Introduction: Fingerprinting is basically to identify a signal based on a short sample for it which usually has its intrinsic features and thus these intrinsic features can be used to identify the different varieties or flavors of the signal. Several applications can be directly spotted for such technique. For example:

- Music industry: Identify a song, a singer voice, a tune.
- Medical diagnosis: identify arrhythmia types in ECG signals.

Description:

1. The whole class is responsible for creating some songs repository:
    - Each group will collect N different songs with something common or similar among them (e.g. same singer voice, or same lyrics, or similar tunes), where N is the number of group members.
    - Extract each song into two separate files (music & vocals) and upload it to the common repository. You are not requested to do the separation by code. This is an advanced topics of components separation that can be found in higher level DSP courses (Blind source separation, ICA topics for those who might be interested to give a look). For now, you need to look for some software to do this job for you. Note the separation will hardly be perfect so do not overdo it.
    - Uploaded files have to named according to the following format: GroupX_SongY_Full, GroupX_SongY_Music, GroupX_SongY_Vocals where X is the group number and Y is the song index (1-N).
    - There should be a sheet that contains the songs names to avoid repetition. When you get your songs, put their names along with their similarity feature in the sheet. Anyone who will put the same song later will be penalized.
2. You need to implement your code that can iterate over the songs in the shared folder to generate the spectrogram for each song (3 spectrograms for: the song, the music and the vocals). Note, these files will be generated on your local folder.
3. Note: the spectrogram will always be generated for the first 1 min of the song regardless of its length. This will reduce the computation a little and will also guarantee that all calculations will be done on the same period in all the songs.

4. For each spectrogram:

- Extract the main features in each spectrogram (Do you search for which features to look for and how to extract them). Collect them in some file along with your spectrogram.
- Use hash functions to hash the collected features into a shorter sequence. Please, make sure to use perceptual hash not regular hash. Do your search for the different hash types mentioned here.
- Both outputs from the previous two items are the fingerprint for each song.

5. Now, given any sound file (either song, or vocals, or music), you should be able to generate its spectrogram features and be able to list the closest songs to it from the shared folder. Your program should be able to generate some similarity index to each song then sort them and output the sorting list along with each similarity check in a nice table in your GUI.

6. Your program should be able to take two files, make a weighted average of them (there should be a slider to control the weighting percentage). Then, your software should treat the new summation as a new file and search for the closest songs to it. One would expect the contributing songs to come with the higher similarity index.

Code practice & Programming skills:

- Same practices from previous tasks (i.e. proper variables/functions names, No code repetition, OOP, Logging) will continue with task 4. No new rules. Just stick with these ones please!
- You need to generate an executable version of your program. An executable is the version that can run on others' computers outside the development environment that does not necessarily exist on your user computer.

Expected subtasks breakout:

- Collect the songs, fill the sheet quickly to reserve them to your group.
- Search for a software that splits a song into vocals and music.
- Generate the split files for all the songs on the drive. Note these will be around 60 files, so better to find an automated way to do it. But you are free ofcourse to do it manually.
- Implement a script that generates the spectrogram for each file in a folder.

- Search for the perceptual hash and generate it for each generated spectrogram.
- Search for spectrogram features and how to extract them. Then, develop your own extraction code for all the songs on the shared folder.
- Develop a simple UI that can take two input songs, generate a weighted sum of them, and start looking for the best match for this song among the other songs in the folder. Output all the songs along with their similarity index into a table in your UI.