**# 在留資格管理システム - インストールと設定マニュアル**

## 1. 前提条件

- Python 3.11/3.12/3.13 のいずれかがインストール済みであること

- プロジェクト一式が taskn-python-app4-master フォルダに揃っていること

- ネットワーク経由で必要なライブラリをインストールできること

## 2. 主要ファイル

- `app.py` - Flask アプリケーション本体

- `requirements.txt` - 必要パッケージ一覧

## 3. 環境設定手順

### 3.1 PowerShell を管理者権限で起動

1. スタートメニューで「PowerShell」と入力します

2. 表示された PowerShell を右クリックし、「管理者として実行」を選択します

3. プロジェクトフォルダに移動します:

```powershell

cd "C:\path\to\taskn-python-app4-master"

```

### 3.2 仮想環境の準備（推奨）

1. 既存の仮想環境がある場合は削除します:

```powershell

Remove-Item .venv -Recurse -Force

```

2. 新しい仮想環境を作成します: 任意

```powershell

python -m venv .venv

```

3. 仮想環境を有効化します:

```powershell

.\.venv\Scripts\Activate.ps1

```

コマンド実行後、プロンプトに `(.venv)` が表示されれば有効化に成功しています。仮想環境を解除する場合は `deactivate` コマンドを実行します。

### 3.3 依存パッケージのインストール

仮想環境が有効になっている状態で、以下のコマンドを順番に実行します:

```powershell

python -m pip install --upgrade pip

python -m pip install -r requirements.txt

```

`requirements.txt` には Flask や pandas など必要なライブラリがまとめて記載されています。schedule ライブラリ（定期実行用）もここでインストールされます。

### 3.4 証明書（SSL）関連の準備（必要に応じて）

Web スクレイピングで証明書エラーが発生する場合、次の順で対応すると効果的です:

1. PC の日時、Windows Update、ブラウザの警告有無を確認します

2. certifi パッケージを更新します:

```powershell

python -m pip install --upgrade certifi

```

3. 必要に応じて、`immigration\_scraper.py` ファイルの `ImmigrationScraper.init` メソッド内で次の設定を行います:

```python

self.session.verify = certifi.where()

```

## 4. アプリケーションの起動と操作

**### 4.1 Flask アプリケーションの起動**

**1. 仮想環境が有効な状態で、次のコマンドを実行します:**

**```powershell**

**python app.py**

**```**

2. 起動に成功すると、コンソールに次の案内が表示されます:

```

================================================================================

在留資格管理システム - Web版

================================================================================

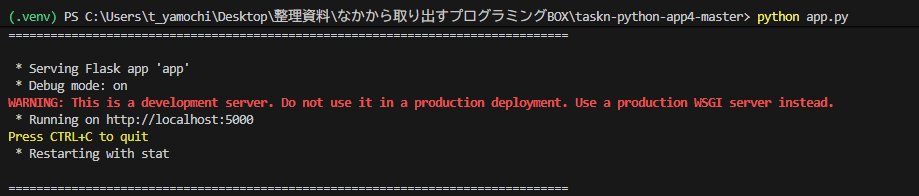
ブラウザで以下のURLにアクセスしてください:

http://localhost:5000

終了するには Ctrl+C を押してください

================================================================================

