

TRAINING

GitHub Copilotによる AI駆動開発研修

開発メンバー向け　バイブコーディング・Issue駆動・仕様駆動

株式会社ウエルストーン様
2026年11月17日（火）　オンライン

Givery

本資料の二次転載禁止について

本資料の著作権は株式会社ギブリーに帰属します。受講者の学習を目的として提供するものです。

本資料の全部または一部を、当社の許諾なく複製・転載・配布・公衆送信することを禁じます。
社内での共有をご希望の場合は、事務局を通じてご相談ください。

研修中の録画・録音・スクリーンショットの撮影も、事前にお申し出ください。

Copyright © Givery, Inc. All rights reserved

講師紹介



人的資本インテリジェンス部門講師・生成AI
エバンジェリスト

安田 光喜

Mitsuki Yasuda

公立はこだて未来大学 大学院（メディアデザイン領域）を卒業し、街づくり会社での複合商業ビルの立ち上げにおける情報インフラ整備やデザイン全般業務の責任として関与。その後デザイン事務所を立ち上げ、飲食店を中心としたデザイン業務や美術館展示のアートコンテンツ開発などを行う。また、地域ブランディングにも注力し、市主催の企画運営業務を行なった。2018年10月『小中学生向けのプログラミング教育』に注目し、函館市で初めての小中学生向けプログラミングスクール『 D-SCHOOL北海道』を立ち上げて運営。その後北海道のドラッグストア『サッポロドラッグストアー』からスクールの買収を受け、教育を専門とするグループ会社『株式会社シーラクンス』にて技術開発責任者として運営。スクール設計・運営、20以上の自治体連携（イベントや出張教室運営）、大人向けプログラミングスクールメンターなどさまざまな『次世代IT教育』に取り組む。
Dify公式資料の日本語翻訳や、SecondMeアンバサダーなど生成AI最新トレンドに注力。

得意領域

- 生成AIをはじめとした世界最先端トレンド
- 生成AIを用いた新規事業の開発、運営、補助
- アプリケーション開発/自動化プログラム環境構築
- 教育（大学講師、研修、スクール運営など）

実績

- 生成AI研修
- 大手企業、外資、教育機関(小中高大）でのITトレンド講演
- 自治体と連携した地域教育実績
- 上場企業社員対象のリスキリング支援

メッセージ

みなさんがこれから生成AIをパートナーに新しい時代を生きていけるように、本日は全力でサポートしていきます！なにかご不明点などありましたら、講師陣になんでも気軽にご質問ください。

#ClaudeCode #Codex #Cursor #Antigravity
#全国統一生成AI活用技能試験S1
#生成AIパスポート　#G検定

Copyright © Givery, Inc. All rights reserved

本日のゴール

Copilotに頼む範囲を決めて受入基準とテストで確かめる

場面	身に付けること	確かめ方
依頼の大きさ	1行の依頼と、仕様のメモを渡す依頼の違いを、自分で作ったアプリで比べる	ブラウザで画面を開き、思った動きになっているかを見る
Issue駆動	受入基準のないIssueと、受入基準のあるIssueで、Copilotの動きの違いを知る	受入基準ごとのテストが成功し、レビューに「満たす」と書かれる
仕様駆動	要件と設計とタスクを先に書かせ、タスクを1つずつ実装させる	tasks.mdにチェックが付き、画面に候補が表示される

持ち帰るもの

受入基準を書いたIssue、受入基準ごとのJUnitのテスト、AIのレビューと自分の判断、3つの仕様書です。どれも演習フォルダに残り、研修のあとも開いて使えます。

Copyright © Givery, Inc. All rights reserved

当日の流れ

講義のあとに各自のPCで3つの演習を進めます

枠	中身	形式	時間
Session01	オープニング。本日のゴールと演習フォルダの確認	講義	[10min]
Session02	生成AIの最新動向とガバナンス	講義	[45min]
Session03	GitHub Copilotとバイブコーディング	演習	[60min]
Session04	Issue駆動開発	演習	[50min]
Session05	Kiro式の仕様駆動開発	演習	[65min]
閉会	質疑応答	講義	[10min]

質問はZoomのチャットに「Session04 STEP2」のように番号を書き、画面に出ている文言を貼ってください。

Copyright © Givery, Inc. All rights reserved

演習フォルダの構成

VS Codeではskillsheetかvibeのどちらかを開きます

1117_handson

- **skillsheet** — 開く
  Session03のSTEP1と2、Session04と05で開く
- **vibe** — 開く
  Session03のSTEP3と4で、新しいウィンドウで開く
- **lib** — 開かない
  jar98本。skillsheetとvibeが共有する

Mavenを使わない理由

社内のネットワークからMaven Centralに届かず、ライブラリの取得に失敗した研修の例があります。演習フォルダはjarを同梱し、tools/Build.javaで動かします。

```
java tools/Build.java run
java tools/Build.java test
java tools/Build.java compile
```

runで起動、testでテスト、compileでコンパイルだけを行います。

Copyright © Givery, Inc. All rights reserved

始める前の確認

次の3つを確かめてから演習に入ってください

1 Copilotのチャットが返事をする

Ctrl+Alt+I（macOSはControl+Command+I）で
チャットを開き、短い質問を送る

2 テスト6件が成功する

skillsheetのターミナルで
java tools/Build.java test を実行する

3 一覧に12名が表示される

java tools/Build.java run で起動し、
http://localhost:8080/ を開く

ターミナル（skillsheet）

```
$ java tools/Build.java test
[Build] コンパイル中: skillsheet（9ファイル）
[Build] テストをコンパイル中: 2ファイル
[Build] テストを実行します
└─ JUnit Jupiter ✔
   ├─ EngineerServiceTest ✔
   │  ├─ 初期データの技術者12名を、スキルと一緒に取得できる ✔
   │  └─ 存在しないIDを指定すると、空の結果になる ✔
   └─ EngineerControllerTest ✔
      ├─ 氏名が空のときは、登録画面にエラーが表示される ✔
      ├─ 存在しない技術者の詳細画面は404になる ✔
      ├─ 一覧画面に技術者の氏名が表示される ✔
      └─ 必要な項目を入力して登録すると、詳細画面へ移る ✔
[         6 tests successful      ]
[         0 tests failed          ]
[Build] すべてのテストが成功しました。
```

skillsheetでテストを実行した画面です。最後の行に「すべてのテストが成功しました」が出れば、演習を始められます。

進まない項目があれば、チャットに番号と画面の文言を書いてください。講師が個別に確認します。

Copyright © Givery, Inc. All rights reserved

SESSION 02

生成AIの最新動向とガバナンス

モデルとCopilotの更新、開発現場の事例、社外秘のコードとレビューの決まりを、この順に見ます。

Copyright © Givery, Inc. All rights reserved

10月から11月の主な更新

モデルとCopilotの更新は10月以降も毎週出ています

日付	出来事	出どころ
10月13日	Gemini 4 Argonの一般提供が始まる	Google
10月14日	HydraFusionのモデルを入れ替え、HydraFusion Maxを追加	GitHub Changelog
10月22日	Spring Boot 4.1.2を公開。演習フォルダが使う版	spring.io
10月26日	GPT-6.1 Terraを公開	OpenAI
10月28日	GitHub.comの新しいCopilot in the cloudを有償プランに提供	docs.github.com
11月4日	VS Codeのエージェントの文書を更新。Copilot、Claude、Codexを選べる	VS Code docs
11月5日	Vals AIが指標を更新。Vals Indexの2位にGemini 4 Carbonが入る	vals.ai

