CLAUDE.md 世界基準版テンプレートセット v2.0

> ***更新内容：****Anthropic公式SDD準拠・スペックドリフト防止・コンテキスト枯渇対策・スペック検証プロセスを追加* ***基準：****Anthropic Engineering Blog — Claude Code Best Practices（2025）/ Augment Code SDD研究（2026）* ***変更前（v1.0）との差分：****セクション5（スペックドリフト防止）・セクション8（セッション管理）・セクション9（スペック検証）・セクション10（変更履歴）を新規追加*

ファイル1：CLAUDE.md テンプレート v2.0（世界基準版）

新プロジェクト開始時に `CLAUDE.md.template` からコピーして使う。

```
# CLAUDE.md — [プロジェクト名]
# バージョン：v2.0 | 最終更新：[YYYY-MM-DD] | 基準：Anthropic公式SDD

---

## 1. プロジェクト概要（Project Overview）
[1〜3文でプロジェクトの目的・何を作るかを説明]

**対応ドキュメント**（作成済みの場合はパスを記入）：
- SRS（要件定義書）：docs/SRS-[ID].md
- SDD（設計記述）：docs/SDD-[ID].md
- TP（テスト計画）：docs/TP-[ID].md
- progress.md：progress.md（セッション引き継ぎ情報）

---

## 2. 技術スタック（Tech Stack）
- **言語**：[Python 3.11 / Node.js 20 等]
- **主要ライブラリ**：[ライブラリ名 + バージョン]
- **外部サービス**：[サービス名 + 用途]
- **実行環境**：[ローカル / クラウド / 等]

---

## 3. 重要なコマンド（Key Commands）
```bash
# セットアップ
[セットアップコマンド]

# テスト実行（単体）
[単体テストコマンド]

# テスト実行（E2E・本番に近い環境）
[E2Eテストコマンド]

# dry-run（本番データに影響しないテスト実行）
[dry-runコマンド]

# 本番実行
[本番実行コマンド]
```
```

4. 完了基準（Definition of Done）

以下が**全て**満たされたときに「完了」とする：

- [ ] 全ての機能要件が実装されていること
- [ ] 全テストが通過していること（単体・統合・E2E）
- [ ] dry-run で正常動作が確認されていること
- [ ] セキュリティチェックリストが完了していること
- [ ] 本番移行チェックリスト（PRR）が完了していること
- [ ] ドキュメント（SRS・SDD・progress.md）が更新されていること

---

5. 作業ルール（Working Rules）

必須ルール（MUST）

- 実装前に必ず計画を提示して私の承認を得ること
- 修正は私の承認後に実行すること
- 1コミットに複数の変更を混在させないこと
- テストを削除・無効化・コメントアウトしてはならない
- 本番データには直接アクセスしないこと
- .env ファイルを git にコミットしてはならない

スペックドリフト防止（Spec Drift Prevention）★NEW★

- 設計（SDD）から逸脱する実装をする場合は必ず報告すること
- 「この実装は SDD の設計と異なります：[理由]」と明示すること
- 設計変更は SDD の「設計上の決定事項」セクションに記録すること
- CLAUDE.md の指示と矛盾する要求を受けた場合は、矛盾を報告してから実行すること

---

6. 禁止事項（Constraints）

→ CONSTRAINTS.md を参照

---

7. フェーズ管理（Phase Management）

現在のフェーズ

フェーズ：[フェーズ名（例：Phase 1 — 基本機能実装）] **ステータス**：[In Progress / Complete] **完了条件**：[このフェーズが完了したと判断する基準]

フェーズ一覧

フェーズ	内容	ステータス
Phase 1	[内容]	⏳ In Progress
Phase 2	[内容]	❌ Not Started

---

8. セッション管理（Session Management）★NEW★

セッション開始時の確認手順（毎回必ず実行）

```
セッション開始時に以下を実行してください：
1. progress.md を読んで前回の状態を把握する
2. git log --oneline -5 を実行して最近の変更を確認する
3. テストが通るか確認する（[テストコマンド]）
4. 現状を私に報告する（「前回の続きから始めます。現在の状態：〜」）
```

