MixList

Developed by Josh Bean, Jeremy Cruz and Gerard Gaimari
CSE 481 I: Sound Design Capstone, Winter 2019

Problem Description

The plethora of music services available allow users to play the songs that they like and collect them into groups called playlists. While these services offer massive catalogues of music, they lack the ability to create intelligent and fluid mixes as a DJ would.

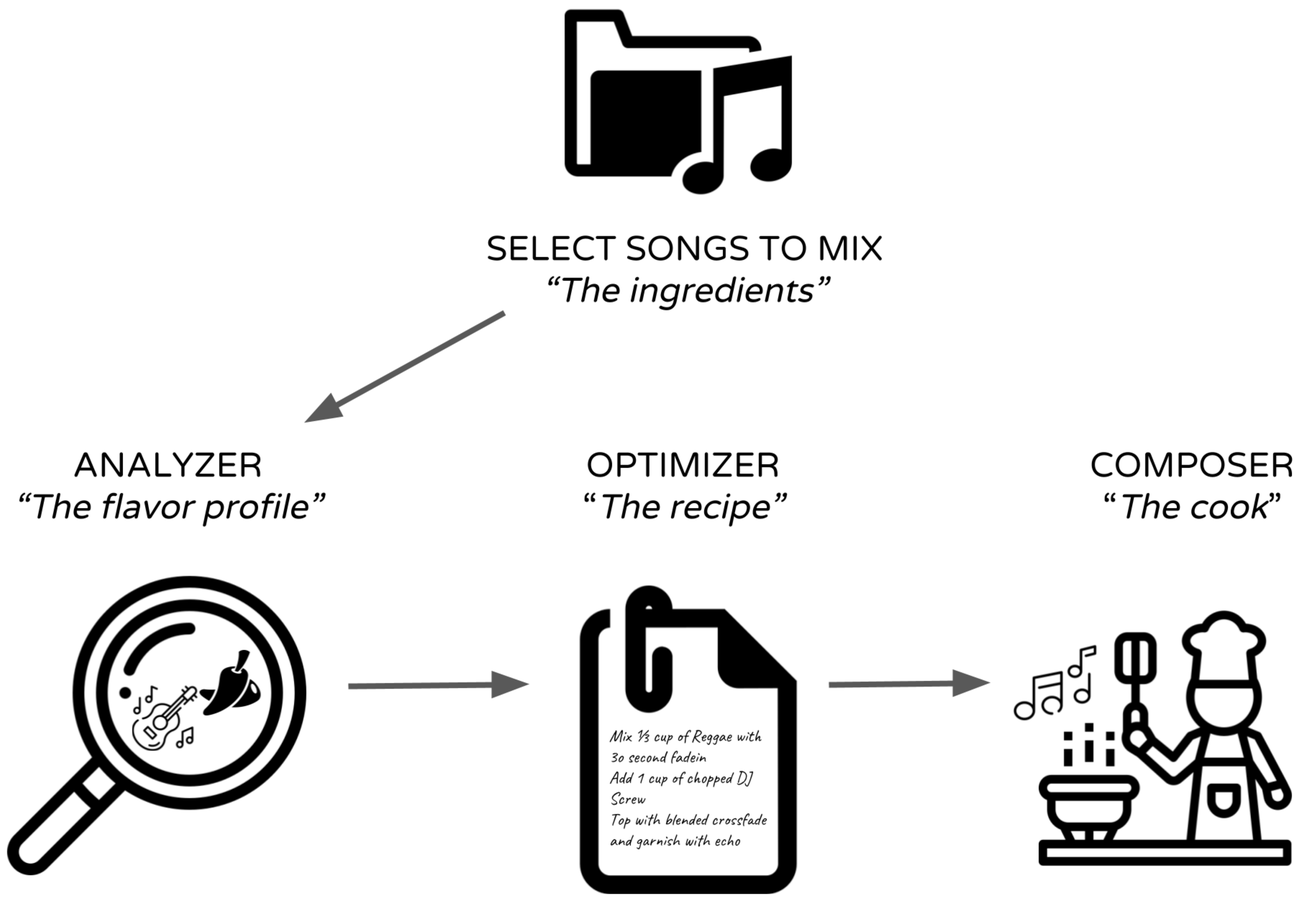
MixList aims to bridge this gap and create a platform that can intelligently mix a users songs with little to no musical knowledge or experience with DJ'ing.

Use Cases

- Don't want to hire a DJ for a party? Use MixList to make a mix instead.
- You want a fast paced workout mix to play when you're at the gym or on a run.
- You curated the best playlist in the world and want to know what it would sound like if a DJ mixed your songs
- Your yoga studio wants you to create a relaxed mix to use for future classes.

Future Work

- Improve on the beat and tempo detection algorithms, account for variations in tempo throughout the songs
- Segment the songs into various sections and analyze the sections separately
  - Label sections as well as "mixable" or "un-mixable" to avoid mixing at unfavorable sections of a song
- Refine criteria for goals which outline a mixtape
- Improve feature comparison and mixtape construction algorithm
- Add different effects and transition types to expand the possible mixes
- Integrate with an existing music platform like Spotify which already has a massive music library and analysis



What are we mixing?

Responsibility: Gather data about the songs selected by the user to in order to provide the optimizer with the musical profiles of each song.
Collects data through:

- Internal analysis (DSP with LibROSA)
- External analysis (query song metadata on Spotify)

Populates an Analysis data structure containing the following features:

- Tempo
- Beat locations (annotated with downbeats)
- Valence (mood)
- Energy
- Danceability
- Musical key

How should it sound?

Responsibility: Use song data provided by the analyzer to create an optimal mix and transition sequence defined by a "style" and high level goals.

- Songs are compared to create "mixes"
- Mixes are scored on probabilistic comparison to style
- Mixes are ordered based on progress to goals
- Transitions are created by taking into account individual song data, mix comparison, style and progress to a goal

Output: A mixtape script to be parsed by the composer containing an ordered list of songs as well as the effects and transitions to apply.

Produce the mix

Responsibility: Parse the optimizer's mix "recipe" into a sequence of instructions to pass to a Digital Audio Workstation (DAW) to perform the audio manipulation.
Our solution uses Audacity along with a scripting language that they have defined in Python.
Steps:

- Imports the audio files into the DAW in the order provided by the recipe
- Aligns the tracks according to where (temporally) they will be mixed
- Applies effects and time stretching from end of the mix to the start to preserve time alignment

Reflections

- The quality of the mixes is strongly dictated by the accuracy of the beats and tempo.
- Digital signal processing problems (musical analysis) are extremely difficult.
- Certain songs are simply not great when mixed together, there's a reason DJ's won't mix certain songs!
- Vocals are difficult to mix, automixers should identify sections of the song with vocals and label them un-mixable

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING