ソースコード診断の報告書

システム技術評価レポート

受発注管理システム

社内の担当者が Claude Code で作成／約40名が利用

見本　このレポートは、架空のシステムを題材にした見本です。実在のお客様の案件ではありません。システムの名前・構成・数値・ファイル名はすべて説明のためにつくったものです。

評価日
2026年10月6日

作成
株式会社Ouver

宛先
〇〇株式会社 御中

1	総合評価	p.02
2	調べた範囲と点数の付け方	p.03
3	項目ごとの評価	p.04
4	変更の履歴から分かること	p.05
5	項目ごとの詳しい評価	p.06
6	見つかった課題	p.08
7	直す順番	p.09
8	保守の体制について	p.09
付録	点数の計算と、用語の説明	p.10

本レポートは、2026年10月6日時点のソースコードと変更の履歴をもとにしています。本番環境への接続や、実際に攻撃を試す検査は行っていません。
ご依頼の範囲とシステムによって、調べる内容と報告の項目は変わります。

このレポートで分かること

AI で作ったシステムを外の会社に保守してもらう場合も、社内で直し続ける場合も、最初に知っておきたいのは次の3つです。

01	このまま保守を引き継げるか	100点満点の点数と、引き継ぐ前に直す所があるかどうか。（1章）
02	どこが危なく、どこは大丈夫か	7つの項目ごとの評価と、変更の履歴から分かる手間のかかりやすい所。（3〜5章）
03	何から、どの順で直すか	すぐ直すこと・1か月以内・あとからでよいことの仕分けと、かかる期間。（6〜7章）

お急ぎの場合は、1章の「総合評価」と7章の「直す順番」だけで、判断に必要なことが分かるようにしてあります。詳しい根拠は3章以降に書きました。分からない言葉は付録の用語の説明をご覧ください。

1 総合評価

このまま保守を引き継げるか



作り直しは要りません。ただ、保守を始める前に直す所があります。

社内の担当者が Claude Code で約4か月かけて作り、約40名が毎日使っている受発注管理システムです。画面・処理・データの分け方が整っていて、設計はしっかりしています。

一方で、ほかの部署の人でも取引先の情報を書き換えられる権限の抜けと、テストが1本もないことが分かりました。保守を引き継ぐ前に、約2週間でセキュリティの問題を直し、続く約1か月でテストを整えることをおすすめします。

80〜100	60〜79	40〜59	20〜39	0〜19
健全	おおむね良好	**要改善**	要注意	危険

急いで直すべき3つのこと

1　ほかの部署の人でも、取引先の情報を書き換えられる

ログインさえしていれば、取引先の一覧を誰でも編集できます。データベースに「行ごとの権限」が設定されていないためです。誤操作でも故意でも、取引先の住所や締め日を書き換えられます。

2　外部サービスの鍵が、ソースコードに直接書かれている

メール送信サービスと会計ソフト連携の鍵が、コードの中にそのまま入っています。コードを見られた時点で、これらのサービスを勝手に使われるおそれがあります。

3　テストが1本もない

直したあとに、ほかの所が壊れていないかを確かめる手段がありません。いまは担当者が画面を触って確かめていますが、受注と請求のように処理が込み入った所では見落としが出ます。

保守しやすい所

- 画面・処理・データの分け方が整っていて、作り直しは要らない
- データベースの設計が、受注・出荷・請求の業務の流れに沿っている
- ライブラリはおおむね新しいバージョンを使っている

引き継ぎの前に手当てが要る所

- 権限の抜けと、コードに書かれた鍵（約2週間）
- テストの整備（約1か月）
- 受注まわりを直せる人が1人しかいない

2 調べた範囲と点数の付け方

何を、どう調べたか。何は調べていないか

調べたシステム

システム	受発注管理システム（社内向け）
主な機能	受注の登録と承認、出荷の指示、請求書の作成と会計ソフトへの連携、取引先の管理、メールでの通知
構成	画面と処理は Next.js（Vercel で公開）、データとログインは Supabase。メール送信サービスと会計ソフトにつながっている
作り方と期間	社内の担当者3名が Claude Code を使って作成。2026年6月〜9月（約4か月）
規模	ソースコード約1万8千行、データベースの表が24、画面が31
使っている人	営業・業務・経理の約40名。平日は毎日使われている

調べ方

コードを読む	エンジニアがソースコードを読み、権限の確かめ方、データの扱い、処理の分け方を確認
自動の検査	書き方の検査、使われていない処理や重複の検出、ライブラリと公表済みの脆弱性との照合
変更の履歴	約4か月分・312回の変更の記録から、よく直されるファイルや担当者の偏りを分析
設定の確認	データベースの権限設定、公開環境の設定、鍵や接続先の置き場所