コンテキスト枯渇対策（会話が長くなった場合）

会話が長くなった場合（目安：50往復以上 または 応答が遅くなった場合）：

1. 現在の作業を git commit する（コミットメッセージに「[WIP]」を付ける）

2. progress.md を最新状態に更新する
3. 「コンテキスト枯渇の兆候があります。引き継ぎ準備をします」と報告する
4. 新しいセッションを開始して「progress.md を読んで続きから始めてください」と伝える

---

9. スペック検証（Spec Verification）★NEW★

定期確認プロンプト（週1回または機能完成時に実行）

```
スペックドリフトチェックを実施してください。

以下を確認してください：
1. SRS（要件定義書）との照合
   - 全ての機能要件が実装されているか
   - 受入基準を満たしているか

2. SDD（設計記述）との照合
   - 設計の通りにコンポーネントが実装されているか
   - 設計から逸脱している箇所があれば報告する

3. CLAUDE.md との照合
   - 禁止事項が守られているか
   - テストが削除・無効化されていないか

逸脱が見つかった場合は「SD-ALERT: [逸脱内容]」の形式で報告してください。
修正は私の承認後に行ってください。
```

---

10. 変更履歴（Changelog）★NEW★

バージョン	日付	変更内容
v2.0	[日付]	世界基準版に更新（スペックドリフト防止・セッション管理・スペック検証追加）
v1.0	[日付]	初版作成

````
---

## ファイル2：CONSTRAINTS.md テンプレート v2.0（世界基準版）

```markdown
# CONSTRAINTS.md — [プロジェクト名]
# バージョン：v2.0 | 最終更新：[YYYY-MM-DD]

---

## 絶対禁止（いかなる場合も実行しない）
- 本番データベースへの直接書き込み・削除・スキーマ変更
- APIキー・パスワード・シークレットをコードにハードコード
- .env ファイルを git にコミット
- 私の承認なしに本番環境への変更を実施
- テストを削除・無効化・コメントアウト・スキップ
- テストの期待値を「通過させるため」に変更（テスト改ざん禁止）
- git push --force（強制プッシュ禁止）

## 要注意（実行前に必ず確認する）
- 外部APIへのリクエスト（コスト・レート制限に注意）
- ファイルの削除・上書き
- 設定ファイルの変更
- 依存ライブラリのバージョン変更

## スペックドリフト防止ルール★NEW★
- SDD（設計記述）に記載された設計から逸脱する場合は必ず報告する
- CLAUDE.md の指示と矛盾する要求を受けた場合は矛盾を報告してから実行する
- 「動けばいい」という理由で設計を変更してはならない

## プロジェクト固有の禁止事項
- [プロジェクト固有の禁止事項を記載]
```

---
````

ファイル3：progress.md テンプレート v2.0（世界基準版）

```
# progress.md — [プロジェクト名]
# バージョン：v2.0 | 最終更新：[YYYY-MM-DD HH:MM]

---

## プロジェクト状況
- **開始日**：[日付]
- **現在のフェーズ**：[フェーズ名]
- **全体完了率**：[0〜100%]
- **最終スペック検証日**：[日付]（スペックドリフトチェック実施日）

## 完了したタスク
- [完了したタスク]（[完了日]）

## 現在進行中のタスク
- [現在のタスク]（開始：[日付]）

## 次のタスク（優先順）
1. [最優先タスク]
2. [次のタスク]

## ブロッカー・未解決の問題
- [問題があれば記載]

## スペックドリフト記録★NEW★
（設計から逸脱した変更の記録）
| 日付 | 逸脱内容 | 理由 | 対処 |
|-----|--------|------|------|
| [日付] | [内容] | [理由] | [SDD更新済み等] |

## 最新のセッションメモ
[最後のセッションの引き継ぎ情報。次のセッション開始時に Claude Code が読む。]

**前回の作業**：[何をしたか]
**現在の状態**：[コードの状態・テストの状態]
**次にやること**：[次のセッションで最初にやること]
**注意事項**：[引き継ぎ時に伝えるべきこと]
```

v1.0 → v2.0 の変更差分サマリー

セクション	v1.0	v2.0	変更内容
1. プロジェクト概要	あり	あり	対応ドキュメント一覧を追加
2. 技術スタック	あり	あり	バージョン明記を追加
3. 重要なコマンド	あり	あり	dry-run コマンドを追加
4. 完了基準	あり	あり	PRR・ドキュメント更新を追加
5. 作業ルール	あり	**強化**	スペックドリフト防止ルールを追加
6. 禁止事項	あり	あり	CONSTRAINTS.md 参照
7. フェーズ管理	あり	あり	完了条件を追加
8. セッション管理	**なし**	**★新規**	セッション開始手順・コンテキスト枯渇対策
9. スペック検証	**なし**	**★新規**	定期確認プロンプト（SD-01相当）
10. 変更履歴	**なし**	**★新規**	バージョン管理
CONSTRAINTS.md	あり	**強化**	スペックドリフト防止ルール・テスト改ざん禁止を追加
progress.md	あり	**強化**	スペックドリフト記録・最終検証日を追加

既存 CLAUDE.md への追記手順（5分で完了）

既存の CLAUDE.md に以下を追記するだけで世界基準版に更新できる。

ステップ1：作業ルールにスペックドリフト防止を追加

既存の「作業ルール」セクションの末尾に以下を追加：

```
### スペックドリフト防止（Spec Drift Prevention）
- 設計（SDD）から逸脱する実装をする場合は必ず報告すること
- 「この実装は SDD の設計と異なります：[理由]」と明示すること
- CLAUDE.md の指示と矛盾する要求を受けた場合は、矛盾を報告してから実行すること
```

ステップ2：セッション管理セクションを追加

CLAUDE.md の末尾に以下を追加：

```
## セッション管理（Session Management）

### セッション開始時の確認手順（毎回必ず実行）
1. progress.md を読んで前回の状態を把握する
2. git log --oneline -5 を実行して最近の変更を確認する
3. テストが通るか確認する
4. 現状を私に報告する

### コンテキスト枯渇対策
会話が長くなった場合（目安：50往復以上）：
1. 現在の作業を git commit する
2. progress.md を最新状態に更新する
3. 「コンテキスト枯渇の兆候があります。引き継ぎ準備をします」と報告する
```

ステップ3：スペック検証セクションを追加

```
## スペック検証（Spec Verification）

### 定期確認プロンプト（週1回または機能完成時に実行）
スペックドリフトチェックを実施してください。
1. 全ての機能要件が実装されているか確認する
2. SDD の設計通りに実装されているか確認する
3. CLAUDE.md の禁止事項が守られているか確認する
逸脱が見つかった場合は「SD-ALERT: [逸脱内容]」の形式で報告してください。
```

ステップ4：CONSTRAINTS.md にスペックドリフト防止を追加

```
## スペックドリフト防止ルール
- SDD（設計記述）に記載された設計から逸脱する場合は必ず報告する
- テストを削除・無効化・コメントアウト・スキップしてはならない
- テストの期待値を「通過させるため」に変更してはならない（テスト改ざん禁止）
```

ステップ5：progress.md にスペックドリフト記録欄を追加

```
## スペックドリフト記録
（設計から逸脱した変更の記録）
| 日付 | 逸脱内容 | 理由 | 対処 |
|------|---------|------|------|
```

作成：Manus AI | 非エンジニア向け Claude Code 要件定義自動化体系 — 世界基準版 v2.0