Task Tracker 7日間開発計画：詳細チェックリスト

（開発者名またはプロジェクトチーム名）

2025年11月

はじめに

本チェックリストは、提供された「Task Tracker 開発要求仕様書」に基づき、React/NestJS/Prisma/JWTを使用したアプリケーションを7日間（1週間）で完成させるための実行計画である。

目標技術スタック:

□ フロントエンド: React, TypeScript, Tailwind CSS

□ バックエンド: NestJS, TypeScript, JWT (認証)

□ データベース: PostgreSQL (Supabase), Prisma (ORM)

1　一日目：環境構築とDB設計（バックエンドの準備）

目標：モノレポのセットアップとデータ構造の確立

□ モノレポ構造の作成（`root/`、`client/`、`server/` ディレクトリ）。

□ Git リポジトリの初期化と基本ファイルのコミット。

□ NestJS (`server/`) プロジェクトの初期設定（Prisma 含む）。

□ Supabase で PostgreSQL データベースを作成し、接続情報を取得。

□ Prisma スキーマ（`User`、`Task` モデル、`TaskStatus` Enum）を仕様書通りに定義。

□ Prisma マイグレーションを実行し、DB にテーブルを反映（`npx prisma migrate dev`）。

2　二日目：バックエンド認証機能の実装

目標：ユーザー登録、ログイン、JWT 認証フローの確立

□ `AuthModule` と `UsersModule` を作成。

□ JWT Strategy、Passport Guard のセットアップ。

□ サインアップ (`POST /auth/signup`) エンドポイントの実装。

　□ パスワードのハッシュ化（`bcrypt` など）を必須とする。

□ ログイン (POST /auth/login) エンドポイントの実装。

- □ パスワード検証と JWT（アクセストークン）の生成。
- □ レスポンスボディでトークンをクライアントに返す。

□ AuthGuard を適用したエンドポイントを仮作成し、認証が機能することを確認。

3 三日目：バックエンド タスク CRUD（一覧・作成）

目標：タスク管理の核となるエンドポイントの実装

□ TasksModule を作成し、AuthGuard を適用。

□ タスク新規登録 (POST /tasks) エンドポイントの実装。

- □ title が必須であることを検証。
- □ 認証情報から userId を自動で紐づけるロジックを実装。
- □ 初期 status を TODO に設定。

□ タスク一覧取得 (GET /tasks) エンドポイントの実装。

- □ ログイン中のユーザーが所有するタスクのみをフィルタリング。
- □ 更新日時順などでソートして返す（推奨）。

4 四日目：バックエンド タスク CRUD（編集・削除）と認可

目標：タスクの編集・削除と所有権チェック（認可）の完了

□ タスク編集 (PATCH /tasks/:id) エンドポイントの実装。

□ タスク削除 (DELETE /tasks/:id) エンドポイントの実装。

□ **所有権チェック（認可）** ロジックの徹底実装。

- □ 取得、編集、削除の全てのエンドポイントで、リクエストユーザー ID とタスクの userId の一致を厳密に検証。

□ バックエンド全体のロジックと API 仕様（第 4 章）の最終チェック。

5 五日目：フロントエンド環境構築と認証 UI/UX

目標：React 環境のセットアップと認証画面の完成

□ React (client/) プロジェクトの初期設定 (TypeScript)。

□ Tailwind CSS の設定と基本レイアウトの作成。

□ 共有型定義 (Task、TaskStatus) を client/src/types/ に定義。

□ 認証画面（サインアップ、ログイン）の UI/UX 実装。

□ Axios を使用した BE 認証 API への接続ロジックを実装。

□ ログイン成功後、JWT（アクセストークン）を保持し、認証状態を管理する Context/Hook を作成。

6　六日目：フロントエンド タスク CRUD UI の実装

目標：タスク管理のメイン UI と操作ロジックの完成

□ 認証後のメイン画面（タスク一覧）のレイアウトとコンポーネントを作成。

□ `GET /tasks` をコールし、取得したタスク一覧を表示するロジックを実装。

□ タスク新規登録モーダル/フォームの実装（`POST /tasks`）。

□ タスクの編集機能の実装（タイトル、詳細、ステータスの更新、`PATCH /tasks/:id`）。

□ タスク削除機能の実装と確認モーダル。

□ **モバイルフレンドリーなレスポンシブ対応**を完了。

7　七日目：統合、デプロイ、提出準備

目標：最終動作確認、デプロイ、提出要件の完了

□ フロントエンドとバックエンドの統合テスト（サインアップから CRUD まで）。

□ **Render** への NestJS API のデプロイと環境変数の設定。

□ **Vercel** への React アプリケーションのデプロイと環境変数の設定。

□ エラーハンドリング（認証失敗、タスク未検出など）の最終チェックと改善。

□ `README.md` の作成/更新（セットアップ手順、使用技術一覧、デプロイ URL、API 仕様書）。

□ GitHub Public リポジトリの確認と最終コミット。

□ 評価観点（設計力、実装力、コード品質、UI）に沿ったセルフレビュー。