

# 開発レポート

- **プロジェクト名:** Machicle - 地域コミュニティイベント管理アプリ
- **作成者:** 綿引佳奈
- **提出日:** 2025年8月15日
- **ドキュメントバージョン:** 1.0

## 目次

1. 活用時の工夫点
2. ChatGPTの活用に関する考察
3. かかった工数
4. 課題を実行するにあたり工夫した点
5. ChatGTP対話履歴
6. 所感

## 1. 活用時の工夫点

- 無料制限があるので、空き時間でやり取りを進め、1日に複数回やり取りをできるように進めた。
- 自分が求めている答えをより、確実に取得できるよう、条件や、背景、目的を伝えるようにした。

## 2. ChatGPTの活用に関する考察

本課題では、開発の初期段階において、指定された通りChatgtpを思考のパートナーとして全面的に活用しました。その過程で得られた効果と、今後の課題について考察します。

### 2-1. 活用時に工夫した点

無料プランには利用回数の制限があるため、思考停止に陥ることなく、効率的に対話を進めるために、以下の2点を常に意識しました。

#### 1. 時間的制約への対応:

対話のタイムラグや利用制限を考慮し、PCに向かえない移動時間や空き時間を

活用してスマートフォンから対話を進めるなど、AIの応答時間を前提としたタイムマネジメントを行いました。これにより、1日あたりの対話の密度を高め、思考を停滞させない工夫をしました。

## 2. 質問の質の向上:

AIからよりの確で質の高い回答を得るため、単に「どうすればいいか?」と問うのではなく、「どのような背景・目的で、どんな制約条件があり、何を達成したいのか」を常に明確に伝えることを心がけました。これにより、AIとの対話の往復回数を最小限に抑え、本質的な議論に集中することができました。

## 2-2. 感じた効果と今後の課題

### 【感じた効果】

- **思考の壁打ちによる、意思決定の迅速化:**

特に、プロジェクト初期の要件定義フェーズにおいて、一人で悩み込んでしまう時間を大幅に削減できました。AIを「相談相手」とすることで、思考を言語化し、客観的な視点を得ながら、迅速に意思決定を行うことができました。

- **制約条件下での、最適な戦略立案:**

「React初心者」「超短期開発」という厳しい制約の中で、MVPスコープの再設定や技術選定におけるリスク回避といった戦略的な判断を行う上で、AIは最適な相談相手となりました。多様な選択肢を提示してくれるAIとの対話を通じて、短期間で成果を最大化するための、現実的かつ効果的な開発計画を立案することができました。

### 【今後の課題】

- **無料プランにおける、対話回数の限界:**

AIとの対話は非常に有効である一方、思考を深めるためにはどうしても対話の往復（キャッチボール）が必要となり、無料プランの利用回数制限が思考の深さを制約する一因となり得ると感じました。より複雑なプロジェクトにおいては、有料プランの活用も視野に入れる必要があると感じています。

## 3. かかった工数

作業内容	時間	ChatGTP活用度
基礎知識学習	7h	中
要件定義	3h	高
DB設計	3.5h	中

UI作成	4h	中
画面設計図作成	1h	低
会員登録画面実装	4.5h	高
ログイン画面実装	2h	高
ホーム画面実装	2h	高
UI調整	1h	中
ドキュメント作成	3h	低
合計	31h	

## 4. 課題を実行するにあたり工夫した点

### 4-1. 段階的スコープ管理と現実的なマイルストーンの設定

#### 【現状認識と課題】

本課題は、地域コミュニティの活性化という広範な目的のもと、多岐にわたる機能要件が含まれていました。しかし、与えられた開発期間は非常に限定的であり、当初の計画通りすべてのMVP機能を実装しようとする、一つ一つの品質が低下し、最悪の場合「動く成果物」を何も提出できないという未完成のリスクが非常に高いと判断しました。

#### 【工夫した点】

そこで、プロジェクト開始後に計画を再評価し、「完璧だが動かないもの」ではなく「不完全でも確実に動くもの」を期限内に届けることを最優先目標としました。具体的には、以下の戦略的な意思決定を行いました。

#### 1. MVP（Minimum Viable Product）の再定義:

当初のMVP要件の中から、アプリケーションの最も核となる「ユーザー登録 → ログイン → イベント一覧表示」というコアなユーザー体験に必須の**3画面にスコープを絞り込み**ました。

