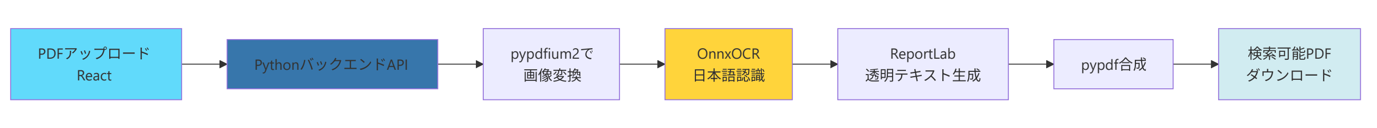
OCR検索可能PDF変換 Webアプリ

License MIT demo GitHub Pages version 1.0.0

スキャンしたPDFをOnnxOCRで高精度にOCR処理し、検索可能なテキストレイヤーを追加するWebアプリケーション

特徴

✅ **OnnxOCR採用** - CPU推論で高速かつ高精度なOCR処理
✅ **Python + Reactハイブリッド** - バックエンドでPython、フロントエンドでReact
✅ **日本語OCR最適化** - PaddleOCRベースの日本語特化モデル
✅ **高精度テキスト抽出** - Tesseract.jsより2-3倍高速で精度も向上
✅ **複数ページ対応** - バッチ処理でリアルタイム進捗表示
✅ **ファイル制限** - フロント側は 10MB まで（バックエンド受信上限は 50MB だが運用上は 10MB）
✅ **透明テキストレイヤー** - ReportLabで完全透明なテキストレイヤーを合成



技術スタック

バックエンド (Python 3.10.11)

- **OnnxOCR**: 高速CPU推論OCRエンジン
- **pypdfium2**: PDFレンダリング
- **pypdf**: PDF合成
- **ReportLab**: 透明テキストレイヤー生成
- **Flask**: REST APIサーバー
- **OpenCV + NumPy**: 画像前処理

フロントエンド

- **React 18**: UIフレームワーク
- **Webpack 5**: モジュールバンドラー

デモ

🌐 **ライブデモ（UIのみ）**: https://j1921604.github.io/OCR-PDF-Converter/

※GitHub Pages（HTTPS）上のフロントエンドから `http://localhost:5000` を呼ぶことは mixed content でブロックされるため、Pages単体ではOCR処理は動きません。OCRを動かす場合はローカル起動してください。

クイックスタート

前提条件

- Python 3.10.11
- Node.js 18以上
- npm または yarn

ワンコマンド起動（PowerShell - 推奨）

```
.\start-full.ps1
```

このスクリプトは以下を自動実行します：

1. Python 3.10.11とNode.jsのインストール確認
2. 依存関係のインストール（Python + npm）
3. Pythonバックエンド起動 (http://localhost:5000)
4. Reactフロントエンド起動 (http://localhost:8080)

サーバーを停止するには `Ctrl+C` を押します。

手動セットアップ

1. バックエンド起動

```
py -3.10 -m venv .venv
.\.venv\Scripts\Activate.ps1
py -3.10 -m pip install --upgrade pip
py -3.10 -m pip install -r requirements.txt
python backend\app.py
```