調べていないもの：本番環境への接続、実際に攻撃を試す検査、利用者が増えたときの速さ、会計ソフト側の設定。これらが必要な場合は別に調べます。

点数の付け方

7つの項目をそれぞれ100点満点で採点し、保守を引き継げるかどうかへの影響の大きさで重みを付けて合計しました。作り直しが要るかを左右する「設計」と、事故につながる「セキュリティ」を重くしています。計算の内訳は付録にあります。

評価	点数	意味
A	80〜100	問題なし。このままでよい
B	60〜79	小さな直しでよい
C	40〜59	直したほうがよい。保守を続けながら直せる
D	0〜39	保守を始める前に直す必要がある

3 項目ごとの評価

どこが良くて、どこが弱いか

弱いのはセキュリティとテストの2つです。設計は B で、作り直しは要りません。残りの4項目は C で、保守を続けながら少しずつ直せば足ります。

設計 重み 25%	B	74	画面・処理・データの分け方は整っている。データベースの表も業務の流れに沿っている
セキュリティ 重み 20%	D	38	ほかの部署の人が取引先の情報を書き換えられる。外部サービスの鍵がコードに書かれている
コードの読みやすさ 重み 15%	C	50	名前の付け方はそろっているが、同じ処理の書き写しが多い
テスト 重み 10%	D	15	テストが1本もなく、直したあとに壊れていないかを確かめる手段がない
使っているライブラリ 重み 10%	C	59	3つに公表済みの脆弱性がある。ほかは新しいバージョンを使っている
資料 重み 10%	C	58	手元で動かす手順はあるが、本番への出し方が書かれていない
AI で直し続けやすいか 重み 10%	C	52	処理が1つのファイルに集まり、AI に渡しても全体を読み切れない

A 問題なし　B 小さな直しでよい　C 直したほうがよい　D 先に直す必要がある

AI で作ったシステムとして見ると

4か月で受注から請求までを一通りそろえ、40名が毎日使える所まで来ています。動く画面を早く作れるのは、AI を使った開発の良い面です。

弱いのは、画面からは見えない部分です。権限の設定、鍵の置き場所、テストは、動かして確かめても問題が表に出ません。AI に頼んで作るとき、頼む側が言わなければ後回しになりやすい所でもあります。今回の課題も、ほとんどがここに集まっていました。

4 変更の履歴から分かること

コードを読むだけでは見えない、手間がかかりやすい所

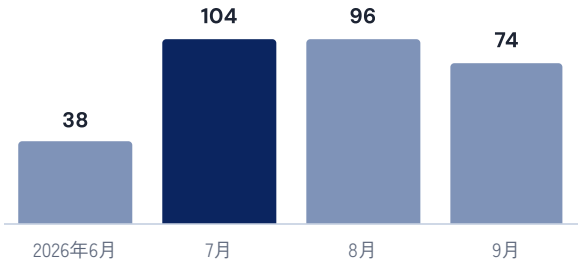
ソースコードの変更の記録（約4か月・312回分）を集計しました。よく直されるファイルは、それだけ不具合が出やすく、保守の手間も集まります。

よく直されるファイル 変更の回数が多い順

ファイル	変更の回数	回数／行数
order/register.ts		18回／2,400行
invoice/export.ts		12回／860行
lib/auth.ts		9回／310行
customers/list.tsx		7回／280行
orders/[id]/page.tsx		6回／420行
lib/mail.ts		5回／190行
shipping/schedule.ts		5回／540行

赤は、よく直されていて行数も多いファイルです。`order/register.ts` は受注の登録・在庫の引き当て・承認の通知を1つのファイルで受け持っていて、2,400行あります。保守の手間がいちばん集まる所なので、テストもここから書きます。

月ごとの変更の回数



7月に受注と請求を作り込み、8月に出荷と通知を足しています。9月は新しい機能が減り、使い始めてから出た不具合の修正が中心で、その約4割が受注と請求の2ファイルに集中しています。

担当者の偏り

82%

受注まわりの変更のうち、1人の担当者が書いた割合

受注まわりの変更の82%を、1人の担当者（担当者A）が書いています。この人が休んだり異動したりすると、受注の処理を直せる人がいなくなります。請求まわりも、担当者Bが68%を書いていて、同じ形の偏りがあります。

担当者	変更の回数	主に書いた所	その所を直せるのは
担当者A	221回（71%）	受注、在庫、ログイン	受注は担当者Aのみ
担当者B	69回（22%）	請求、会計ソフトとの連携	請求は担当者Bが中心
担当者C	22回（7%）	取引先の画面、メールの文面	ほかの2人も直せる

一緒に直されることが多いファイル

コードの上ではつながりが見えないのに、片方を直すと、もう片方も直す必要が出る組み合わせです。引き継いだ人が知らずに片方だけ直すと、不具合になります。

ファイル	相手のファイル	一緒に直された回数	理由
`order/register.ts`	`invoice/export.ts`	9回	受注に項目を足すと、請求書の出力も直す必要がある
`order/register.ts`	データベースの変更の記録	7回	受注の表を変えるたびに、登録の処理も書き換えている
`lib/auth.ts`	`customers/list.tsx`	5回	画面ごとに、ログインの確かめ方を個別に書いている

AI で作ったシステムに多い跡

23回
一度に500行以上の変更
AI にまとめて書かせた変更です。何を変えたかを後から追いにくく、不具合の原因を探すときに時間がかかります。

41個
使われていない処理
作り直しの途中で古いほうが残ったものです。読む人が「どちらが本物か」で迷います。

6か所
同じ処理の書き写し
金額の計算が6か所に書き写されています。消費税の扱いを変えるとき、全部を直す必要があります。

5 項目ごとの詳しい評価

良い点、直したい点、保守への影響

B 設計　74 点

画面・処理・データの3つが分かれていて、どこに何が書いてあるかを追えます。データベースの表も、受注・出荷・請求という業務の流れに沿って作られています。作り直す必要はありません。

良い点
- 画面ごとにフォルダが分かれ、置き場所の決まりが守られている
- 受注と出荷と請求の表が、業務の単位で分かれている
- データベースの変更が、記録として残されている

直したい点
- 受注の登録が1つのファイル（約2,400行）に集まっている
- ログインの確かめ方を、画面ごとに個別に書いている
- 在庫の引き当てが、受注の処理の中に埋まっている

保守への影響 全体としては保守しやすい作りです。受注の登録だけは、直すたびに別の所が壊れやすいので、次に機能を足すときに合わせて役割ごとに分けます。

D セキュリティ　38 点

ログインの仕組みには Supabase の機能を正しく使っています。ただ、ログインしたあとに「誰が何をしてよいか」を絞る設定が抜けていました。

すぐ直す：取引先の表に「行ごとの権限」が設定されておらず、ログインしていれば誰でも取引先の情報を書き換えられます。画面にボタンを出していなくても、ブラウザから直接データベースに書き込みを送れば、そのまま通ってしまいます。

すぐ直す：メール送信サービスと会計ソフト連携の鍵が、`lib/mail.ts` と `invoice/export.ts` に直接書かれています。

良い点
- パスワードの管理を Supabase に任せていて、自前で持っていない
- 通信はすべて暗号化されている

ほかに直したい点
- 受注と請求の表も、部署ごとの権限が粗い
- ログインの失敗が続いても止める仕組みがない

保守への影響 保守を始める前に直す必要があります。権限の設定は、データベースの表ごとに「誰が読めて、誰が書けるか」を決める作業で、約1週間で終わります。

C コードの読みやすさ　50 点

名前の付け方や書き方はそろっていて、読み始めるのに苦労はしません。気になるのは、同じ処理があちこちに書き写されていることです。

良い点

- 変数やファイルの名前が、業務の言葉とそろっている
- 書き方の自動の検査が設定されている

直したい点

- 金額の計算が6か所に書き写されている
- 使われていない処理が41個残っている
- 型の検査を外している所が37か所ある

保守への影響　書き写された所では、1か所だけ直して残りを直し忘れる事故が起きます。金額の計算は1つにまとめておきます。

D テスト　15 点

テストのコードが1本もありません。テストを動かす道具は入っていますが、使われていませんでした。いまは、直すたびに担当者が画面を触って確かめています。

良い点

- テストを動かす道具は、すでに入っている
- 公開の前に、型の検査は自動で動いている

直したい点

- 受注・請求の金額の計算を確かめるテストがない
- 権限の設定を確かめるテストがない
- 公開の前にテストが動く仕組みがない

保守への影響　テストがないと、どんな小さな直しでも、壊れていないかを人の手で確かめることになります。受注と請求の処理から順に書き、公開の前に自動で動くようにします。