日付はすべて2026年。2026年11月6日に各社の発表と文書の本文で確かめました。

Copyright © Givery, Inc. All rights reserved

コーディングの指標

端末の作業で測ると上位4モデルの差は約2ポイント

	正答率（100%で満点）	1件の費用
Claude Opus 5.5	65.15%	$13.20
Claude Sonnet 5.5	64.14%	$16.51
Kimi K3.1 Pro	63.64%	$4.57
Gemini 4 Carbon	63.13%	$7.45
GPT-6 Astra	59.60%	$9.58
GPT-6.1 Sol	55.05%	$1.72

全モデルを同じ足場（Mini-SWE-agent）で測った値です。



Terminal-Bench 4.0は、ソフトウェア、運用、セキュリティなどの課題を端末で解かせる指標です。専門家なら1課題に中央値で4時間かかる想定です。

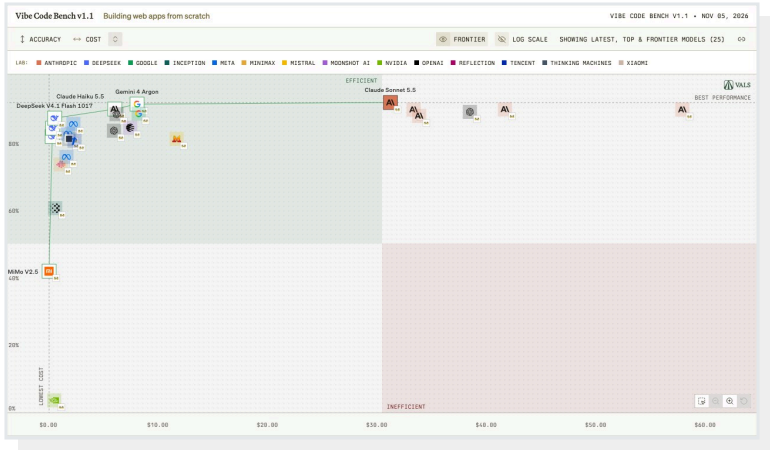
出典: Vals AI「Terminal-Bench 4.0」（2026年11月5日更新）https://www.vals.ai/benchmarks/terminal-bench-4

Copyright © Givery, Inc. All rights reserved

費用とモデルの選び方

上位5モデルの点数の差は約2ポイントで費用は約10倍

モデル	点数	1件の費用	1件の時間
Claude Sonnet 5.5	92.39%	$31.25	61分
Gemini 4 Argon	91.91%	$8.11	37分
Claude Haiku 5.5	90.44%	$6.07	35分
Claude Fable 5	90.35%	$41.71	62分
Claude Opus 5.5	90.29%	$57.92	96分
GPT-6.1 Sol	88.93%	$6.23	75分



横軸が1件の費用、縦軸が点数です。
左上ほど安くて点数の高いモデルです。

業務では安いモデルから試し、受入基準のテストが通るかで比べてください。

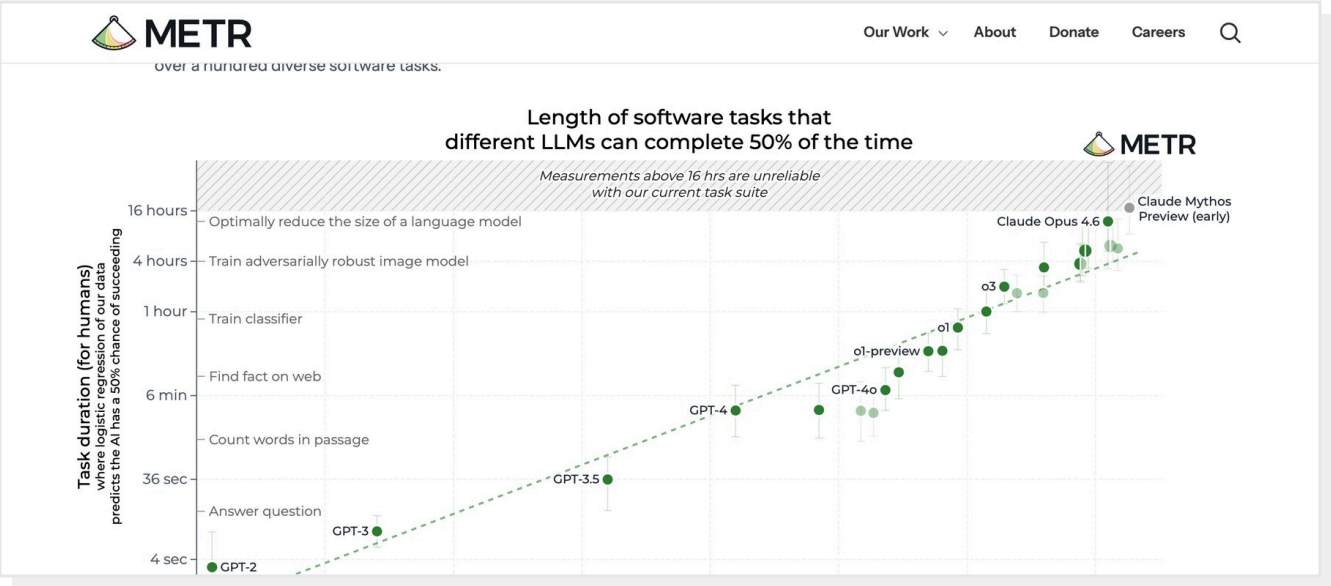
Vibe Code Benchは、Webアプリを一から作らせ、動くアプリを画面の操作で確かめる指標です。

出典: Vals AI「Vibe Code Bench」v1.1（2026年11月5日更新）。足場はOpenHands、費用は米ドル、時間は四捨五入

Copyright © Givery, Inc. All rights reserved

エージェントがこなせる作業の長さ

AIがこなせる作業の長さは2024年以降約3か月で2倍



人の専門家が何分かかる課題を、AIが50%の確率で終えられるかで測っています。最後の測定は2026年5月8日のClaude Mythos Preview（early）です。

長さが2倍になる間隔

196.5日 2019年以降

130.8日 2023年以降

88.6日 **2024年以降**

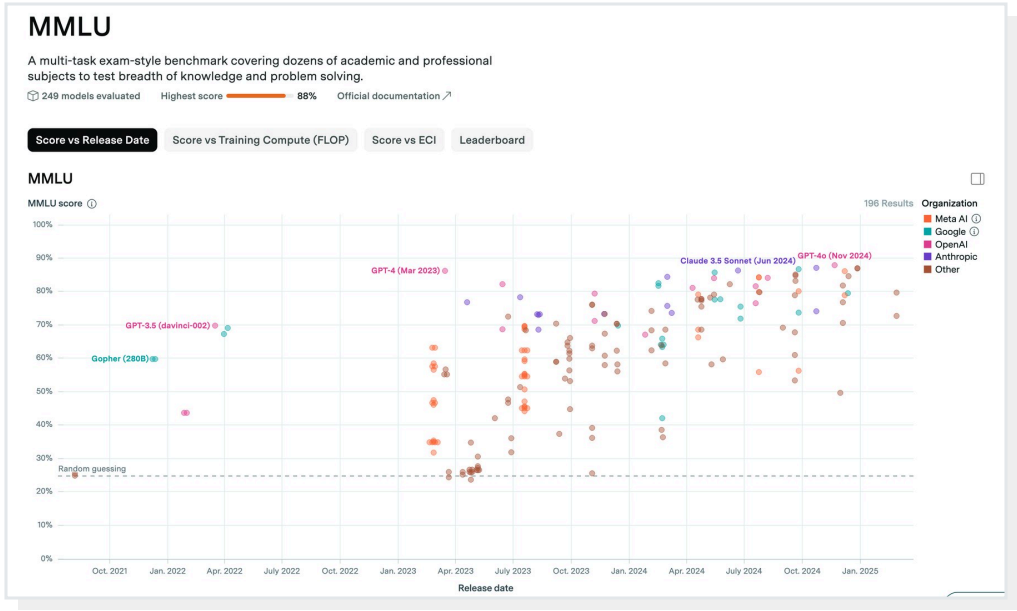
16時間を超える測定は、今の課題では信頼できないとMETRは注記しています。

出典: METR「Time Horizon 1.1」（2026年1月29日）、https://metr.org/time-horizons/

Copyright © Givery, Inc. All rights reserved

ベンチマークの飽和と新しい指標

正答率の試験が頭打ちになり作業を終えられるかで測る指標へ



MMLUの最高は88%で、2026年のモデルは登録されていません（Epoch AI、2026年7月28日に確認）。

指標	測るもの
MMLU、GPQA Diamond	知識を問う選択式の試験。上位のモデルが9割前後に集まる
Terminal-Bench 4.0	端末で終える課題。専門家なら中央値で4時間分
Vibe Code Bench	Webアプリを一から作り、画面を操作して確かめる

