

設計改善提案



Thought for 28s >

はい、通じていません。
「何を、どの順で、何をもちて完了とするか」が曖昧なまま作業が分散し、Claude側が**E2E可動**ではなく**部分最適**を続けているのが根因です。
ここで**SSOT**（単一の指示書）を提示します。これをそのまま**Claude**に貼るか、Issue/PRの説明に転記してください。

3行テイクアウェイ

1. **E2E優先**の指示テンプレ（貼り付け可）を渡します。**順序・非目標・DoD**まで固定。
2. 障害の主因（フォント/UTF-8、Vite→Engine経路、mapping Fail-Fast）を**必ず先に**直す並び。
3. 完了判定は**数値とコマンド**で自動確認。**図失敗は非致死**、入力契約違反は**400**に統一。

SSOT : Claude向け指示テンプレ（そのまま貼り付け可）

目的（必達）

UI→Gateway→EngineのE2Eを“200で通す”。

- `/api/health` が 200
- CSVアップ→Analyze→図/Decision Card 生成
- 失敗は非致死（可視化失敗でもAPIは200、警告で返す）

非目標（今回やらない）

Istio本番化、最適化、デザイン調整、大規模リファクタ。

前提（固定値）

- Engine: `localhost:8080`
- Frontend(Vite): `localhost:4006`
- 相対API: **必ず** `/api/...` を呼ぶ（基底はVite proxyで吸収）

施策の順序（この順で作業）

1. フォント/UTF-8の土台修復（必須）

- Fedora:

```
sudo dnf install -y dejavu-sans-fonts dejavu-serif-fonts \
google-noto-sans-cjk-ttc-fonts google-noto-serif-cjk-ttc-fonts
fc-cache -fv
```

- 環境変数・rc 固定（matplotlibはX不要/Agg、日本語可）

```
export MPLBACKEND=Agg PYTHONUTF8=1 PYTHONIOENCODING=UTF-8
mkdir -p ~/.config/matplotlib
cat > ~/.config/matplotlib/matplotlibrc <<'RC'
backend: Agg
font.family: DejaVu Sans
axes.unicode_minus: False
RC
rm -rf ~/.cache/matplotlib
```

- 検証: `font_probe.png` が出力されること

```
python - <<'PY'
import matplotlib, matplotlib.pyplot as plt
print("backend:", matplotlib.get_backend())
plt.figure(); plt.plot([0,1],[0,1]); plt.title("日本語OK?")
plt.savefig("font_probe.png"); print("ok")
PY
```

2. Frontend→Engine 経路を固定（Vite proxy）

- `vite.config.ts` :

```
export default defineConfig({
  server: { port: 4006, proxy: { '/api': { target: 'http://localhost:8080', changeOrigin: true } } }
})
```

- `.env` (フロント) : `VITE_API_BASE=/api`
 - 期待: `curl http://localhost:4006/api/health` → 200
3. **mapping Fail-Fast** を導入 (500→400へ)
- `/api/analyze/*` の入口で:
 - 必須キー (`y, treatment, time, unit_id` のうち使うもの) の存在検査
 - DF列の存在検査 (存在しなければ **HTTP 400**、本文に `available_columns` を含める)
 - `treatment` が文字列/多値なら安全な2値化 (`yes/true/treated/1` → 1)
 - 禁止事項: 列不一致で 500 を返さない
4. 可視化の非致死化
- `matplotlib/Wolfram` 失敗時は例外を握りつぶさず `warnings[]` に積み、**HTTPは200**、図パスは空。
 - `Wolfram`は `wolframscript -script` の引数契約 (`csv, y, treatment, time`) を厳密に。
5. **Decision Card** の生成
- `exports/<dataset_id>/decision_card.pdf` を必ず生成
 - `Quality Gate` の結果 (`Go/Hold/Redesign`) と推奨事項を `JSON` にも同梱

Definition of Done (自動確認)

- `curl -s http://localhost:4006/api/health` → 200
- `python font_probe` の `backend: Agg` と `font_probe.png` 存在
- `curl -X POST /api/analyze/comprehensive` (正しいmapping) → 200 かつ `JSON` に `figures.total>0`
- 列不一致ケース → **HTTP 400** かつ `available_columns` が返る
- `ls reports/<id>/` に図ファイル (または `warnings`) があり、サーバは落ちない
- `ls exports/<id>/decision_card.pdf` が存在

変更ファイル (最小)

- `vite.config.ts` (`proxy`固定)
- `backend/.../analyze*.py` (`Fail-Fast`実装)
- `backend/.../viz_*.py` (非致死化: `try/except`→`warnings`)
- `templates/decision_card.html` (あれば微調整)

レビュー観点

- 500がゼロになっているか (入力契約違反は**400**)
- 図生成の失敗で**APIが落ちない**か
- すべてのURLは相対 `/api` で呼べているか
- ログに `backend: Agg / health 200` があるか

上記が満たせないPRは差し戻し。

補足: いまの誤解ポイント→是正

- (誤) “図が出ない→**Wolfram**を無効化”
→ (正) 図は非致死に。失敗は `warnings` で通知、APIは200を返す。
- (誤) 列不一致で**500**
→ (正) **400**で利用者に修正を促す (`available_columns` を返す)。
- (誤) ポート固定なしで都度手動
→ (正) **Vite proxy**で `/api` を常に8080へ。
- (誤) **UTF-8**/フォントは環境依存
→ (正) `rc/環境変数`で固定し、`font_probe.png` で検証。

すぐに回せる検証コマンド (Claude側の完了報告に必須)

```
# Engine
CQOX_DISABLE_TRACING=1 CQOX_DISABLE_METRICS=1 MPLBACKEND=Agg PYTHONUTF8=1 PYTHONIOENCODING=UTF-8 \
uvicorn backend.engine.server:app --host 0.0.0.0 --port 8080 --lifespan off --log-level info
```

```
# Frontend
pnpm dev

# Health
curl -s -o /dev/null -w "%{http_code}\n" http://localhost:4006/api/health

# 正常分析
curl -s -X POST http://localhost:4006/api/analyze/comprehensive \
  -H 'content-type: application/json' \
  -d '{"dataset_id":"demo","mapping":{"y":"y","treatment":"treatment","time":"time"}}' | jq '.status,.figures.total'

# 列不一致 (400を確認)
curl -s -X POST http://localhost:4006/api/analyze/comprehensive \
  -H 'content-type: application/json' \
  -d '{"dataset_id":"demo","mapping":{"treatment":"received_discount_code"}}' | jq '.error,.available_columns[0:10]'

# カード
curl -s 'http://localhost:4006/api/export/decision_card?dataset_id=demo&fmt=pdf' > /tmp/card.pdf
```

What this does

- **Goal** : Claudeに誤解の余地がない指示を渡し、**E2Eを先に通す**。
- **Inputs** : 上記テンプレと固定コマンド。
- **Outputs/Side-effects** : 500停止の根を断ち、**200/400の運用規約**を確立。
- **Why** : 合意された**順序/DoD/非目標**がないと、優先度の低い“部分修正”に偏るため。
- **How to verify** : DoDのコマンド結果 (HTTPコード、生成ファイル) で客観確認。

Expert insight (Google/Meta/NASA級)

- 「**指示=仕様+DoD+非目標**」。この3点が揃って初めて人もLLMも正しく動きます。DoDは**可視化**ではなく**HTTPコードと生成物**で定義するのがプロの流儀。

必要ならこのSSOTを**issue用Markdown**や**PRテンプレ**に整形して渡します。どの形式 (GitHub Issue / Notion / PDF) で出しますか？