Code Flow and RAG Architecture Overview

1. Code Flow

The application's core functionality is orchestrated through `app.py`, which serves as the main entry point for the web interface. It interacts with `rag_engine.py` to handle the Retrieval-Augmented Generation (RAG) logic, while `utils.py` provides utility functions to support various operations.

Here's a high-level overview of the code flow:

1. **User Interaction (`app.py`):**
    - The `app.py` file sets up a web interface (likely using Flask or a similar framework) that allows users to upload PDF documents and ask questions.
    - When a user uploads a PDF, `app.py` likely calls a function to process the document, which might involve `rag_engine.py` for document ingestion and vectorization.
    - When a user submits a query, `app.py` passes this query to the RAG engine for processing.
2. **RAG Engine (`rag_engine.py`):**
    - `rag_engine.py` encapsulates the core RAG logic. It receives user queries from `app.py`.
    - It's responsible for retrieving relevant information from the vectorized document store (vector database).
    - It then uses an LLM to generate a coherent and accurate answer based on the retrieved information.
    - The generated answer, along with source references, is returned to `app.py`.
3. **Utility Functions (`utils.py`):**
    - `utils.py` contains helper functions that are used by both `app.py` and `rag_engine.py`.

- These might include functions for PDF parsing, text chunking, embedding generation, or other data processing tasks.

2. Important Blocks in RAG Architecture

Based on the provided code and the proposed features, the RAG architecture includes the following important blocks:

1. **Document Ingestion and Vectorization:**
    - **Purpose:** To process raw PDF documents and convert their content into a format suitable for retrieval.
    - **Components:** PDF parser, text chunker, embedding model (to convert text chunks into numerical vectors).
    - **Output:** Vector embeddings stored in a Vector Database.
2. **User Query Processing:**
    - **Purpose:** To prepare the user's query for effective retrieval and response generation.
    - **Components:**
        - **Spell Check Module:** Corrects spelling errors in the user query to improve search accuracy.
        - **Guardrails (User Query):** Filters out inappropriate, toxic, or cybersecurity-related queries before processing.
        - **Conversation History Module:** Maintains the context of previous interactions, allowing the system to understand follow-up questions and maintain conversational awareness.
3. **Retrieval Module:**
    - **Purpose:** To find the most relevant document chunks from the Vector Database based on the user's query.

- **Components:** Vector Database (e.g., ChromaDB, Pinecone), similarity search algorithm.
- **Output:** Top-k relevant document chunks.

4. **LLM-based Reranker Module:**
   - **Purpose:** To re-rank the retrieved document chunks for higher relevance and to reduce the number of tokens passed to the main LLM, thereby optimizing cost and performance.
   - **Components:** A smaller, specialized LLM or a re-ranking algorithm.
   - **Output:** A refined set of highly relevant document chunks.
5. **Generative Module (LLM):**
   - **Purpose:** To synthesize a coherent and accurate answer from the re-ranked document chunks and the user's query.
   - **Components:** Large Language Model (LLM).
   - **Output:** Generated answer and source references.
6. **Response Processing:**
   - **Purpose:** To ensure the generated response is safe, accurate, and user-friendly.
   - **Components:**
     - **Guardrails (LLM Response):** Filters out toxic, biased, or cybersecurity-related content from the LLM's response.
     - **Real-time Feedback Module:** Captures user feedback (thumbs up/down, comments) on the generated response. This feedback loop is crucial for continuous improvement of the RAG system.

This architecture allows for a robust and intelligent RAG application capable of providing accurate and context-aware responses to user queries about the Cognizant Annual Report.