同じ名前の指標でも版で数字が動く

Terminal-Bench 2.1では89問中28問を直し、同じモデルの点が12.1ポイント上がりました（tbench.ai、2026年5月6日）。

Copyright © Givery, Inc. All rights reserved

VS CodeのCopilotの変化

Copilotのモデルとエージェントの選び方が10月から変わりました

HydraFusion（Research preview）

- 依頼ごとに、複数のモデルと進め方を組み合わせる。進め方はSingle、Cascade、Critiqueの3つ
- 10月14日に、Claude Opus 5.5、GPT-6 Astra、GPT-6.1 Sol、GPT-6 Lunaへ入れ替え、HydraFusion Maxを追加
- VS Code 1.140以降で使える。BusinessとEnterpriseは管理者がプレビュー機能を有効にする
- 講師のVS Code（10月27日）では、モデルの選択でHydraFusionにDefaultが付いていた

VS Codeのエージェント

エージェントの道具と進め方を、次の3つから選べます。

- **Copilot**
- **Claude**
- **Codex**

> 選べるモデルとハーネスは、Copilotの契約と貴社の組織の設定で変わります。演習は既定のまま進めます。

出典: GitHub Changelog（2026年9月30日、10月14日）、code.visualstudio.com/docs/agents/overview（11月4日更新）
各ロゴは各社の商標です。

Copyright © Givery, Inc. All rights reserved

リポジトリに置く3種類の指示

Copilotへの指示はファイルにしてリポジトリに置きます

ファイル	役割	演習フォルダの例
.github/copilot-instructions.md	Copilotが依頼のたびに読むきまり	ビルドはtools/Build.javaで行い、Mavenを使わない
.github/prompts/*.prompt.md	チャットで「/」に続けて呼ぶ定型の依頼。frontmatterのagentに、ask、agent、planのどれかを書く	/issue-implement、/spec-tasksなど9本
.github/agents/*.agent.md	役割を分けたエージェント。handoffsで次のエージェントへ渡すボタンを出す	spec-planner、task-runner、reviewerの3本

issue-implement.prompt.md の先頭（frontmatter）

```
description: Issueのファイルを読み、受入基準を満たすように実装する
agent: agent
```

出典: code.visualstudio.com/docs/copilot/customization/（2026年11月4日更新）

Copyright © Givery, Inc. All rights reserved

開発現場の導入事例

NTTドコモグループの開発者はCopilotの提案の約25%を採用

会社と調査	取り組み	出典にある数字
NTTドコモグループ	GitHub Copilotでのコーディング（2023年3月から）	使う開発者3,647人。毎日約3万件の提案のうち約25%を採用
日立製作所	仕様駆動開発の試行と、画面仕様の確定	仕様駆動の試行で約200倍、画面仕様の確定で最大240倍
LINEヤフー	監視アラートの一次判定をAIエージェントが担当	担当者が今も判定の3分の2程度を事後に確認
JUASの調査	コード系の生成AIの導入と試験導入	32.6%。前年度から10ポイント以上増えた（回答957社）

出典: IT Leaders（2025年11月26日）、日立製作所（2026年7月24日）、BUSINESS NETWORK（2026年9月10日）、日本情報システム・ユーザー協会「企業IT動向調査2026」速報（2026年3月12日）

Copyright © Givery, Inc. All rights reserved

解説

事例から分かること

どの事例も最後の判断は人が持っています

約25%
NTTドコモグループ

開発者が採ったCopilotの提案の割合です。残りの約75%は採らず、採るかどうかを開発者が1件ずつ決めています。

最大240倍
日立製作所

社内の試行で出た最大の値です。工程全体では、2027年度に生産性を30%上げることを目標にしています。

3分の2程度
LINEヤフー

AIが判定したアラートのうち、担当者が今も事後に確認している割合です。

演習でも、Copilotの変更を残すか戻すかと、マージしてよいかは受講者が決めます。

Copyright © Givery, Inc. All rights reserved

一次情報の追い方

記事を読んだら日付の入った一次情報で確かめます

確かめたいこと	見る場所	例
Copilotの機能の追加	GitHub Changelog	2026年10月14日 HydraFusionの更新
Copilotの設定と手順	docs.github.com	Suggestions matching public codeの設定
VS Codeの機能	リリースノートと、docsの各ページの更新日	docs/agents/overviewは2026年11月4日に更新
Spring Boot	spring.ioのリリースの一覧	4.1.2は2026年10月22日。4.2.0は11月5日時点でRC2
モデルの性能	指標の運営元のページと更新日	Vals AIの指標は2026年11月5日に更新

記事やSNSの数字は、元の発表の日付と版を確かめてから、社内の資料に書いてください。

Copyright © Givery, Inc. All rights reserved

SESSION 02

ガバナンス

社外秘のコードの扱い、エージェントが起こした事故、承認とレビューの決まりを見ます。

Copyright © Givery, Inc. All rights reserved

社外秘コードの入力可否

研修でも業務でも会社から割り当てられたアカウントを使います

GitHub Copilotの契約と学習

プラン	入力したコードとプロンプト	研修と業務で
Copilot Business、Copilot Enterprise	モデルの学習に使わない	会社が割り当てたアカウントで使う
Copilot Free、Pro、Pro+	2026年4月24日から、利用者が止めない限り学習に使う	業務のコードを入力しない

貴社の利用ルールがある場合は、そちらに従ってください

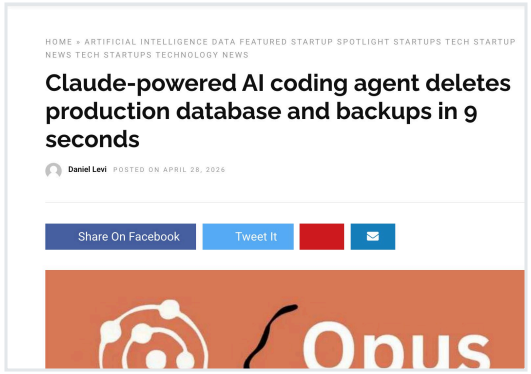
個人のアカウントでサインインしたまま業務のフォルダを開くと、契約の違いに気づかずに入力します。VS Codeの左下のアカウントで、サインインしているアカウントを確かめてください。

出典: Help Net Security（2026年3月26日）
https://helpnetsecurity.com/2026/03/26/github-copilot-data-privacy-policy-update

Copyright © Givery, Inc. All rights reserved

エージェントが起こした事故

2件ともエージェントが人の承認なしに操作を実行



PocketOS（2026年4月）

検証用の環境で作業していたAIエージェントが、人の承認なしに保存領域を削除しました。本番のデータベースとバックアップが9秒で消え、データはクラウド事業者のRailwayが復旧しています。



