DJ PHAT ENCHANTAH

Olin College of Engineering | Software Design Spring '15
Hannah Fowler | Kai Levy | Hieu Nguyen | Wilson Tang

Background

Our team set out to build our own program to mix and match music. However, before we could create this, we had to understand how a DJ thinks. After researching and talking to Scott Mackinlay '18, we determined that a DJ's skills consist of the following:

- choosing which songs to play depending on key factors
- subtly transitioning from one song to another
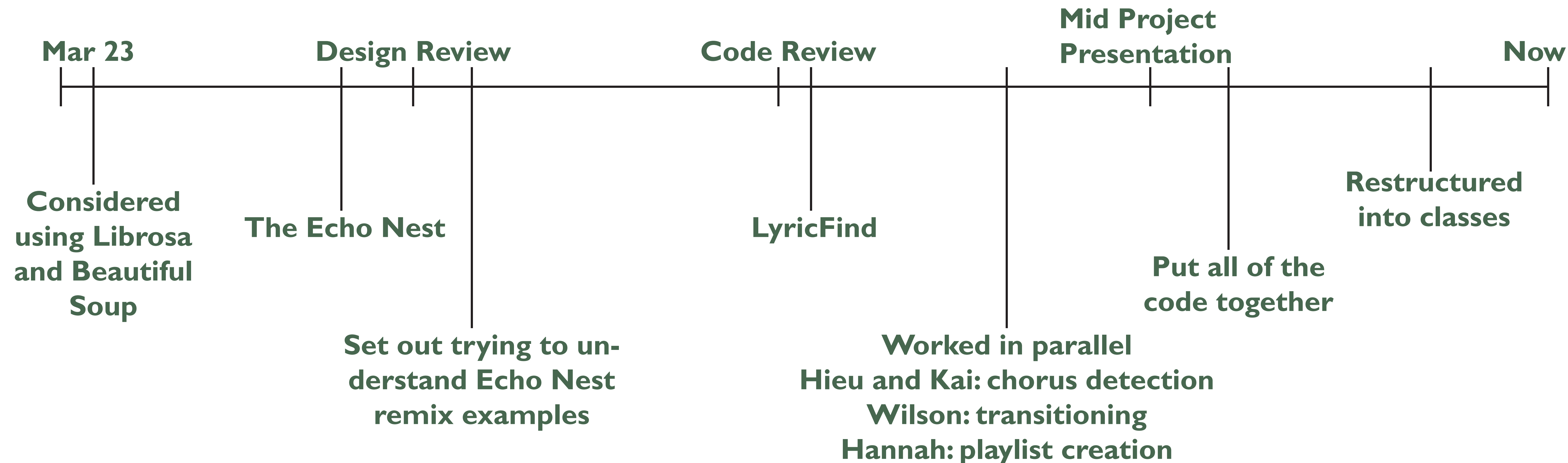- maintaining the energy in the space

