Documentation for `routes/favorites.py`

This module contains the endpoints for the favorites service. It provides routes for adding, removing, and listing favorite songs for a user.

`add_song_to_favorites(song, user=Depends(login_manager), db=Depends(get_db))` `async`

Add a song to the authenticated user's list of favorites.

- **song**: SongPath - The path of the song to be added to favorites.
- **user**: User - The authenticated user who is adding the song to favorites.
- **db**: Session - The database session for querying and updating the database.
- **return**: Returns a message indicating the song was successfully added to favorites or if it was already in favorites.

Source code in `routes/favorites.py`

```python
@router.post("/add", tags=["favorites"])
async def add_song_to_favorites(song: SongPath, user: User =
Depends(login_manager), db: Session = Depends(get_db)):
    """
    Add a song to the authenticated user's list of favorites.

    - **song**: SongPath - The path of the song to be added to favorites.
    - **user**: User - The authenticated user who is adding the song to
favorites.
    - **db**: Session - The database session for querying and updating the
database.
    - **return**: Returns a message indicating the song was successfully
added to favorites or if it was already in favorites.
    """
    user = db.merge(user)
    db.refresh(user)

    if len(user.favorites) >= 9:
        # Remove the oldest song from the favorites
        user.favorites.pop(0)

    music_id = get_song_id_by_filepath(db, song.file_path)
    if not music_id:
        raise HTTPException(status_code=404, detail="Song not found")
    music = db.query(MusicLibrary).get(music_id)

    # Check if the song is already in the user's favorites
    if music in user.favorites:
        return {"message": "Song is already in favorites"}

    user.favorites.append(music)
    db.commit()
    return {"message": "Song added to favorites"}
```

`delete_song_from_favorites(song, user=Depends(login_manager), db=Depends(get_db))` `async`

Remove a song from the authenticated user's list of favorites.

- **song**: SongPath - The path of the song to be removed from favorites.
- **user**: User - The authenticated user who is removing the song from favorites.
- **db**: Session - The database session for querying and updating the database.
- **return**: Returns a message indicating the song was successfully removed from favorites or if the song was not found in favorites.

Source code in `routes/favorites.py`

```python
@router.delete("/delete", tags=["favorites"])
async def delete_song_from_favorites(song: SongPath, user: User =
Depends(login_manager), db: Session = Depends(get_db)):
    """
    Remove a song from the authenticated user's list of favorites.

    - **song**: SongPath - The path of the song to be removed from
favorites.
    - **user**: User - The authenticated user who is removing the song from
favorites.
    - **db**: Session - The database session for querying and updating the
database.
    - **return**: Returns a message indicating the song was successfully
removed from favorites or if the song was not found in favorites.
    """
    user = db.merge(user)
    db.refresh(user)
    music_id = get_song_id_by_filepath(db, song.file_path)
    if not music_id:
        raise HTTPException(status_code=404, detail="Song not found")
    music = db.query(MusicLibrary).get(music_id)
    for favorite in user.favorites:
        if favorite.id == music.id:
            user.favorites.remove(favorite)
            db.commit()
            return {"message": "Song removed from favorites"}
    raise HTTPException(status_code=404, detail="Song not found in
favorites")
```

`get_favorites(user=Depends(login_manager), db=Depends(get_db))` async

Retrieve the list of favorite songs for the authenticated user.

- **user**: User - The authenticated user whose favorites are to be retrieved.
- **db**: Session - The database session for querying the database.
- **return**: Returns a list of the user's favorite songs.

Source code in `routes/favorites.py`

```python
@router.get("/", tags=["favorites"])
async def get_favorites(user=Depends(login_manager), db: Session =
Depends(get_db)):
    """
    Retrieve the list of favorite songs for the authenticated user.

    - **user**: User - The authenticated user whose favorites are to be
retrieved.
    - **db**: Session - The database session for querying the database.
    - **return**: Returns a list of the user's favorite songs.
    """
    # The merge() function is used to merge a detached object back into the
session.
    # It returns a new instance that represents the existing row in the DB.
    # This is necessary because the 'user' object might have been created
in a different session and we want to associate it with the current
session.
    user = db.merge(user)

    # The refresh() function is used to update the attributes of the 'user'
instance with the current data in the DB.
    # This is necessary because the 'user' object might have stale data and
we want to ensure we're working with the most recent data.
    db.refresh(user)
    return user.favorites
```