Givery

HANDS-ON GUIDE ／ ハンズオンガイド

AI協働開発ハンズオン講座 情報系 Day2 ハンズオンガイド

AI協働開発ハンズオン講座　情報系トラック　Day2

> Day2で手を動かす7つの演習を、手順とつまずいたときの戻し方までまとめた資料です。投影スライドは流れていきますが、この資料は手元に残ります。手が止まったら、いまやっている演習のページに戻ってください。

この日にすること

Day1は自分で書いた5行の仕様から worklog.html を作りました。Day2は逆で、他人が書いた index.html を読ませてから手を入れます。午後はテストの話に移り、観点の洗い出しからケース表、テストデータ、報告書のPDFまでを通します。

時間帯	ブロック	分数
午前	Day1の振り返りと今日の地図	10分
午前	モデルの使い分け（演習1を含む）	70分
午前	休憩	10分
午前	既存アプリへの機能追加（演習2・演習3）	90分
午後	規約ファイルのチーム展開（演習4）	50分
午後	テスト業務とAIの分担（手を動かしません）	50分
午後	休憩	10分
午後	観点抽出とテストケース生成（演習5）	110分
午後	テストデータ生成と報告書のPDF出力（演習6・演習7）	70分
午後	Day2クロージング	10分

各演習の分数は、演習1が20分、演習2が22分、演習3が17分、演習4が19分、演習5が44分、演習6が16分、演習7が13分です。合計すると自分の手を動かす時間は151分になります。

> 分数は目安です。当日の進み具合で前後します。戻る時刻は休憩に入る直前に口頭でお伝えします。この資料に時刻は書いていません。

Day3では、今日作ったテストケース表を単体テストとE2Eテストに落として、ブラウザの中で実際に動かします。今日は1件も実行しません。

使うもの

配布フォルダの中身

配布ZIPに入っているのは次の2つだけです。

ファイル	役割
index.html	作業記録アプリの本体です。午前の機能追加の題材で、午後のテスト設計の対象にもなります

ファイル	役割
README.txt	展開先と、今日この中に自分で作るファイルの一覧が書いてあります

展開先

デスクトップの tpro-work フォルダの中に、ZIPの中身をすべて展開してください。Day1で tpro-work を作った方は、そのフォルダに上書きで展開して構いません。見当たらない方は、デスクトップに tpro-work という名前のフォルダを新しく作り、その中へ展開してください。Day1のファイルが無くても今日の演習は最後まで進みます。

デスクトップに書き込めない端末では、ドキュメントフォルダの直下に tpro-work を作ってください。以降この資料に出てくる「デスクトップの tpro-work」は、ご自身の展開先に読み替えてください。

展開したらVSCodeの「ファイル」から「フォルダーを開く」で tpro-work フォルダを開きます。ファイルを1つだけ開くと、GitHub Copilot が周辺のファイルを読めません。必ずフォルダで開いてください。

作業記録アプリの中身

index.html はHTMLとCSSとJavaScriptが1枚に収まったファイルです。日付、作業種別、内容、工数(時間) の4つの入力欄があり、作業種別は 設計 / 実装 / レビュー / 試験 から選ぶ形です。データは records という配列に date, kind, memo, hours の4項目で入っています。開いた時点で8件の記録が入っています。

保存の仕組みはありません。追加した記録はブラウザを再読み込みすると消え、最初の8件に戻ります。まだ無い機能は、作業種別での絞り込み、工数の合計、工数の入力チェックの3つです。午前の演習で上の2つを足します。入力チェックは午後にテストケースだけを設計し、Day3で実装します。

今日この中に自分で作るファイル

いずれも配布物には入っていません。演習の中でご自身で作り、tpro-work フォルダの中に置いてください。

ファイル	どの演習で作るか
model-memo.txt	演習1と演習3。依頼文、モデルの違い、検算の結果を書きます
index.html への追記	演習2と演習3。絞り込み欄と合計の表を足します
before.txt / after.txt	演習4。規約ファイルを置く前と後の回答を貼ります
.github/copilot-instructions.md	演習4。tpro-work の直下に .github フォルダを作って置きます
AGENTS.md	演習4の発展課題。tpro-work の直下に置きます
testcases.md / prompt-memo.txt	演習5。テストケース表と、効いた一文の1行メモです
data.csv / data-memo.txt	演習6。テストデータ31行と、指定と違った点のメモです
report.md / report.html / report.pdf	演習7。テスト設計の報告書です

書く場所と動かす場所

VSCodeは書く場所です。index.html を書き換えて Ctrl と S で保存します。ここでは画面は動きません。ブラウザ（Edge等）が動かして確かめる場所です。保存した index.html を開き、直したら Ctrl と Shift と R を押して読み直します。保存せずに読み直しても画面は変わりません。

index.html をエクスプローラー（Windowsのファイル表示画面）でダブルクリックすると、既定のブラウザで開きます。アドレスバーが file:/// から始まっていれば正しく開けています。別のアプリが開いてしまう場合は、右クリックして「プログラムから開く」からブラウザを選んでください。

1週間ぶりの確認

前回から1週間空いています。VSCode右下のCopilotのアイコンに斜線が入っていたら、サインインが切れています。左下のアカウントアイコンからGitHubに入り直してください。

演習1 モデルを切り替えて同じ依頼を2回送る

どんな場面か

派遣先では、使えるモデルが組織の設定で決まっていることがあります。選べる現場でも、切り替えて何がどう変わるのかを自分の目で見たことがないと、うまくいかないたびにモデルを変える癖がつきます。同じ文を2回投げて、変わる部分と変わらない部分を手元で確かめます。

