タスク管理アプリ

～Python 版～

作成日

2024/12/02

作成者

都野浩実

プロジェクト概要

1. 目的
   - 自分専用のタスク管理システムを開発し、タスクの作成、編集、削除、および閲覧を効率的に行えるようにする。
2. 背景
   - 自分好みのタスク管理アプリを作成したかった
3. 目標
   - タスクの効率的な管理
   - 使いやすいインターフェース
   - 信頼性とパフォーマンス

ターゲットユーザー

1. ユーザー層
   - 一般のユーザー
2. ユーザーのニーズ
   - カスタマイズ性:テーマと配色の変更・インターフェースの柔軟性
   - 機能性:タスクの効率的な管理・カテゴリ管理・検索機能
   - 視覚的な魅力:美しいデザイン・テーマの切り替え
   - データの保護:作成されたタスクのデータが安全に保存される

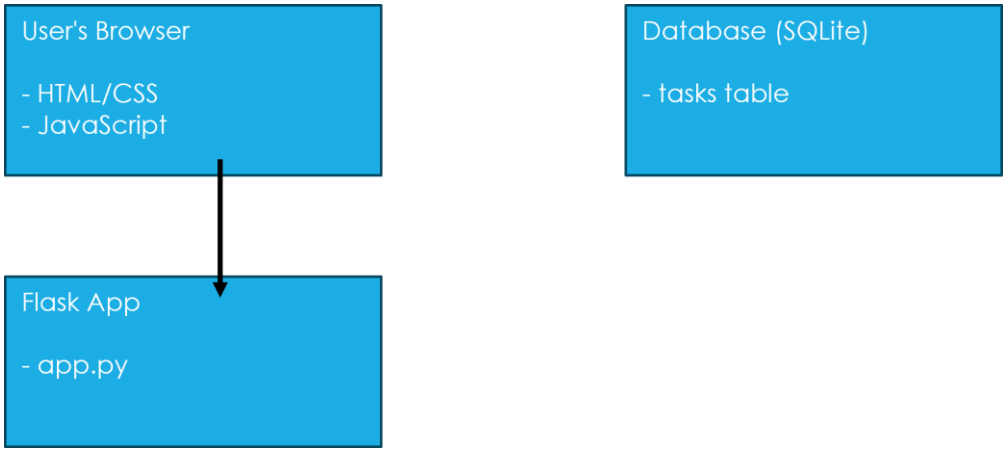
機能要件

1. 主な機能
   - タスク管理
     - タスクの作成： ユーザーは新しいタスクを作成できる。
       タイトル、説明、優先度、および期限日を入力する
       フォームが提供される。
     - タスクの編集：ユーザーは既存のタスクを編集できる。
       完了状態の更新が可能。
     - タスクの削除：ユーザーは不要なタスクを削除できる。
     - タスクの閲覧：ユーザーはタスクの一覧を閲覧できる。
       タスクは表形式で表示、各タスクの詳細が確認できる。
     - タスクの並び替え：ユーザーはタスクを期限日（昇順および降順）や
       優先度（昇順および降順）で並び替えできる。
     - 詳細表示：モーダルウィンドウを使用して、各タスクの
       詳細説明を表示できる。
     - テーマ切り替え：ユーザーはダークテーマとライトテーマを
       切り替えられる。
     - 差し色選択：ユーザーはインターフェースの差し色を選択できる。

2. 非機能要件
   - 可用性
     - アプリケーションは 24 時間 365 日稼働することを目標とする。
   - ユーザビリティ
     - ユーザーインターフェースは直感的で使いやすいものであること。
   - 保守性
     - コードはコメントやドキュメントが充実しており、他の開発者が容易に理
       解できるようにする。

システム構成

1. アーキテクチャ



2. 技術スタック
   - クライアントサイド
     - HTML5
     - CSS3
     - JavaScript
   - サーバーサイド
     - Python3
     - Flask
     - SQLite
   - 開発ツール
     - Spyder

開発スケジュール

日付	フェーズ	詳細
12/2	要件定義・設計	プロジェクト目的と目標の確認
		機能要件と非機能要件の最終決定
		システムアーキテクチャの設計
		データベース設計
12/3	フロントエンド開発	index.html の作成
		style.css、dark-theme.css、light-theme.css の作成
		ユーザーインターフェースの設計と実装
		JavaScript (script.js)の実装
12/4	バックエンド開発	Flask アプリケーションのセットアップ
		app.py の作成
		データベース操作(SQLite)の実装
		ルーティングとビューの設定
12/5	統合とテスト	フロントエンドとバックエンドの統合
		ユニットテストの作成と実行
		バグ修正と機能の最終調整
		ユーザビリティテスト
12/6	デプロイと最終チェック	最終動作確認
		ドキュメントの作成
		プロジェクトの振り返りと今後の計画