```

仮想環境は

赤文字が出ますが問題ないです

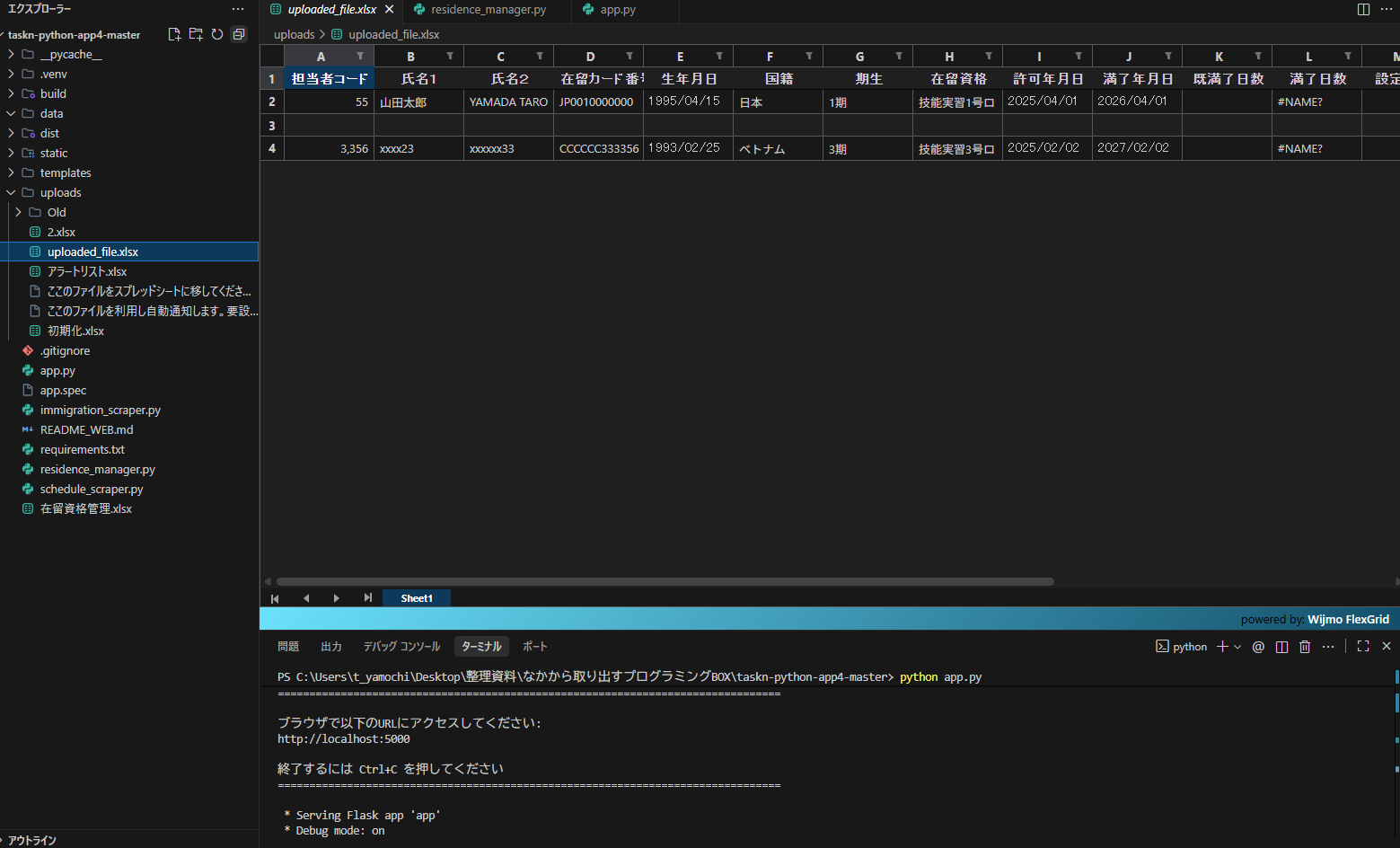
3. ブラウザで `http://localhost:5000` にアクセスし、画面が表示されることを確認します