ここで言うモデルとは、チャットの回答を作っているAIの種類のことです。Copilot Chat には Claude系（Anthropic）、GPT・Codex系（OpenAI）、Gemini系（Google）と、Copilot側が自動で選ぶ Auto が同じ場所に並んでいます。並ぶ顔ぶれは数か月で入れ替わるので、順位を覚えても次の現場では使えません。覚えるのは、切り替え口の場所と、切り替えても変わらないことの2つです。

切り替えると変わるのは、返答の言い回し、コードの書き方の癖、長いファイルをどこまで一度に扱えるか、返ってくるまでの体感の速さです。変わらないのは、渡していない情報です。既存のコードを添付していなければ、どのモデルを選んでも既存の作りを外した提案が返ってきます。

やること

1. VSCodeで tpro-work をフォルダごと開き、その中の worklog.html を開きます。Day1のファイルが手元に無い方は、配布の index.html を使ってください。
2. tpro-work の中に model-memo.txt を新しく作り、依頼文を1行貼って控えます。文は次のとおりにそろえます。

```
このファイルの動きを、初めて見る人に向けて説明してください。日本語で回答してください。
```

3. Copilot Chat の入力欄で半角の # を打ちます。ファイルの候補が出るので、対象のファイルを選んで添付します。入力欄の上にファイル名が並べば添付できています。
4. モデル名は触らずにそのまま送ります。返ってきた説明は最初の5行だけ読みます。全部読まなくて構いません。
5. 入力欄の中に出ているモデル名をクリックすると選択肢の一覧が開きます。ここが切り替え口です。別の提供元の系統に切り替えて、控えた文を貼ってもう一度送ります。文は1文字も変えません。
6. どこが違ったかを1行、model-memo.txt に書き足します。どちらが優れていたかは書きません。

依頼文とモデルを同時に変えると、どちらが効いたのか分からなくなります。変えるのは片方ずつです。

違いを書くときは、説明の長さ（短くまとまった／細かく並んだ）、コードの出し方（全文を出した／変更箇所だけ出した）、前提の確認（先に質問してきた／そのまま答えた）のどれかで書くと言葉になります。

できたかの確認

- model-memo.txt に、送った依頼文がそのまま1行残っている。2回とも同じ文を送ったことが後から確認できる
- 1回目は既定のまま、2回目は別の提供元の系統で送っている。提供元が変わっていることを一覧で確認できる
- 違いを書いた1行が入っており、優劣ではなく「どこが違ったか」の形で書かれている
- 違いが分からなかった場合は、「今回の依頼では差が分からなかった」と書けていれば達成です

うまくいかないとき

症状	戻し方
# を打っても候補が出ない	全角の＃になっています。半角に切り替えて打ち直します
添付できていない	入力欄の上にファイル名が出ているかを見ます。出ていなければ添付されていません
モデルの一覧が開かない	サインインが切れています。左下のアカウントアイコンからGitHubに入り直します
一覧に系統が1つしか出ない	組織の設定で絞られています。既定のまま1回だけ送り、隣の方の画面で違いを見せてもらいます
2回目で依頼文をつい足してしまった	控えた1行に戻して、新しいチャットで送り直します
切り替えたのに提供元が同じ	同じ提供元の別バージョンを選んでいます。一覧で提供元の名前を見て選び直します

同じモデルで2回送っても答えが少し変わることがあります。毎回まったく同じ答えが返るわけではありません。それも違いとして記録して構いません。

時間が余ったら

- 3つ目の系統でも同じ文を送り、3つを並べて違いを1行増やします
- 同じモデルで2回続けて同じ文を送り、同じモデルの中でも返答が揺れるかを見ます
- 依頼文の末尾に「200字以内で」を足して送り直し、字数の指定がどこまで効くかを見ます

午前の理解度チェック

配布された既存のWebアプリに、作業種別で絞り込む機能を足したいとします。最初にやることとして最も適切なものはどれでしょうか。

1. 性能が高いとされる系統のモデルに切り替えてから依頼する
2. 対象のファイルをチャットに添付して、いまの作りを説明させる

3. 完成後の画面イメージを絵に描いてチャットに添付する
4. 「絞り込みをつけて」と送り、返ってきたコードをそのまま貼る

> 自分で選んでから読んでください。
>
> 正解は2です。既存コードに手を入れる依頼は、いまの作りが渡っているかどうかで結果が変わります。先に読ませてから頼むと、既存の関数名とデータの持ち方をそのまま使った変更が返り、どこに足すかまで書かれます。モデルの切り替えは詰まってからで間に合います。
>
> 1が外れる理由は、前提が渡っていない状態ではどの系統を選んでも既存の作りを外した提案になるからです。3は見た目の情報だけでは既存の関数名やデータの持ち方が伝わらず、貼る場所が決まりません。4は既存と噛み合わない書き直しが返り、貼ると一覧が二重に出るなどの手戻りになります。

依頼文の最後に「まだコードは書かないでください」を添えると、説明だけを先に受け取れます。

演習2 作業種別での絞り込みを足す

どんな場面か

派遣先で最初に任される改修は、たいてい他人が書いた画面への小さな追加です。仕様書は無く、動く画面と口頭の依頼だけが手元にあります。その状態から、既存を壊さずに1機能を足すところまでを、配布の作業記録アプリで一度通します。

同じ依頼でも、先に読ませたかどうかで返ってくるものが変わります。「作業種別で絞り込めるようにして」とだけ送ると、既存の render を使わない書き直しが返り、貼ると一覧が二重に出たり追加ボタンが反応しなくなったりします。先に index.html を添付して説明させてから同じことを頼むと、既存の render と records をそのまま使った変更が返ります。

先に読ませる

機能を足す前に、いまの作りを説明させます。返ってきた説明はコードの中で自分で確かめます。

1. index.html をブラウザで開き、一覧が出ることと1件追加できることを確認します。壊す前の状態を自分の目で見ておきます。
2. Copilot Chat の入力欄で半角の # を打ち、index.html を選びます。
3. 次の依頼文をそのまま送ります。ここではまだコードを書かせません。

```
#index.html
このアプリの役割と、画面に一覧が出るまでの流れを説明してください。
次の3点は必ず含めてください。
- 作業記録のデータがどこに入っているか
- 一覧を描画している関数の名前
- 追加ボタンを押したときに呼ばれる関数の名前
日本語で回答してください。まだコードは書かないでください。
```

4. 返ってきた関数名を、実際に index.html の中で検索して探します。見つからなければ、その説明は当てになりません。

3点を指定しているのは、返ってきた説明を自分で検算できる形にするためです。関数名は検索すれば当たりを確かめられます。

やること

1. 仕様を3行でチャットに書きます。一覧の上に作業種別を選ぶ欄を置くこと、選ぶと一覧がその作業種別だけになること、「すべて」を選ぶと元に戻ること。この3行です。
2. 半角の # で index.html を添付し、3行の仕様を送ります。末尾に「既存の render をそのまま使ってください」と一文足します。
3. 返答に書かれた場所を先にVSCodeで開いて見つけてから貼ります。分からなければ「どの関数の下に入れるか教えてください」と聞き返します。コードは返答の右上のコピーボタンで丸ごとコピーしてください。一部だけ手で選ぶと貼り漏れます。
4. Ctrl と S で保存し、ブラウザで Ctrl と Shift と R を押して読み直します。選択を切り替えて一覧が変わるかを見ます。
5. 動かなければ F12 を押し、出ているエラー文をそのままチャットに貼ります。自分の言葉に言い換えないでください。

1回に1つだけ変えてください。2つ同時に頼むと、どちらが原因で壊れたか分からなくなります。

> 動いた形は残しておきます。機能が1つ動いたら index.html をコピーして、ファイル名の末尾に _ok1 のように付けた別名で保存します。手に負えなくなったら、そこへ戻して依頼文を書き直します。同じ文で投げ直しても同じところで詰まります。本研修ではGitを使わないので、コピーで代用しています。

できたかの確認

- 一覧の上に作業種別を選ぶ欄が出ており、設計を選ぶと設計の行だけが残る
- 「すべて」に戻すと、絞り込む前の件数に戻る
- 絞り込んだ状態でも、フォームから1件追加できる。前からある機能が壊れていない
- index.html の中に、既存の render を呼ぶ形で追記されている。render が丸ごと別の関数に置き換わっていない

足した機能が動くことと、前からある機能が壊れていないことは、別に見てください。

うまくいかないとき

症状	戻し方
選択欄は出たが一覧が変わらない	イベントの登録が漏れています。「選択を変えたときに一覧が更新されません」とだけ伝えて直させます
画面が真っ白になった	F12 を押して赤い行の1行目をそのまま貼ります。原因は推測しません
保存したのに画面が変わらない	ブラウザのキャッシュです。Ctrl と Shift と R で読み直します
追加ボタンが反応しなくなった	既存の render が置き換わっています。貼る前に戻して、既存の render を使うよう書き足して頼み直します

症状	戻し方
貼る場所が分からない	「どの関数の下に入れるか教えてください」と聞き返せば、行の位置まで返ってきます

動いた時点で、どの一文が効いたと思うかを1行メモしておいてください。午後の演習4で使う材料になります。

時間が余ったら

- 選択中の件数を、一覧の上に「表示 12 件」のように出します
- 作業種別に加えて、日付の範囲でも絞り込めるようにします。ただし1つずつ足してください
- 絞り込んだ状態で追加したとき、その1件が一覧に出るかどうかを確かめ、出ない場合の仕様を自分で決めます

演習3 作業種別ごとの工数合計を足す

どんな場面か

集計はAIがすぐ書いてくれる部類の処理です。だから中身を見ずに通してしまいやすく、桁が違っていても画面はそれらしく見えます。出てきた数字を人が1つだけ検算する動作を、ここで手癖にします。

画面に数字が出たことではなく、その数字が合っていることを見ます。

やること

1. 何を出すか決めます。作業種別ごとの工数の合計を、一覧の下に表で出します。絞り込みが効いているときは、その結果だけを合計します。
2. 演習2と同じように index.html を添付して仕様を送ります。末尾に「前の絞り込み機能を壊さないでください」と一文足します。
3. 作業種別を1つ選び、一覧に出ている工数を自分で足して、画面の合計と一致するかを見ます。ここは必ず人がやります。
4. 作業種別を切り替えて、合計も一緒に変わるかを確認します。変わらなければ、その症状をそのまま伝えて直させます。
5. 検算の結果を1行、model-memo.txt に書き足します。合っていたか、ずれていたか、ずれた原因は何だったかを書きます。

できたかの確認

- 一覧の下に、作業種別ごとの工数合計が表で出ている
- 作業種別を1つ選んで手で足した数と、画面の合計が一致している。一致を自分で確認した
- 絞り込みを切り替えると、合計も一緒に変わる
- model-memo.txt に検算の結果が1行増えている。ずれていた場合は、ずれた原因も書かれている

うまくいかないとき

症状	戻し方
合計が全件のままで、絞り込みと連動しない	「作業種別を切り替えても合計が変わりません」とだけ伝えます
合計が 1.52.5 のように横につながる	文字列として足されています。症状をそのまま貼れば直ります
合計の表が一覧の中に入り込んで崩れる	どこに出すかを仕様に書いていません。「一覧の下に別の表として出してください」と足します
前の絞り込みが消えた	手順2の一文を書き忘れています。貼る前の形に戻してから頼み直します
小数の合計が0.1ずれる	小数の丸めによるものです。今日は深追いしません。表示する桁を丸める指示を足せば済みます

