

機能仕様書: Todo App - template-no-delete.tsx ベースアプリケーション

機能プラン: [@todo-app-spec](#)

作成日: 2025-11-13

ステータス: Draft

入力: [AI_input](#) フォルダ内の全ドキュメントに基づく包括的なTodoアプリケーション仕様

ユーザーシナリオ & テスト

ユーザーストーリー-1 - 個人用Todoページの作成と基本操作 (優先度: P1) MVP

概要: 開発者が自分専用のTodoページを作成し、タスクの追加/完了切り替え/フィルタリングができる。

この優先度の理由: これが最も基本的な機能であり、すべての開発者がまず習得すべき機能です。他の機能はこの基礎の上に構築されます。

独立テスト: UIから新規ページを追加し、タスクを追加/完了切り替えなどで、完全に機能するMVPとして価値を提供します。

受ナシリスト:

- Given 開発者ユーザーが起動している、When サイドバーの「+ 新規タスク追加」ボタンをクリックし、名前を入力して追加、Then サーバー再起動後、サイドバーに新しいタスクが表示される
- Given 個人用Todoページが開いている、When タスクテキストを入力して「追加」ボタンをクリック、Then タスクリストに追加され、LocalStorageに保存される
- Given タスクが複数存在する、When タスクのチェックボックスをクリック、Then 完了状態が切り替わり、取り消し線とスタイルが適用される
- Given タスクが完了と完了済みの両方存在する、When フィルターボタン（すべて/進行中/完了済み）をクリック、Then 選択したフィルターに応じてタスクが表示される
- Given タスクが追加・変更された状態、When ページをリロード、Then データがLocalStorageから復元され、すべてのタスクが保持される

ユーザーストーリー-2 - タスク削除機能の実装 (優先度: P2)

概要: 開発者がタスク削除機能を自分で実装し、配列操作とReact状態管理を学習する。

この優先度の理由: MVP機能の次に重要な学習課題であり、React開発者としての基礎スキルを身につけるために必須です。

独立テスト: 削除機能を実装し、個別削除と一括削除が正常に動作することで、完全なタスク管理機能を提供します。

受ナシリスト:

- Given タスクリストに複数のタスクが存在する、When タスクの削除ボタンをクリック、Then 該当タスクがリストから削除され、LocalStorageからも削除される
- Given 完了済みタスクと未完了タスクが混在する、When 「完了済みクリア」ボタンをクリック、Then 完了済みタスクのみが削除され、未完了タスクは保持される
- Given 完了済みタスクが1つも存在しない、When ページを表示、Then 「完了済みクリア」ボタンは表示されない（条件レンダリング）

ユーザーストーリー-3 - ページ管理機能（編集・削除） (優先度: P3)

概要: ユーザーが作成したTodoページを編集・削除でき、削除時にはタスクデータも完全にクリーンアップされる。

この優先度の理由: ページ管理機能は便利だが、基本的なTodo機能が動作すれば価値を提供できるため、優先度は低めです。

独立テスト: ページの編集・削除機能を実装し、データクリーンアップが正しく行われることで、完全なページラフサイクル管理を提供します。

受ナシリスト:

- Given サイドバーにページが表示されている、When ページの「編集」ボタンをクリックして新しい名前を入力、Then ページ名が更新され、サーバー再起動後も反映される
- Given ページにタスクデータが存在する、When ページの「削除」ボタンをクリックして削除、Then ページ情報とタスクデータ（LocalStorage）が完全に削除される
- Given ページを削除した後、When 同じ名前で新しいページを追加、Then 以前のタスクデータは復元されず、クリーンな状態でスタートする

ユーザーストーリー-4 - ワンコマンド起動とGitHub Pagesデプロイ (優先度: P1)

概要: 開発者が [start.ps1](#) でワンコマンドで開発環境を起動でき、GitHub Pagesにデプロイできる。

この優先度の理由: 開発体験を大幅に向上させ、プロジェクトの再現性とアクセシビリティを確保するため、MVPレベル的重要性があります。

独立テスト: ワンコマンドで起動し、デプロイスクリプトが正常にGitHub Pagesに公開できることで、完全な開発フローを提供します。

受ナシリスト:

- Given プロジェクトディレクトリで、When [\start.ps1](#) を実行、Then 依存関係インストール、サーバー起動、プラウザオープニングが自動実行され、5秒後にPowerShellが終了する
- Given アプリケーションがビルド済み、When [npm run deploy](#) を実行、Then GitHub Pagesにデプロイされ、公開URLでアクセス可能になる
- Given GitHub Pagesでアプリを開く、When ページ操作を行う、Then ローカル環境と同様にすべての機能が動作する

要件

機能要件

- FR-001: システムは、開発者がUIから新規Todoページを追加できる機能を提供しなければならない（名前入力、アイコン自動設定）
- FR-002: システムは、各Todoページのタスクを追加・完了切り替え・フィルタリング（すべて/進行中/完了済み）ができるなければならぬ
- FR-003: システムは、タスクデータをlocalStorageに自動保存し、ページロード後も復元しなければならぬ
- FR-004: システムは、タスクデータをlocalStorage（別個のID）で初期状態（すべて）にリセットされる
- FR-005: システムは、タスクデータをlocalStorage（別個のID）で初期状態（すべて）にリセットされる
- FR-006: システムは、ページの編集・削除機能（日本語文字、絵文字、特殊文字が正しく保存・表示される
- FR-007: サーバー再起動・編集・削除・削除後も復元しなければならぬ
- FR-008: システムは、GitHub Pagesへのデプロイ機能（[npm run deploy](#)）を提供しなければならぬ
- FR-009: システムは、テスカーライズム（日本語文字・タスク名）を正しく処理・保存・表示できなければならぬ
- FR-010: システムは、テスカーライズム（日本語文字・タスク名）を正しく処理・保存・表示しなければならぬ（106テスト）

主要エンティティ

TodoItem

- id: 一意識別子（タイムスタンプ）
- text: タスク内容
- completed: 完了状態（true/false）
- createdAt: 作成日時

UserPage

- name: ページ名（日本語対応）
- icon: アイコン（フォント）
- path: ルート URL
- component: 対応するReactコンポーネント

StorageKey

- パスワード: [\start.ps1](#)
- 例: [\test\Todos\Todos](#), [\test\Todos\Todos](#)

FilterType

- 定型値: 'all' | 'active' | 'completed'
- 初期値: 'all'
- 動作: タスクリストの表示内容を制御

成功基準

測定可能な成果

- SC-001: 開発者が30分以内に自分専用のTodoページを作成し、タスク追加・完了切り替えができる
- SC-002: ページロード時間 < 12秒（106テスト）
- SC-003: [start.ps1](#) 実行からプラウザまで5秒以内に完了する
- SC-004: GitHub Pagesで106テストが12秒以内に完了し、すべてにパスする
- SC-005: GitHub Pagesで106テストが12秒以内に完了する
- SC-006: 日本語ページ名とタスク名が日本語、絵文字、特殊文字が正しく保存・表示される
- SC-007: ページ削除時の確認ダイアログが表示される
- SC-008: ページ削除時の確認ダイアログが表示される
- SC-009: ページ削除時の確認ダイアログが表示される
- SC-010: 10,000個のタスクを登録してもページが正常に表示される（パフォーマンス目標）

アーキテクチャ概要

技術スタック



コンポーネント構造



データフロー



タスク状態遷移図



前提条件

- FR-001: システムは、開発者がUIから新規Todoページを追加できる機能を提供しなければならない（名前入力、アイコン自動設定）
- FR-002: システムは、各Todoページのタスクを追加・完了切り替え・フィルタリング（すべて/進行中/完了済み）ができるなければならぬ
- FR-003: システムは、タスクデータをlocalStorageに自動保存し、ページロード後も復元しなければならぬ
- FR-004: システムは、タスクデータをlocalStorage（別個のID）で初期状態（すべて）にリセットされる
- FR-005: システムは、タスクデータをlocalStorage（別個のID）で初期状態（すべて）にリセットされる
- FR-006: システムは、ページの編集・削除機能（日本語文字、絵文字、特殊文字が正しく保存・表示される
- FR-007: サーバー再起動・編集・削除・削除後も復元しなければならぬ
- FR-008: システムは、GitHub Pagesへのデプロイ機能（[npm run deploy](#)）を提供しなければならぬ
- FR-009: システムは、テスカーライズム（日本語文字・タスク名）を正しく処理・保存・表示できなければならぬ
- FR-010: システムは、テスカーライズム（日本語文字・タスク名）を正しく処理・保存・表示しなければならぬ（106テスト）

