### **Document Vectorizer (doc\_vectorizer.py) Analysis**

This table breaks down the functions responsible for processing, extracting content from, and vectorizing various file types into a Chroma vector database.

| **Category** | **Function/Method Name** | **Input Parameters** | **Output [Datatype]** | **Comments** |
| --- | --- | --- | --- | --- |
| **Utility** | get\_filename | file\_path: str | str | A simple helper to extract the base name of a file without its extension (e.g., "report.pdf" -> "report"). |
| **Utility** | get\_file\_hash | file\_path: str | str or None | Generates a SHA-256 hash of the file's content, useful for detecting duplicate files. |
| **Utility** | get\_file\_size | file\_path: str | int or None | Returns the size of the file in bytes. |
| **Utility** | create\_base\_metadata | file\_path: str, company\_name: str, file\_type: str | Dict | Creates a standardized dictionary of metadata (company, filename, timestamp, hash, etc.) that is applied to all chunks of a document. |
| **Utility** | get\_supported\_file\_types | None | Dict | Returns a dictionary mapping supported document categories (e.g., 'word\_document') to their file extensions (e.g., ['.doc', '.docx']). |
| **Utility** | validate\_file\_type | filepath: str | (bool, str) | Checks if a given file's extension is in the list of supported types and returns its category if it is. |
| **File Routing** | file\_router | file: str | str | The core routing logic. It inspects a file's extension and content to determine its type (e.g., 'powerpoint', 'imagepdf', 'pdf') to decide which processing function to use. |
| **Image Processing** | encode\_image | image (PIL Image object) | str | Converts a PIL image object into a base64-encoded string, which is required for the Gemini vision model. |
| **Image Processing** | image\_summarize | model, base64\_image: str, prompt: str | str | Sends the base64 image and a text prompt to the Gemini model and returns the AI-generated text summary of the image. |
| **Image Processing** | image\_handler | image (PIL Image object) | str | A wrapper that encodes an image, gets its summary, and saves the summary to a new text file (example.txt). |
| **Image Processing** | image\_handler\_append | image (PIL Image object) | str | Similar to image\_handler, but it appends the image summary to example.txt instead of overwriting it. Used for multi-page image PDFs. |
| **Content Extraction** | extract\_ppt\_content | filepath: str | (str, int) | Extracts all text from a PowerPoint file, including from shapes and tables. It has a fallback mechanism to use a different loader if the primary method fails. Returns the text and slide count. |
| **Content Extraction** | extract\_word\_content | filepath: str | (str, int) | Extracts all text from a Word document (.doc or .docx), including from tables. It uses different libraries based on the file extension and has a fallback loader. Returns the text and paragraph/chunk count. |
| **Vectorization** | vectorize\_text | text: str, company\_name: str, filename: str, base\_metadata: dict | Chroma (VectorStore) | The core vectorization function for any text. It splits the text into chunks, adds rich metadata to each chunk, and stores them as embeddings in a persistent ChromaDB collection. |
| **Vectorization** | vectorize\_powerpoint | filepath: str, company\_name: str | Chroma (VectorStore) | Orchestrates PowerPoint processing. It calls extract\_ppt\_content and then passes the resulting text and specific metadata to vectorize\_text. |
| **Vectorization** | vectorize\_word\_document | filepath: str, company\_name: str | Chroma (VectorStore) | Orchestrates Word document processing. It calls extract\_word\_content and then passes the resulting text and metadata to vectorize\_text. |
| **Vectorization** | vectorize\_single\_image | image, company\_name: str | Chroma (VectorStore) | Orchestrates single image processing. It calls image\_handler to get a summary and then passes the summary and metadata to vectorize\_text. |
| **Vectorization** | vectorize\_multiple\_images | image\_path: str, company\_name: str | Chroma (VectorStore) | For image-only PDFs. It converts each PDF page to an image, gets a summary for each, concatenates the summaries, and passes the result to vectorize\_text. |
| **Vectorization** | vectorize\_docs | filepath: str, company\_name: str | Chroma (VectorStore) | The standard processor for text-based PDFs. It uses PyPDFLoader to load the document, splits it into chunks, adds metadata, and creates the Chroma vector store. |
| **Entry Point** | vectorize | filepath: str, company\_name: str | Chroma (VectorStore) | **The main entry point of the script.** It takes a file path, calls file\_router to determine the file type, and then calls the appropriate vectorize\_\* function to handle the ingestion. |
| **Metadata Querying** | search\_by\_metadata | vectorstore, metadata\_filter: dict, query: str, k: int | List | A function to search the vector store not just by semantic similarity (query) but also by filtering on the metadata fields (e.g., find all chunks where company\_name is "ABC Corp"). |
| **Metadata Querying** | get\_document\_metadata\_summary | vectorstore | Dict or None | An administrative function to query the entire vector store and return a summary of its contents, such as the total number of documents and unique file types. |