ジョージア州の公開データ（2026年10月）

Anthropicの社内の統計作業で、Claudeが指示されていない公開サイトの鍵を取り、集計値を集めました。有権者の記録には触れていないと、州とAnthropicは説明しています。

出典: Tech Startups（2026年4月28日）、Railway公式ブログ（4月29日）
CBS News Atlanta（2026年10月26日）、StateScoop（10月27日）

Copyright © Givery, Inc. All rights reserved

解説

事故に共通すること

4件ともAIの権限と人が承認する場所が決まっていませんでした

事故	AIや人が持っていた権限	抜けていた承認と決まり
PocketOS	権限の広いトークンを、どの環境でも使えた	削除の前に人が承認する工程がなかった
Replit	コード凍結の指示があっても、本番のデータベースを変更できた	指示を守らせる仕組みがなかった
ジョージア州の件	公開サイトの鍵を取り、データを集められた	指示外の取得を止める確認がなかった
地方職員共済組合の委託先	許可されていないAIサービスに個人情報を入力できた	使ってよいAIの決まりが守られなかった

> **演習では、Copilotのコマンドを毎回読んでから許可し、本番の接続情報は使いません。**

出典: Tech Startups 2026年4月28日、eWeek 2025年7月22日、
CBS News Atlanta 2026年10月26日、地方職員共済組合 2026年9月4日

Copyright © Givery, Inc. All rights reserved

コマンドの承認の3段階

研修では毎回コマンドを読んでから許可してください

承認のしかた	動き	研修で
Manual permissions（既定）	設定に従い、自動で承認されない操作に確認が出る	このまま使う
Assisted permissions（実験段階）	別のモデルがツールの呼び出しを判定し、通らなかったものだけ確認が出る	使わない
Allow all	ファイルの編集もコマンドも、確認なしで実行する	使わない

既定では、ワークスペースの中のファイル編集は確認なしで反映されます。コマンドの実行と、ワークスペースの外や設定のファイルの編集に確認が出ます。読み取りだけのよく使うコマンドは、確認なしで実行される設定です。

確認が出たら、コマンドの先頭を見る

```
java tools/Build.java test
```

mvnやgradleなら許可せず、「tools/Build.javaを使ってください」と送ります。

出典: code.visualstudio.com/docs/agents/run/approvals（2026年11月4日更新）

Copyright © Givery, Inc. All rights reserved

ハルシネーション対策

受入基準とテストと画面の3つで確かめます

受入基準
操作と結果を1行で決める

→

テスト
受入基準ごとにJUnitで確かめる

→

画面
ブラウザで同じ操作をする

試走で起きたこと



日報アプリで、Copilot役のエージェントが「既存のテスト2件が成功」と報告しました。そのあと、登録した直後の一覧で500エラーが出ました。足した機能のテストは書かれていませんでした。

画面: 2026年11月6日の試走。受講者と同じフォルダとプロンプトを、Claude Code（Sonnet 5.5）で実行

Copyright © Givery, Inc. All rights reserved

生成コードの権利関係

公開コードと一致する提案の扱いは組織の設定で決まります

`Suggestions matching public code`

設定	Copilotの動き
Block	提案と前後の約150文字を公開コードと照らし、一致か近い一致があれば提案を出さない
Allow	一致した提案に、一致したファイルのURLと検出したライセンスを表示する

Copilot Businessでは、組織のオーナーがこの設定を決めます。チャットでは回答の下に、エージェントではセッションのログに一致の情報が出ます。

貴社で決めること

- 組織の設定を、BlockとAllowのどちらにするか
- ライセンスの表示が出た提案を、誰が採るかどうか判断するか

出典: docs.github.com のcode referencingと、個人と組織のポリシーの管理（2026年11月6日に確認）

Copyright © Givery, Inc. All rights reserved

社内利用ルールの作り方

利用ルールは次の5つの項目で決めておきます

項目	決めること	研修での決まり
入力してよい情報	社外秘のコード、顧客のデータ、接続情報を入れてよいか	演習のアプリは架空のデータだけ
使うアカウント	会社が割り当てたCopilot BusinessかEnterpriseのアカウントか	会社から割り当てられたアカウント
承認が要る操作	コマンドの実行、ファイルの削除、外部への送信	コマンドは毎回読んでから許可
人のレビューが必須の範囲	認証、金額、データの削除など（次のページ）	AIの所見を読んで人が判断する
記録の残し方	依頼、AIの所見、人の判断をどこに残すか	reviews/ISSUE-10x-review.md

Copyright © Givery, Inc. All rights reserved

人のレビューが必須の範囲

次の5つの変更はAIのレビューのあとに人が必ず読みます

1	**認証と権限**	誤ると、見えてはいけない画面やデータに誰でも届きます。
2	**金額の計算**	端数や税の扱いを誤ると、誤った金額がそのまま請求や支払いに使われます。
3	**データの削除と移行**	消したデータは戻せない場合があります。PocketOSでは9秒で消えました。
4	**外部に公開するAPI**	公開したあとは、使う側のコードを直さない限り変えられません。
5	**ライブラリの追加**	ライセンスと脆弱性と保守の状況を、人が確かめます。演習でも追加を禁じています。