#### 2. 段階的開発計画への移行:

まずは再定義した3画面の実装を「フェーズ1」として確実に完了させ、残りの画面（イベント作成、チャット等）は「フェーズ2」として、時間に余裕が生まれた場合に着手する、という段階的な開発計画へと切り替えました。

#### 【得られた効果】

この計画見直しにより、以下の効果が得られました。

- **開発の焦点の明確化:**

限られた時間の中で、取り組むべきタスクが明確になり、Reactの学習と実装という、最も重要な活動に集中することができました。

- **リスクの低減:**

予期せぬエラーや環境構築での遅延が発生しても、プロジェクト全体が破綻するリスクを大幅に低減させることができました。

- **モチベーションの維持:**

達成不可能な目標ではなく、現実的なマイルストーンを設定したことで、一つ一つのタスクを着実に完了させる達成感を得られ、高いモチベーションを維持して開発に取り組むことができました。

---

## 4-2. マルチAI活用による、超高速プロトタイピング戦略

### 【現状認識と課題】

前述の通り、本課題はReact初心者というスキルセットと、お盆期間中という極めて限られた開発時間の中で、高品質なプロトタイプを完成させるという、非常に高い挑戦でした。この制約を乗り越えるため、「複数のAIを、それぞれの特性に合わせて戦略的に活用する」という、マルチAI開発アプローチを全面的に採用しました。

### 【工夫した点】

#### 1. AIツールの戦略的活用

本課題では「ChatGPTを活用すること」が要件として定められていましたが、「他のAIツールの利用を禁止する」との明記はなかったため、以下の通り、複数のAIをそれぞれの強みに合わせて活用し、開発効率を最大化しました。

(もし、この解釈が課題の意図と異なっておりましたら、何卒ご容赦ください。)

- **UI生成AI v0.devの活用（デザインとコーディングの同時並行化）:**

UI開発において最も時間がかかる「デザイン→コーディング」の工程を劇的に短縮するため、UI生成AIであるv0.devを全面的に活用しました。対話形式でUIを洗練させると同時に、高品質なReactコードを即座に入手することで、圧倒的な開発スピードを実現しました。

- **対話型AIの併用による、継続的な開発サポート:**

課題の要件通り、プロジェクトの初期段階ではChatGPTを思考のパートナーとして活用しました。しかし、無料プランの利用制限に達したため、開発を停滞させないことを最優先し、その後はGoogleのGemini 2.5 Proに切り替えて、AIによる学習支援と問題解決を継続しました。これにより、常に高度なテクニカルサポートを受けられる環境を維持し、開発の勢いを止めることなく課題に取り組むことができました。

## 2. AI生成コードに対する、主体的な問題解決

AIが生成するコードは、必ずしもそのまま動くわけではありませんでした。特にv0.devが生成するコードは、最新のNext.js/TypeScript環境を前提としていました。

これを、今回の課題要件であるVite/JavaScript環境で動作させるため、**AIの生成物を鵜呑みにせず**、自らコードを分析。TypeScriptの型定義を除去し、Next.jsの独自コンポーネントを標準のHTMLタグやreact-router-domの部品に手動で置き換える作業を行いました。

また、Tailwind CSSの環境構築で発生した依存関係の問題など、解決が難しい実践的な問題に対し、AIと壁打ちをしながら、原因の特定から解決までを自力で完遂しました。

### 【効果】

このマルチAI活用アプローチにより、React初心者でありながら、**わずか数日間**で、環境構築から3画面のUI実装、そしてGit/GitHubでの納品までを完了させることができました。これは、AIを単なる「コード生成機」としてではなく、「思考と開発を加速させるための、複数の戦略的パートナー」として主体的に活用したからこそ達成できた、本課題における最大の工夫点です。

## 5. ChatGPT対話履歴

### • 8/9

主に、アイデア整理、どんなアプリケーションを作成するか？の整理を会話ベースで行いました。

あなた:

以下の課題を解決するための地域コミュニティイベント管理アプリを開発します。まずは、要件定義を行いたいのですが、その前に、どんなものを開発すべきか、アイデアを出してください。

課題

- 地域住民のつながりが希薄化
- 祭りや行事の参加者減少、担い手不足により維持困難
- 住民がイベント情報を気軽に入手できない
- 多様な世代で交流できる場が欲しい
- 自分のスキルや経験を地域で活かしたい