時間が余ったら

- 合計の多い順に並べ替えます
- 合計行に件数を併記し、何件で何時間かが分かるようにします
- 合計を1つだけわざと壊してもらい、検算で気づけるかを試します。壊す前のファイルは必ずコピーしてから行います

演習4 規約ファイルを置いて前後を比べる

どんな場面か

派遣先では、同じ画面のコードを複数人が触ります。前置きを丁寧に打つ人と省く人がいると、同じ依頼でも返ってくるコードの形がそろわず、レビューの指摘が人によって変わります。規約ファイルを1枚置いたとき、その差がどこまで埋まるのかを自分の手元で確かめます。

新しいチャットのスレッドを開くと前の会話は引き継がれないので、使う言語、禁止したいライブラリ、コメントの書き方を毎回打ち直すことになります。tpro-work/.github/copilot-instructions.md に置くと、Copilot Chat が会話のたびに自動で読みます。意識して貼る必要はありません。書き方を変えたいときは、このファイル1か所を直せば済みます。

.github はGitHub由来の決まったフォルダ名です。GitHubはソースコードを共有する場所、GitHub Copilot はVSCodeの中で動く別の製品で、規約ファイルを読むのは Copilot のほうです。

やること

1. デスクトップの tpro-work フォルダをVSCodeで開き、Copilot Chat に次の文だけを送ります。言い回しは変えません。

```
工数の入力チェックの関数を追加して
```

2. 出てきたコードを tpro-work フォルダの中の before.txt に貼って保存します。
3. VSCodeのエクスプローラーで tpro-work の行を1回クリックして選び、新しいフォルダーのアイコンで .github を作ります。先頭のドットを忘れると読まれません。tpro-work の1つ上に作っても読まれません。
4. .github を選んだ状態で新しいファイル copilot-instructions.md を作り、次の全文を貼ります。そのうえで自分の現場で足したい行を2行だけ追記して、Ctrl と S で保存します。
5. チャットを新しいスレッドで開き直し、手順1とまったく同じ文言をもう一度送ります。出てきたコードを after.txt に貼って保存します。
6. before.txt と after.txt を左右に並べて開き（タブを右端にドラッグすると画面が2分割になります）、変数名、コメントの言語、DOMの取得方法の3点を見比べます。

規約ファイルの全文

手順3まで終えてから写してください。手順1を先に済ませないと、前後の比較になりません。

作業記録アプリ向けに書いた copilot-instructions.md の全文です。前提、言語と書き方、レビュー観点、変更の出し方の4つの見出しで区切っています。禁止したいことまで書くのが要点です。

```
# このプロジェクトの前提
- 単一の index.html に HTML と CSS と JavaScript を書く。ファイルを分割しない。
- ブラウザで index.html を直接開いて動かす。ビルド手順は無い。
- fetch は使わない。file:// では動かないため、ファイルはドラッグ＆ドロップで受け取る。

# 使う言語と書き方
- JavaScript（ES2020）で書く。TypeScript は使わない。
- 外部ライブラリを追加しない。同梱しているものだけ使う。
- 変数名と関数名は英語、コメントは日本語で書く。
- DOM の取得は document.getElementById と querySelector に統一する。
- 既存の render 関数は消さない。画面の描き直しは render を呼んで行う。

# レビューで見るところ
- 入力値のチェックを通ったあとに records へ追加しているか。
- 画面に出す文言をコードの各所に散らさず、1か所にまとめているか。
- 例外を握りつぶさず、画面のメッセージ領域に文言を出しているか。

# 変更の出し方
- 変えたファイル名と関数名を先に1行で書く。
- 該当する関数は省略記号を使わず全体を出す。
```

2つの規約ファイル

発展課題で使う AGENTS.md との違いです。読む側が違うので、置き場所と書く粒度も変わります。

項目	copilot-instructions.md	AGENTS.md
置き場所	tpro-work/.github/ の中	tpro-work の直下
読む側	GitHub Copilot。Chat が会話のたびに読みます	Copilot を含む複数のAIコーディングツールが読む共通の書式です

項目	copilot-instructions.md	AGENTS.md
入れる中身	書き方の指定。言語、命名、禁止事項、レビュー観点	プロジェクトの前提。構成、動かし方、触ってよい範囲
分量の目安	20行前後。守ってほしい順に上から	30行前後。事実だけを書き、判断は書かない

同じ内容を両方に写さないでください。片方を直したときにもう片方が古くなり、結果が安定しなくなります。

できたかの確認

- 回答の上に出る参照ファイルの一覧に copilot-instructions.md が並んでいる。ここまでで「読まれた」ことが確認できます
- after.txt のコメントが日本語で、DOMの取得が getElementById か querySelector になっている
- 規約に書いた項目のうち、何個が反映されて何個が反映されなかったかを数で言える
- before.txt と after.txt が tpro-work フォルダの中に保存されている

読まれたかの確認と、守られたかの確認は別の作業です。参照ファイルの一覧に名前が出ていても、全部の行が守られるわけではありません。

うまくいかないとき

症状	戻し方
.github フォルダが作れない	Windowsのエクスプローラーはドットで始まる名前を弾くことがあります。VSCodeのエクスプローラーで新しいフォルダーを押し、名前に .github と入力します
参照ファイルの一覧に名前が出ない	保存していないか、tpro-work の1つ上に .github を作っています。タブの丸印と、エクスプローラー上の階層の2つを確認します
前後で差が出ない	手順1の依頼文に細かい指定を足しています。規約と内容が重なって差が消えます。文言を元に戻してもう一度送ります
2回目を同じスレッドで続けてしまった	前の会話に引きずられて差が見えません。新しいスレッドを開いて送り直します
ファイル名を copilot-instruction.md と書いた	instructions は複数形です。1文字違うと読まれません
タブ名の横に丸印が付いたまま	保存されていません。Ctrl と S で保存します