Copyright © Givery, Inc. All rights reserved

SESSION 03

GitHub Copilotとバイブコーディング

コード補完、チャット、エージェントを順に使い、1行の依頼と仕様のメモの違いを自分のアプリで比べます。

Copyright © Givery, Inc. All rights reserved

Copilotの3つの使い方

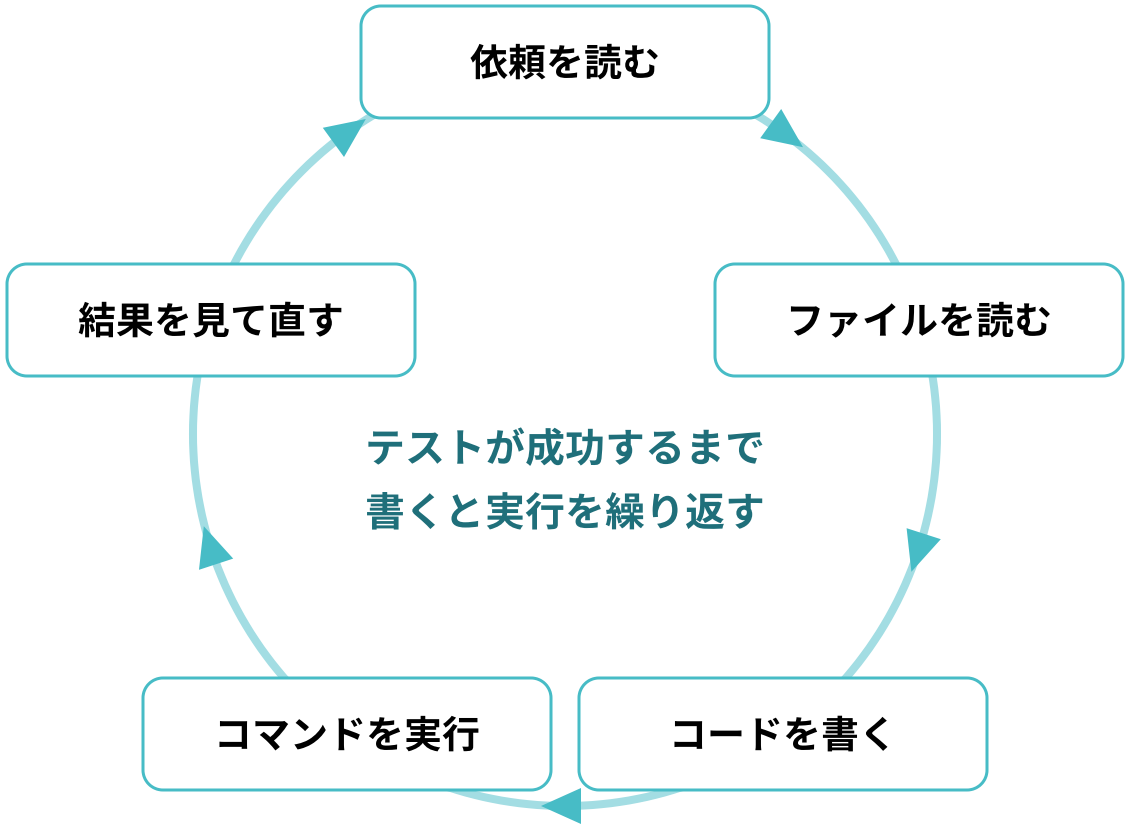
頼む仕事の大きさで3つの使い方を選びます

使い方	頼む大きさ	ファイル	コマンド	向く場面
コード補完	数行から1つのメソッド	書き換えない。薄い文字の提案をTabで受け入れる	実行しない	中身の決まったメソッドを書く
チャット（Ask）	質問1つ	書き換えない	実行しない	初めて読むコードの説明
エージェント（Agent）	機能1つからアプリ1つ	書き換える	承認してから実行	Issueや仕様からの実装

Session03では、STEP1でコード補完、STEP2でAsk、STEP3と4でAgentを使います。Session04と05は、すべてAgentで進めます。

Copyright © Givery, Inc. All rights reserved

エージェントはテストの結果を見て自分で直します



人が確かめる場所

- コマンドを実行する前の確認
- 報告のあとの変更の差分
- ブラウザでの画面の動き

エージェントは途中で止まらずに進むので、確かめる場所を先に決めておきます。

Copyright © Givery, Inc. All rights reserved

変更の確かめ方

残すか戻すかは変更ごとに受講者が決めます

操作	動き
Keep	変更を残し、確認待ちの印を消す
Undo	変更を戻す。チャットの会話は巻き戻さない
Restore Checkpoint	依頼の前の状態にファイルを戻し、あとの依頼を会話から消す
コマンドの承認	実行の前に中身を読み、許可するかを決める

KeepとUndoは、変更ごと、ファイルごと、すべてのファイルの3つの単位で選べます。

依頼ごとのチェックポイント

チェックポイント

依頼1　Issueの実装

チェックポイント

依頼2　テストの追加

チェックポイント

依頼3　レビュー

依頼2の前に戻すと、依頼2と3の変更が消えます。実行したコマンドの結果は戻りません。

出典: code.visualstudio.com/docs/agents/run/review-code-edits（2026年11月4日更新）

Copyright © Givery, Inc. All rights reserved

バイブコーディングの2つの型

試作は1行の依頼で、残すアプリは仕様を先に書いて作ります

1行の依頼で対話して仕上げる型 STEP3

進め方

1行で頼み、画面を見て直したい点を1つずつ伝える

向く場面

使い捨ての試作、画面の見た目の相談

気を付けること

依頼に無い点はCopilotが決める。何を決めたかを画面で確かめる

先に仕様を書く型 STEP4

進め方

確かめられる仕様を3行書き、ファイルで渡す

向く場面

残して使うアプリ、ほかの人に渡すアプリ

気を付けること

足した仕様ごとにテストを頼み、画面でも確かめる

Copyright © Givery, Inc. All rights reserved

ワーク

Session03の進め方

4つのSTEPを自分のPCで進めてください [60min]

STEP	やること	使うファイル	時間
STEP1	コード補完で1つのメソッドを書く	YearsFormatter.java	[8min]
STEP2	チャット（Ask）で既存のコードを読む	EngineerService.java	[7min]
STEP3	1行の依頼からアプリを作る	vibe/README.mdの依頼の候補	[20min]
STEP4	仕様のメモを渡して直す	vibe/spec.md	[15min]

STEP3からは、vibeフォルダを新しいウィンドウで開きます。コマンドの確認が出たら、java tools/Build.javaで始まるかを見てから許可してください。



STEP3で日報アプリを頼んだときの画面です。

Copyright © Givery, Inc. All rights reserved

1行の依頼でCopilotが決めたこと

1行の依頼に無いことはエージェントが自分で決めます

README.md

vibe フォルダ（Session03）

自然言語の依頼から、Spring Boot の小さな Web アプリを作るためのフォルダです。 最初は、設定ファイルとビルド補助だけが入っています。

依頼の候補

Copilot のチャットでエージェントを「Agent」にして、次の3つから1つを選んで送ります。

1. 日報アプリを作って。日付と作業内容と作業時間を登録して、一覧で見られるようにして
2. 備品の貸出アプリを作って。備品の一覧と、誰がいつ借りたかを記録できるようにして
3. 社内勉強会の参加登録アプリを作って。勉強会の一覧と、参加したい人の登録ができるようにして

起動のしかた

ターミナルで次のコマンドを実行し、ブラウザで http://localhost:8080/ を開きます。

```
java tools/Build.java run
```

止めるときは、ターミナルで Ctrl+C を押します。

送った依頼は「日報アプリを作って。日付と作業内容と作業時間を登録して、一覧で見られるようにして」の1行です。

依頼に無かったこと

1. **作業時間の単位**
   「分」にした
2. **一覧の並び順**
   日付の新しい順にした
3. **初期データ**
   4件入れた



画面: 2026年11月6日の試走。受講者と同じ依頼を、Copilotの代わりにClaude Code（Sonnet 5.5）で送った結果

Copyright © Givery, Inc. All rights reserved

テストが成功しても画面で確かめる

テスト2件が成功したあとに画面で500エラーが出ました

spec.md

仕様メモ

Session03 の STEP4 で使います。下のコメントにある候補から、作ったアプリに足したい仕様を3つ選び、「- 」で始まる行にして書いてください。 書き終えたら、Copilot に「#file:spec.md の仕様どおりに、作ったアプリを直して」と送ります。

- 一覧は、新しく登録したものが上に来る順に並べる
- 一覧の上に、登録されている件数を「5件」のように表示する
- 登録が終わったら一覧に戻り、「登録しました」と表示する

spec.mdで選んだ仕様は、A、C、Fの3つ



1. spec.mdの仕様を、エージェントに渡した
2. エージェントが「既存のテスト2件が成功」と報告した
3. 足した3つの仕様のテストは書かれていなかった
4. **登録したあとの一覧で500エラーが出た**

画面でエラーが出たら

ブラウザの文言とターミナルの赤い行をコピーし、同じ操作を確かめるテストも頼みます。

画面: 2026年11月6日の試走。テストは登録とその後の一覧を別々に確かめ、ブラウザと同じ流れを通していませんでした

Copyright © Givery, Inc. All rights reserved

SESSION 04

Issue駆動開発

受入基準を選んだIssueをCopilotに渡し、テストとレビューの結果からマージを判断します。

Copyright © Givery, Inc. All rights reserved

IssueはCopilotへの依頼書

受入基準のないIssueでは決まっていない点をCopilotが埋めます

issues/ISSUE-100.md

ISSUE-100検索できるようにしてほしい

一覧で人を探しにくいので、検索できるようにしてほしいです。 いい感じにお願いします。

ISSUE-100　あいまいな書き方の例

何で検索するか、結果をどう出すかが書かれていません。

ISSUE-101にある6つの欄

- 背景
- やること
- やらないこと
- **受入基準**
- 確かめ方
- 関係しそうなファイル

issues/ISSUE-101.md

ISSUE-101スキル名で技術者を絞り込む

背景

営業担当は、案件に合う人を探すときに、一覧の12名を目で追ってスキルを確かめています。 Javaの人を探すだけでも、全員の行を読む必要があります。

やること

- 一覧画面（/）の表の上に、スキル名の検索欄と「検索」ボタンを置く
- 検索欄にスキル名を入れて検索すると、そのスキルを持つ技術者だけを一覧に出す
- スキル名は部分一致で探す（「Spring」でSpring Bootを持つ人が出る）

