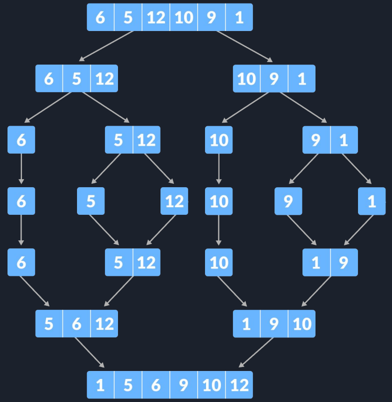
MergeSortMusic Final Report

COSC340 Final Project Presentation

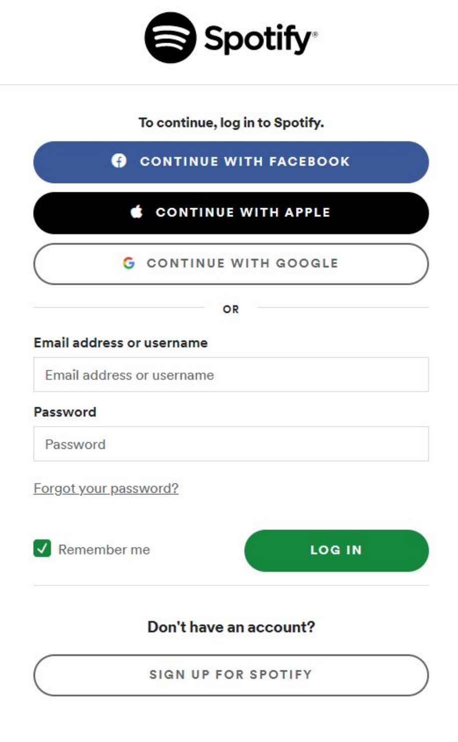
Project/Members

- Team Name: MergeSortMusic
- Team Members: John Carmack, Georgia Stricklen, Jonathan Ting, Shreyank Patel
- We attempted to meet weekly and sometimes biweekly via discord. Discord was also used to keep in close contact when we did not schedule a meeting or in between meetings.



Introduction to Project

- Current playlist creators/sorters: have complicated interfaces, lack in depth personalization of playlist, ineffective/few filters.
- Our solution: create a website that imports playlist from Spotify, the application then implements a merge sort algorithm which eventually prompts user to choose between two songs (this happens until all of the songs have been sorted), and the user exports the playlist back to Spotify.



Spotify

To continue, log in to Spotify.

Email address or username

Email address or username

Password

Password

Forgot your password?

☑ Remember me

Don't have an account?

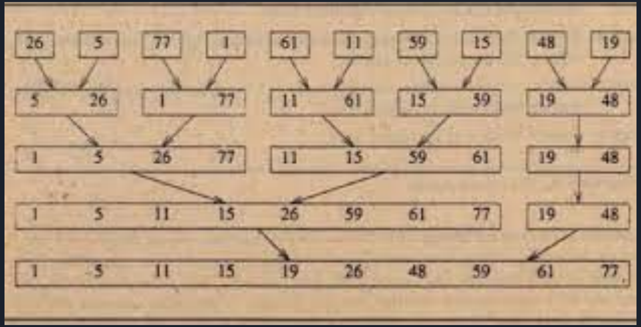
Customer Value

- We maintained that our project would be different from other sorting providers by our more personalized and efficient approach.
- A different sorting method had to be implemented due to time constraints but given enough time merge sort would have been implemented.

Technology

Merge Sort Algorithm:

- Recursive to Iterative
- Keep track of amount of comparisons to be made
- Backend to Frontend



Backend:

- Django - a Python Web framework
- Django REST framework - a toolkit for building RESTful Web APIs

Frontend:

- React - A JavaScript library for building user interfaces
- React Router - A collection of navigational components to be used with React.
- Material-UI - A popular React UI framework
- Babel - A JavaScript compiler
- webpack - A module bundler

Testing

Merge Sort Algorithm:

- Written first in C++ for command line
- Written in Python
- Written in Javascript

Backend:

- Tested backend API calls by using the tools built into Django REST framework for calling API handles without a frontend.
- Created test cases for API calls for most scenarios to verify functionality
    - Such as when the user is not authenticated with Spotify yet, or if user has no playlists.
- Once the API calls were verified individually, they are tested by calling them from a frontend.

Frontend:

- Merged with backend and tested functionality by logging into Spotify, importing playlist to website, and sorted the songs.

Team Management

- Roles remained static
- Jonathan Ting: led development of backend
- Georgia Stricklen: led development of Merge Sort algorithm
- Shreyank Patel and John Carmack: led development of frontend

Goals

- We did not implement collaborative sorting or the ability to view the amount of comparisons the user had yet to make.
- We implemented Selection sort instead of Merge Sort. due to shortage of time
- We did accomplish the vast majority of our goals for this project.

Demonstration

Reflection

Went well:

- Completion of a functional product
- Meeting regularly
- Locating resources to fill in gaps of knowledge

Could use improvement:

- Communicating concepts online

Success: Yes

- We were able to produce a functional product that fulfilled the majority of our initial goals and requirements, learn new programming/development concepts and software, and acquired experience working with others online using GitHub.