リスクと対策

1. 要件変更問題
   - リスク：要件が頻繁に変更される可能性があるため、スケジュールに遅れが生じる
   - 対策：事前にすべてのステークホルダーと要件を明確にし、要件が変更される場合は変更管理プロセスを導入する。変更が発生した場合のスケジュール調整を迅速に行う。
2. テスト不足問題
   - リスク：十分なテストを行わずにリリースした場合、バグや不具合が残る可能性がある。
   - 対策：詳細なテスト計画を作成し、ユニットテスト、統合テスト、ユーザビリティテストを徹底的に実施する。

品質と管理

1. テスト戦略
   - ユニットテスト
     - 目的：各機能が個別に正しく動作することを確認します。
     - 対象：Python の関数、Flask のルート、JavaScript の機能など。
   - 結合テスト
     - 目的：複数のコンポーネントが連携して正しく動作することを確認。
     - 対象：フロントエンドとバックエンドの連携、データベース操作など。
   - ユーザビリティテスト
     - 目的：ユーザーインターフェースが直感的で使いやすいことを確認
     - 対象：UI の操作、入力フォーム、モーダルウィンドウ、テーマ切替
2. 品質基準
   - 機能性
     - すべての主要機能が仕様通りに動作する。
   - 保守性
     - コードベースが読みやすく、ドキュメントが充実している。
   - ユーザビリティ
     - ユーザーからのフィードバックを基に継続的に UI を改善。

納品物

1. 納品物の概要
   - タスク管理アプリの完全版
2. 納品予定日
   - 2024/12/18

レビューと反省

作業を始めるといろいろな機能を足したくなった。
設計の段階で、どこまでの機能を実装をするか明確にすること。

詳細設計書

機能構成

	機能一覧	機能の説明
1	タスクの作成	ユーザーは新しいタスクを作成できる。 タイトル、説明、優先度、期限日を入力
2	タスクの編集	タスクの完了状態の更新が可能。
3	タスクの削除	ユーザーは不要なタスクを削除できる。
4	タスクの閲覧	ユーザーはタスクの一覧を閲覧できる。 各タスクの詳細が確認可能。
5	タスクの並び替え	ユーザーはタスクを期限日 （昇順および降順）、優先度（昇順および降順） で並び替えることができる。
6	詳細表示	モーダルウィンドウを使用して、 各タスクの詳細説明を表示する機能
7	テーマ切り替え	ユーザーはダークテーマとライトテーマを 切り替えることができる。
8	色選択	ユーザーはカラーピッカーを使用して、 インターフェースの差し色を変更できる
9	フォームバリデーション	ユーザーが入力した期限日などの情報の形式 が正しいかをチェックするバリデーション機能。
10	セッション管理	ユーザーセッションを管理し、 セッションの有効期間を設定する機能。

シーケンス

- ユーザー -> ブラウザ: タスク作成リクエスト送信
- ブラウザ -> サーバー: HTTP リクエスト送信
- サーバー -> データベース: 新しいタスクを保存
- サーバー -> ブラウザ: タスク作成成功メッセージ
- ブラウザ -> ユーザー: タスク作成成功通知

- ユーザー -> ブラウザ: タスク編集リクエスト送信
- ブラウザ -> サーバー: HTTP リクエスト送信
- サーバー -> データベース: タスクの完了状態を更新
- サーバー -> ブラウザ: タスク編集成功メッセージ
- ブラウザ -> ユーザー: タスク編集成功通知

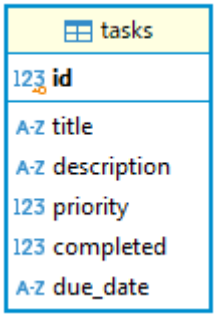
- ユーザー -> ブラウザ: タスク削除リクエスト送信
- ブラウザ -> サーバー: HTTP リクエスト送信
- サーバー -> データベース: タスクを削除
- サーバー -> ブラウザ: タスク削除成功メッセージ
- ブラウザ -> ユーザー: タスク削除成功通知

- ユーザー -> ブラウザ: タスク一覧リクエスト送信
- ブラウザ -> サーバー: HTTP リクエスト送信
- サーバー -> データベース: タスク一覧を取得
-
- サーバー -> ブラウザ: タスク一覧データ送信
- ブラウザ -> ユーザー: タスク一覧表示