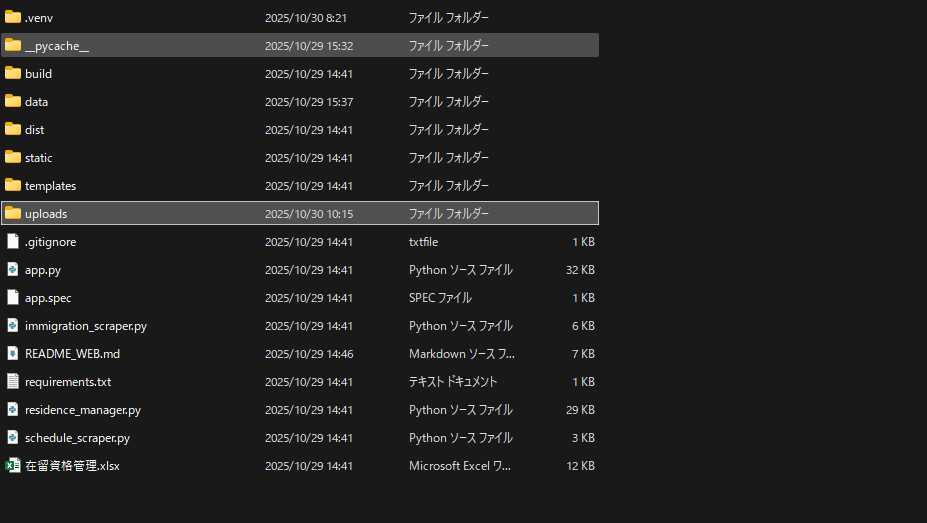
4. アプリケーションを終了するには、PowerShell 上で `Ctrl + C` キーを押します

### 4.2 Excel データの読み込み

1. アプリケーション画面から `.xlsx` ファイルをアップロードします

アップロード用は \taskn-python-app4-master\uploads





- アップロードすると、サーバ側で `load\_excel()` → `process\_data()` の順に処理が実行されます

- これらの処理は `app.py` と `residence\_manager.py` で定義されています

2. アップロード成功時には PowerShell に読み込んだ行数などの `[DEBUG]` ログが表示されます

3. 読み込みに失敗した場合は、表示されるエラーメッセージを参照し、以下の点を確認します:

- ファイルが `.xlsx` 形式であるか

- ファイルが他のアプリケーションで開かれていないか

- 必須列（例：満了年月日）が欠けていないか

**## 5. スクレイパー実行（任意）**

### 5.1 手動でのスクレイピング実行

1. 次のコマンドを実行します:

```powershell

python immigration\_scraper.py

```



2. 実行結果:

- 成功時: 取得件数がログに表示され、`data/immigration\_status\_updates\_YYYYMMDD.csv` ファイルが生成されます

- エラー発生時（SSL など）: 前述の「3.4 証明書関連の準備」セクションを参照してください

INFO - 在留資格関連の更新情報を取得中...

INFO - ニュースを取得中: https://www.moj.go.jp/...

INFO - 更新情報を 3 件見つけました: data/immigration\_status\_updates\_20251029.csv

### 5.2 定期実行（スケジューラ）の設定

1. 次のコマンドを実行します:

```powershell

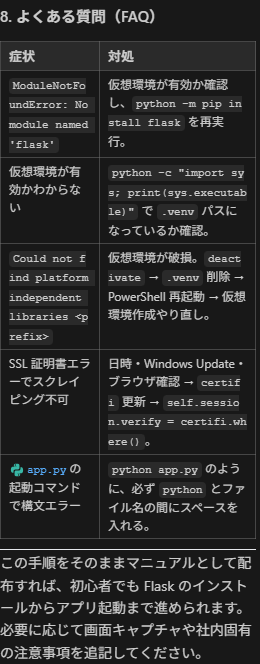
python schedule\_scraper.py

