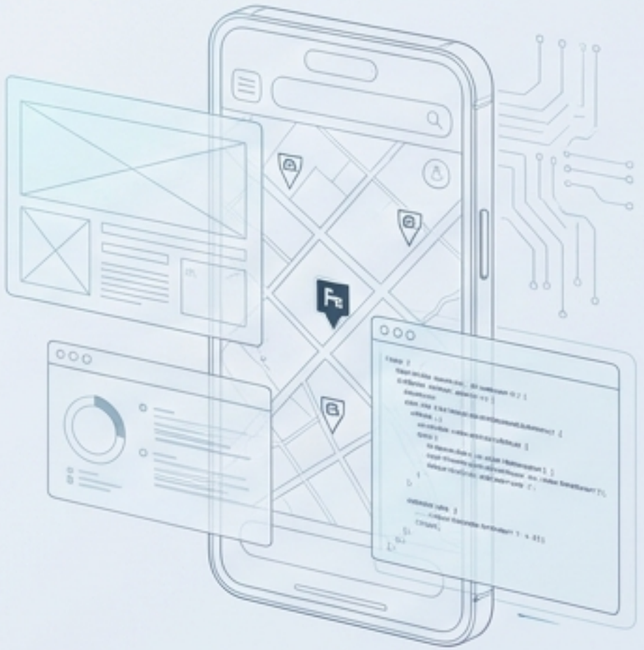
[STATUS: DEPLOYED]

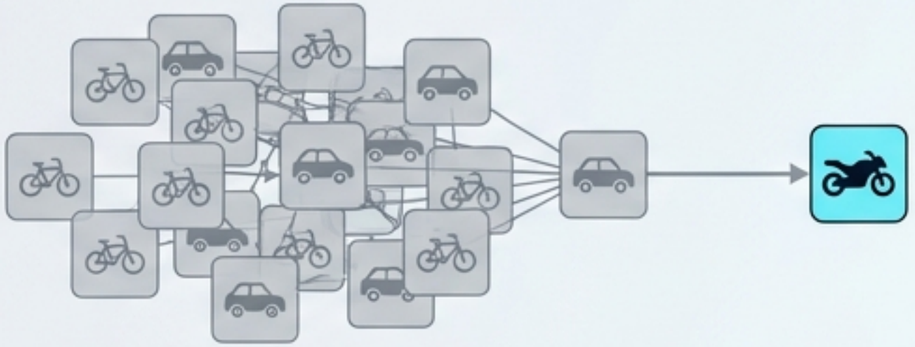
AI-Driven PWA Development // 東京大型入・バイク駐車場マップ開発録

Claude Codeによる要件定義ドリブン開発と非同期ワークフローの構築



本ドキュメントは、Claude Codeを活用したモダンなアプリケーション開発のユースケースを記述する。東京における大型バイク駐車場検索のペインポイントを解決するPWAの開発を通じ、人間からAIへの「コーディングの委譲」と、要件定義・非同期ワークフローへパラダイムシフトを視覚的に解剖する。

検索のノイズとUXの分断



既存のソリューションにおける決定的な欠陥：

1. **Google Mapsのノイズ:** 「バイク 駐車場」で検索しても自転車の駐輪場が混在し、大型バイク（自動二輪）専用の選別が不可能。

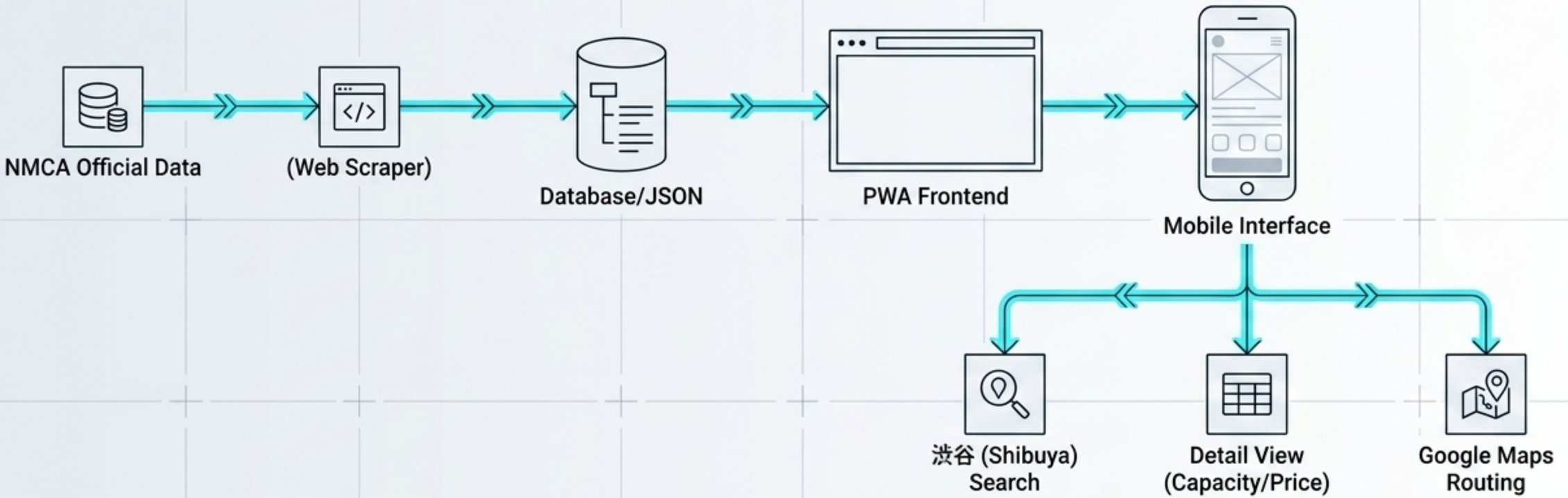
2. **公式データのUX限界:** 二輪車普及安全協会（NMCA）が公開する公式マップは存在するが、そこからGoogleマップへのナビゲーション動線が存在せず、モバイル環境での利用において極めて高いストレスを発生させていた。