2. フロントエンド起動（別ターミナル）

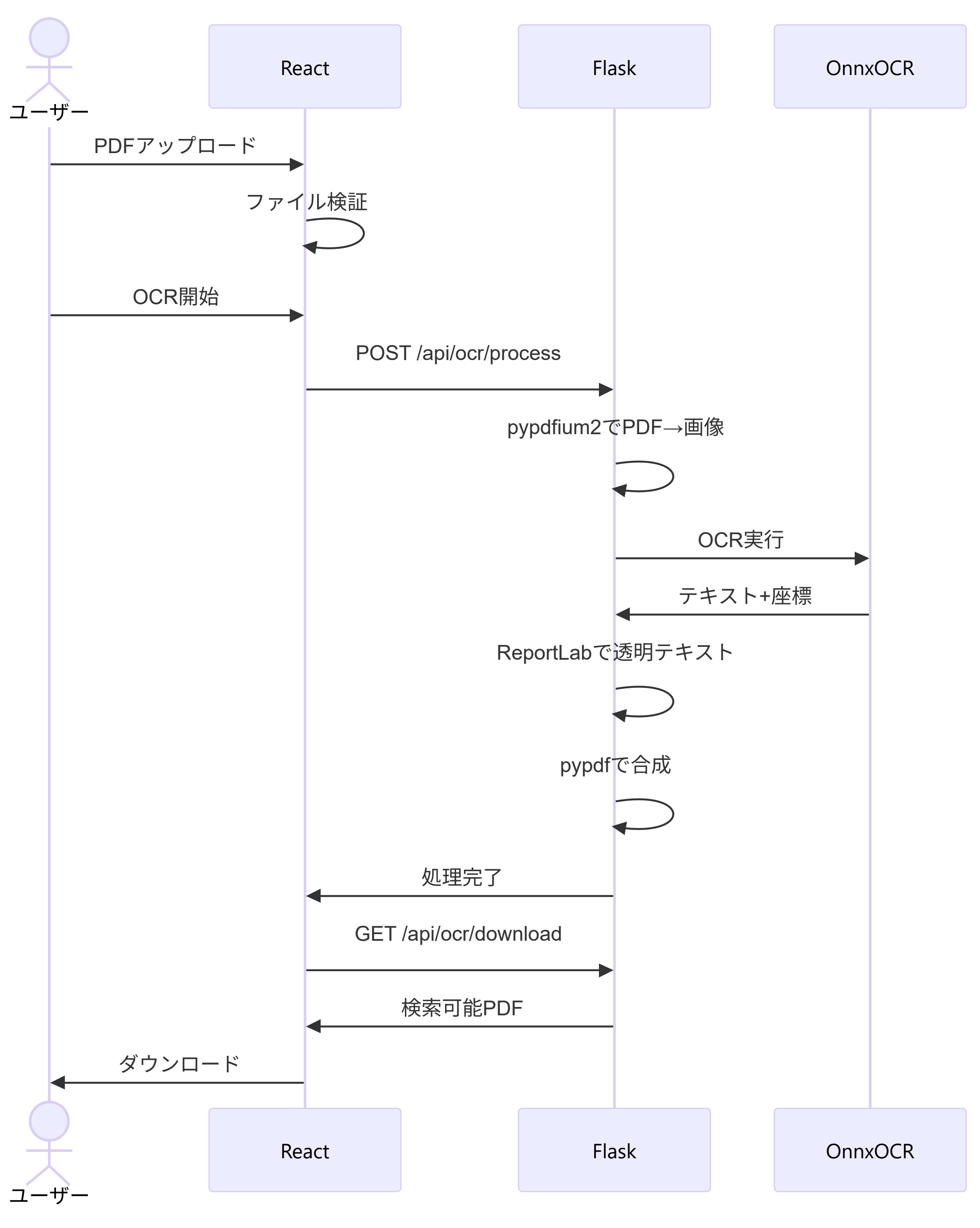
```
npm install
npm start
```

ブラウザで `http://localhost:8080` を開きます。

使い方

1. **ファイルを選択**
   「ファイルを選択」ボタンをクリックし、スキャンしたPDFファイル（10MB以下）を選択します。
   対応形式: PDF / JPEG / PNG / TIFF（画像はフロント側でPDFに変換してから送信します）

2. **OCR変換開始**
   「OCR変換開始」ボタンをクリックすると、Pythonバックエンドで高精度OCR処理が開始されます。



技術スタック

バックエンド (Python 3.10.11)

カテゴリ	ライブラリ	バージョン	用途
OCRエンジン	OnnxOCR	2025.5+	高速CPU推論OCR
PDFレンダリング	pypdfium2	4.30+	PDF→画像変換
PDF合成	pypdf	5.1+	PDFページ操作
テキストレイヤー	ReportLab	4.2+	透明テキスト生成
Web API	Flask	3.0+	REST APIサーバー
画像処理	OpenCV + NumPy	4.10+	前処理

フロントエンド

カテゴリ	ライブラリ	バージョン	用途
UIフレームワーク	React	18.2+	SPA構築
モジュールバンドラー	Webpack	5.104+	ビルドツール

進捗バーでリアルタイムに処理状況を確認できます。

3. **検索可能PDFをダウンロード**
   処理完了後、「ダウンロード」ボタンから検索可能なPDFファイルを保存します。

4. **テキスト検索**
   ダウンロードしたPDFをPDFビューアー（Adobe Acrobat Reader等）で開き、 `Ctrl+F` （Windows）または `Cmd+F` （Mac）でテキスト検索が可能です。

プロジェクト構造

```
OCR-PDF-Converter/
├── backend/                   # Pythonバックエンド
│   ├── app.py                # Flask APIサーバー
│   ├── main.py               # OCRエンジン実装
│   └── （注）requirements.txt はリポジトリ直下
├── specs/                     # 仕様ドキュメント
│   └── 001-OCR-PDF-Converter/
│       ├── spec.md           # 機能仕様
│       ├── requirements.md   # 技術要件
│       ├── tasks.md          # タスクリスト
│       └── checklists/       # 品質チェックリスト
├── src/                       # Reactフロントエンド
│   ├── components/           # UIコンポーネント
│   ├── hooks/                # カスタムHook
│   ├── services/             # API連携サービス
│   └── utils/                # ユーティリティ関数
├── public/                    # 静的ファイル
├── cypress/                   # E2Eテスト
│   └── e2e/
│       └── ocr-converter.cy.js
├── tests/                     # 単体テスト
│   └── unit/
├── .github/                   # GitHub設定
│   └── workflows/
│       └── pages.yml         # GitHub Actionsデプロイ
├── start-full.ps1            # ワンコマンド起動スクリプト
├── package.json              # npm依存関係
└── README.md                 # このファイル
```

開発

ブランチ戦略

プロジェクト憲法（ `.specify/memory/constitution.md` ）に従い、基本は **main単一運用**です。 必要に応じて短命の作業ブランチ（例: `wip/<topic>` ）を切ってもよいですが、最終的にmainへマージしてブランチは削除します。

