Task 1: Document Parsing
Leveraging the `Retrieval` class, documents within a specified directory are processed, pairing technical text files with corresponding metadata using generated IDs. The technical text is segmented into sections and paragraphs while metadata files are converted to Python structures. Related metadata is associated with every passage in the technical texts, and the resultant data is saved as a DataFrame in a CSV file.

Task 2: Passage Embeddings
SentenceTransformer('all-MiniLM-L6-v2') was utilized due to its extensive training on diverse English datasets, ensuring the capture of semantic nuances in the English language.

Task 3: ElasticSearch Integration
A mapping was designed in ElasticSearch to store passages, metadata, and embeddings. These constituents were chosen as they are pivotal for subsequent retrieval and analysis, offering an exhaustive contextual overview of the stored documents.

Task 4: Document Retrieval
A unique approach was adopted where the embeddings of the question were used as a query to search in ElasticSearch. Cosine Similarity was deployed to map question embeddings with document embeddings, enhancing the precision of document retrieval.

Task 5: Containerization
Two Docker containers were configured: one for ElasticSearch and the other for the Flask API, streamlining deployment and ensuring environment consistency.

Task 6: API Deployment with Flask

Two POST routes were established in Flask; one for receiving questions and the other for parsing documents, catering to different facets of user interaction and data processing.

This summary provides a succinct overview of the six tasks, outlining the key activities and the rationale behind certain choices made during implementation.