## ChatGPT:

いいですね！

課題から逆算すると、「情報が届かない」「交流の場が少ない」「担い手が不足」「スキル活用の機会がない」という4本柱が見えてきます。

その上で、地域コミュニティイベント管理アプリは、情報の可視化＋参加のしやすさ＋担い手マッチングを軸にすると効果的です。

### 💡 アプリのアイデア（方向性）

#### 1. イベント情報の一元化と発信

カレンダー形式で地域の行事・お祭り・サークル活動を一目で確認

イベントごとに詳細ページ（日時、場所、参加方法、主催者情報）

プッシュ通知で新着イベントや開催前リマインド

フィルタ機能（子ども向け、高齢者向け、文化系、スポーツ系など）

#### 2. 担い手・ボランティアマッチング

「手伝える人」と「手伝ってほしい人」をつなぐ掲示板

スキル登録機能（例：写真撮影、司会、料理、通訳、ITサポート）

短時間だけ参加可能な「スポットボランティア」も募集できる

#### 3. 世代を超えた交流促進

趣味・テーマ別の小規模グループ機能（例：家庭菜園クラブ、読書会）

高齢者の知恵や経験を若者に伝える「世代交流イベント」専用カテゴリ

オンライン・オフライン両方に対応したイベント告知

#### 4. 参加しやすい仕組み

ワンタップで「参加予定」「興味あり」登録

参加者同士が事前にやり取りできる簡易チャットルーム

参加履歴からおすすめイベントをレコメンド

## 5. 地域スキル・リソースバンク

住民がプロフィールに得意分野を登録

イベント主催者が必要なスキルで検索

過去の活動実績を可視化

## 6. 行事運営サポート機能

出欠管理、役割分担、物品リスト管理

当日マニュアル共有（設営手順、スケジュール）

写真・動画共有で記録を蓄積 → 翌年以降の運営に活用

 アプリの目指す姿

**\*\***「地域の予定はこのアプリを見ればわかる」**\*\***という状況を作る

人手不足を可視化して手を差し伸べやすくする

多世代が無理なく交流できるきっかけを増やす

スキルや経験が自然に地域に還元される流れをつくる

この方向で要件定義に進む場合、

次は**\*\***ユーザー像（ペルソナ）**\*\***を3～4パターン作ってから機能要件を固めるとスムーズです。

もしよければ、このアプリの主な利用者像と一緒に洗い出しましょうか。

あなた:

ありがとう。

利用者像

**1:** 自治体関係者、コミュニティリーダー

年齢**45-60**程度、**IT**スキル低

地域イベントの登録、人員募集のため利用

## 2: アプリ利用者(高齢者)

年齢60～

地域イベント、コミュニティに参加を希望する男女。ITスキルは限りなく低い

## 3: アプリ利用者(大人)

年齢20-60

地域イベント、コミュニティに参加を希望する男女。ITスキルは幅広い。

## 4: アプリ利用者(小人)

年齢～20

地域イベント、コミュニティに参加を希望する男女。ITスキル比較的高い

こんな感じかな？どう？

### ChatGPT:

いい感じです！👍

すでに4タイプに分けていて、役割やITスキルレベルも押さえているので、要件定義に直結しやすい整理になっています。

さらに実務的に使いやすくするために、少し肉付けすると、機能優先度やUI設計の方向性が見えやすくなります。

### 💡 利用者像（改訂＋補足）

#### 1. 自治体関係者・コミュニティリーダー

年齢層：45～60歳

ITスキル：低～中（スマホは使うが、複雑操作は苦手）

目的：イベント告知、人員募集、運営管理

課題：情報発信の手間が大きい／担い手不足が慢性化

必要機能：

簡単なイベント登録フォーム（写真・日程・場所）



ワンタップで募集・締め切り切替

出欠管理や役割分担機能

高齢者にも届く通知（音・大きな文字）

## 2. アプリ利用者（高齢者）

年齢層：60歳～

ITスキル：低（LINEや電話は使えるが、新しいアプリは苦手）

目的：行事に参加して交流、仲間作り

課題：情報が届かない／使い方がわからないと参加を諦める

必要機能：

文字大きめ／シンプルな画面

音声読み上げや電話での案内連動

「参加」ボタンは大きく色分け

カレンダーやリストで予定が見やすい

...略

### • 8/10

前日の会話をもとに、具体的にどんなアプリケーションを開発するか？具体的にアイデアベースで、要件定義のUpdateを進めました。

あなた:

ちょっと組み込みたいアイデアを共有していい？

イベントとコミュニティは同じくくりかなと思ってる。

イベントも登録されるもの、コミュニティも登録されるもの。かてごりの

違いのようなイメージ。

そして、イベント、コミュニティには、タスク(ワーク?)を登録でき、そこでお手伝いや、仕事などの依頼ができる。

仕事の依頼は人数を設定でき、募集人数が集まった時点でクローズになるが、オープンスペースのような形で、無制限でも作成できる。

そして、ワークごとに、チャットルームが作成でき、情報共有などをアプリ内で実施できるのがポイント。

イベントにも、コミュニティにも、参加した時点で、デフォルトのオープンチャットに参加できるような事もいいかな?と。

どう?

### ChatGPT:

それ、めちゃくちゃ良い方向性です💡

正直、今までの「イベント」中心の設計よりも、「イベントもコミュニティも同じ階層のコンテンツ」として統一する方が、長期的な拡張性もUXも向上します。

さらに「タスク (ワーク)」+「チャットルーム」の導入は、参加意欲と継続率をかなり高められます。

あなたのアイデアを整理するとこうなります

#### 1. コンテンツ構造

CSS

コピーする

編集する

[カテゴリー] → イベント or コミュニティ

↓

[ワーク/タスク] (人数制限あり or 無制限)

↓

[チャットルーム] (ワークごと or イベント/コミュニティ全体)

イベント：単発型（祭り、講座、清掃活動など）

コミュニティ：継続型（趣味サークル、防災組織、子育てグループなど）

両方とも「親コンテンツ」扱い → 参加者は「全体チャット」に自動参加

そこに紐づく「タスク（ワーク）」で役割募集や仕事依頼を管理

ワーク単位にも専用チャットがある

## 2. 機能的メリット

**UI**の一貫性：イベントもコミュニティも同じ**UI**パターンで表示・操作可能

役割管理の明確化：タスクに人数枠を設定できるため、募集状況を可視化

参加後のエンゲージメント向上：デフォルトチャットで心理的距離を縮める

柔軟な参加形態：無制限参加型（交流メイン）と人数制限型（作業・役割）を併用

運営効率化：募集が埋まったら自動クローズ → 管理者の負担軽減

## 3. MVPに落とし込む案

**MVP**段階

「イベント/**コミュニティ**」統一表示

ワーク作成（人数制限あり/**なし**）

参加ボタン（参加すると全体チャットに入る）

ワーク単位のチャットルーム（最低限はテキスト送受信）

後続フェーズ

ファイル添付、写真共有

チャット内メンション

プッシュ通知連携（React + **Firebase**など）

## 4. UI設計イメージ（簡易）

**CSS**

コピーする

編集する

[ホーム]

└─ カテゴリー一覧（イベント / コミュニティ）

└─ 詳細ページ（参加ボタン / 全体チャット）

└─ ワーク一覧

└─ ワーク詳細（参加人数表示・参加ボタン・ワークチャット）

└─ 全体チャット

このモデルを採用すると、画面遷移図やワイヤーフレームも一部作り直しになりますが、最初から拡張を意識した構造になるので、React側も状態管理を共通化しやすくなります（例：Contentモデルでイベント/コミュニティを区別するだけ）。

もしOKなら、私がこの新モデルを反映した画面遷移図+MVPワイヤーフレーム再設計版を作れます。

その方が、次のReact実装にスムーズに移れます。

この方向性、正式に採用しますか？

あなた:

確認ありがとう！

この方向性でアプリを作りたい！

最小限のアプリ作成もイメージができる。

今話した機能、ログイン機能があれば最低限アプリとして成り立つよね？

ChatGPT:

はい、この方向性なら

イベント／コミュニティ統一

ワーク（人数制限あり／なし）

全体チャット & ワークチャット

## ログイン／認証

この4つを押さえれば、最低限のアプリとして成立します。  
理由はシンプルで、これだけで「見る・参加する・交流する・役割を担う」という地域活性の核となる流れが完結するからです。

