画像整理アプリ 要件定義書 v2.7 (初回起動時ガイダンス追加)

本書について 本書は、「画像整理アプリ（仮称：イメージクリーンアップアシスタント）」の開発にあたり、アプリが実現すべき機能や性能、品質などを明確にすることを目的とした文書です。 このアプリ開発に新たに参加された方や、アプリの概要を初めて知る方にも、どのようなアプリを目指しているのかをご理解いただけるように記述しています。

はじめに 1.1. アプリの名称（仮称） イメージクリーンアップアシスタント 1.2. 開発の背景と目的 背景: 多くの方が、パソコンの中にたくさんの写真を保存しています。しかし、写真が増えすぎると、以下のような問題が起こりがちです。 パソコンの保存容量（ストレージ）がいっぱいになってしまう。 似たような写真や、手ブレしてしまった写真などがたくさんあり、見返すのが大変になる。 目的: この「イメージクリーンアップアシスタント」は、上記のような問題を解決するために開発されます。 具体的には、パソコンに保存されている画像の中から、 手ブレやピンボケなどで上手く撮れていない「ブレ画像」 同じものが複数あったり、ほとんど同じ内容の「類似画像（重複画像を含む）」 を自動的に見つけ出し、ユーザーが簡単に削除したり、別の場所に整理したりする手助けをします。 これにより、パソコンのストレージ容量を確保し、写真ライブラリをスッキリと見やすくすることを最終的な目標とします。

1. このアプリを利用する主なユーザー 主に、以下のような方を想定しています。 パソコンのローカルストレージ（ハードディスクやSSDなど）に多くの画像を保存している方。 ストレージ容量の不足を感じている方や、写真が多くて整理に困っている一般のパソコンユーザー。
2. アプリの主な機能（何ができるようになるか） このアプリが提供する主要な機能は以下の通りです。 3.1. 画像スキャン機能（問題のある画像を見つける機能） ユーザーが指定したフォルダ内にある画像を調べて、問題のある画像（ブレ画像や類似画像）をリストアップします。 3.1.1. スキャン対象の指定 ユーザーは、パソコンの中からスキャンしたいフォルダを自由に選べます。 メイン画面ヘッダーの「対象フォルダ選択」ボタンから、一般的なフォルダ選択ダイアログを開いて指定します。 選択されたフォルダのパスは、ボタンの隣（または下）に表示されます。 フォルダを直接アプリの画面にドラッグ＆ドロップして指定することもできます。（詳細はUI/UX検討タスクリスト参照） 選んだフォルダの中にあるサブフォルダ（フォルダの中のフォルダ）も一緒にスキャンするかどうかを選べます（初期設定ではスキャンする設定で、これは後述する設定画面で変更可能です）。 3.1.2. スキャン実行と進捗表示 ユーザーがメイン画面ヘッダーの「スキャン開始」ボタンを押すと、画像の調査が始まります。 スキャン中は、ボタンの表示が「スキャン中...」に変わり、非活性化されることを検討します。 メイン画面のフッターエリア（または専用のステータスバー領域）に、今どのファイルを調べているか、何枚の画像を見つけ終わったか、あとどれくらいで終わりそうか（パーセンテージや残り時間の目安）などが表示されます。 もしスキャンを途中でやめたくなった場合、キャンセルすることもできます（キャンセルボタンの配置と動作をUI/UXで検討）。 3.1.3. ブレ画像の検出 以下のようないわゆる「失敗写真」を見つけ出します。 手ブレ: カメラを持った時の手の揺れで画像が不鮮明になっているもの。 ピンボケ: ピントが合っておらず、被写体がぼやけているもの。 被写体ブレ: 動いている被写体がブレて写っているもの。 検出の仕組みと表示: アプリ内部では、画像のブレ具合を0から100の数値（スコア）で評価します。数値が高いほどブレが大きいことを意味します。 検出アルゴリズムとして、ラプラシアンフィルタを適用した画像の分散値を基にブレを評価する手法を採用します。具体的には、カーネルサイズ3のラプラシアンフィルタを適用し、得られた分散値を0-100のスコアに正規化します（プロトタイプ検証では、分散値が約180でブレ大（スコア100相当）、約3000で鮮明（スコア0相当）となる傾向）。 スキャン結果のリストには、このブレの度合い（0-100に正規化されたスコア）が表示されます。 