For our project specifically, we focused on transitions between songs. In this process, we had to consider multiple components of a song (i.e., bpm, pitch)
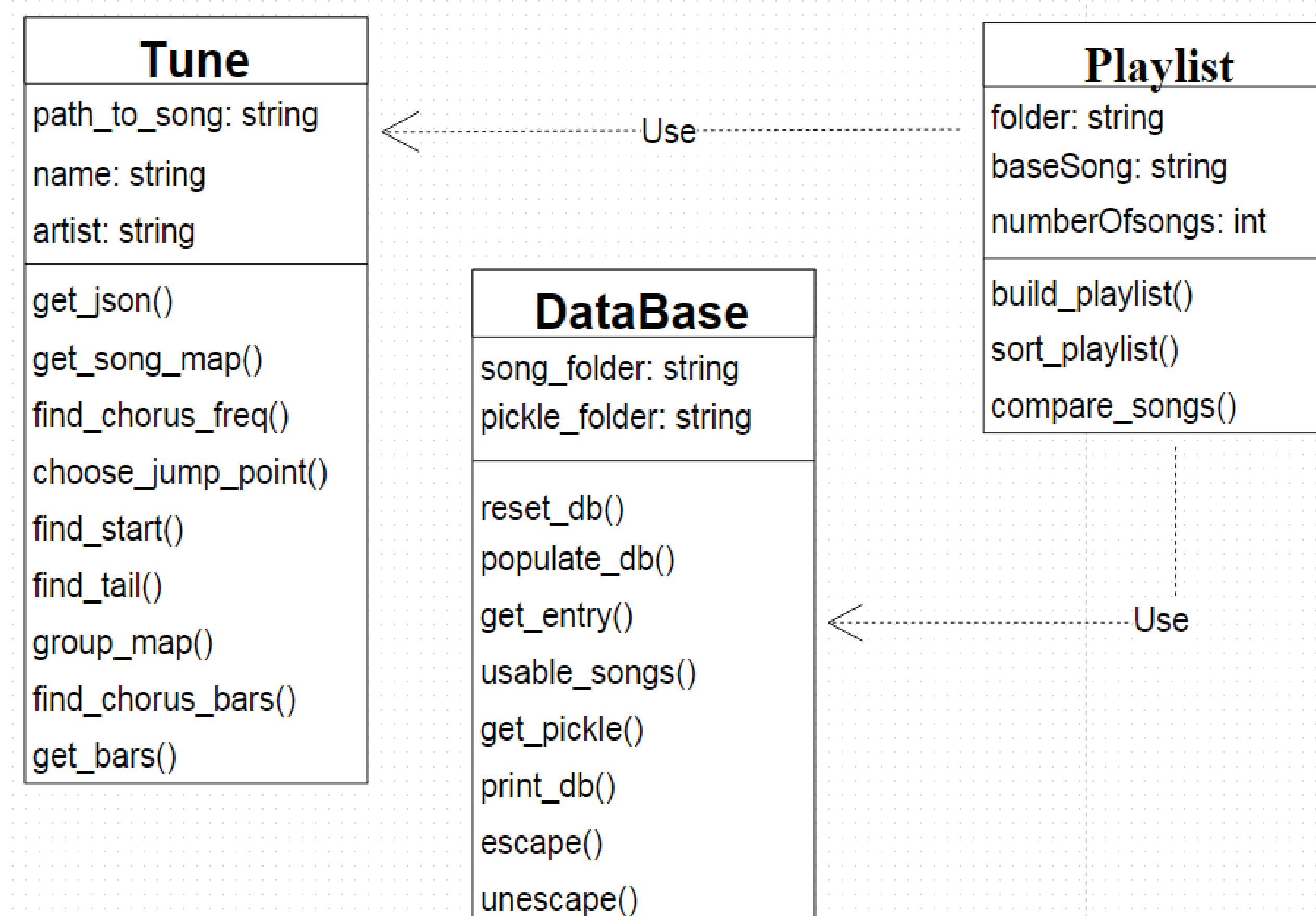
Libraries Used

LyricFind

the echonest

Our Project Evolution

Mar 23 — **Considered using Librosa and Beautiful Soup**

Design Review — **The Echo Nest** / **Set out trying to understand Echo Nest remix examples**

Code Review — **LyricFind** / **Worked in parallel**
Hieu and Kai: chorus detection
Wilson: transitioning
Hannah: playlist creation

Mid Project Presentation — **Put all of the code together**

Now — **Restructured into classes**

Code Architecture

Tune
path_to_song: string
name: string
artist: string

get_json()
get_song_map()
find_chorus_freq()
choose_jump_point()
find_start()
find_tail()
group_map()
find_chorus_bars()
get_bars()

Playlist
folder: string
baseSong: string
numberOfsongs: int

build_playlist()
sort_playlist()
compare_songs()

— Use →

DataBase
song_folder: string
pickle_folder: string

reset_db()
populate_db()
get_entry()
usable_songs()
get_pickle()
print_db()
escape()
unescape()

— Use →

Flow Chart: Our Final Code

Playlist → Input Song → Using information of all songs in database, build playlist of length N which are compatible with input song → Splice Songs based on SONG_MAPS

Database → Parses song collection → Stores information in Database

Tune → Input Song → Calls LyricFinder and Echonest API for data → Returns information + SONG_MAP → Chase_jump_point() → Playlist Class