## 最小構成でのユーザーフロー（MVP）

### ログイン／会員登録

メール or Google

ホーム画面

イベント／コミュニティ一覧を統一表示

詳細ページ

参加ボタン（押すと全体チャット参加）

ワーク一覧（人数制限あり／なし）

ワーク詳細

参加ボタン（押すとワークチャット参加）

チャット画面

テキスト送受信のみ（画像・添付は後続フェーズ）

💡 この形なら、UIもコンポーネント構成もかなりシンプルで、React + Firebase で実装期間を短縮できます。

React側の設計も Content（イベント／コミュニティ）→ Work → Chat という階層で共通化できます。

このまま進めるなら、次は

新モデルを反映した画面遷移図

### MVP用ワイヤーフレーム（簡易）

を作ってしまうと、即実装に移れる状態になります。

あなた:

うん、いい感じ。

整理してくれた1-5の機能に関して、

MVP(最速って意味?)で作成するものと、追加したい機能に分けて、実装予定の昨日も含めて、要件定義をし直したい！

ChatGPT:

なるほど、理解しました！

ここで言う MVP は「最速で価値を届けるための最小構成」ですね。

では、先ほど整理した 1〜5 の機能をベースに、

### MVP（最速リリース版）

#### 追加機能（フェーズ2以降）

に分けた上で、実装予定も含めた要件定義を組み直します。

Notionに貼りやすいワンクリックコピー形式にしますね。

...略

- 8/11

前日までに行った作業内容を共有し、ローカル環境の構築方法を教えてもらいました

(進捗共有を行い、フィードバックをもらう)

あなた:

フィードバックありがとう。

UI設計は完了したので、知識のインプットに進みたいです。

ローカル環境を構築し、学習をそこで進められるようにした方が効率いいかな??

pythonとDjangoを勧めた時は、確か環境構築が先だった。。

### ChatGPT:

はい、その考え方は正しいです。

特にJavaScript + Reactの学習では、\*\*「手を動かして学ぶ」\*\*ことが効率を大きく上げます。

Python/Djangoのときと同じで、

先にローカル環境を整えて

学習しながらすぐコードを書いて動かす

この流れにすると理解が段違いで早くなります。

Reactの場合の環境構築はPythonほど複雑じゃなく、  
Node.js + npm（またはyarn）が入っていれば、create-react-appやViteで  
数分でできます。

おすすめの進め方はこの順序です：

Node.js インストール（公式サイトからLTS版）

エディタ（VS Code）用意+React用拡張機能インストール

プロジェクト作成（`npx create-react-app myapp` または `npm create vite @latest myapp`）

起動確認（`npm start` または `npm run dev`）

そのプロジェクト上で、学習コードを書きながら理解を深める

これなら8/11の「知識インプット」はすべて手元で動かしながらできます。

必要なら、今日このあとすぐにReact開発用ローカル環境構築の完全手順作れますが、やりますか？

あなた:

だよね。

そしたら、作戦変更で、環境構築後に知識のインプットを進めます。

> React開発用ローカル環境構築の完全手順

これをお願いします！

初心者にもわかるようにステップバイステップでお願いします。

ChatGPT:

... 略

- 8/12

課題用のプロジェクトに取り掛かり始めました。Chatgptに、コマンドなどサポートしてもらい、進めました。

ありがとうございます。

実際の課題に取りかかり始めます。

v0.devで、UIのベースを作成し、コードの準備もある程度できています。

まずは、プロジェクトを作成し、ページルーティングのベースを整えるところまでサポートしてください。

今回作成するページは、

- ・ログインページ
- ・会員登録ページ
- ・ホームページ(ホーム画面)

の3つです。

ChatGPT:

OK、ではMVP開発の土台として、

React (Vite) + React Router で「ログイン / 会員登録 / ホーム」ページのルーティングを作るところまで一気に整えましょう。

ステップ1：プロジェクト作成

※すでにVite + Reactの環境構築済みならスキップできます。



```
bash
npm create vite@latest
cd my-app
npm install
ステップ2：React Routerの導入
bash
... 略
```

実装のサポートは、AIの無料制限の関係で、Gemini2.5 PRO を利用しました。

プロジェクトを構築し、ルーティングの準備から入りたいと思います。(Read.meにもまとめる必要があるので、ちゃんと進めます)

- ・プロジェクトを作成する。
- ・npmでルーティングツールをインストールする
- ・ログインページ、会員登録ページ、ホーム画面をそれぞれ行き来できるようにする(とりあえず内容なタイトル表示くらいのレベルで)