ユーザーは、表示されたブレ画像の中から、ブレのスコアが特定の範囲内にあるものだけを右ペインのフィルター機能（スライダーまたは数値入力）で絞り込んで表示することができます。 補足: 以前のバージョンでは設定画面でブレ検出の「感度」を調整する案がありましたが、これを廃止し、検出されたすべてのブレ画像をスコアと共に表示し、ユーザーが結果を見て判断・フィルタリングする方式とします。 3.1.4. 類似画像（重複画像を含む）の検出 内容がよく似ている画像や、完全に同じ画像を見つけ出します。 完全に同一の画像: ファイル名や保存場所が違っていても、中身が全く同じ画像。 判定方法: SHA256ファイルハッシュ値を比較します。ハッシュ値が一致するものは内容が完全に同一と判断します。 内容が類似している画像: 同じような構図で撮影された連続写真、リサイズやトリミング、軽微な編集が施された画像など。 判定方法: 知覚ハッシュの一種であるahash（Average Hash、ハッシュサイズ8x8=64bit）を計算し、画像間のハッシュ値のハミング距離を基に類似度を評価します。ハミング距離が小さいほど類似度が高いと判断します。(プロトタイプ検証では、ハミング距離3以下で類似度100%、ハミング距離6で約50%、ハミング距離15以上で類似度0%となるようなスコアリングを採用。) 検出の仕組みと表示: スキャン結果のリストには、見つかった類似画像のペア（またはグループ）ごとに、どれくらい似ているかの度合い（類似度：0%～100%）が表示されます。完全に同一の画像は類似度100%として扱われます。 ユーザーは、表示された類似画像の中から、類似度が特定の範囲内にあるものだけを右ペインのフィルター機能（スライダーまたは数値入力）で絞り込んで表示することができます。 補足: 以前のバージョンでは設定画面で類似度検出の「閾値（基準値）」を調整する案がありましたが、これを廃止し、検出されたすべての類似画像を類似度と共に表示し、ユーザーが結果を見て判断・フィルタリングする方式とします。 3.1.5. スキャン結果の表示 スキャンが終わると、見つかった画像がメイン画面中央ペインのタブに分かれてリスト形式で表示されます。 「ブレ画像」タブ: サムネイル（小さな画像）、ファイル名、ファイルサイズ、更新日時、撮影日時、画像の大きさ（解像度）、ブレのスコア（0～100）が表示されます。 各タブには検出されたアイテム数が表示されます（例: ブレ画像 (102件)）。 「類似画像」タブ: 類似している画像のペア（またはグループ）ごとに表示されます。 各ペアについて、ペア全体を選択するチェックボックス、1つ目の画像のファイル名・サムネイル・解像度、2つ目の画像を選択するチェックボックス、2つ目の画像のファイル名・サムネイル・解像度、そして類似度（%）が表示されます。 アプリが「こちらを残した方が良いのでは？」と判断した画像（例：ファイルサイズが大きい、解像度が高いなど）を自動で提案し、リスト上で分かりやすく示す機能も検討します（UI/UXで詳細検討）。 「エラー」タブ: スキャン中に何らかの問題で処理できなかったファイルがあった場合、そのファイル名、エラーの具体的な内容、ファイルの場所（パス）が表示されます。 エラーの種類でフィルターする機能（ドロップダウンリスト）を右ペインに設けます。 どのリストも、各項目名（例：ファイル名、サイズなど）をクリックすることで、その項目で並べ替えることができます（昇順・降順）。 3.2. 画像整理機能（見つかった画像をどうするか決める機能） スキャン結果を見て、不要な画像を削除したり、別の場所に移動したりできます。 3.2.1. プレビュー機能 リストで選んだ画像を、アプリ左ペインで大きく表示して確認できます。 ブレ画像の場合: プレビューエリア1つを使い、画像を大きく表示します。プレビュー倍率の操作UI（スライダー、数値入力、ボタンなど）を設けます（例: 1%～150%）。 類似画像グループの場合: プレビューエリアを2つ使い、それぞれの画像を表示して見比べることができます（画像の配置は固定で、ユーザーは変更できません）。 プレビュー時には、その画像の詳細情報（ファイル名、ファイルサイズ、解像度、撮影日時、Exif情報から取得できるものなど）も表示します。 