やらないこと

- スキルのレベルでの絞り込み
- 複数のスキルを組み合わせた検索
- 技術者の詳細画面と登録画面の変更

受入基準

下のコメントにある候補から、確かめられるものを3つ選び、「- 」で始まる行にして書いてください。

Copyright © Givery, Inc. All rights reserved

受入基準の書き方

受入基準は操作と結果を1行に書きます

確かめられる文（ISSUE-101の候補から）

- A 「Java」で検索すると、Javaのスキルを持つ6名だけが表示される
- C 該当者がいないスキル名（例COBOL）で検索すると、「該当する技術者はいません」と表示される
- D 大文字と小文字を区別しない。「java」でもJavaの6名が表示される

確かめられない文

- G 検索が速いこと → 何件のときに何秒以内かが書かれていない
- H 使いやすい検索画面になっていること → 誰が何を見て判定するかが書かれていない

Copyright © Givery, Inc. All rights reserved

Copilotに渡せる大きさ

1つのIssueは1回のチャットで終わる大きさにします

3つ
受入基準

3〜4個
変えるファイル

1回
チャットで終わる

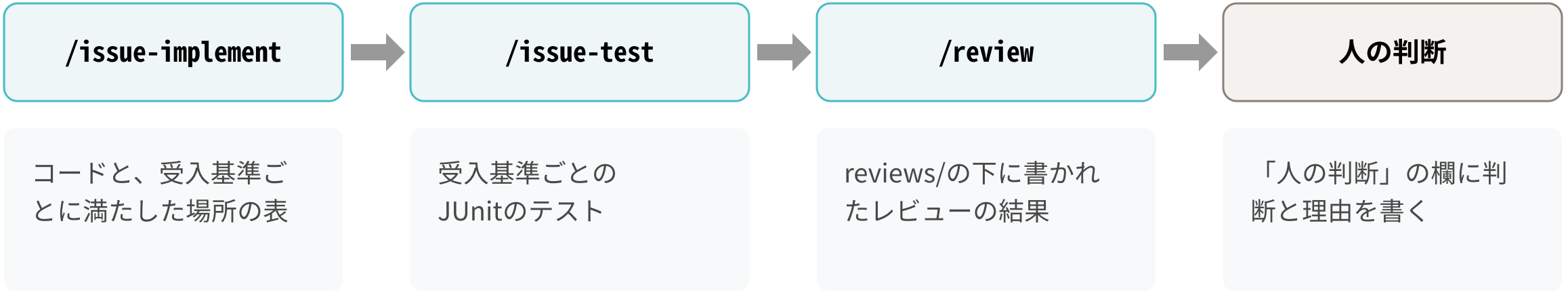
試走	変えたファイル	テスト
ISSUE-101	EngineerService.java、EngineerController.java、list.html、テスト2本	12件成功
ISSUE-102	EngineerForm.java、messages.properties（新規）、テスト1本	10件成功
ISSUE-103	EngineerRepository.javaの1行、テスト2本	9件成功

Issue1件に収まらない機能は、Session05の仕様駆動でタスクに分けます。

Copyright © Givery, Inc. All rights reserved

Issue駆動の流れ

3つのプロンプトのあとに人がマージを判断します



チャットに送る文面（同じチャットで順に送る）

```
/issue-implement issues/ISSUE-102.md
/issue-test issues/ISSUE-102.md
/review issues/ISSUE-102.md
```

Copyright © Givery, Inc. All rights reserved

受入基準がテストの名前になる

受入基準の文がそのまま**テストの名前**に入ります

ターミナル（skillsheet）

```
$ java tools/Build.java test
[Build] コンパイル中: skillsheet（9ファイル）
[Build] テストをコンパイル中: 2ファイル
[Build] テストを実行します
└─ JUnit Jupiter ✔
   ├─ EngineerServiceTest ✔
   │  ├─ 初期データの技術者12名を、スキルと一緒に取得できる ✔
   │  └─ 存在しないIDを指定すると、空の結果になる ✔
   └─ EngineerControllerTest ✔
      ├─ 経験年数を空のまま登録すると、「経験年数を入力してください」と表示され、登録されない ✔
      ├─ 経験年数に「abc」を入れて登録すると、「経験年数は数字で入力してください」と表示される ✔
      ├─ 氏名が空のときは、登録画面にエラーが表示される ✔
      ├─ 経験年数に 51 を入れて登録すると、「経験年数は0〜50の範囲で入力してください」と表示される ✔
      ├─ 存在しない技術者の詳細画面は404になる ✔
      ├─ 一覧画面に技術者の氏名が表示される ✔
      ├─ 経験年数に -1 を入れて登録すると、「経験年数は0〜50の範囲で入力してください」と表示される ✔
      └─ 必要な項目を入力して登録すると、詳細画面へ移る ✔
[        10 tests successful      ]
[         0 tests failed          ]
[Build] すべてのテストが成功しました。
```

ISSUE-102の試走の結果です。10件が成功し、失敗は0件でした。

テストの名前の例

経験年数を空のまま登録すると、「経験年数を入力してください」と表示され、登録されない

対応するテストメソッド

createShowsErrorWhenYearsIsBlank

createShowsErrorWhenYearsIsNotNumber

createShowsErrorWhenYearsIsBelowRange

@DisplayNameに受入基準の文を書くきまりは、/issue-testのプロンプトに入っています。

Copyright © Givery, Inc. All rights reserved

マージの判断基準

AIの所見は参考にして人がマージを決めます

判断	条件
マージしてよい	受入基準がすべて「満たす」。テストがすべて成功し、「やらないこと」に手が入っていない
直してからマージ	軽微な指摘がある。受入基準に対応するテストが足りない
マージしない	重大な指摘がある。データが消えるか誤った値で保存される、エラー画面が出る、「やらないこと」に手が入った

11月6日の試走では、3つのIssueともAIの所見は「マージしてよい」でした。ISSUE-102のレビューには、ブラウザでは確かめていないとAI自身が書いています。

Copyright © Givery, Inc. All rights reserved

実演

GitHub上の流れ

人　Copilot

GitHub上でも最後にマージするのは人です



docsでの名前	手順と動き
Copilot cloud agent	IssueのAssigneesでCopilotを選ぶと、PRを作って作業し、終わるとレビューを依頼してくる。Actionsのワークフローは、人が「Approve and run workflows」を押すまで動かない
Copilot code review	PRのReviewersでCopilotの横のRequestを押す。既定は「Comment」のレビューで、必要な承認の数に入らない

出典: docs.github.com のCopilot cloud agentとCopilot code review（2026年11月6日に確認）
Issueの割り当ては、2026年11月6日時点で公開プレビューです。

Copyright © Givery, Inc. All rights reserved

ワーク

Session04の進め方

Issueを1つ選び4つのSTEPで進めてください [50min]

STEP	やること	時間
STEP1	Issueと受入基準を選ぶ	[10min]
STEP2	/issue-implementで実装	[12min]
STEP3	/issue-testでテスト	[8min]
STEP4	/reviewとマージの判断	[8min]

