Documentation for `routes/openl3.py`

This module contains the endpoints for the OpenL3 service. It provides routes for extracting audio embeddings using the OpenL3 model. The embeddings can then be used to perform similarity searches on the embeddings using the Milvus service.

`get_embeddings(file_path, user=Depends(login_manager), db=Depends(get_db))`

Retrieves the embeddings for a specified audio file.

This function loads a model from MinIO, retrieves the specified audio file as a temporary file, computes the embeddings using the loaded model, and then cleans up the temporary file. If successful, it returns an EmbeddingResponse object containing the file name and its embeddings. If the process fails, it raises an HTTPException with status code 500.

Parameters: - file_path (str): The path to the audio file for which embeddings are to be computed. - user: The current user object, automatically provided by the login_manager dependency. - db: The database session, automatically provided by the get_db dependency.

Returns: - EmbeddingResponse: An object containing the file name and its computed embeddings.

Source code in `routes/openl3.py`

```python
@router.post("/embeddings/", response_model=EmbeddingResponse, tags=
["OpenL3"])
def get_embeddings(file_path: str, user=Depends(login_manager), db: Session
= Depends(get_db)):
    """
    Retrieves the embeddings for a specified audio file.

    This function loads a model from MinIO, retrieves the specified audio
file as a temporary file,
    computes the embeddings using the loaded model, and then cleans up the
temporary file. If successful,
    it returns an EmbeddingResponse object containing the file name and its
embeddings. If the process fails,
    it raises an HTTPException with status code 500.

    Parameters:
    - file_path (str): The path to the audio file for which embeddings are
to be computed.
    - user: The current user object, automatically provided by the
login_manager dependency.
    - db: The database session, automatically provided by the get_db
dependency.

    Returns:
    - EmbeddingResponse: An object containing the file name and its
computed embeddings.

    """
    print(f"Starting to get embeddings for file: {file_path}")
    try:
        embedding_512_model = load_model_from_minio()
        temp_file_path = get_temp_file_from_minio(file_path)

        # Compute embeddings using the temporary file path
        vector = embedding_512_model.compute(temp_file_path)
        embedding = vector.mean(axis=0)

        # Clean up the temporary file
        os.unlink(temp_file_path)

        print(f"Successfully processed embeddings for file: {file_path}")
        return EmbeddingResponse(file_name=file_path,
embedding=embedding.tolist())
    except Exception as e:
        print(f"Failed to get embeddings for file: {file_path}. Error:
{e}")
        raise HTTPException(status_code=500, detail=f"Failed to process the
request: {e}")
```