「エラー」タブで項目を選んだ場合: 画像プレビューは行わず、ファイル情報やエラー内容をプレビューエリアに表示します。 3.2.2. 整理対象の選択 リストに表示された画像の中から、整理したい画像をチェックボックスを使って個別に、または複数まとめて選択できます。 右ペインの操作エリアに「全件選択」「選択解除」ボタンを設けます。 類似画像のリストでは、ペアになっている画像のうち、どちらを整理対象にするか（例：ファイル2だけを削除対象にするなど）を、それぞれのチェックボックスで細かく選べます。 3.2.3. 削除機能 選択した画像を削除します。画面下部のアクションボタンエリアにある「選択した画像をゴミ箱へ」「選択した画像を完全に削除」ボタンを使用します。 削除方法として、以下のどちらかを選べます（どちらを標準動作にするかは設定画面で記憶できます）。 パソコンの「ゴミ箱」へ移動する。 ゴミ箱を経由せず、直接完全に削除する（この場合は、本当に削除して良いか確認メッセージを表示します）。 削除を実行する前には、何枚の画像を削除しようとしているかと、それらの合計ファイルサイズを画面下部（または確認ダイアログ内）に表示し、最終確認を促します。 もし削除処理中に何か問題（例：ディスクがいっぱいで削除できないなど）が起きた場合は、エラーメッセージを表示します。 3.2.4. 移動機能 選択した画像を、ユーザーが指定した別のフォルダへ移動します。画面下部のアクションボタンエリアにある「選択した画像を移動...」ボタンを使用します。 移動先のフォルダは、メイン画面ヘッダーの「移動先フォルダ選択」ボタンから事前に指定しておくか、移動操作の都度指定することができます。 移動を実行する前には、何枚の画像を移動しようとしているかと、移動先のフォルダ名を表示し、最終確認を促します。 もし移動処理中に何か問題が起きた場合は、エラーメッセージを表示します。 3.2.5. エラータブ固有の操作 「エラー」タブで選択したアイテムに対して、「選択を無視」（リストから非表示にするなど）、「再スキャン試行」といった操作を画面下部のアクションボタンで行えるようにします。 右ペインの操作エリアに「エラーログをエクスポート」ボタンを設けることも検討します。 3.3. ユーザーインターフェース・操作フローの概要（画面構成と基本的な使い方） （このセクションは、提供されたHTMLモックアップ v1.4に基づき更新） アプリケーションは単一のウィンドウで構成され、主に以下のエリアに分かれます。 ヘッダーエリア: 「対象フォルダ選択」ボタンと選択パス表示エリア 「移動先フォルダ選択」ボタンと選択パス表示エリア 「スキャン開始/スキャン中」ボタン 「設定」ボタン（歯車アイコン） メインコンテンツエリア（3ペイン構成）: 左ペイン: プレビューエリア 選択された画像の拡大表示、またはエラー情報の表示。 画像情報（ファイル名、サイズ、解像度、Exif等）の表示。 ブレ画像プレビュー時の倍率調整UI。 中央ペイン: 結果リストエリア 「ブレ画像」「類似画像」「エラー」のタブ切り替えUI。各タブには件数を表示。 各タブに応じた画像/ファイルリスト。カラムヘッダーでのソート機能。 アイテム選択用チェックボックス。 右ペイン: 操作・フィルターパネル 選択操作エリア: 「全件選択」「選択解除」ボタン。エラータブでは「エラーログをエクスポート」ボタンも検討。 フィルター条件エリア: 各タブに応じたフィルターUI（ブレ画像: ブレスコア範囲、類似画像: 類似度範囲、エラー: エラーの種類）。「適用」「リセット」ボタン。 フッターエリア (画面下部アクションボタンエリア): 選択中のアイテム数と合計サイズの表示（ブレ画像/類似画像タブ）。エラータブでは選択アイテム数を表示。 ブレ画像/類似画像タブ用アクションボタン: 「選択した画像をゴミ箱へ」「選択した画像を完全に削除」「選択した画像を移動...」 エラータブ用アクションボタン: 「選択を無視」「再スキャン試行」 スキャン進捗表示やステータスメッセージ表示エリア。 バージョン情報表示。 基本的な操作フロー: ユーザーはヘッダーエリアで「対象フォルダ選択」ボタンを押し、スキャンしたいフォルダを指定する。 