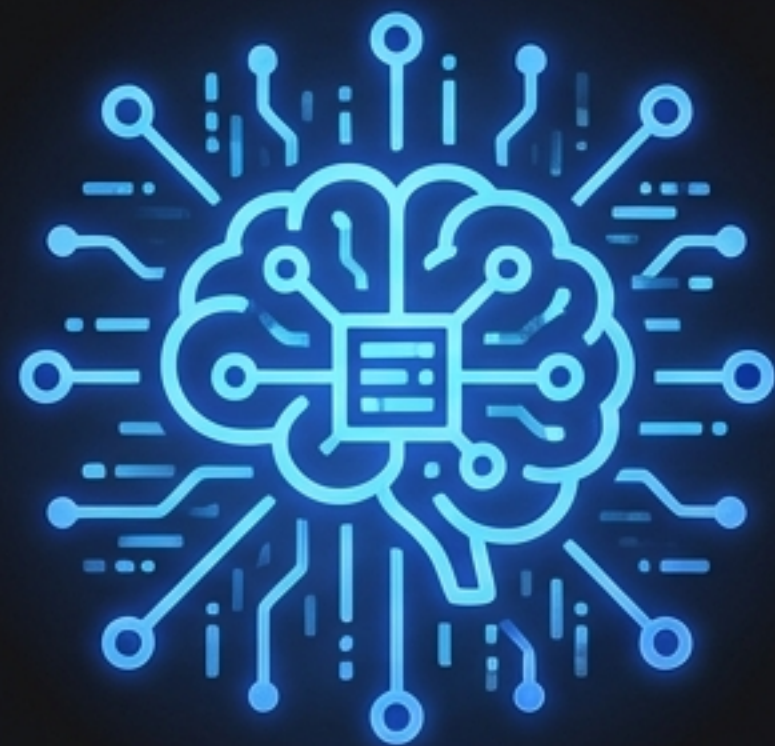
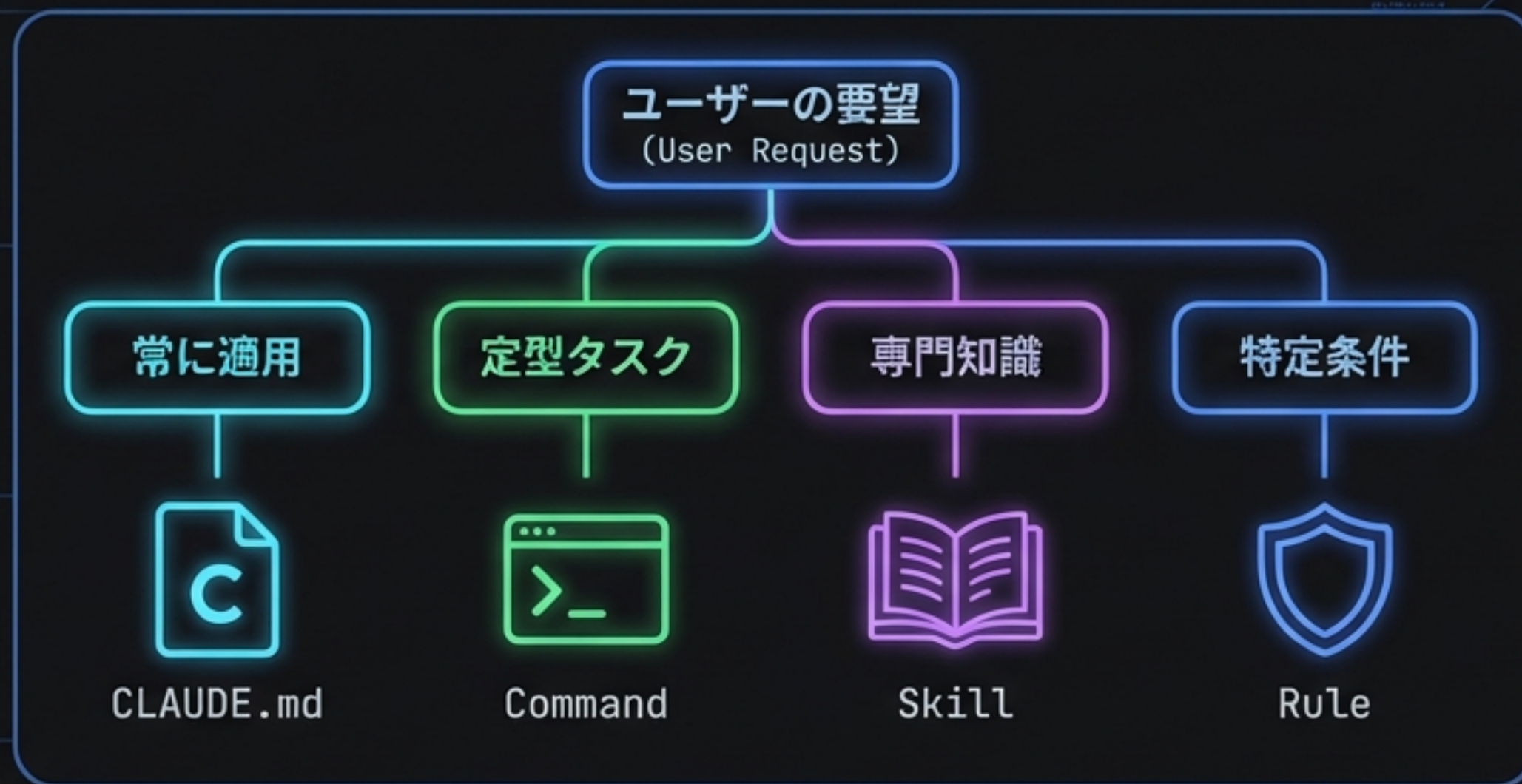
JetBrains Mono
commit-check