開発ワークフロー

1. **憲法確認**: https://github.com/J1921604/OCR-PDF-Converter/blob/main/.specify/memory/constitution.md を読む
2. **仕様作成**: `specs/001-OCR-PDF-Converter/spec.md` で要件定義
3. **実装**: `feature/impl-001-OCR-PDF-Converter` ブランチで開発
4. **テスト**: 単体テスト → 統合テスト → E2Eテスト
5. **レビュー**: コードレビューと仕様整合性確認
6. **マージ**: 仕様ブランチ → main

コマンド

```
# 開発サーバー起動
npm start

# ビルド（本番用）
npm run build

# テスト実行
npm test

# Lint実行
npm run lint

# フォーマット
npm run format
```

GitHub Pages デプロイ

GitHub Actionsで自動デプロイされます。

※デプロイされるのはフロントエンド（静的ファイル）のみです。OCR処理にはバックエンドが必要です。

```
# .github/workflows/pages.yml
on:
  push:
    branches: [ main ]
```

`main` ブランチにプッシュすると、自動的にビルド→デプロイされます。

トラブルシューティング

1. サーバーが起動しない

症状: `npm start` または `.\start-dev.ps1` 実行後、サーバーが自動停止する

解決方法:

- まず `\.\start-full.ps1` の利用を推奨します（同一ウィンドウ内で安定起動/停止します）。
- 8080/5000番ポートが他プロセスで使用中の場合は、先に停止してから再実行してください。

2. CSP violation エラー

症状: ブラウザコンソールに "Content Security Policy directive" エラーが表示される

原因: CSP設定が不足している

確認方法:

- F12キー → Console タブでエラーメッセージを確認
- `public/index.html` の CSP meta tagを確認

補足: API接続先（ `connect-src` ）やワーカー（ `worker-src` ）の許可が不足していると発生します。

3. PDFファイル読み込みエラー

症状: PDFをアップロード後、エラーメッセージが表示される

原因:

- ファイルサイズが10MBを超えている
- 破損したPDFファイル
- 暗号化されたPDFファイル

解決方法:

- ファイルサイズを確認（10MB以下）
- 別のPDFで試す
- 暗号化を解除してから再試行

4. OCR精度が低い

原因:

- スキャン解像度が低い（推奨: 300dpi以上）
- 画像が斜めになっている
- 低品質なスキャン画像

解決方法:

- OnnxOCRは自動的に300dpiで処理します
- 高解像度でスキャンし直す
- コントラストを高める

パフォーマンス指標

- **1ページPDF処理時間**: 5秒以内（P95、OnnxOCR CPU推論）
- **10ページPDF処理時間**: 50秒以内（P95）
- **メモリ使用量**: Python 512MB、React 256MB（ピーク時）
- **ファイルサイズ制限**: 10MB（フロント制限。バックエンドは50MB設定だが運用上10MB）

よくある質問（FAQ）

Q1: バックエンドサーバーはどこで動作しますか？
A: Pythonバックエンドはローカル環境（localhost:5000）で動作します。サーバーへのファイル送信は行われません。

Q2: 処理できるファイルサイズの上限は？
A: フロント側は 10MB までです（バックエンドは 50MB まで受け付けますが、運用上は 10MB を上限としています）。

Q3: 日本語以外の言語も対応していますか？
A: OnnxOCRは多言語対応（日本語、英語、中国語）ですが、現在は日本語に最適化しています。

Q4: 商用利用は可能ですか？
A: はい。MITライセンスで公開しているため、商用利用可能です。

Q5: オフラインで使用できますか？
A: Python環境とOnnxOCRモデルが事前にインストールされていれば、完全オフラインで使用可能です。

ブラウザサポート

ブラウザ	最小バージョン
Chrome	100+
Firefox	100+
Edge	100+
Safari	15+

ライセンス

MIT License

コントリビューション

プルリクエストを歓迎します！

謝辞

このプロジェクトは以下のオープンソースライブラリを使用しています：

- OnnxOCR - 高速CPU推論OCRエンジン
- pypdfium2 - PDFレンダリング
- pypdf - PDF操作
- ReportLab - PDFテキストレイヤー生成
- React - UIフレームワーク

リンク

- 📖 **仕様書**: https://github.com/J1921604/OCR-PDF-Converter/blob/main/specs/001-OCR-PDF-Converter/spec.md
- 🛠 **技術要件**: https://github.com/J1921604/OCR-PDF-Converter/blob/main/specs/001-OCR-PDF-Converter/requirements.md
- ✅ **チェックリスト**: https://github.com/J1921604/OCR-PDF-Converter/blob/main/specs/001-OCR-PDF-Converter/checklists/requirements.md
- 📜 **プロジェクト憲法**: https://github.com/J1921604/OCR-PDF-Converter/blob/main/.specify/memory/constitution.md

---

作成日: 2026-1-15
バージョン: 1.0.0
メンテナ: J1921604