Project Title

OCRify: Advanced Optical Character Recognition (OCR) and Image Analytics System

1. Introduction

OCRify is a comprehensive desktop Graphical User Interface (GUI) application designed to bridge the gap between static imagery and digital text. Built using Python, it leverages the Tesseract engine to perform Optical Character Recognition (OCR) on various image formats.

Beyond basic text extraction, OCRify distinguishes itself with an integrated analytics suite that provides real-time word frequency analysis, unique word detection with dictionary definitions via API, and a detailed image metadata/EXIF viewer. The application features a modern, tabbed interface with drag-and-drop capabilities, making it suitable for researchers, data analysts, and general users who require deep insights into image-based text.

2. Functional Requirements

These requirements define the specific behaviors and functions the system must support.

2.1. File Management & Input

- **Image Loading:** Users must be able to load images via a file dialog or by dragging and dropping files into the workspace.
- **Supported Formats:** The application must support PNG, JPG, JPEG, GIF, BMP, TIFF, and TIF.
- **Recent History:** The system must store the last 10 accessed files, accessible through a "Recent Files" menu.
- **Image Preview:** The interface must display a scaled preview of the loaded image.

2.2. Text Extraction (OCR)

- **Core Extraction:** The system must use the `pytesseract` wrapper to interface with the Tesseract engine and extract text from the loaded image.
- **Text Manipulation:** Users must be able to view extracted text in a scrollable area, copy it to the clipboard, or clear the display.
- **Automation:** The system must support an **Auto-extract** mode that triggers OCR immediately after loading an image.

2.3. Text Analytics

- **Statistical Counts:** Calculate and display total words, characters, and lines in the extracted text.
- **Frequency Analysis:** Identify and list the top 10 most frequently used words, including their count and percentage.
- **Unique Word Detection:** Identify words appearing only once and having a minimum length of 4 characters.
- **Dictionary Integration:** Fetch definitions, phonetics, and parts of speech for unique words using the external API (`api.dictionaryapi.dev`).

2.4. Metadata & EXIF Analysis

- **File Attributes:** Extract filename, path, file size (bytes and MB), creation date, and modification date.
- **Image Properties:** Report image dimensions, format, color mode, aspect ratio, and megapixel count.
- **EXIF Data:** Display available EXIF tags (e.g., Camera Make/Model, ISO, Exposure Time, Orientation).
- **Color Analysis:** Attempt to calculate the number of unique colors and detect the dominant color.

2.5. Data Export

- **Result Saving:** Users must be able to save extracted text and metadata as `.txt` or `.json`.
- **Metadata Export:** Allow exporting metadata as a standalone JSON file.

3. Non-Functional Requirements

These define the system's performance, reliability, and user experience attributes.

3.1. Performance & Concurrency

- **Asynchronous Processing:** Long-running tasks (OCR and API requests) must run on separate threads to keep the GUI responsive.
- **Progress Feedback:** Provide a progress bar and status indicators during processing.

3.2. User Interface (UI) & Experience (UX)

- **Modern Styling:** Use ttk styling (themes such as *vista* or *clam*) with a blue/cyan/white palette.
- **Responsiveness:** Use grid-based layout management for dynamic resizing.
- **Accessibility:** Provide keyboard shortcuts (e.g., Ctrl+O, Ctrl+E) and tooltips.

3.3. Reliability & Error Handling

- **Dependency Checking:** Verify Tesseract installation on startup; disable OCR features if not found.
- **Network Resilience:** Handle API timeouts or failures without crashing.
- **Input Validation:** Ensure only supported image formats are processed.

3.4. System Dependencies

- **Libraries:** The system uses Pillow for image processing, Pytesseract for OCR, Requests for API communication, and `tkinterdnd2` for drag-and-drop support.
- **External Engine:** Tesseract OCR must be installed and accessible via the system PATH or manual configuration.