```

2. 実行すると、毎月1日と15日の9:00に自動的にスクレイピングが実行されます

3. スケジューラを停止するには `Ctrl + C` キーを押します

## 6. トラブルシューティング（よくある質問）



７.画面の説明

ファイルを開く。

→### 4.2 Excel データの読み込み のもの

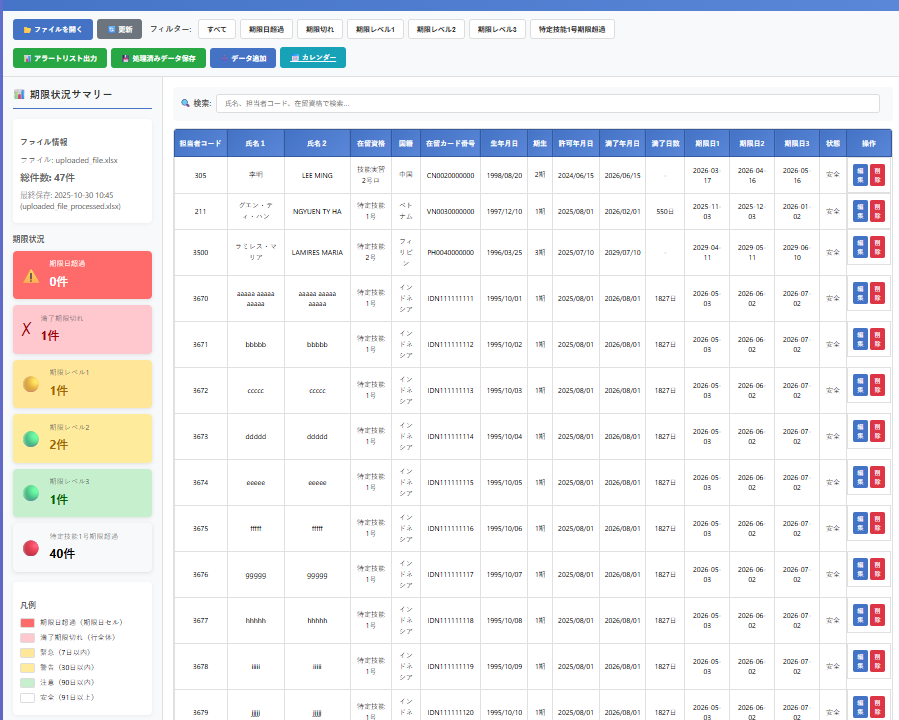
以下ファイルパスのエクセルを選択

\taskn-python-app4-master\uploads

テスト環境ではテスト環境読み込みエクセルシート.xlsxを

こちらのエクセルに





期限日クリックすることでカレンダーの日付にジャンプ

直接表の値を手直しも可能。

セーブする前にアプリを終了する前は警告がはいるため処理済みデータ保存で保存するように

勝手にuploadsのフォルダにいれます。

こちらがPythonを利用した理由

Javascriptだけでは実現出来ないことが多い・・・

（パソコンのダウンロードフォルダを凄く圧迫します）

詳しくは動画（タスクン.gif)を見てください

7.1各ボタンの紹介

アプリ上の各ボタンの詳細説明

このアプリには、リスト画面（メイン画面）とカレンダー画面の2つの主な画面があります。各画面のボタンを以下に詳しく説明します。

リスト画面（メイン画面）のボタン

ツールバー上部のボタン（左側）

📂 ファイルを開く

機能: Excelファイル（.xlsx形式）を選択してアップロード・読み込み。

動作: クリックするとファイル選択ダイアログが開き、選択したExcelファイルをサーバーに送信。データが読み込まれ、テーブルに表示される。

注意: 初回使用時は必ずこのボタンからデータを読み込む必要がある。アップロード成功後、データが自動的にテーブルに反映される。

🔄 更新

機能: 現在のデータを再読み込み。

動作: サーバーから最新のデータを取得し、テーブルを更新。変更が反映されない場合やデータをリフレッシュしたい場合に使用。

注意: 未保存の変更がある場合は警告が表示される（処理済みデータ保存を推奨）。

ツールバー中央のフィルターボタン（中側）

すべて

機能: フィルターを解除し、全データを表示。

動作: クリックすると全データが表示され、ステータスバーに「フィルター: すべて表示」と表示。

期限日超過

機能: 期限日1/2/3のいずれかが超過しているデータのみ表示。

動作: 満了年月日までの日数が設定期限以下の場合に該当。ステータスバーに該当件数が表示。

期限切れ

機能: 満了年月日が過ぎているデータのみ表示。

動作: 満了年月日までの日数が0未満の場合に該当。最も緊急度の高いデータ。

期限レベル1

機能: 設定期限1（デフォルト90日）以内のデータのみ表示。

動作: 最も余裕の少ない分類。ステータスバーに件数が表示。

期限レベル2

機能: 設定期限2（デフォルト60日）以内のデータのみ表示。

動作: 中程度の余裕がある分類。

期限レベル3

機能: 設定期限3（デフォルト30日）以内のデータのみ表示。

動作: 最も余裕がある分類。

特定技能1号期限超過

機能: 特定技能1号の在留資格で、満了日数 + 184 > 1825 の条件を満たすデータのみ表示。

動作: 特定技能1号の最大在留期間（5年 + 184日 = 1825日）を超過する可能性があるデータ。

ツールバー右側のボタン（右側）

📊 アラートリスト出力