私の「開発チーム」名簿 (The Agents)



管理職エージェント: Config

仕組みを作るための仕組み



「ユーザーの要望を診断し、適切な設定に振り分けます」

委譲チェーン (The Relay Race)



Implementer

実装完了
&
テスト追加

**Auto
Handoff**



委譲しないと
完了にならない
(Absolute Rule)



QA Engineer

ブラウザ動作確認

**Auto
Handoff**



委譲しないと
完了にならない
(Absolute Rule)



Code Reviewer

承認
&
マージ

自動引き継ぎにより、人間が介在せずに品質プロセスが回る。

実装担当: Implementer

テストを書かないコードは、コードではない

完了定義:

- [] 受け入れ条件を全て満たしている
- [] 新規コードにテストを追加
- [] 追加したテストが全てパス
- [] `code-reviewer` に委譲した (必須)

Must-have
constraint

Forces the
Relay Chain

Basic Philosophy: YAGNI / DRY / KISS

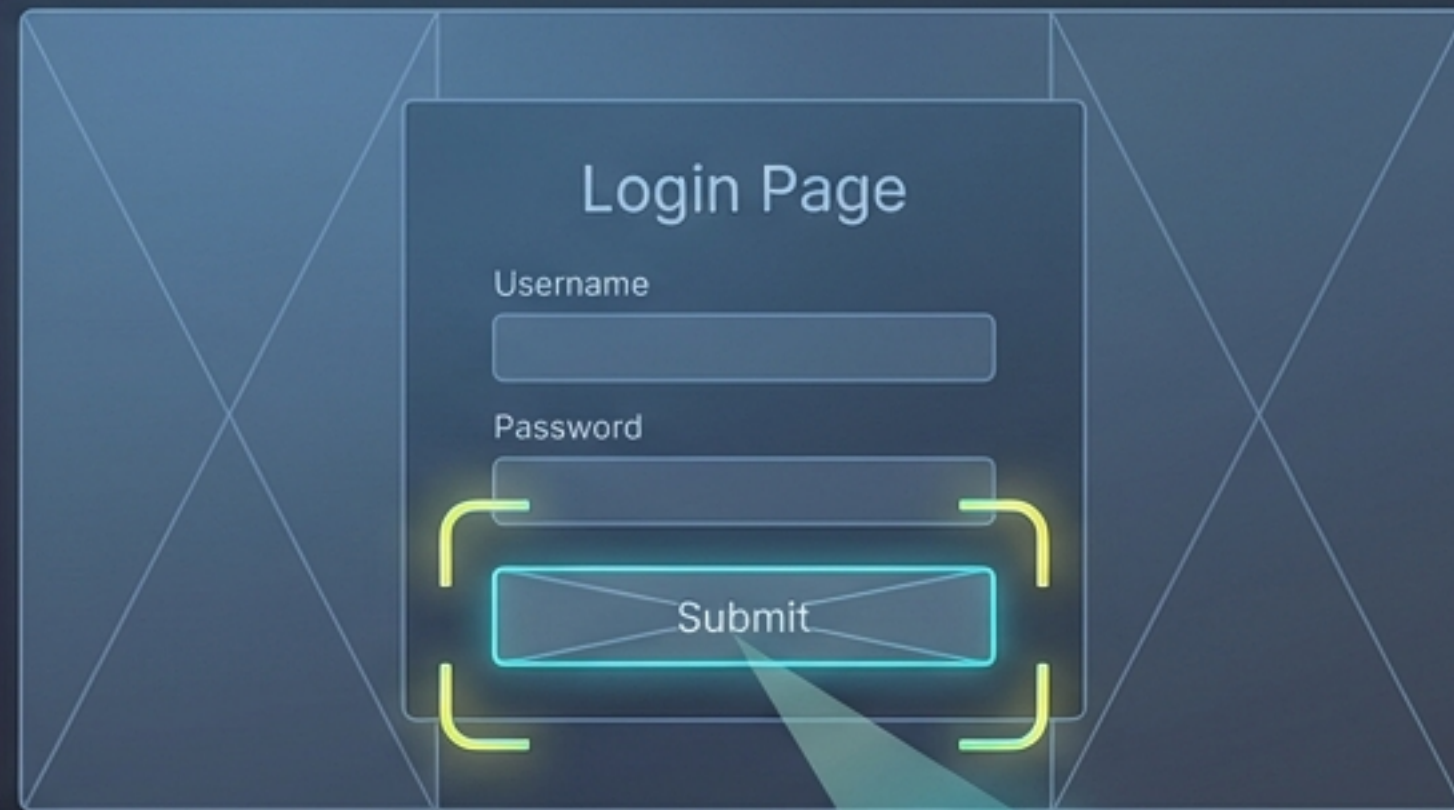
品質保証: QA Engineer

Claude in Chrome MCP で「見て」確認

> Claude Terminal

```
> mcp__claude-in-chrome__find(query: 'ログインボタン')  
>   
> mcp__claude-in-chrome__click(element: 'btn_login')  
> taking_screenshot...
```

Chrome - QA Test Session



UI変更やバグ修正後の動作を、人間のように検証する。

門番: Code Reviewer

甘えを許さないブロッキング基準



Blocking Criteria

- ✗ テストカバレッジ不足（新機能にテストがない）
- ✗ セキュリティリスク（SQLi/XSS）
- ✗ 重大なバグ（ロジックエラー）

➡ Uses: `~/.bun/bin/codex review --uncommitted`

修正が完了するまで「承認不可」

コマンド: チームを動かすスイッチ