時間が余ったら

- 規約に「1関数30行以内」を追記して3回目を送り、行数の指定がどこまで効くかを見ます
- 禁止事項の行だけを消した版で同じ依頼を送り、否定形があるときとないときで結果が変わるかを見ます
- AGENTS.md を tpro-work の直下に置き、参照ファイルの一覧に2つとも出るかを見ます。中身は次のとおりです

```
# 作業記録アプリ
日付・作業種別・内容・工数を入力して一覧に表示する、1画面のアプリです。

## 構成
- index.html … 画面、スタイル、処理を1ファイルに収めています
- .github/copilot-instructions.md … コーディング規約

## 動かし方
- index.html をブラウザで開きます。サーバは立てません。
- 直したら Ctrl+S で保存し、ブラウザで Ctrl+Shift+R を押して読み直します。

## 触ってよい範囲
- records に入れる項目（date, kind, memo, hours）の名前は変えない。
- 既存の render を消して作り直さない。追記で足す。

## 制約
- 外部と通信する処理は書かない。
- 実在する会社名・人名・案件名をデータに書かない。
```

テスト業務とAIの分担

この50分は手を動かしません。話題がテストに移り、ここから Day2 の残りと Day3 は全部テストの話になります。

テストの4工程

工程	やること	今日の扱い
計画	何をどこまで確認するかを決めます。範囲と、ここまでやれば終わりという条件を書きます	扱いません
設計	観点を並べ、条件を組み合わせ、ケース表にします	休憩のあとの110分はここだけです
実行	手順どおりに動かし、画面を目で見て結果を記録します	動かすのは Day3 です
報告	件数と未実施を数え、読む人に合わせて要約します	今日の最後にPDFで出します

この4工程は、単体テストでも結合テストでも同じ順に並びます。いま何の作業をしているのか分からなくなったら、この表に戻ってください。

AIが出すのは候補まで

どの工程でも、AIが出すのは候補です。計画では過去の不具合の傾向から確認したい範囲の候補、設計では観点の洗い出しとケース表の下書き、実行では操作手順の文章化と再現手順の書き起こし、報告では件数の集計と表の整形。ここに挙げたものは全部、間違っていても画面上は同じ見た目で返ってきます。

人が決めるのは、計画なら出荷してよいかの線引き、設計なら仕様の解釈と落としてよいケースの判断、実行なら実際に動かして画面を目で見る確認、報告なら不具合の重大度とリリース可否です。業務への影響を知っている人にしか決められません。

「テストの網羅性を保証したのはAIです」と書ける報告書は存在しません。ケースが足りていませんでした、と説明する場面で、AIが出したので、は通りません。出したものに自分の名前を出せる範囲までが、AIに任せてよい範囲です。

返る量を止める1行

一度に100件のケースを出させると、1件ずつ見る気力がなくなります。分けて出させる最初の1手は、書かせないものを1行で禁止することです。

```
作業記録アプリの工数の入力チェックについて、テスト観点を出してください。
テストケース、期待結果、操作手順は書かないでください。
```

効いているのは2行目です。この2行に対象の絞り込みと個数と書式を足した完成形は、演習5で使います。

GitHub公式のベストプラクティスのページでも、生成結果をレビューせずに使わないことと、依頼を小さく分けることが挙げられています。記載は更新されるので、社内に展開する前にご自身でページを確認してください。

理解度チェック

テスト業務でAIに任せてよい範囲の線引きとして、最も適切なものはどれでしょうか。

1. 観点の洗い出しは人がやり、ケース表の清書だけをAIに任せる
2. 過去の不具合の傾向を渡し、どこまで確認すれば出荷してよいかの判断まで任せる
3. 候補を出す作業はAIに任せ、どれを落とすかと不具合の重大度は人が決める
4. 100件のケースを一度に出させ、レビューは不具合が出た箇所だけに絞る

> 自分で選んでから読んでください。
>
> 正解は3です。計画・設計・実行・報告の4工程のどこを見ても同じ形になります。AIは候補を量産できますが、落としてよいケースの判断と不具合の重大度は、その現場と業務への影響を知っている人にしか決められません。
>
> 1は分担が逆です。観点の洗い出しはAIが速い作業で、人に残すべきなのは清書ではなく、並んだ観点から何を落とすかの判断です。2は傾向から範囲の候補を出させるところまでは使えますが、終了条件の判断は人に残ります。4は一度に大量に出させるとレビューが形だけになり、そもそもケースが抜けていることに気づけません。

演習5 観点表とテストケース表を作る

どんな場面か

派遣先で最初に任されるテスト設計は、たいてい他人が作った画面の一部です。仕様書は途中までしかなく、動く画面だけが手元にあります。その状態から観点を出して、他の人が実行できる形のケース表に落とすところまでを、配布の

作業記録アプリで一度通します。

今日いちばん長い110分のブロックです。うち44分が自分で手を動かす時間です。今日は実行しません。実行はDay3です。

4段の構造

観点からケース表までを4段に分けて、順に細かくします。段を飛ばして一気に頼むと、画面全体の話とボタン1つの話が混ざった表が返ってきます。

段	何を書くか	配布アプリでの例
段1 観点	何を確かめたいかの切り口	工数の入力チェック
段2 条件	値の範囲や状態への分解	0以下 / 0.5から24 / 24超 / 空欄 / 全角数字
段3 ケース	1行の操作	工数に 0 を入れて追加ボタンを押す
段4 期待結果	そのとき画面がどうなるか	工数欄の下に「0.5以上24以下で入力してください」が出る。一覧は増えない

