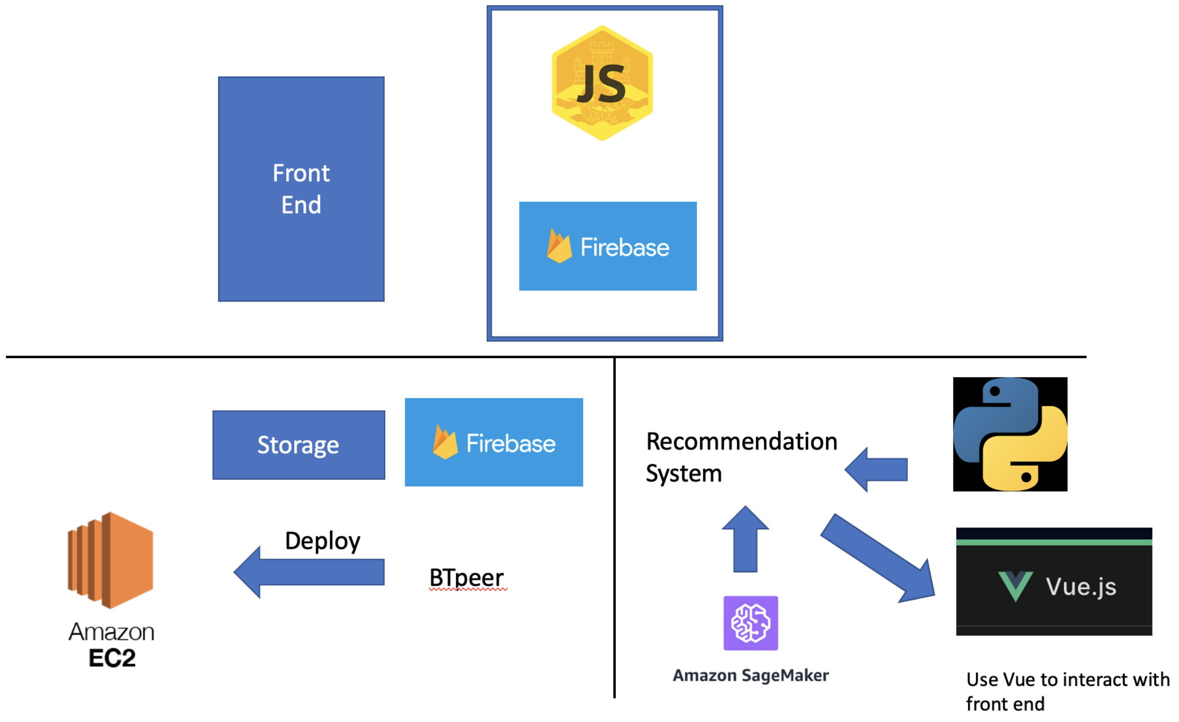
Cloud computing final project part2 report
Music recommendation application

Bohan Zhang, Yuteng Zhang, Yang He, Tianyi Li

Description:

Nowadays music has become part of our lives, and takes place as an important role of joy and entertainment. However, our time of enjoying music is limited, as more and more songs have been created and released, choosing the right music to listen to becomes more and more important. To address this, we propose to build a cloud based music recommendation system that would provide high quality music recommendation based on the users' own taste and interests. The service will be built using a peer to peer network, with a user friendly interface and powerful algorithms.



Architecture design:

Cloud platform used:

1.Google Cloud Platform
2.Amazon Web Services

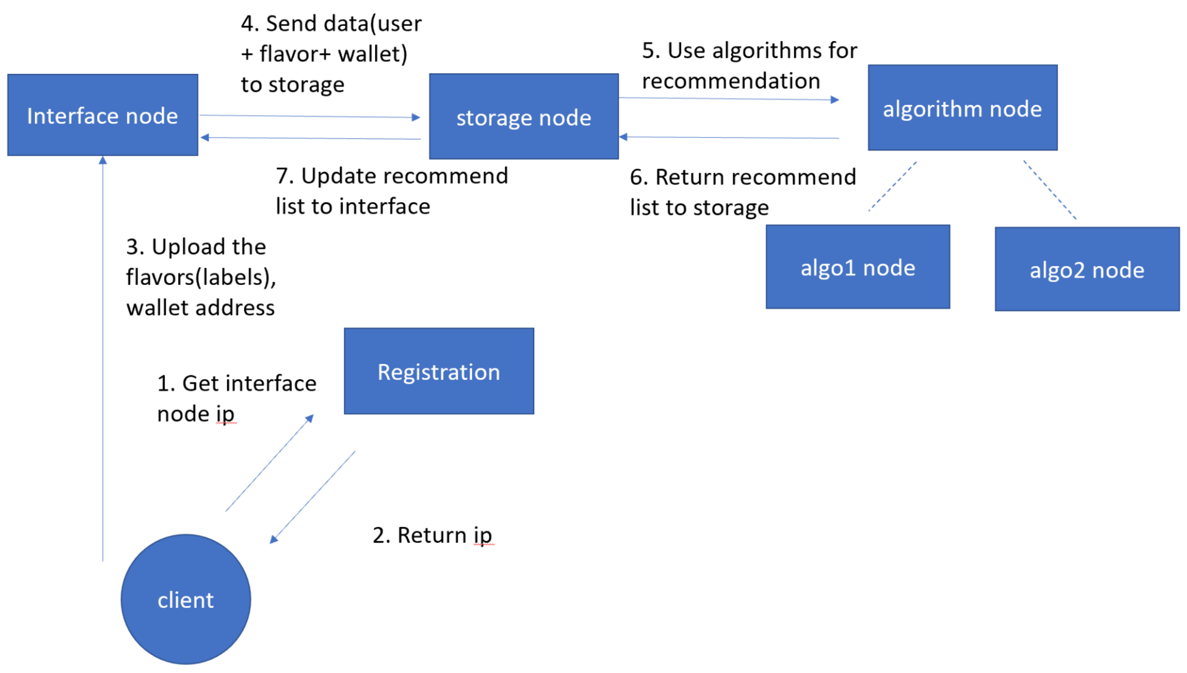
Cloud Components:

1. AWS EC2
2. Firebase
3. Amazon SageMaker

Framework:

BTpeer: https://cs.berry.edu/~nhamid/p2p/

Architecture overview:



Interface node
storage node
algorithm node
algo1 node
algo2 node
Registration
client
4. Send data(user + flavor+ wallet) to storage
5. Use algorithms for recommendation
7. Update recommend list to interface
6. Return recommend list to storage
3. Upload the flavors(labels), wallet address
1. Get interface node ip
2. Return ip

Project implementation:

We will run multiple EC2 instances that will act as nodes in our P2P network. We will use the BTPeer framework to implement said communication. Each node will be responsible for providing one of the following services:

-Interface node.

-recommendation system node.

-storage node

Nodes:

Interface node

Users upload flavor labels and pay for the service through Interface nodes. The IP of an interface node can be found by querying the registration server. Each interface node will run a small web server (port 8080) that serves a front end (HTML and JS) allowing the user to upload their information(account, ip, flavor labels). Then return the list of recommendations. Additionally, each interface node will run their btpeer implementation in a child process, on port 1119.

The interface node will listen to the following message types:

The web server (port 8080) will send an "INIT" message to the p2p node running on port 1119 when a user has successfully paid and uploaded (flavor labels).

The p2p interface node process will then send a (JSON) tasks to storage.
We have chosen to decouple the web server from the BTPeer logic, as it meant we could write our web server in javascript using express. It also creates a natural boundary between front end user interface (presentation layer HTML), “front end interface node” (But backend presentation layer. Is web server hosted on the node) and “backend interface node” (BTPeer implementation).
All block chain logging and BTPeer messaging is performed server side on the node and in theory the web server and BTPeer logic could all have been written in a single python file.
Storage node
Storage node will publish messages from interface nodes to algorithm node and back to interface node. Our storage database is firebase and the node is on port 3000.