C 使っているライブラリ　59 点

使っているライブラリ（他の人が作って公開している部品）は68個で、大半は新しいバージョンです。そのうち3つに、公表済みの脆弱性がありました。

良い点

- Next.js と Supabase の部品は、最新に近いバージョンを使っている
- 版を固定する仕組みがあり、勝手に変わらない

直したい点

- 日付の扱い、PDF の作成、表の読み込みに使う3つに脆弱性がある
- 同じ役割のライブラリが2つ入っている所がある

保守への影響　知られた手口で攻撃される余地が残ります。更新して、請求書の PDF の見た目が変わらないかを確かめます。

C 資料　58 点

手元のパソコンで動かす手順は、分かりやすく書かれています。本番に出す手順と、外部サービスとのつなぎ方は書かれていませんでした。

良い点

- 手元で動かすまでの手順が、順を追って書かれている
- データベースの表ごとに、項目の説明がある

直したい点

- 本番への出し方と、戻し方が書かれていない
- 会計ソフトとのつなぎ方が、担当者Bの頭の中にしかない
- AI への指示に使った資料が、古いまま残っている

保守への影響　不具合が出たとき、戻し方が分からないと復旧に時間がかかります。引き継ぎの打ち合わせで担当者から聞き取り、手順として書き起こします。

C AI で直し続けやすいか　52 点

保守を続けるあいだも、AI に手伝わせて直していくことを前提に見ました。型が付いていて、ファイルの置き場所も決まっているので、AI が読みやすい作りです。ただ、大きなファイルとテストのなさが足を引っ張っています。

良い点

- TypeScript で書かれ、データの形が型で分かる
- フォルダの決まりがそろっていて、AI が置き場所を間違えにくい

直したい点

- 2,400行のファイルは、AI に渡しても全体を読み切れない
- テストがないので、AI が直した内容が正しいかを確かめられない
- AI への指示の決まりごとが、資料にまとまっていない

保守への影響　テストを整え、大きなファイルを分ければ、AI に任せられる直しの幅が広がります。AI に守らせる決まりごとを1つの資料にまとめておくと、誰が頼んでも同じ品質で直せます。

この章のまとめ

7つの項目のうち、保守を始める前に手当てが要るのはセキュリティとテストの2つです。どちらも、作ったものを壊さずに足す作業で、作り直しにはなりません。ほかの項目は、保守を続けながら機能を足すついでに直していけます。

6 見つかった課題

すぐ直すこと、あとから直せばよいこと

見つかった課題は5件です。放っておいたときの影響と、直すのにかかる手間で順番を付けました。

順番	見つかった課題	放っておくと	直し方	手間の目安
すぐ	**取引先の一覧を、ログインしていればほかの部署の人も書き換えられる** データベースの取引先の表	誤って、または故意に取引先の情報が書き換えられる	行ごとの権限を設定し、部署ごとに書き換えられる範囲を絞る	約5日
すぐ	**外部サービスの鍵がソースコードに直接書かれている** lib/mail.ts、invoice/export.ts	コードを見られると、外部サービスを勝手に使われる	鍵を作り直し、環境変数に移す	約1日
1か月以内	**テストが1本もない** システム全体	直すたびに、ほかの所が壊れていないかを人の手で確かめることになる	受注と請求の処理から順にテストを書き、公開の前に自動で動かす	約3週間
1か月以内	**使っているライブラリのうち3つに、公表済みの脆弱性がある** 日付の扱い、PDF の作成、表の読み込み	知られた手口で攻撃される余地が残る	更新して、動きが変わらないかを確かめる	約3日
あとから	**受注を登録する処理が1つのファイル（約2,400行）に集まっている** order/register.ts	直すたびに別の所が壊れやすく、改修に時間がかかる	次に機能を足すときに合わせて、役割ごとに分ける	機能追加と一緒に

手間の目安は、Ouver のエンジニア1名が作業した場合の日数です。テストの作成は、受注と請求の主な処理を対象にしています。

課題に入れなかったこと

同じ処理の書き写し、使われていない処理、本番への出し方の資料がないことは、5章の詳しい評価に書きました。急いで直すほどではないため、課題の一覧には入れず、保守を続けながら直す作業に回しています（7章）。

7 直す順番

安心して直し続けられる状態になるまで

最初の約2週間	次の約1か月	そのあと
セキュリティの修正	**テストの整備**	**通常の保守**
権限の設定、鍵の作り直し、ライブラリの更新。ここが終われば、事故の心配は大きく減ります。	受注と請求の処理のテストを書き、変更のたびに自動で動くようにします。	問い合わせへの対応、軽い修正、機能の追加、月ごとの報告。