セキュリティ考慮事項

データ保護

- LocalStorage 平文保存: 機密情報（パスワード、個人情報、クレジットカード番号など）は保存しない
- データ削除: LocalStorageはリマインダー（未読メール）とデータ削除（定期削除）で常に隔離されている
- データ削除: ページ削除時にデータを完全に削除する

アーキテクチャセキュリティ

- XSS対策:
 - エスケープ: [\start.ps1](#) の内の変数は自動エスケープ
 - ディレクティブ: [dangerouslySetInnerHTML](#) は使用しない
 - ユーザー入力: フィルタリング（タスク名、ページ名）はすべてエスケープ処理される

デプロイメントセキュリティ

- HTTPS強制: GitHub Pagesは自動的にHTTPSで提供
- 依存関係の脆弱性: [npm audit](#) を定期的に行い、脆弱性を修正
- 環境変数: APIキーなど機密情報は環境変数で管理（現在、外部APIなし）
- template-no-delete.tsx で自動リロード設定を追加（検討中）

パフォーマンス要件

- 初期表示時間: < 10秒（106テスト）
- ページ読み込み時間: < 12秒（106テスト）
- 単体テスト時間: < 5秒
- ハードウェア要件: ノートパソコンで実行可能
- メモリ使用量: < 1GB
- CPU使用率: < 10% (CPU usage)
- バッファリング: < 100ms (Buffering)
- リクエストレスポンス: < 100ms (Response time)

想定される課題と回避策

- 誤ったタスク登録: ページ登録時にタスク登録が複数回実行される場合、タスク登録が複数回実行される
- ページ表示: ページ表示が遅くなる場合、ページ表示が遅くなる
- データ削除: データ削除が遅くなる場合、データ削除が遅くなる

課題3: 日本語文字化け

- start.ps1 でUTF-8エンコーディングを明示的に指定: [\[Console\]:outputEncoding = \[System.Text.Encoding\]:UTF8](#)

影響範囲: すべてのユーザー

- 回避策: [vite.config.ts](#) で "charset": "utf8" を設定

課題4: テストシナリオリスト

- テストシナリオ: コード変更時にテストが壊れやすい
- テスト実行: テスト実行が遅くなる場合、テスト実行時間を短縮する
- テストカバレッジ: テストカバレッジが低くなる場合、カバレッジを高める
- テスト失敗: テスト失敗が頻繁になる場合、失敗するテストを削除する

想定される課題と回避策

- ページ表示: ページ表示が遅くなる場合、ページ表示が遅くなる
- データ削除: データ削除が遅くなる場合、データ削除が遅くなる

課題5: デプロイメント不整合

- ページ表示: ページ表示が遅くなる場合、ページ表示が遅くなる
- データ削除: データ削除が遅くなる場合、データ削除が遅くなる

課題6: GitHub Pagesルーティング問題

- GitHub Pagesでクロスオリジンリソース资源共享（CORS）エラーが発生する場合、CORSエラーが発生する

影響範囲: GitHub Pages プロジェクト

- 回避策: [vite.config.ts](#) で "base": "https://todo-app.netlify.app" を設定

課題7: フィルタリング問題

- フィルタリングが正常に動作しない場合、フィルタリングが正常に動作しない

影響範囲: 全てのユーザー

- 回避策: [vite.config.ts](#) で "base": "https://todo-app.netlify.app" を設定

課題8: フィルタリング不整合

- フィルタリングが正常に動作しない場合、フィルタリングが正常に動作しない

影響範囲: 全てのユーザー

- 回避策: [vite.config.ts](#) で "base": "https://todo-app.netlify.app" を設定

課題9: フィルタリング問題

- フィルタリングが