Algorithm node
The machine learning algorithm to generate my music list json.
1. Data Pre-processing

- Remove duplicate
- Select useful columns

2. Feature Generation

- Sentiment analysis
- One-hot encoding
- TF-IDF
- Normalization
- Feature generation

3. Content-based Filtering Recommendation

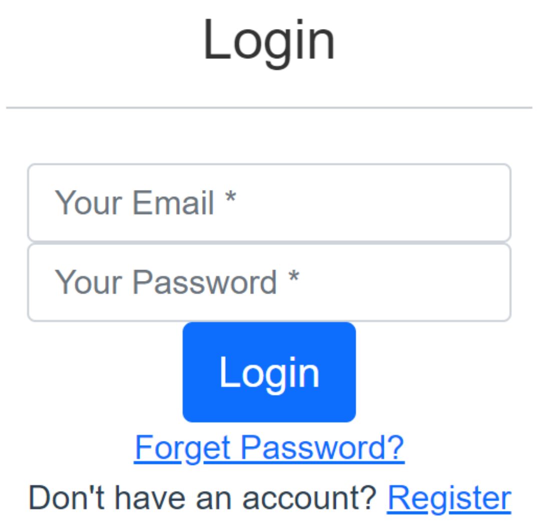
- Getting playlist info
- Generate features
- Generate recommendation

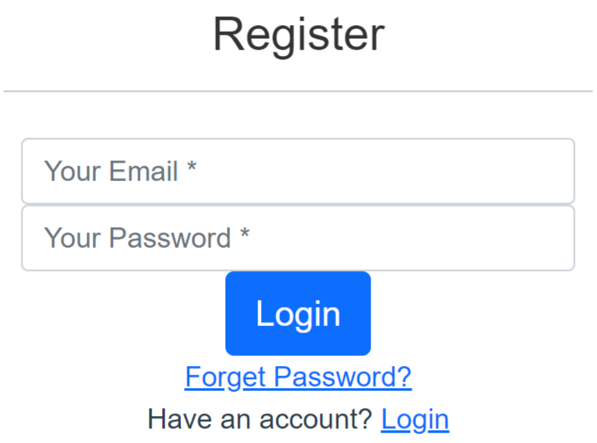
User Flow - “front end”
1. The user queries the registration server using curl (IP known) and asks for the address of an Interface node (within the p2p network).
a) The registration server checks if there exists a node of each type: Interface, storage, algorithm
i) If yes the service is up and the registration server returns the IP of a random Interface node.
ii) If not, the service is down and the registration server lets the client know.
3. The user is redirected to the IP address of the interface node and is asked to upload flavor labels.
4. On successful payment the interface node publishes a “JSON” task to a storage node.

5. The user will receive the recommendation on the interface.

Operation Flow - “back end”
1. A storage node listens for “JSON” messages from an interface node.
2. Interface node send a “JSON” message and hits firebase endpoint. Storage node then sends out a “JSON” message.
3. A Algorithm node will receive a “JSON” message – import data for machine learning
4. Storage node receive messages of “JSON” from algorithm node.
5. Interface node receive message of “JSON” and extract the data to frontend.

Result:





Register

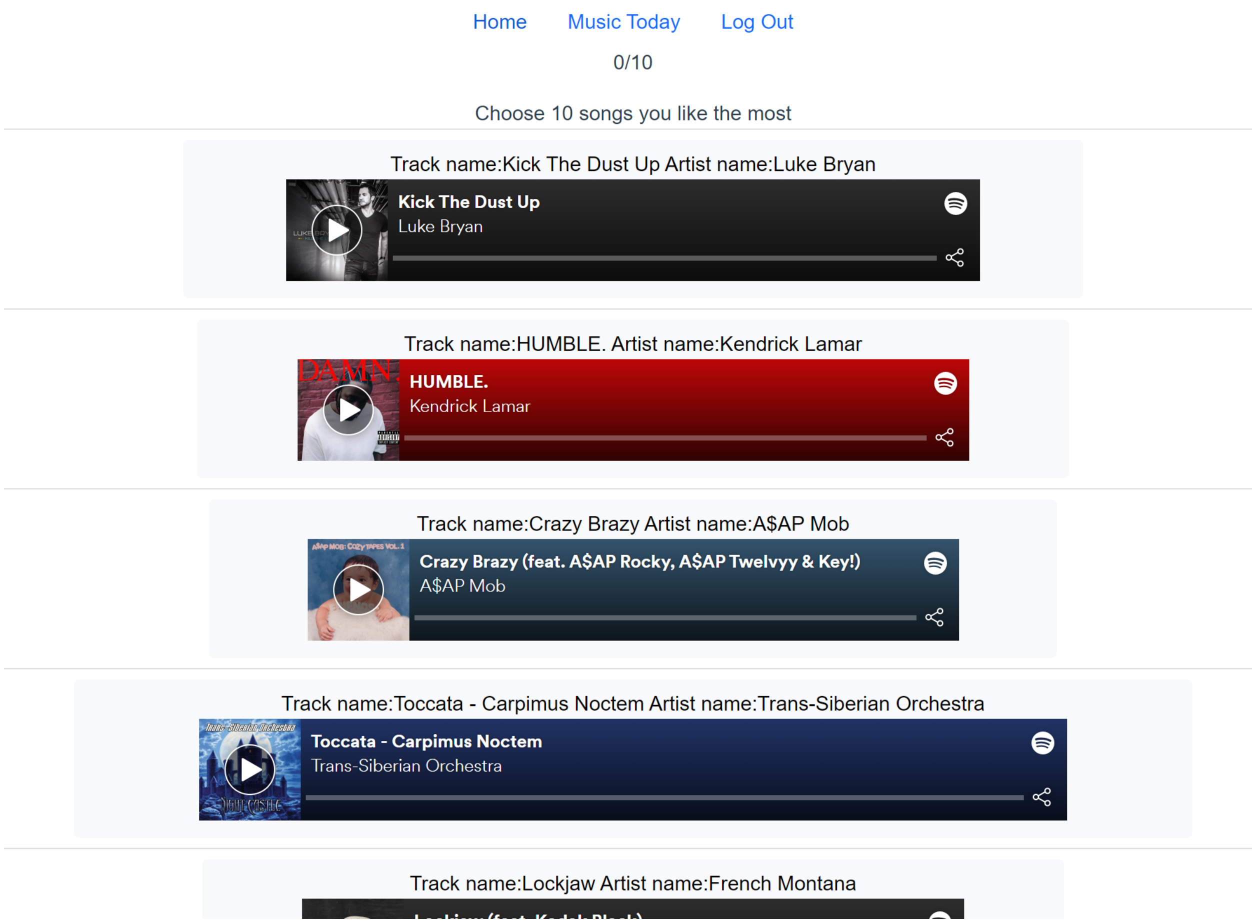
Your Email *

Your Password *

Login

Forget Password?