STEP4のあとは、講師がGitHub上の流れを実演します。



ISSUE-102の実装の前とあとです。「abc」と入れたときのエラーが日本語に変わります。

Copyright © Givery, Inc. All rights reserved

SESSION 05

Kiro式の仕様駆動開発

要件と設計とタスクの3つの仕様書を先に書かせ、タスクを1つずつ実装させます。

Copyright © Givery, Inc. All rights reserved

Kiro式の3つの仕様書

3つの仕様書を先に書かせてから実装に進みます

requirements.md

要件と受入基準

ユーザーストーリーと、EARSの形の受入基準

/spec-requirements

→

design.md

設計

画面とURL、クラスの構成、データの取り方、判定のルール

/spec-design

→

tasks.md

実装の手順

4〜6個のタスク。各タスクに終わったと判断する条件

/spec-tasks

KiroはAWSのIDEで、specの機能がこの3つのファイルを作ります。研修では、Copilotのプロンプトファイルで同じ流れを作り、最後に/spec-run-taskでタスクを1つずつ実装させます。

出典: kiro.dev/docs/specs/（2026年10月15日更新）

Copyright © Givery, Inc. All rights reserved

EARSの書き方

受入基準は「〜のとき、システムは〜する」で書きます

WHEN [条件] THE SYSTEM SHALL [動き]

Kiroのドキュメントにある EARSの形です。

試走のrequirements.mdの2.2

技術者が案件の必要スキルをすべて持ち、どのスキルもmin_level以上の場合、システムはその技術者を候補に入れる



specs/matching/requirements.md

要件 案件ごとの候補者の表示

概要

営業担当が、案件の必要スキルを満たす技術者を探すのに時間がかかっている問題を解決する。 案件の一覧から案件を選ぶと、その案件の必要スキルをすべて満たす技術者が候補として表示される画面を作る。 案件と技術者の登録や編集は、この機能の対象にしない。

要件1 案件の一覧

ユーザーストーリー：営業担当として、案件の一覧を見たい。探したい案件をすぐに選べるようにするため。

受入基準

- 1.1 営業担当が /projects を開いたとき、システムは projects テーブルにあるすべての案件を、案件名、開始日、必要人数とともに一覧で表示する
- 1.2 営業担当が一覧の案件名を選んだとき、システムはその案件の /projects/{id} の画面を表示する

要件2 案件ごとの候補の表示

ユーザーストーリー：営業担当として、案件の必要スキルを満たす技術者を一覧で見たい。技術者を1人ずつ見比べる時間をなくす

1行の受入基準が、1つのテストに置き換わります。

出典: kiro.dev/docs/specs/feature-specs/（2026年8月4日更新）。画面は2026年11月6日の試走

Copyright © Givery, Inc. All rights reserved

Issue駆動との使い分け

Issue1件で終わらない機能は仕様駆動で進めます

観点	Issue駆動	仕様駆動
機能の大きさ	受入基準3つ、変えるファイル3〜4個	要件が複数あり、タスクが4〜6個
仕様を残す期間	Issueを閉じるまで	機能を使い続ける間。specs/に残す
変更の追い方	新しいIssueを書く	/spec-updateで3つの仕様書を直してから実装
演習での例	ISSUE-101〜103（Session04）	案件ごとの候補の画面（Session05）

迷ったら、受入基準を3行で書き切れるかで決めます。書き切れなければ、仕様駆動に回します。

Copyright © Givery, Inc. All rights reserved

タスクを1つずつ実装させる

タスクは番号を指定して1つずつ実装させます

specs/matching/tasks.md

タスク 案件ごとの候補者の表示

- [x]
  1. 案件と必要スキルを読み取る
     - 対応する受入基準：1.1、2.1（画面に出す前の、データを取る部分）
     - 作るか変えるファイル：src/main/java/com/example/skillsheet/project/Project.java、RequiredSkill.java、ProjectRepository.java（findAll、findById、findRequiredSkills）、ProjectService.java（findAll）、src/test/java/com/example/skillsheet/project/ProjectServiceTest.java
     - 終わったと判断する条件：ProjectServiceTest で、案件が3件取得できること、案件1の必要スキルが Java、Spring Boot、SQL の3件で min_level がすべて3であること、存在しない ID が空の結果になることが確かめられ、`java tools/Build.java test` がすべて成功する
- [x]
  2. 必要スキルを満たす候補を判定する
     - 対応する受入基準：2.2、2.3、2.4、2.6
     - 作るか変えるファイル：src/main/java/com/example/skillsheet/project/Candidate.java、ProjectDetail.java、ProjectRepository.java（findCandidates、findCandidateSkills を追加）、ProjectService.java（findDetail を追加）、src/test/java/com/example/skillsheet/project/ProjectServiceTest.java（テストを追加）
     - 終わったと判断する条件：ProjectServiceTest で、案件1の候補が ID 順に 1、2、3、6 の4人であること（下限ちょうどの人が入り、スキルが無い人と下限未満の人が入らない）、案件2が 4 の1人、案件3が 5 の1人であること、テスト内で足した必要スキル0件の案件が候補0人になることが確かめられ、`java tools/Build.java test` がすべて成功する
- [x]

1つのタスクに書かれていること

- 対応する受入基準の番号
- 作るか変えるファイル
- 終わったと判断する条件（成功するテスト、画面で見えるもの）

チャットに送る文面

```
/spec-run-task specs/matching 1
```

テストが成功すると、Copilotがtasks.mdのチェックを付けます。報告とチェックを見てから、番号を2に変えて送ります。

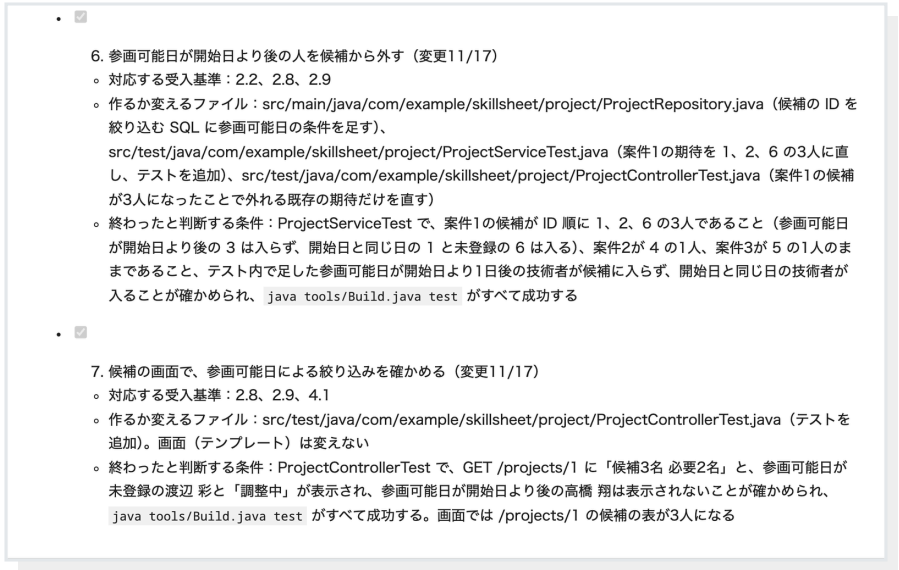
Copyright © Givery, Inc. All rights reserved