すぐ
〜2週間

- [ ] 取引先・受注・請求の表に、部署ごとの「行ごとの権限」を設定する
- [ ] メール送信サービスと会計ソフト連携の鍵を作り直し、環境変数に移す
- [ ] 古い鍵が使われていないことを、各サービスの管理画面で確かめる
- [ ] 脆弱性のある3つのライブラリを更新し、請求書の PDF の見た目を確かめる
- [ ] 権限の設定が効いているかを確かめるテストを書く

1か月以内
2週目〜6週目

- [ ] 受注の登録と、請求の金額の計算にテストを書く
- [ ] 公開の前にテストが自動で動き、失敗したら公開を止める仕組みを入れる
- [ ] 6か所に書き写された金額の計算を1つにまとめる
- [ ] 本番への出し方・戻し方と、会計ソフトとのつなぎ方を資料にする

あとから
保守と並行

- [ ] 受注の登録を、在庫の引き当て・承認の通知と分けて、役割ごとのファイルにする
- [ ] 使われていない処理41個を消す
- [ ] ログインの確かめ方を1か所にまとめる
- [ ] AI に守らせる決まりごとを1つの資料にまとめる

8 保守の体制について

誰が、どのくらい関われればよいか

必要な人と関わり方

役割	関わり方	求める経験
保守の担当エンジニア	最初の6週間は週3日ほど。そのあとは月に数日	Next.js と Supabase。権限の設定に慣れていること
社内の窓口	週に1回、30分ほどの打ち合わせ	受注から請求までの業務の流れが分かる方
担当者A・B	最初の2週間に、合わせて半日ほどの聞き取り	受注と請求の処理の作りを説明していただく

担当者AとBへの聞き取りは、引き継ぎでいちばん大事な作業です。2人が忙しくなる前に、早めに時間をいただくことをおすすめします。

AI で直し続けるために

保守に入ってからも、AI に手伝わせて直すのは良いやり方です。ただし、AI が直した内容が正しいかを確かめる手段がないうちは、任せる範囲を小さくしておくべきです。

テストが整ったら任せやすいこと

- 画面の文言や項目の追加
- テストのコードを書くこと
- 大きなファイルを分ける作業の下書き
- 資料の書き起こし

人が必ず確かめること

- 権限の設定を変える直し
- 金額の計算にかかわる直し
- データベースの表の作りを変える直し
- 会計ソフトへの連携

総合所見

約4か月で、受注から請求までの業務を1つのシステムにまとめた点は高く評価できます。設計もしっかりしていて、作り直しは要りません。

課題は、権限の設定とテストという「動かしても見えない部分」に集まっていました。最初の6週間でここを手当てすれば、そのあとは機能を足しながら、AI も使って安心して直し続けられる状態になります。

この見本について

実際の報告書の中身は、ご依頼の範囲とシステムによって変わります。点検の進め方や費用は、ホームページの「AIで作ったシステムの保守・改修」をご覧ください。

株式会社Ouver
ouver.co.jp

付録　点数の計算と、用語の説明

総合点の出し方と、本文に出てくる言葉

総合点の計算

項目	重み	評価	点数	寄与
設計	25%	B	74	18.5
セキュリティ	20%	D	38	7.6
コードの読みやすさ	15%	C	50	7.5
テスト	10%	D	15	1.5
使っているライブラリ	10%	C	59	5.9
資料	10%	C	58	5.8
AI で直し続けやすいか	10%	C	52	5.2
合計	100%			**52.0点**

総合点の目安：80〜100点「健全」このまま引き継げる／60〜79点「おおむね良好」小さな直しをしながら保守できる／40〜59点「要改善」保守を始める前に直す所がある／20〜39点「要注意」大きな直しが前提／0〜19点「危険」作り直しをおすすめする

用語の説明

Supabase
データベースとログインの仕組みをまとめて借りられるサービス。AI で作るシステムでよく使われます。

行ごとの権限
データベースの表の1行ずつに「誰が読めて、誰が書けるか」を決める設定。Supabase ではこれを設定しないと、ログインした人なら誰でもデータを書き換えられる状態になります。

鍵（API キー）
外部のサービスを使うための合言葉。知られると、他人がそのサービスをあなたの名前で使えてしまいます。

環境変数
鍵などをソースコードの外に置いておく場所。コードを見られても鍵は漏れません。

テスト
「この入力ならこの結果になる」を自動で確かめるプログラム。直したあとに、ほかの所が壊れていないかをすぐに確かめられます。

ライブラリ
ほかの人が作って公開している部品。日付の計算や PDF の作成など、よく使う処理を自分で書かずに済みます。

脆弱性
攻撃に使われるおそれのある欠陥。公表済みのものは手口も知られているため、早めに更新します。