Have an account? Login



Home Music Today Log Out

0/10

Choose 10 songs you like the most

Track name:Kick The Dust Up Artist name:Luke Bryan

Kick The Dust Up
Luke Bryan

Track name:HUMBLE. Artist name:Kendrick Lamar

HUMBLE.
Kendrick Lamar

Track name:Crazy Brazy Artist name:A$AP Mob

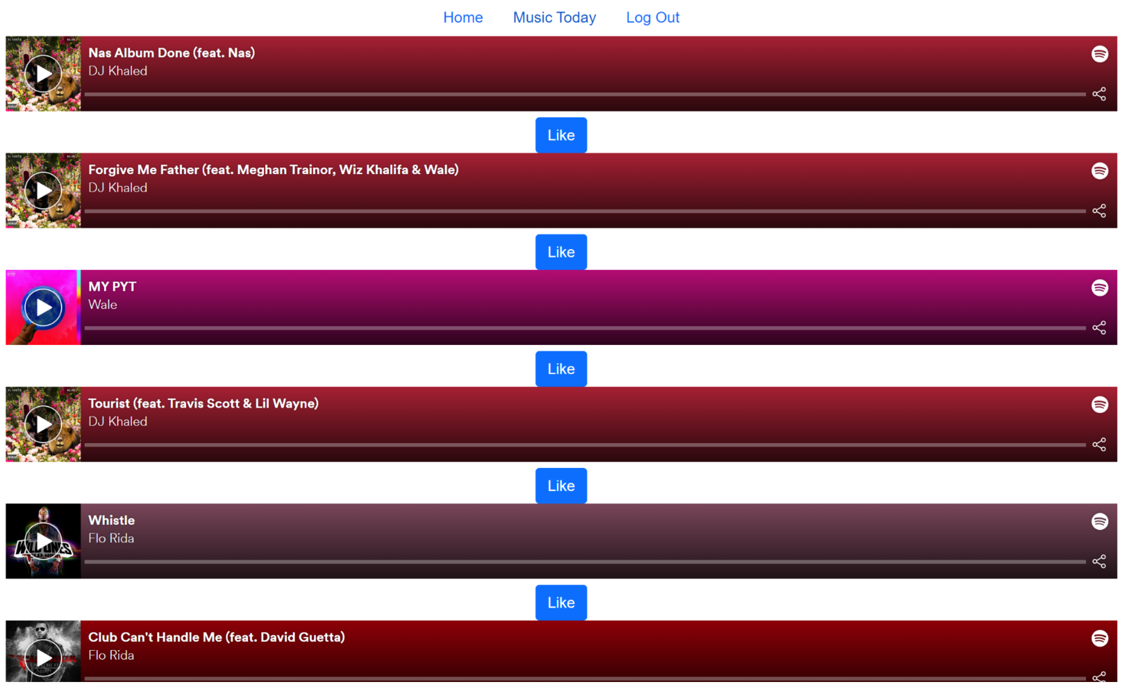
Crazy Brazy (feat. A$AP Rocky, A$AP Twelvyy & Key!)
A$AP Mob

Track name:Toccata - Carpimus Noctem Artist name:Trans-Siberian Orchestra

Toccata - Carpimus Noctem
Trans-Siberian Orchestra

Track name:Lockjaw Artist name:French Montana

Lockjaw (feat. Kodak Black)


Home Music Today Log Out

Nas Album Done (feat. Nas)
DJ Khaled

Like

Forgive Me Father (feat. Meghan Trainor, Wiz Khalifa & Wale)
DJ Khaled

Like

MY PYT
Wale

Like

Tourist (feat. Travis Scott & Lil Wayne)
DJ Khaled

Like

Whistle
Flo Rida

Like

Club Can't Handle Me (feat. David Guetta)
Flo Rida

Future work:

There is still a lot we could do to improve our project.

Algorithm Optimization
The algorithm we currently use is still quite basic and we could do a lot to improve it. We can add more layers and compare with different models to justify the best results.

UI beautify
For now we only had a very simple UI, nothing fancy and pretty. If we have time we could add more varieties to the ui and make it more beautiful

P2P masternode switch
We now set up a masternode, but what if the masternode leaves the network?
We can use blob to get the master node from the P2P network and connect it to the web server.

Node exit
We only have 3 nodes in our P2P network now, but if one of the nodes exits, the application will be down. So we need replaceable nodes and try to join the network.

Reference:

BTPeer:
https://github.com/trozler/interface_nodes_btpeer/blob/master/btpeer/btpeer.py

BTPeer interface node:
https://github.com/trozler/interface_nodes_btpeer/blob/master/btpeer/btpeer_init_handler.py

Recording of sample:
https://nyu.zoom.us/rec/play/NGBIwqEykYI6PnoeEAhuokJxDrzGq_9HLxNpzLIk8ZORQwdz_vG2PKCQXLuOHjN1kY8dyF4C_-w6ZD98.L9G0qif9WS6g3grE?continueMode=true

GitHub:
https://github.com/garyleelty/project