> Type a command...

/orc - Orchestrator: 主要エントリーポイント。エージェントを統括。

/commit - Commit Check: コミット作成前に品質チェックを実行。

/quick - Emergency: 委譲なしで直接実装（緊急対応用）。

/pr-ready - Status Check: PR準備状況を確認。

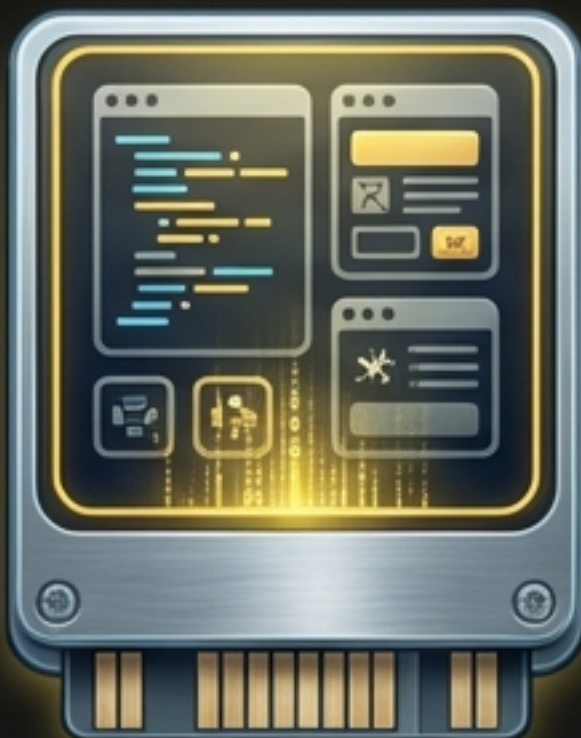


スキル: 共有される手順書

📁 `.claude/skills/{name}/SKILL.md`

frontend-design

本番品質のUI作成ルール



commit-check

コミット前品質チェック



codex-cli

Codex CLI連携・レビュー
基準



plugin-dev

Claude Code設定ファ
イルの書き方



導入ステップ: 小さく始める

1 ディレクトリ作成

```
mkdir -p .claude/{agents,commands,skills}
```

2 最初の1人 (Implementer)

`implementer.md`を作成。
「テストを書く」ルールだけ入れる。

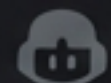
3 徐々に拡張

委譲ルールを追加 → QA担当を追加
→ コマンドを追加

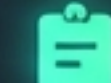
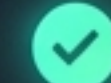
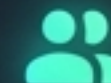
Iterative Process

\$200 の価値: チームを持つということ

Before (Copilot)

-  指示待ち (Passive)
-  文脈をすぐ忘れる
-  テストは人間が書く
-  Role: Junior Coder

After (Claude Team)

-  自律的に委譲 (Active)
-  役割とルールを遵守
-  テスト・レビューを強制
-  Role: CTO & Senior Team

自分が「コーダー」から「CTO」になる。



トークンを燃やせ。
チームを動かせ。

Let's Burn \$200 🔥

github.com/noricha-vr