必要であれば「移動先フォルダ選択」ボタンで、整理後の画像の移動先フォルダを指定する。 「スキャン開始」ボタンを押すと、スキャンが実行される。進捗はフッターエリアに表示される。 スキャン完了後、中央ペインの各タブに結果が表示される。 ユーザーはタブを切り替え、リストを確認する。 リスト内のアイテムを選択すると、左ペインにプレビューや詳細情報が表示される。 右ペインのフィルター機能で、表示するアイテムを絞り込むことができる。 整理したい画像を選択し、フッターエリアのアクションボタン（削除、移動など）で整理を実行する。 設定を変更したい場合は、ヘッダーの「設定」ボタンから設定画面を開く。 3.4. UI/UXデザインに関する詳細検討（画面の見やすさ・使いやすさについて） （このセクションは、HTMLモックアップ v1.4 および 設定画面モックアップ v1.3に基づき更新） アプリ全体の見た目の方針、ユーザー操作の詳細、エラー発生時の表示方法、設定画面の内容などは、別途「UI/UXデザイン詳細検討タスクリスト」および「デザイン原則案」で詳細化します。 HTMLモックアップで示されているデザイン（Tailwind CSSベース、フォントInter、角丸の使用など）を基本方針とします。 3.4.4. 設定画面 アプリの各種設定は、メイン画面のヘッダーにある歯車型のアイコンボタンから開かれる、1つのモーダルダイアログ画面でまとめて行います。 スキャン設定: 「サブフォルダを常にスキャン対象に含める」チェックボックス。 整理操作設定: 「『削除』ボタンのデフォルト動作」を「ゴミ箱へ移動」または「直接完全に削除」から選択するラジオボタン。 ログ設定: 「ログレベル」を「エラーのみ」「通常（エラーと主要な操作）」「詳細（デバッグ用）」から選択するドロップダウンリスト。 「ログファイルの場所」の表示と、それを変更するための「変更...」ボタン。 ダイアログ下部には「キャンセル」「適用」「OK」ボタンを配置します。 「適用」：設定を適用しダイアログは開いたまま。 「OK」：設定を適用しダイアログを閉じる。 「キャンセル」：変更を破棄しダイアログを閉じる。 注意: 以前の案にあった「ブレ検出の感度」や「類似画像検出の類似度」をユーザーが細かく調整する設定項目は、操作をシンプルにするため廃止しました。これらの判定はアプリ内部の固定ロジックで行い、結果のリスト表示とフィルター機能でユーザーが取捨選択できるようにします。 **3.5. 初回起動時ガイダンス機能** アプリケーションの初回起動時に、ユーザーが主要な操作をスムーズに理解できるよう、簡単なガイダンスを表示します。
   * **目的**: ユーザーに以下の主要ステップを理解させる。
     + 対象フォルダの選択方法
     + スキャンの開始方法
     + 結果の確認方法（タブ、プレビュー、リスト）
     + 画像の整理方法（主要アクションボタン）
   * **形式**: 複数ステップのモーダルダイアログ形式。各ステップで画面の該当箇所をハイライトし、簡潔な説明文を表示。
     + 例: ステップ1「ようこそ！まずスキャンするフォルダを選びましょう。」（「対象フォルダを選択」ボタンをハイライト）など、主要4ステップ＋導入・完了メッセージで構成。
   * **操作**: 各ステップに「次へ」ボタン、最終ステップに「開始する」または「閉じる」ボタン。最初のステップに「スキップ」ボタンを配置。
   * **表示タイミング**: 初回起動時、メインウィンドウ表示直後に自動表示。
   * **終了条件**: ユーザーが「開始する」または「スキップ」ボタンを押下、またはダイアログを閉じる。
   * **「次回から表示しない」オプション**: ガイダンスの最初または最後に「□ 次回からこのガイダンスを表示しない」チェックボックスを設け、選択された場合は設定を永続化する。