[TARGET_PLATFORM]:
Mobile

[REQUIRED_SPEC]:
大型バイク専用

[CRITICAL_PATH]:
シームレスなGoogle Maps遷移

[SOLUTION_TRIGGER]:
自作によるUXの完全掌握



クローリングからPWAへの統合アーキテクチャ

散在する公式データ（NMCA）をクローリングによって構造化データとして抽出し、独自のデータベースを構築。フロントエンドはモバイル最適化されたPWAとして出力し、地名検索（例: 渋谷）、詳細確認（料金・台数）、そしてGoogleマップへの即時ナビゲーションという一連のユーザー体験を単一のインターフェースに統合した。

Native iOS / Android

[FRICTION]
Apple/Googleの厳格な審査プロセス

[DEPLOYMENT]
ストア依存によるタイムラグ

[COMPATIBILITY_WITH_AI]
✖ 低。即時生成・即時テストのサイクルが審査によって分断される。

PWA (Progressive Web App)

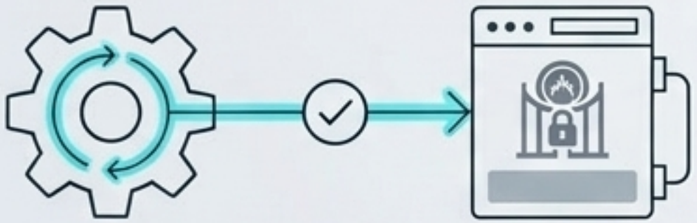
[FRICTION]
審査なし。Webサイトとして即時公開

[DEPLOYMENT]
URL共有のみ。ブラウザから「ホーム画面に追加」でネイティブ同等のUX

[COMPATIBILITY_WITH_AI]
🔵 最適。Claude Codeが出力したものを即座にデプロイし、手元のスマホで検証可能。

要件定義（RFP）ドリブン開発

AIにいきなりコードを書かせるアプローチは破綻しやすい。重要なのは、人間がまず「何を、どういう技術を使って、どう作るか」という仕様書　（RFP = 要件定義書）をAIとすり合わせること。コード生成の前に仕様を構造化することで、大規模な手戻りを防ぐ。




```
> INITIATING RFP PROTOCOL...

[USER]: 以下の要件で大型バイク専用駐車場の
PWAを構築するRFPを作成せよ。
{
  "Tech_Stack": "React, Tailwind, PWA",
  "Core_Feature": "Map integration,
Route to Google Maps",
  "Data_Source": "Crawled NMCA JSON"
}

[CLAUDE CODE]: 了解しました。要件定義書
（仕様書）を生成します...
```

VOICE-TO-TEXT PROTOCOL ACTIVATED

入力のボトルネックを破壊する「音声入力ハック」

> TARGET: Aqua Voice (Voice-to-Text)
> STATUS: カフェで思考をそのまま喋り続ける
> SYSTEM_RESPONSE: 完璧に構造化されたRFPの自動生成

キーボードでのタイピングは、人間の思考スピードに対して遅すぎる。 AIのポテンシャルを最大化するためには、音声入力（Aqua Voice等）を起動し、欲しい機能や仕様を「喋り倒す」のが最適解。AI VOCODE-TEXT Primer
等、欲しい機能や仕様を「喋り倒す」のが最適解。カフェで不審に思われても、得られる圧倒的な開発スピードがそれを凌駕する。

夜間バッチ型AIペアプログラミング

[22:00] Human Input

要件定義（RFP）の最終調整。最も重く、時間のかかる実装タスクをAI（Claude Code）に指示・委譲する。

[00:00 - 06:00] AI Autonomous Run

人間が睡眠・休息中（AIとの同期的なキャッチボールによる腰痛の回避）。AIが自律的にコード生成、ビルド、エラー解決を実行。

[07:00] Human Review

翌朝、完成したアウトプットを確認。指揮官として成果物をレビューし、デプロイメントを決定。

[REVIEW_STYLE]

Output-Based

AIが生成した膨大なMarkdown仕様書を人間がいちいち読む必要はない。実際に動く成果物（アウトプット）を触り、違Headers感（UXの引っかかり）がある部分のみを修正指示する。

[DEV_TIME]

数時間 (Hours)

従来のスクラッチ開発であれば数週間を要するシステム統合・フロントエンド構築が、AIへの適切な委譲によりわずか数時間に短縮される。

[HUMAN_EFFORT]

Minimal

人間の労力は「初期の仕様決定」と「最終の動作確認」に極振りされる。途中のタイピングやデバッグ作業はシステムに吸収される。

[SYSTEM_UPDATE: DEVELOPER ROLE REASSIGNED]

From Coder to Orchestrator

開発者の主たる任務は「コードを書くこと」から完全にシフトした。

AI時代のエンジニアの役割は、解決すべき課題を見極め、AIが動くための完璧な要件定義（RFP）を設計し、非同期で生成されたシステムを厳格にレビューする「オーケストレーター」である。

コーディングは機械に委ねよ。人間は「何を創るか」に集中する。