Markdown to PDF

Contents

1　マークダウン →PDF 変換 GUI 管理ツール開発計画作成プロンプト

以下の要件に基づいて、マークダウンファイルを PDF に変換するための GUI 管理ツールの開発計画を作成してください。

1.1　現状の問題点

1. **大量のマークダウンファイル**: `/Users/yuto/itphy` 配下に 30 以上のマークダウンファイルが存在し、それぞれ異なるディレクトリに配置されている
2. **個別の変換方法**: 各ファイルごとに異なる変換スクリプト（`convert_to_pdf.sh`、`convert_to_pdf_simple.s`など）や設定が必要
3. **管理の複雑さ**: 変換方法が統一されておらず、手動でスクリプトを実行する必要がある
4. **設定の分散**: テンプレートファイル（`pandoc_template.tex`）、ヘッダーファイル（`pandoc_header.tex`）が各ディレクトリに散在している可能性

1.2　既存の変換方法

現在、以下の技術スタックが使用されています：

- **変換エンジン**: Pandoc + XeLaTeX
- **フォント**: Hiragino Sans（日本語対応）
- **テンプレート**: LaTeX テンプレート（`pandoc_template.tex`）
- **ヘッダー**: LaTeX ヘッダー（`pandoc_header.tex`）
- **主要オプション**:
  - `--pdf-engine=xelatex`
  - `--from=markdown+tex_math_dollars+raw_tex`
  - 目次生成（`--toc`, `--toc-depth=2`）
  - ハイパーリンク対応（`colorlinks=true`）
  - 図の相対パス対応（`figures/`ディレクトリ）

1.3　要件

1.3.1　基本機能

1. **GUI インターフェース**: 直感的で使いやすいグラフィカルユーザーインターフェース
2. **マークダウンファイル選択**: ファイルブラウザまたはドラッグ & ドロップでマークダウンファイルを選択
3. **自動 PDF 変換**: 選択したマークダウンファイルを自動的に PDF に変換
4. **変換履歴の管理**: 変換したファイルの履歴を記録・表示

1.3.2　技術的要件（重要）

以下の要素を適切に処理できる必要があります：

1. 特殊文字の処理

- 日本語文字（ひらがな、カタカナ、漢字）の正確な表示
- 絵文字や特殊記号（☒、☒ など）の適切な変換
- 各種記号の文字化け防止

2. LaTeX 数式の処理

- インライン数式（`$...$`）の正確なレンダリング
- ブロック数式（`$$...$$`）の適切な配置
- 複雑な数式環境（`align`、`equation` など）の対応

- 数式の改行処理（`\allowdisplaybreaks`）

3. 図と画像の処理

- 相対パスでの画像参照（`figures/`ディレクトリなど）
- 絶対パスでの画像参照
- PNG、JPEG、SVG などの複数フォーマット対応
- 画像のサイズ調整と配置
- 画像が見つからない場合のエラーハンドリング

4. 目次（Table of Contents）の生成

- 自動目次生成機能
- 目次の深さ設定（`--toc-depth`）
- 目次のスタイリング（色、フォントなど）

5. ハイパーリンクの処理

- 内部リンク（アンカーリンク）の保持
- 外部 URL リンクの保持
- リンクの色設定と視認性
- PDF 内でのリンクの動作確認

1.3.3 追加機能（推奨）

1. **設定の保存**: 変換設定（フォント、マージン、目次深度など）をプロファイルとして保存
2. **バッチ変換**: 複数のマークダウンファイルを一度に変換
3. **プレビュー機能**: 変換前のマークダウン内容のプレビュー
4. **エラーログの表示**: 変換エラーの詳細表示とトラブルシューティング支援
5. **出力先の指定**: PDF の保存先ディレクトリを指定可能
6. **テンプレート管理**: 複数の LaTeX テンプレートを管理・選択可能

1.4 開発計画に含めるべき項目

以下の観点から詳細な開発計画を作成してください：

1. **技術スタックの選定**
   - GUI フレームワーク（Python: Tkinter/PyQt、Electron、Web アプリなど）
   - 推奨技術とその理由
2. **アーキテクチャ設計**
   - コンポーネント構成
   - データフロー
   - 設定管理の方法
3. **実装フェーズ**
   - フェーズ 1: 基本機能（ファイル選択、単一変換）
   - フェーズ 2: 高度な機能（バッチ変換、設定保存）
   - フェーズ 3: UI 改善とエラーハンドリング
4. **特殊文字・LaTeX・図・目次・リンクの処理方法**
   - 各要素の処理アルゴリズム
   - 既存の Pandoc 設定の活用方法
   - エッジケースの対応
5. **テスト計画**
   - テストケース（特殊文字、複雑な数式、複数画像など）
   - 既存マークダウンファイルでの検証方法

6. **既存スクリプトとの統合**
    - 既存の `convert_to_pdf.sh` などの活用方法
    - 後方互換性の確保
7. **ユーザビリティ**
    - UI/UX 設計の考慮事項
    - エラーメッセージの分かりやすさ
8. **将来の拡張性**
    - 他の変換形式への対応（HTML、Word など）
    - プラグイン機能の検討

1.5 出力形式

開発計画は以下の形式で出力してください：

- **概要**: プロジェクトの全体像
- **技術選定**: 推奨技術スタックと理由
- **詳細設計**: 各機能の実装方法
- **実装スケジュール**: フェーズごとの実装内容と期間見積もり
- **リスクと対策**: 想定される問題と解決策
- **テスト計画**: 検証方法とテストケース

このプロンプトに基づいて、実用的で保守性の高い GUI 管理ツールの開発計画を作成してください。なお、わからないことはすべて質問してください。