3. 非機能要件（性能や品質など、機能以外の重要なこと） アプリが快適かつ安全に動作するために満たすべき、機能以外の要件です。 4.1. 対応プラットフォーム（どのパソコンで動くか） OS（オペレーティングシステム）: 主に Windows 10 (64bit版) 以降のバージョンで動作することを想定しています。 ハードウェア（パソコンの性能）: CPU（頭脳部分）: Intel Core i3 プロセッサ、またはそれと同等以上の性能を持つAMD社製プロセッサ。 メモリ（作業スペース）: 最低4GB、快適に使うためには8GB以上を推奨します。 ストレージ（保存領域）: アプリ本体のインストールに最低100MBの空き容量が必要です。処理する画像の数によっては、一時的な作業のためにもう少し空き容量が必要になることがあります。 ディスプレイ（画面）: 1280x720ピクセル以上の解像度が表示できる画面。 必要なソフトウェア: （4.8. 開発技術スタックで詳述） 4.2. 対応画像形式（どんな種類の画像ファイルを扱えるか） JPEG (.jpeg, .jpg) PNG (.png) GIF (.gif) TIFF (.tiff, .tif) 将来的に: HEIC（iPhoneなどで使われる新しい形式）や、デジタルカメラのRAW画像形式（より専門的な画像データ）の読み込みにも対応できるよう検討する可能性があります。 **4.3. パフォーマンス（どれくらい速く、軽く動くか）**
   * **スキャン速度**:
     + **目標環境**: Intel Core i5 (第8世代以降)相当のCPU、8GB RAM、SATA SSD搭載のWindows 10/11 PC。
     + **対象データ**: 平均ファイルサイズ2MBのJPEG画像10,000枚が含まれる単一フォルダ（サブフォルダは含まない条件での計測）。
     + **目標時間 (初回スキャン)**: キャッシュがない状態での初回フルスキャン完了まで **10分以内**。
     + **目標時間 (2回目以降、キャッシュ利用時)**: 同一フォルダ構成でファイル変更がない場合（将来的にキャッシュ機能を実装した場合の目標）は **1分以内**。 (v1ではキャッシュ機能は見送りのため、参考目標)
     + **プログレスバーの更新頻度**: スキャン中は、最低でも**5秒に1回以上**、または処理ファイル**100件ごと**のいずれか早いタイミングで進捗状況がユーザーにフィードバックされること。
     + スキャン処理は、アプリの裏側（バックグラウンド）で行い、スキャン中もユーザーが他の操作をできるなど、画面が固まらないようにします。
   * **メモリ使用量**:
     + **アイドル時**: アプリケーション起動後、スキャンを実行せず、結果も表示されていない安定状態で **200MB以下**。
     + **スキャン中のピークメモリ使用量**: 上記「対象データ」のスキャン実行中における最大メモリ使用量が **700MB以下**。
     + **結果表示時のメモリ使用量**: 10,000件のブレ画像または10,000ペアの類似画像がリストに表示されている状態（仮想スクロール有効）で **500MB以下**。
     + 非常に多くの画像を一度に処理する場合でも、パソコン全体の動作が不安定にならないよう、メモリを効率的に使うようにします。
   * **UI（画面操作）の応答性**:
     + **リストスクロール**: 10,000件のアイテムが表示されているリスト（仮想スクロール有効）をスクロールする際に、目に見えるカクつきがなく、体感として**ほぼ60fpsに近い滑らかさ**を維持すること。
     + **リストアイテム選択からプレビュー表示までの時間**: リストでアイテムを選択してから、左ペインにそのアイテムのプレビュー（画像または情報）が表示されるまでの時間が **300ミリ秒以内**。
     + **フィルター適用時間**: 10,000件のアイテムが表示されているリストに対してフィルター条件を適用し、結果が再描画されるまでの時間が **2秒以内**。
     + **タブ切り替え時間**: 「ブレ画像」「類似画像」「エラー」タブ間の切り替えにかかる時間が **200ミリ秒以内**。
   * **起動時間**:
     + **コールドスタート**: OS起動後、初めてアプリケーションを起動し、メインウィンドウが表示され操作可能になるまでの時間が **7秒以内**。
     + **ウォームスタート**: アプリケーションを一度終了した後、再度起動し、メインウィンドウが表示され操作可能になるまでの時間が **3秒以内**。 4.4. ユーザビリティ（どれくらい使いやすいか） 直感性: アプリを初めて使う人でも、説明書を読まなくても主要な機能（フォルダを選ぶ、スキャンを始める、結果を見る、整理する）がだいたい理解できて操作できること。 学習容易性: 何回か使えば、ほとんどの機能をスムーズに使いこなせるようになること。 効率性: ユーザーがやりたいことを達成するための操作手順を、できるだけ少なくします。まとめて選択したり、条件に合うものを一括で選んだりする機能を提供し、作業が早く終わるようにします。 誤操作防止: ファイルの削除（特に元に戻せない完全削除）など、取り返しのつかない重要な操作の前には、必ず「本当に実行しますか？」といった確認メッセージを表示します。 今どの画像を選んでいるのかがはっきり分かるように、選択した項目は色を変えるなどして目立たせます。 フィードバック（応答の表示）: ユーザーがボタンをクリックしたり、項目を選んだりといった操作をしたら、ボタンの色が変わるなど、システムが反応したことをすぐに視覚的に伝えます。 時間のかかる処理（スキャン、大量のファイルの削除や移動）の途中では、進捗バー（進行状況を示す棒グラフ）やメッセージで、今どういう状況かを分かりやすく伝えます。 アクセシビリティ（誰にでも使いやすくするための基本的な配慮）: マウスだけでなく、キーボードだけでも主要な機能が使えるようにします（例：Tabキーで項目を移動し、EnterキーやSpaceキーで決定するなど）。 画面の文字や背景の色は、コントラスト（明暗差）を十分につけて見やすくします。 文字の大きさは、読みやすい適切なサイズにします。 4.5. 信頼性（どれくらい安心して使えるか） 検出精度: ブレ画像や類似画像を見つけるアルゴリズムは、間違って正常な画像を問題ありと判断したり、逆に見つけるべき問題のある画像を見逃したりすることが、できるだけ少なくなるようにします。 安定性: 長時間連続して使ったり、非常に多くのファイルを一度に処理したりしても、アプリが途中で強制終了したり、固まったりしないこと。 データ保全: 画像ファイルの削除や移動は、ユーザーが指示した通りに確実に行い、間違って違うファイルを操作したり、ファイルを壊してしまったりしないこと。 エラー耐性: 予期せぬエラー（例：ファイルにアクセスできない、ディスクの空き容量がないなど）が起きた場合でも、アプリが安全な状態を保ち、可能であれば処理を中断したり、ユーザーにどうすれば良いか適切な対応を促したりすること。 4.6. 保守性（将来の修正や機能追加のしやすさ） コード品質: プログラムのコードは、他の開発者が見ても読みやすく、理解しやすいように記述します。適切なコメント（説明文）も付けます。 モジュール性: 機能ごとに部品（モジュール）のように分け、それぞれの独立性を高めます。これにより、一部を修正したり新しい機能を追加したりする際に、他の部分への影響が少なくなるようにします。 テスト容易性: プログラムが正しく動くかを確認するテスト（ユニットテストや結合テスト）がしやすいように設計します。 ドキュメント: アプリの主要な設計や仕様に関する資料を整備します。 4.7. セキュリティ（安全性について） ファイルアクセス: アプリがパソコン内のファイルにアクセスするのは、ユーザーが明示的に指定したフォルダとその中にあるフォルダ・ファイルだけに限定します。それ以外の場所に勝手にアクセスすることはしません。 外部通信: 原則として、このアプリがユーザーの許可なくインターネット上の外部サーバーと通信することはありません。もし将来、アップデートの確認などで通信が必要になる場合は、その目的をユーザーに明確に伝え、許可を得るようにします。 脆弱性対策: アプリ開発に使用するライブラリやフレームワーク（開発を助けるための部品群）に、セキュリティ上の問題点（脆弱性）が見つかっていないか定期的に確認し、必要であれば最新版にアップデートします。 4.8. 開発技術スタック（採用候補） 本アプリケーションの開発にあたり、以下の技術スタックを採用候補とします。 4.8.1. プログラミング言語: Python 選定理由: 学習コストが比較的低く、迅速な開発が期待できます。 