```
from typing import Optional
from pathlib import Path
from loguru import logger
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader


class LlamaIndexDB:
    """A class to manage document indexing and querying using LlamaIndex.

    This class provides functionality to add documents from a directory and query the indexed documents.

    Args:
        data_dir (str): Directory containing documents to index. Defaults to "docs".
        **kwargs: Additional arguments passed to SimpleDirectoryReader and VectorStoreIndex.
            SimpleDirectoryReader kwargs:
                - filename_as_id (bool): Use filenames as document IDs
                - recursive (bool): Recursively read subdirectories
                - required_exts (List[str]): Only read files with these extensions
                - exclude_hidden (bool): Skip hidden files

            VectorStoreIndex kwargs:
                - service_context: Custom service context
                - embed_model: Custom embedding model
                - similarity_top_k (int): Number of similar docs to retrieve
                - store_nodes_override (bool): Override node storage
```

```
"""

def __init__(self, data_dir: str = "docs", **kwargs) -> None:
    """Initialize the LlamaIndexDB with an empty index.

    Args:
        data_dir (str): Directory containing documents to index
        **kwargs: Additional arguments for SimpleDirectoryReader and VectorStoreIndex
    """
    self.data_dir = data_dir
    self.index: Optional[VectorStoreIndex] = None
    self.reader_kwargs = {
        k: v
        for k, v in kwargs.items()
        if k
        in SimpleDirectoryReader.__init__.__code__.co_varnames
    }
    self.index_kwargs = {
        k: v
        for k, v in kwargs.items()
        if k not in self.reader_kwargs
    }

    logger.info("Initialized LlamaIndexDB")
    data_path = Path(self.data_dir)
    if not data_path.exists():
```

```
            logger.error(f"Directory not found: {self.data_dir}")
            raise FileNotFoundError(
                f"Directory {self.data_dir} does not exist"
            )

        try:
            documents = SimpleDirectoryReader(
                self.data_dir, **self.reader_kwargs
            ).load_data()
            self.index = VectorStoreIndex.from_documents(
                documents, **self.index_kwargs
            )
            logger.success(
                f"Successfully indexed documents from {self.data_dir}"
            )
        except Exception as e:
            logger.error(f"Error indexing documents: {str(e)}")
            raise

    def query(self, query: str, **kwargs) -> str:
        """Query the indexed documents.

        Args:
            query (str): The query string to search for
            **kwargs: Additional arguments passed to the query engine
                - similarity_top_k (int): Number of similar documents to retrieve
```

```
            - streaming (bool): Enable streaming response
            - response_mode (str): Response synthesis mode
            - max_tokens (int): Maximum tokens in response

        Returns:
            str: The response from the query engine

        Raises:
            ValueError: If no documents have been indexed yet
        """
        if self.index is None:
            logger.error("No documents have been indexed yet")
            raise ValueError("Must add documents before querying")

        try:
            query_engine = self.index.as_query_engine(**kwargs)
            response = query_engine.query(query)
            print(response)
            logger.info(f"Successfully queried: {query}")
            return str(response)
        except Exception as e:
            logger.error(f"Error during query: {str(e)}")
            raise


# # Example usage
```

```
# llama_index_db = LlamaIndexDB(
#     data_dir="docs",
#     filename_as_id=True,
#     recursive=True,
#     required_exts=[".txt", ".pdf", ".docx"],
#     similarity_top_k=3
# )
# response = llama_index_db.query(
#     "What is the medical history of patient 1?",
#     streaming=True,
#     response_mode="compact"
# )
# print(response)
```