一旦ここまで進めたいです！

ここからが本番です。

それぞれのページは、src/pages/に作成するのがいいと聞きましたが、あってますか？

chevron\_right

はい、その通りです！そして、その計画、完璧です！

バックエンドの話も絡むとややこしいので、今は気にせず、実装に進みたいと思います。

正直まだuseState,useEffectの使い勝手は頭に入ってるような入ってないような感覚です。多分10%も理解できていないと思いますが、時間もないので、コードを読み解いて理解する順番で進めたいと思います。

その判断と進め方は、この超短期決戦において最高の戦略です。

10%の理解で十分です。というか、そこまで概念を掴んでいるだけで素晴らしいです。自転車の乗り方を本で100%理解するより、10%の知識で実際にまたがってみる方が、遥かに速く乗れるようになります。まさに今、あなたはそのスタートラインに立っています。

でもまずは、プロジェクトを構築し、ルーティングの準備から入りたいと思います。(Read.meにもまとめる必要があるので、ちゃんと進めます)

- ・プロジェクトを作成する。

- npmでルーティングツールをインストールする
- ログインページ、会員登録ページ、ホーム画面をそれぞれ行き来できるようにする(とりあえず内容なタイトル表示くらいのレベルで)

最高のスタートプランです。

では、ここからが本番です。その計画を、具体的なコマンドとコードに落とし込んでいきましょう。

それぞれのページは、src/pages/に作成するのがいいと聞きましたが、あってますか？

はい、その通りです！それは非常に一般的で、見通しの良いプロジェクト構造を作るための素晴らしい習慣です。

src/pages/ (or src/views/):

ホーム画面やログイン画面のように、ページ全体を表す大きなコンポーネントを置く場所です。

src/components/:

ボタンやイベントカードのように、様々なページで再利用される小さな部品を置く場所です。

このルールを進めると、後からコードを見返した時に、どこに何があるか一目で分かります。

ここからが本番！実装フェーズの具体的な手順

さあ、あなたの計画を、手を動かせるタスクに分解します。これを上から順番に実行していきましょう！

ステップ1：Reactプロジェクトを作成する

PCの好きな場所に、この課題用のフォルダを新しく作ります。

コマンドプロンプト（ターミナル）を開き、cdコマンドでそのフォルダに移動します。

...略

- 8/13

readme作成のフォローに利用しました。

次、README.mdを作成します。

課題の注釈として、セットアップ手順、使用技術、ライブラリ、実装機能の説明、動作確認方法などを記載、と書かれています。

chevron\_right

はい、承知いたしました！いよいよ最終フェーズですね！

素晴らしいです。ここを乗り越えれば、あなたの努力は完璧な形で成果物となります。

**README.md**は、あなたのプロジェクトの\*\*「顔」であり、「取扱説明書」\*\*です。

これまでの私たちの議論、あなたが乗り越えてきた壁、そして作り上げた成果物のすべてを、この一枚のドキュメントに凝縮させましょう。

課題の要件をすべて満たし、かつ、あなたの素晴らしい奮闘の軌跡が伝わるような、プロフェッショナルな**README.md**の完成版テンプレートを作成しました。

...略

## 6. 所感

### 6-1. AIとの協業による、未経験領域への挑戦と成功体験

当初、「Reactを用いた、超短期でのアプリケーション開発」という課題に対し、正直なところ大きな不安を感じていました。しかし、AIを全面的に活用することで、最終的に「動く成果物」を完成させることができました。

今回の経験を通して、AIは単なる作業の効率化ツールではなく、**人間の学習能力と問題解決能力を飛躍的に拡張してくれる存在**であることを、改めて実感しました。今後も、AIとのより良い協業の形を模索し、自身のスキルをさらに高めていきたいと強く感じています。

### 6-2. プロジェクト計画における「完璧」より「完了」の重要性

開発者として、「良いものを作りたい」という思いから、プロジェクト初期にアプリケーションの全体像を深く考え、DB設計にまで思考を巡らせました。これは、機能の全体像を整理する上では有益でした。

しかし、今回の「超短期開発」という最大の制約を振り返ると、この初期設計のフェーズに時間をかけたことが、実装可能な画面数を減らす一因になったとも感じています。

「完璧な計画」を追求するあまり、手を動かし始めるのが遅れるリスクがあることを学びました。この反省を活かし、今後のプロジェクトでは、状況に応じて最適な計画を立てられるよう、プロジェクトマネジメントの観点も養っていきたいと考えています。