画像処理、ファイル操作、GUI開発など、本アプリに必要なライブラリが豊富に存在します。 OpenCVやPillow、ImageHashといった主要な画像処理ライブラリとの親和性が高いです。 Windows、macOS、Linuxなど、主要なOSで動作するため、将来的なクロスプラットフォーム展開の基盤となります。 4.8.2. 主要ライブラリ: 画像処理・数値計算: OpenCV, NumPy, Pillow ハッシュ計算: ImageHash (知覚ハッシュ用), hashlib (ファイルハッシュ用) 選定理由: これらはPythonにおける画像処理やハッシュ計算の標準的なライブラリであり、豊富な機能とドキュメント、活発なコミュニティに支えられています。プロトタイプ開発においてもこれらのライブラリで良好な結果が得られています。 4.8.3. UIフレームワーク: 採用フレームワーク: Electron (HTML, CSS, JavaScript + Python連携) 選定理由: Web技術を用いてUIを構築できるため、提供されているHTMLモックアップのデザインやコンポーネントを最大限に活かすことができます。 Tailwind CSSなどのモダンなCSSフレームワークを利用しやすく、デザインの自由度が高いです。 クロスプラットフォーム（Windows, macOS, Linux）への対応が比較的容易です。 バックエンドのコアロジック（画像判定など）はPythonで実装し、ElectronのUIと連携させることで、それぞれの技術の長所を活かします。 代替候補: PyQt (または PySide) 検討理由: ネイティブに近いパフォーマンスやOSとのより深い統合が必要となった場合の代替選択肢として残します。
4. 今後の検討事項（これから決めていくこと） この要件定義書は基本的な方針を示すものであり、今後開発を進める中で、以下の点をより具体的に検討・決定していきます。 UI/UXデザイン詳細: これまでのワイヤーフレームやモックアップを元に、実際の画面デザインや操作感を試作（プロトタイピング）しながら、さらに使いやすく改善していきます。 特に、Electronを採用するため、HTML/CSS/JavaScriptでの具体的なUI実装方針を固めます。（「UI/UXデザイン詳細検討タスクリスト」参照） エラー処理詳細: ファイルアクセスエラー、メモリ不足、サポートされていない画像形式への対応など、さまざまなエラー発生時の具体的な対処方法とユーザーへの通知方法（メッセージ内容、表示方法など）を定義します。 設定項目詳細: 設定画面で設定できる各項目（サブフォルダのスキャン有無、削除の標準動作、ログレベルなど）の具体的なUI（見た目や操作方法）と、それがアプリの動作にどう影響するかを設計します。（「UI/UXデザイン詳細検討タスクリスト」参照） 文言（マイクロコピー）: アプリ内で表示される各ボタンのラベル名、確認メッセージ、エラーメッセージ、通知メッセージなどの具体的なテキスト内容を詰めていきます。 ~~初回起動時のガイダンス: アプリの初回起動時に、簡単な操作説明やチュートリアルを表示するかどうか検討します。~~ 対応OSとバージョン: まずはWindowsを対象としますが、具体的にWindowsのどのバージョンからサポートするか（例：Windows 10のバージョンXXXX以降など）を決定します。 macOSへの対応も視野に入れています。 インストールとアップデート方法: アプリをどのようにユーザーのパソコンにインストールしてもらうか、また、新しいバージョンが出た時にどのようにアップデートしてもらうかを決定します。 ログ機能詳細: ログファイルが際限なく大きくなるのを防ぐ仕組み（ログローテーション）や、保存するログの最大サイズなどを検討します。 セッション保存・復元機能詳細: アプリを一度終了しても前回の状態を復元できるようにする場合、具体的にどの情報（スキャンしたフォルダ、結果リスト、選択状態、フィルター条件など）を保存し、どのように復元するかを詳細に設計します。 キーボードショートカット: 主要な操作に対するキーボードショートカットを定義します。 その他 （このアプリに関して、上記以外に何か特別な要望やイメージがあれば、今後の話し合いでここに追記していきます。）