仕様の変更の追い方

仕様書を先に直して足されたタスクだけを実装します

requirements.md → **design.md** → **tasks.md** → **/spec-run-task**

2.8と2.9を追加 | SQLと判定のルール | タスク6と7を追加 | 6と7だけを実装



- [x]
  6. 参画可能日が開始日より後の人を候補から外す（変更11/17）
     - 対応する受入基準：2.2、2.8、2.9
     - 作るか変えるファイル：src/main/java/com/example/skillsheet/project/ProjectRepository.java（候補の ID を絞り込む SQL に参画可能日の条件を足す）、src/test/java/com/example/skillsheet/project/ProjectServiceTest.java（案件1の期待を 1、2、6 の3人に直し、テストを追加）、src/test/java/com/example/skillsheet/project/ProjectControllerTest.java（案件1の候補が3人になったことで外れる既存の期待だけを直す）
     - 終わったと判断する条件：ProjectServiceTest で、案件1の候補が ID 順に 1、2、6 の3人であること（参画可能日が開始日より後の 3 は入らず、開始日と同じ日の 1 と未登録の 6 は入る）、案件2が 4 の1人、案件3が 5 の1人のままであること、テスト内で足した参画可能日が開始日より1日後の技術者が候補に入らず、開始日と同じ日の技術者が入ることが確かめられ、`java tools/Build.java test` がすべて成功する
- [x]
  7. 候補の画面で、参画可能日による絞り込みを確かめる（変更11/17）
     - 対応する受入基準：2.8、2.9、4.1
     - 作るか変えるファイル：src/test/java/com/example/skillsheet/project/ProjectControllerTest.java（テストを追加）。画面（テンプレート）は変えない
     - 終わったと判断する条件：ProjectControllerTest で、GET /projects/1 に「候補3名 必要2名」と、参画可能日が未登録の渡辺 彩と「調整中」が表示され、参画可能日が開始日より後の高橋 翔は表示されないことが確かめられ、`java tools/Build.java test` がすべて成功する。画面では /projects/1 の候補の表が3人になる



スキルシート

在庫管理システムの画面改修

概要	倉庫ごとの在庫を一覧する画面を、Spring Boot で作り直す。既存の Oracle Database を参照する
開始日	2026/12/01
必要人数	2名
必要スキル	Java 3以上 / Spring Boot 3以上 / SQL 3以上

候補

候補3名 必要2名

氏名	所属	経験年数	参画可能日	必要スキルのレベル
佐藤 健太	開発1部	8年	2026/12/01	Java 5 / Spring Boot 4 / SQL 4
鈴木 美咲	開発1部	3年	2026/11/25	Java 3 / Spring Boot 3 / SQL 3
渡辺 彩	開発1部	5年	調整中	Java 4 / Spring Boot 3 / SQL 4

案件一覧に戻る

案件1の候補が4名から3名になり、高橋 翔が外れました。

画面: 2026年11月6日の試走。変更の内容は「参画可能日が案件の開始日より後の人は、候補から外す」

Copyright © Givery, Inc. All rights reserved

試走の結果

7本とも4分以内に終わりテストはすべて成功しました

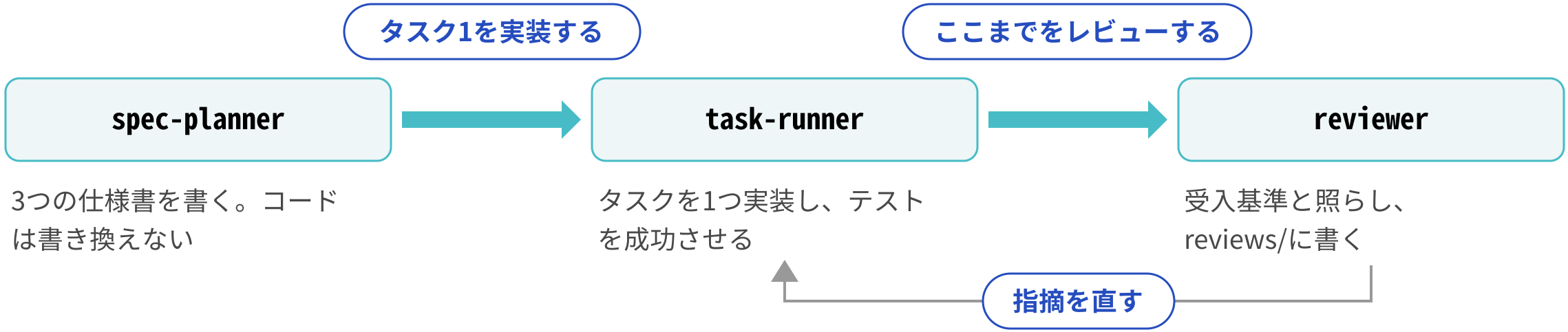
演習	依頼	かかった時間	テスト
Session03	1行の依頼で日報アプリ	56秒	2件成功
Session03	仕様のメモで直す	39秒	2件成功。画面で500エラー
Session04	ISSUE-101 スキル名の検索	67秒	12件成功
Session04	ISSUE-102 経験年数の入力チェック	55秒	10件成功
Session04	ISSUE-103 参画可能日の順	69秒	9件成功
Session05	案件マッチング タスク1〜5	225秒	20件成功
Session05	仕様の変更	90秒	23件成功

2026年11月6日に、受講者と同じフォルダとプロンプトで、Claude Code（Sonnet 5.5）を使って通した結果です。
Copilotでは、かかる時間と書かれるコードが変わります。

Copyright © Givery, Inc. All rights reserved

役割を分けたエージェントでつなぐ

ボタンで次のエージェントへ仕事を渡します



spec-planner.agent.md の一部

```
handoffs:
 - label: タスク1を実装する
 agent: task-runner
 send: false
```

send: falseなので、ボタンを押しても送信されず、入力欄に依頼の文面が入ります。受講者が読んでから送ります。handoffsが使えるのは、Localのハーネスだけです。

Copyright © Givery, Inc. All rights reserved

ワーク

Session05の進め方

4つのSTEPで案件ごとの候補の画面を作ってください [65min]

STEP	やること	時間
STEP1	request.mdで要件を2つ選び、/spec-requirements	[12min]
STEP2	/spec-designと/spec-tasks	[10min]
STEP3	/spec-run-taskでタスクごとに実装	[20min]
STEP4	/spec-updateで仕様を変える	[10min]

STEP3で時間が足りないときは、終わったタスクまでで止めてください。
講師が完成例のフォルダをお渡しします。



追加の要件にAとEを選んだときの案件1の画面です。「候補4名 必要2名」が出ています。

Copyright © Givery, Inc. All rights reserved

研修のあとに

skillsheetの.githubは貴社のリポジトリにそのまま置けます

ファイル	置いたときの働き	直すところ
copilot-instructions.md	Copilotが毎回読むきまり	ビルドのコマンド、使うライブラリ、パッケージの構成
ISSUE_TEMPLATE/feature.md	GitHubでIssueを作るときのテンプレート	確かめ方の欄のコマンド
pull_request_template.md	PRの説明文の型	そのまま使える
prompts/*.prompt.md	チャットで「/」に続けて呼び出すプロンプト9本	テストのコマンドと、既存のテストクラスの名前
agents/*.agent.md	仕様、実装、レビューに分けたエージェント3本	そのまま使える

置き場所は、貴社のリポジトリの一番上にある.githubフォルダです。

Copyright © Givery, Inc. All rights reserved

質問

質疑応答

ご質問をチャットか音声でお寄せください

たとえば、次のような質問を受け付けます

- 業務のリポジトリに、プロンプトとエージェントを置くときの直し方
- 社内の利用ルールで、最初に決める項目
- Issueの受入基準を、チームで書きそろえる方法
- HydraFusionと、ほかのモデルの選び方

研修のあとのご質問は、事務局を通じてお寄せください。

Copyright © Givery, Inc. All rights reserved



株式会社ギブリー (Givery, Inc.)
〒150-0036 東京都渋谷区南平台町15-13 帝都渋谷ビル8F