Coding exercise:

Write a Flask / FastAPI / Django Web API that simulates the behavior of an audio file server while using a MongoDB / SQL database.

Requirements: You have one of three audio files which structures are defined below

Audio file type can be one of the following:

1 – Song
2 – Podcast
3 – Audiobook

Song file fields:

- ID – (mandatory, integer, unique)
- Name of the song – (mandatory, string, cannot be larger than 100 characters)
- Duration in number of seconds – (mandatory, integer, positive)
- Uploaded time – (mandatory, Datetime, cannot be in the past)

Podcast file fields:

- ID – (mandatory, integer, unique)
- Name of the podcast – (mandatory, string, cannot be larger than 100 characters)
- Duration in number of seconds – (mandatory, integer, positive)
- Uploaded time – (mandatory, Datetime, cannot be in the past)
- Host – (mandatory, string, cannot be larger than 100 characters)
- Participants – (optional, list of strings, each string cannot be larger than 100 characters, maximum of 10 participants possible)

Audiobook file fields:

- ID – (mandatory, integer, unique)
- Title of the audiobook – (mandatory, string, cannot be larger than 100 characters)
- Author of the title (mandatory, string, cannot be larger than 100 characters)
- Narrator - (mandatory, string, cannot be larger than 100 characters)
- Duration in number of seconds – (mandatory, integer, positive)
- Uploaded time – (mandatory, Datetime, cannot be in the past)

Implement create, read, upload, and delete endpoints for an audio file as defined below:

Create API:

The request will have the following fields:

- audioFileType – mandatory, one of the 3 audio types possible
- audioFileMetadata – mandatory, dictionary, contains the metadata for one of the three audio files (song, podcast, audiobook)

Delete API:

- The route will be in the following format: "<audioFileType>/<audioFileID>"

Update API:
- The route be in the following format: "<audioFileType>/<audioFileID>"
- The request body will be the same as the upload

Get API:
- The route "<audioFileType>/<audioFileID>" will return the specific audio file
- The route "<audioFileType>" will return all the audio files of that type

The response of these methods should be one of the following:
- Action is successful: 200 OK
- The request is invalid: 400 bad request
- Any error: 500 internal server error

Recommendations:
- Create only four endpoints (make them generic and usable for all audio file types, do not create four endpoints for each of them)
- The classes should be written in such a way that they are easy to test.
- Write as many tests as you think is enough to be certain about your solution works
- Use SOLID principles.
- Use design patterns where you find it suitable