### **Pain Point Extractor (pain\_points\_extractor.py) Analysis**

This table breaks down the function responsible for analyzing a document and extracting key business challenges or "pain points" using a RAG pipeline.

| **Function/Method Name** | **Location of the API/Function** | **Input Parameters** | **Output [Datatype]** | **Comments** |
| --- | --- | --- | --- | --- |
| vectorize | Document\_Upload\_Vectordb.doc\_vectorizer | file: str, company\_name: str | VectorStore object | This external function takes a file path, processes the document (likely splitting it into chunks), and creates a vector embedding for each chunk, storing them in a searchable vector database. |
| format\_docs | Document\_Upload\_Vectordb.doc\_xtraction\_utils | docs | str | An external utility function that takes a list of retrieved document chunks and formats them into a single string to be used as context for the LLM prompt. |
| clean\_to\_list | Document\_Upload\_Vectordb.doc\_xtraction\_utils | result: str | str | An external utility function designed to clean the raw string output from the LLM, likely removing markdown formatting (like ```json) to ensure it's a valid JSON string. |
| get\_pain\_points | pain\_points\_extractor.py | file: str, company\_name: str | list or dict | This is the main function. It orchestrates the entire RAG process: <br> 1. **Initializes**: Sets up the prompt template, LLM, and output parser. <br> 2. **Vectorizes**: Calls vectorize to create a retriever from the input file. <br> 3. **Builds Chain**: Constructs a LangChain (LCEL) pipeline that retrieves context, inserts it into a prompt, calls the LLM, and parses the output. <br> 4. **Invokes & Cleans**: Runs the chain with a query, then cleans and parses the resulting JSON string into a Python list or dictionary. |