対象は画面全体にしません。入力欄1つ、または選択欄1つまで絞ります。

同値クラスと境界値

同値クラスは、同じ結果になる値のかたまりのことです。境界値は、そのかたまりの境目にある値です。工数を 0.5 から 24 の範囲とすると、0 は入らない値の代表、0.5 は入る値の下の境目、8 はかたまりの真ん中の代表、24 は上の境目、24.5 は入らない値の代表になります。

線に載らない値も同値クラスの1つとして拾います。空欄、全角数字、マイナスの3つです。全部の組み合わせは作りません。1項目ずつ動かし、関係しそうな項目だけ組み合わせます。

入力項目	同値クラス	境界値として拾う値
工数	範囲内 / 範囲より小さい / 範囲より大きい / 数値でない	0、0.5、24、24.5
内容	文字が入っている / 空 / 上限を超えている	空、1文字、上限、上限＋1文字
日付	当日以前 / 未来日 / 書式が違う	前日、当日、翌日
作業種別の選択欄	特定の工程を選択 / すべて	すべてに戻したとき

やること

1. 対象を1つに絞ります。午前に機能追加した配布アプリから、作業種別の絞り込み、作業種別ごとの工数合計、工数の入力チェックのどれか1つを選びます。画面全体は対象にしません。
2. 対象部分のHTMLをチャットに貼り、次の文で観点だけを出させます。返ってきた観点に、自分の観点を2つ足します。

```
次のWebアプリの一部について、テスト観点だけを箇条書きで出してください。
- 対象は工数の入力欄と追加ボタンだけです
- 表にせず、観点の名前だけを1行ずつ書いてください
- 10個以内にしてください
- 観点の名前は「○○の入力チェック」のような短い名詞で書いてください
（この下に、対象部分のHTMLを貼ります）
```

3. 観点ごとに同値クラスと境界値を出させます。項目、同値クラス、境界値の3列で頼みます。
4. 次の見出し行をそのままコピーして渡し、Markdownの表で出させます。列を先に指定しないと、AIは毎回違う形で返してきます。

```
| ID | 観点 | 前提 | 操作 | 期待結果 | 優先度 |
|---|---|---|---|---|---|
| C-01 | 工数の入力チェック | 一覧に初期データが表示されている | 工数に 0 を入れて追加を押す | 「0.5以上24以下で入力してくだ
| C-02 | 工数の入力チェック | 同上 | 工数に 0.5 を入れて追加を押す | 一覧が1件増え、入力欄が空に戻る | 高 |
| C-03 | 工数の入力チェック | 同上 | 工数を空のまま追加を押す | 「入力してください」が出て、カーソルが工数欄に移る | 中 |
| C-04 | 作業種別の絞り込み | 一覧に初期データが表示されている | 作業種別を「実装」に切り替える | 実装の行だけが残る。すべて
```

5. 出てきた表を、デスクトップの tpro-work フォルダの中に testcases.md という名前で保存します。
6. 改善サイクルを2周します。仕様のメモと画面を見ながら表にない操作を探し、「同じ作業種別で同じ日に2件登録したときが抜けている」のように観点の名前で1文にして、既存の表に追記させます。「作り直して」とは言いません。作り直させるとIDがずれます。
7. どの一文が効いたと思うかを1行だけ、tpro-work フォルダの中の prompt-memo.txt に書きます。

AIが1回で出すのは、1操作ずつの独立したケースです。渡したHTMLは1画面の止まった状態なので、時間の経過や操作の順番は材料に含まれていません。だから出てきません。登録してから削除する、前の入力が残ったまま次を開く、タブを2つ開いて交互に操作する。この手の切り口を人が文章で渡せば、行を書くのはAIができます。

> チャットに入れてよいのは配布アプリの中身だけです。客先名、製品名、実データは入れません。

できたかの確認

- testcases.md に10行以上のケースがあり、そのうち2行以上は自分が抜けに気づいて足した行である
- どの行にも前提、操作、期待結果がそろっている
- 期待結果が、画面に何が出るかで書かれている。「正常に動作する」「エラーになる」は不可です
- 優先度が全部「高」になっていない。落ちたら業務が止まるものだけが高になっている

期待結果が仕様から判断できない行が出てきたら、それを見つけたこと自体が成果です。仕様の穴なので、ケース表に「確認中」と書いて残してください。実務でも同じ扱いになります。

うまくいかないとき

症状	戻し方
対象が絞れず観点が30個出た	入力欄1つ、選択欄1つまで絞り直します

症状	戻し方
午前の機能追加が途中	絞り込み欄が画面に出ているところまでを対象にして構いません。期待結果は自分が決めた仕様で書きます
観点だけ頼んだのにケース表が返ってくる	「観点だけを箇条書きで、表にしないで」を書き足します
Markdownの表が崩れて見える	Ctrl と Shift と V でプレビューを開くと整形された形で見えます
ケースが5行しか出ない	条件の表に戻り、境界値の列から拾い直します
追記を頼んだのに似た行が増えた	「もっと追加して」では増えるだけです。抜けている操作の名前を書いて渡します
testcases.md が見つからない	保存先が別のフォルダになっています。デスクトップの tpro-work フォルダの中に統一します

時間が余ったら

- 登録してから削除、削除してから同じ内容で再登録、の順に操作するケースを3行足します
- 優先度が高の行だけを抜き出した実行順リストを、別の表としてもう1つ作ります
- 作業種別の絞り込みと工数の入力チェックを組み合わせたケースを2行足します