テーブル定義

カラム名	データ型	制約	説明
id	INTEGER	PRIMARY KEY AUTOINCREMENT	タスクの一意識別子
title	TEXT	NOT NULL	タスクのタイトル
description	TEXT		タスクの詳細説明
priority	INTEGER	NOT NULL	タスクの優先度(1～5)
completed	BOOLEAN	NOT NULL CHECK (completed IN (0, 1))	タスクの完了状態 (0:未完了、1:完了)
due_date	TEXT		タスクの期限日 (YYYY-MM-DD 形式)

ER 図



テスト表

テストケース番号	テスト項目	テスト内容	テストデータ	期待結果	実行結果
TC001	タスク作成	新しいタスクを作成し、データベースに正しく保存されるか	タイトル: "買い物", 説明: "牛乳を買う", 優先度: 3, 期限日: "2024-12-10"	タスクがデータベースに保存され、一覧に表示される	○
TC002	タスク編集	既存のタスクの完了状態を更新し、データベースに反映されるか確認	タスクタイトル: 買い物, 完了状態: "完了"	タスクの完了状態がデータベースに更新され、一覧に反映される	○
TC003	タスク削除	既存のタスクを削除し、データベースから正しく削除される	タスクタイトル：買い物	タスクがデータベースから削除され、一覧からも消える	○
TC004	タスク並び替え	タスク一覧を期限日や優先度で並び替えできるか確認する	並び替え基準: 期限日昇順	タスクが指定した基準で正しく並び替えられる	○
TC005	詳細表示	タスクの詳細をモーダルウィンドウで表示できるか確認する	タスク ID: 1	タスクの詳細がモーダルウィンドウで表示される	○
TC006	テーマ切り替え	ダークテーマとライトテーマを正しく切り替えできるか確認する	テーマ: ダーク、ライト	テーマが正しく切り替わり、画面に反映される	○
TC007	色選択	カラーピッカーで差し色を変更し、画面に正しく反映されるか確認する	色：レッド	差し色が変更され、画面に反映される	○
TC008	フォームバリデーション	期限日が正しい形式で入力されているか確認する	期限日: "2024-12-32"	不正な期限日が入力された場合、エラーメッセージが表示される	○
TC009	セッション管理	ユーザーセッションが正しく管理されているか確認する	セッション開始、セッション終了	セッションが有効期間内に管理され、セッション切れがない	○

操作手順書

スタート画面

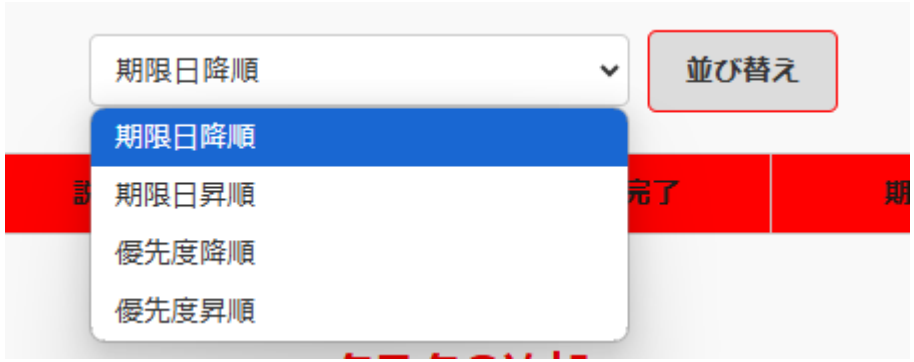
① 背景・差し色の変更
② 並べ替え
③ タスクリスト
④ 新規タスク入力画面

① 背景・差し色の変更

背景を light-theme、差し色をレッドにしたスタート画面

② 並べ替え

- タスクを並べ替える機能



- 期限日降順：期限日が遠い順に並べ替えます
- 期限日昇順：期限日が近い順に並べ替えます
- 優先度降順：優先度が高い順（5→1）に並べ替えます
- 優先度昇順優先度が低い順（1→5）に並べ替えます

※ 必ず「並び替え」ボタンをクリックしてください

③ タスクリスト

- 例）タイトル：買い物
  説明：牛乳を買う
  優先度：3
  期限：2024/12/10
  - 「詳細」ボタンをクリックすると、入力したタスクの説明を確認することができます。

- 「更新」ボタンの隣のチェックボタンをクリックし、
  「更新」ボタンをクリックすると情報が更新され、青いチェックが入る
  ※必ず「更新」ボタンをクリックしてください。

- 「削除」ボタンをクリックするとタスクが削除されます

④　新規タスク入力画面

① タイトル入力フォーム

- 追加したいタスクのタイトルを追加してください
  ※必ず入力してください
  例）買い物

② 説明入力フォーム

- 追加したいタスクの説明を入力してください
  ※必ず入力してください
  例）牛乳を買う

③ 優先度

- 1～5 を選択できるようにしてあります

④ 期限日入力フォーム

- 締切の日付を入力してください
  ※必ず入力してください

⑤ 追加ボタン

- すべての入力確認後にクリックするとタスクが追加されます。

プロジェクト概要

プレゼン資料

目的と背景

目的

- 自分専用のタスク管理システムを開発し、タスクの作成、編集、削除、および閲覧を効率的に行えるようにする。

背景

- 市販のタスク管理アプリにはたくさんの選択肢がありますが、すべてのユーザーのニーズを完全に満たすものは少ないです。特に、自分の好みに合わせてカスタマイズできるアプリケーションは限られています。そこで、個々のユーザーが求める機能やデザインを取り入れ、自分好みのタスク管理システムを開発することを背景に、このプロジェクトを立ち上げました。

ターゲットユーザー

ユーザー

- ユーザー層
  - 一般のユーザー
- ユーザーのニーズ
  - カスタマイズ性の高いインターフェース
  - 効率的なタスク管理
  - 視覚的に魅力的なデザイン

機能要件

プレゼン資料

主な要件

タスクの作成	タスクの編集	タスクの削除	タスクの閲覧
タスクの並び替え	詳細表示	テーマ切り替え	色選択

非機能要件

ユーザビリティ

ユーザーインターフェースは直感的で使いやすく、モバイルデバイスにも対応しています。

保守性

コードは読みやすく、適切にコメントとドキュメントが付けられています。

システム構成

技術スタック

クライアントサイド	サーバーサイド	開発ツール
・HTML5 ・CSS3 ・JavaScript	・Python 3 ・Flask ・SQLite	・Spyder

アーキテクチャ

クライアントサイド	サーバーサイド	サーバーインフラ
・HTML/CSS: ユーザーインターフェースの構築に使用されます。 ・JavaScript: ユーザーインターフェースの動作やインタラクションを制御します。	・Flask: Pythonベースのマイクロウェブフレームワークで、アプリケーションのロジックとルーティングを処理します。 ・SQLite: 軽量なデータベース管理システムで、タスクデータを永続的に保存します。	・Webサーバー: Flaskアプリケーションをホストするためのサーバー

開発スケジュール

フェーズごとのスケジュール

プレゼン資料

リスクと対策

リスク	対策
・要件変更のリスク ・要件が頻繁に変更される可能性があり、スケジュールに遅れが生じるリスクがあります。 ・テスト不足のリスク ・十分なテストを行わずにリリースした場合、バグや不具合が残る可能性があります。	・要件変更のリスク ・対策: 事前にすべてのステークホルダーと要件を明確にし、要件が変更される場合は変更管理プロセスを導入します。変更が発生した場合のスケジュール調整を迅速に行います。 ・テスト不足のリスク ・対策: 詳細なテスト計画を作成し、ユニットテスト、統合テスト、ユーザビリティテストを網羅的に実施します。

リスクと対策

品質基準とテスト

テスト戦略

ユニットテスト
- 目的: 各機能が個別に正しく動作することを確認します。
- 対象: Pythonの関数、Flaskのルート、JavaScriptの機能など。

統合テスト
- 目的: 複数のコンポーネントが連携して正しく動作することを確認します。
- 対象: フロントエンドとバックエンドの連携、データベース操作など。

ユーザビリティテスト
- 目的: ユーザーインターフェースが直感的で使いやすいことを確認します。
- 対象: UIの操作、入力フォーム、モーダルウィンドウ、テーマ切り替えなど。

プレゼン資料

デモ

品質基準

機能性
- すべての主要機能が仕様通りに動作する。
- ユーザーインターフェースが直感的で使いやすい。

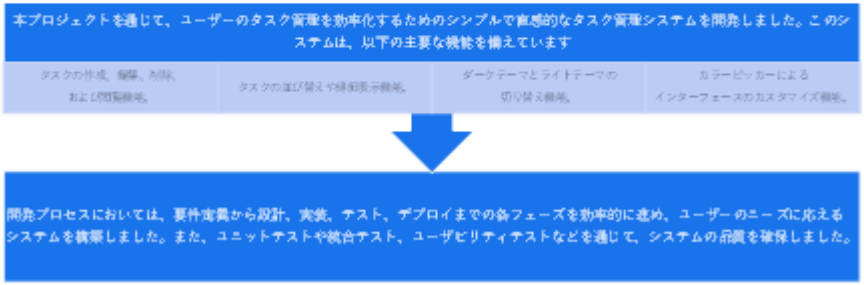
保守性
- コードベースが読みやすく、ドキュメントが充実している。

ユーザビリティ
- ユーザーからのフィードバックを基に継続的にUIを改善。

結論

プレゼン資料

総括



改善点