機能: アラート対象データをExcelファイルとしてダウンロード。

動作: サーバーからアラートリストを生成し、ブラウザのダウンロード機能で保存。ファイル名は自動生成。

💾 処理済みデータ保存

機能: 計算済みのデータをExcelファイルとして保存・ダウンロード。

動作: サーバーで計算式を含むExcelファイルを生成し、ブラウザ経由でダウンロード。未保存の変更がある場合、ボタンが強調表示される。

注意: この保存を実行すると、未保存変更のマークがクリアされる。

➕ データ追加

機能: 新しいデータを追加するためのフォームを開く。

動作: クリックするとモーダルウィンドウが開き、各フィールドを入力可能。保存するとサーバーに送信され、テーブルに追加。

📅 カレンダー

機能: カレンダー画面へ遷移。

動作: クリックすると/calendarページへ移動。期限日をカレンダー形式で視覚化。

各行の操作ボタン（テーブル内）

編集

機能: 該当行のデータを編集。

動作: クリックするとモーダルウィンドウが開き、既存のデータを編集可能。保存で更新。

削除

機能: 該当行のデータを削除。

動作: クリックすると確認ダイアログが表示され、2段階の確認を経て削除。元に戻せない。

カレンダー画面のボタン

ツールバー上部のボタン（左側）

📂 ファイルを開く

機能: リスト画面と同様、Excelファイルを選択してアップロード。

動作: カレンダーデータも同時に更新される。

🔄 更新

機能: カレンダーデータを再読み込み。

動作: 現在の年月でカレンダーを更新。

ツールバー中央のナビゲーションボタン（中側）

◀◀ 前の年

機能: 表示年を1年戻す。

動作: 現在の年 - 1 の年を表示。

◀ 前の月

機能: 表示月を1ヶ月戻す。

動作: 月が1月の場合は前年の12月へ自動遷移。

次の月 ▶

機能: 表示月を1ヶ月進める。

動作: 月が12月の場合は翌年の1月へ自動遷移。

次の年 ▶▶

機能: 表示年を1年進める。

動作: 現在の年 + 1 の年を表示。

今日

機能: 現在の年月に戻る。

動作: システムの日付に基づいて今月を表示。

ツールバー右側のボタン（右側）

📊 アラートリスト出力

機能: リスト画面と同様、アラートデータをExcel出力。

💾 処理済みデータ保存

機能: リスト画面と同様、計算済みデータをExcel保存。

📋 リストに戻る

機能: リスト画面へ戻る。

動作: クリックするとメイン画面へ遷移。

モーダルウィンドウ内のボタン（データ追加・編集時）

保存

機能: 入力したデータをサーバーに送信・保存。

動作: フォームのバリデーションを行い、成功するとテーブル更新。特定技能1号の場合は既満了日数の必須チェックあり。

キャンセル

機能: モーダルを閉じて変更を破棄。

動作: 確認なしでウィンドウが閉じる。

追加機能：期限日クリック（新機能）

期限日セル（期限日1/2/3の値）

機能: カレンダー画面へジャンプ。

動作: クリックすると該当の日付へ遷移し、該当日が点滅強調表示。空セル（'-'）はクリック不可。

直近ではアスクルがランサムウェアで停止しましたし

SQL インジェクションの対策のためAIにも外部にも上げておりません。Pythonも使用したため外部サーバーに上げることは可能ですが時間と知識が足りません。 全ユーザーが見れるようにしたら個人情報の取扱もあり危険性もあがりますしね。いずれです。