演習6 テストデータをCSVで30行作る

どんな場面か

テストデータを人が手で作ると、正常系ばかりの30行になります。境界値の行は面倒なので後回しになり、結局そこが漏れます。件数と分布を先に指定して作らせる書き方を、配布アプリの列で一度やります。

CSVは、値をカンマで区切って並べた表形式のテキストファイルです。1行目に列の名前を置きます。

正常系、境界値、異常系は、同値クラスと境界値の言い換えです。正常系はどの同値クラスにも素直に収まる値で、hours なら 8 のような真ん中の値です。境界値はかたまりの境目で、hours なら 0.5 と 24 です。異常系は同値クラスの外にある値で、hours なら 0 や 24.5、それに空欄と全角数字です。内訳を指定しないと、正常系ばかりの当たり障りのないデータが返ってきます。

やること

1. 次の文をチャットにコピーします。前のスレッドにケース表が残っていれば、同じスレッドで続けます。

```
次のテストケース表に対応する入力データを CSV で30行作ってください。
条件:
- 列は date,kind,memo,hours の4つ
- 正常系20行、境界値5行、異常系5行の内訳にする
- hours には 0、0.5、8、24 を必ず1行ずつ含める
- kind は 設計 / 実装 / レビュー / 試験 の4種だけにする
- memo は架空の作業内容にし、20文字以内にする
- date は YYYY-MM-DD でそろえる
```

```
- 実在しそうな企業名や人名は使わない
- 先頭行はヘッダにし、コードブロックだけを出力する
```

2. hours の条件を自分のケース表に合わせて直します。0、0.5、8、24 のうち、自分の表に出てこない値があれば足します。
3. 返ってきたCSVをコピーし、VSCodeで新規ファイルを作って、デスクトップの tpro-work フォルダの中に data.csv として保存します。
4. 行数を数えます。data.csv を開くと左端に行番号が出ます。31行になっていなければ、足りない内訳を名指しして追加を頼みます。
5. hours 列に 0 と 24 が入っているか、kind 列が 設計 / 実装 / レビュー / 試験 の4種だけかを目で確認します。
6. 指定と違った項目を1行だけ、tpro-work フォルダの中の data-memo.txt に書きます。何行足りなかったか、どの条件が無視されたかを書きます。

返ってきたCSVを目で信じないでください。数えてください。この演習の主眼は生成ではなく検算です。

できたかの確認

- data.csv が31行（ヘッダ1行とデータ30行）ある
- hours 列に 0、0.5、8、24 が各1行以上入っている
- kind が 設計 / 実装 / レビュー / 試験 の4種に収まっている
- memo に実在しそうな企業名や人名が1件も入っていない
- data-memo.txt に、指定と違った点が1行書かれている

うまくいかないとき

症状	戻し方
30行と頼んで15行で止まる	「残りも続けて」で続きが出ます。出た行数は毎回数えます
コードブロックの前後に説明文が付く	手で消して構いません。最後の1行の指定で減りますが、毎回消えるわけではありません
hours に 1.25 のような値が混ざる	0.5刻みだと条件に書いていないのが原因です。「0.5刻みだけにしてください」を足します
kind に調査や打合せが混ざる	指定しても守られないことがあります。名指しで直させ、その事実を data-memo.txt に書きます
保存したのにエクスプローラーに出てこない	保存ダイアログの場所が別フォルダです。デスクトップの tpro-work を選び直します
行数が数えにくい	最終行にカーソルを置くと、右下のステータスバーに行番号が出ます

31行にならない方が多数派です。半分に届かないのが普通なので、そのまま名指しで追加を頼んでください。

時間が余ったら

- 同じ作業種別で同じ日に2件登録した行を3行足すよう頼み、重複を含むデータに広げます
- 作業種別ごとの hours の合計をAIに計算させてから、1種類だけ自分で検算します
- date に来月の日付を3行混ぜるよう頼み、未来日のデータを足します

演習7 報告書をPDFで出す

どんな場面か

設計だけで終わった作業は、上に報告するときにいちばん説明しにくいところです。件数と未実施の理由を書いた1枚があれば、進捗の説明が口頭で済みます。ケース表を報告書にしてPDFにするところまでを一度通します。

今日の報告書には合否の欄を作りません。実行していないので、書ける数字がありません。書くのは、概要、対象機能、設計したケース件数、未実施と理由、所見の5つです。

やること

1. testcases.md をチャットに添付し、次の文で report.md を書かせて、デスクトップの tpro-work フォルダの中に保存します。

```
添付の testcases.md をもとに、テスト設計の報告書を Markdown で書いてください。
見出しは 概要 / 対象機能 / 設計したケース件数 / 未実施と理由 / 所見 の順にしてください。
条件:
- 設計したケース件数には、合計と優先度別（高・中・低）の内訳を表で入れる
- 未実施と理由には「全件未実施。Day3で実行予定」と書く
- 所見は3行以内にする
- 合否や不具合件数の欄は作らない
```

2. testcases.md を開いて行を数え、報告書の合計件数と優先度別の内訳が一致しているか突き合わせます。違っていたら手で直します。数えるときはヘッダ行と区切り行を除きます。
3. 「この Markdown を1枚の HTML にして、印刷用の CSS も入れてください」と頼み、tpro-work フォルダの中に report.html として保存します。印刷用のCSSには、表の行が途中で分かれない指定と、余白の指定を入れてもらいます。
4. report.html をブラウザのウィンドウにドラッグして開き、Ctrl と P を押します。送信先を「PDFに保存」にして、tpro-work フォルダの中に report.pdf として保存します。
5. プレビューで表の右端と見出しの位置を見ます。切れていたら用紙を横向きにするか、文字を小さくするCSSを足すよう頼みます。

手順2を飛ばさないでください。ここを飛ばすと、この演習は単なるファイル変換の練習になります。

> 報告書に客先を特定できる語が入っていないかを、保存する前にご自身で見直してください。

できたかの確認

- A4の1枚か2枚のPDFになっており、表の右端が切れていない

- 見出しがページの最終行に取り残されていない
- 報告書の合計件数と優先度別の内訳が、testcases.md を数えた数と一致している
- 未実施と理由に「全件未実施。Day3で実行予定」と書かれている

うまくいかないとき

症状	戻し方
印刷プレビューが真っ白になる	詳細設定の「背景のグラフィック」にチェックを入れます
report.html をダブルクリックすると別のブラウザで開く	ウィンドウにドラッグして開きます
AIが書いた件数が実際と合わない	よくあります。数えた数で上書きし、合わなかったことを所見に書いても構いません
表が横に切れる	まず用紙を横向きにし、それでも切れるなら列を減らします。全部の列をPDFに載せる必要はありません
見出しがページの最後に残る	見出しの後ろで改ページさせない指定をCSSに足すよう頼みます

PDF化は今日はブラウザの印刷で完結します。Markdown PDF 拡張を使う方法もありますが、インストールの可否は職場のルールに従ってください。設定を入れないと変換用のブラウザを取りに行き、社内ネットワークでは止まったままになります。使う場合は設定の markdown-pdf.executablePath に、PCに入っているEdgeの実行ファイルのパスを入れます。

時間が余ったら

- 優先度が高の行だけを抜き出した表を、報告書にもう1枚足します
- 報告書の冒頭に3行の要約（対象機能、設計件数、人が追記した件数）を付けます
- 同じ report.md から、社内向けと客先向けで文の硬さを変えた2種類を作らせて読み比べます

うまくいかないときに見るところ

演習をまたいで起きる困りごとです。演習ごとの症状は各セクションの表を見てください。

症状	見るところ
Copilot Chat が開かない	アクティビティバーのCopilotのアイコンを押します。見当たらないときは Ctrl と Alt と I でも開きます
Copilotが応答しない、モデル一覧が出ない	サインインが切れています。VSCode右下のアイコンに斜線が入っていないかを見て、左下のアカウントアイコンから入り直します
# を打ってもファイル候補が出ない	全角の＃です。半角に切り替えます

症状	見るところ
添付したはずのファイルが読まれていない	入力欄の上にファイル名が出ているかを見ます。出ていなければ添付されていません
既存の作りを無視した回答が返る	対象のファイルを添付していません。読ませてから頼み直します
保存したのにブラウザの画面が変わらない	保存できていないか、キャッシュです。タブ名の横の丸印を消してから Ctrl と Shift と R で読み直します
画面が真っ白になった	F12 を押し、コンソールの赤い行の1行目をそのままチャットに貼ります。原因は推測しません
前は動いていた機能が壊れた	残しておいたコピーに戻し、依頼文を書き直してから頼み直します
どこまで戻せばいいか分からない	1回に1つだけ変えるルールに戻ります。何を1つ変えるかを決めてから頼みます
作ったファイルが見つからない	保存先が tpro-work 以外になっています。VSCodeのエクスプローラーで tpro-work の中を見て、無ければ保存し直します
途中で席を外して手順が分からなくなった	近くの方に聞くか、休憩時に個別でお声がけください。途中から合流できます

問い合わせ先は（要確認：メールアドレス、当日の緊急連絡先）です。演習中と休憩時間は講師が教室内を回っています。その場で声をかけてください。

今日の持ち帰り

午前で身についたのは、他人が書いたコードに手を入れるときの順序です。読ませてから頼む。うまくいかないときは、モデルより先に渡した情報を疑う。この2つが午前の結論です。AIが出した数字は人が1つだけ検算する、という手癖も演習3で作りました。

午後で身についたのは、毎回の前置きをファイル1枚に移す方法と、テスト設計の型です。計画・設計・実行・報告の4工程のどこでも、AIが出すのは候補で、通す基準を書くのは自分です。ケース表の価値は行数ではなく、期待結果が判定できるかで決まります。

明日以降どこで使えるかは、既存改修のときの読ませる依頼文、チームで頼み方をそろえたいときの規約ファイル、テスト設計のときの4段構造とケース表の列見出し、報告書をブラウザの印刷でPDFにする経路の4つです。PDFの経路はテスト以外の報告書でもそのまま使えます。

次回持参するもの

tpro-work フォルダ一式です。中に index.html、.github/copilot-instructions.md、testcases.md、data.csv が入っていることを画面で確認してください。Day3ではこのケース表を単体テストとE2Eテストに落として、ブラウザの中で実際に動かします。セキュリティの観点も足します。

model-memo.txt、prompt-memo.txt、data-memo.txt、report.md、report.html、report.pdf は自分で読み返す用です。持ってこなくてもDay3は進みます。

Day1で受け取った lib フォルダは今日は開きませんが、Day3で使うので消さずに残しておいてください。

守っていただくこと

チャットに入れてよいのは、配布の作業記録アプリの中身と、自分で書いた文章だけです。客先の実際の仕様書やコードは開かず、貼らないでください。次の4つを守ってください。

1. 客先名、製品名、プロジェクト名、取引先名は、業界や工程のレベルに丸めて書きます
2. 実データ、実ファイル、スクリーンショットは使いません
3. 数値や仕様は具体値を避け、目的、課題、期待効果の粒度で書きます
4. 提出前に、社外に出ても問題ない内容かをご自身で確認します

今日の提出物はありません。提出が発生する場合も、出せるのは言語化したテキストだけです。コードと実データは提出できません。

このアプリのデータはすべて架空のものです。