Final Project Check Point - SI507

> Author: Haolin Li
>
> Email: haolinli@umich.edu

>
> Last Updated: 11/29/2023
>
> Github Repository: https://github.com/HumblePasty/SI507

Project Code Repository

Course Github Repository: https://github.com/HumblePasty/SI507

- See project folder: https://github.com/HumblePasty/SI507/tree/master/FinalProject

Data Sources

Description:

- **MusicBrainz** is an open music encyclopedia that collects music metadata and makes it available to the public. It is similar to the freedb project, but with richer data and a more advanced underlying structure that allows for better data retrieval and understanding.
- **Documentation**

Data Format:

The expect data fetching method for this project is through the provided API by MusicBrainz.

The API's architecture follows the REST design principles. Interaction with the API is done using HTTP and all content is served in a simple but flexible format, in either **XML or JSON**.

Query Example:

There are three different types of requests:

```
lookup:   /<ENTITY_TYPE>/<MBID>?inc=<INC>
browse:   /<RESULT_ENTITY_TYPE>?<BROWSING_ENTITY_TYPE>=<MBID>&limit=<LIMIT>&offset=<OFFSET>&inc=<INC>
search:   /<ENTITY_TYPE>?query=<QUERY>&limit=<LIMIT>&offset=<OFFSET>
```

for example, if I want to query `U2` in the artists database and get the data in json format, I should request:

```
https://musicbrainz.org/ws/2/artist?query=Coldplay&fmt=json
```

the returning result of the API would be (sample)

```
{
```

```
    "created":"2023-11-30T02:18:54.006Z",
    "count":48,"offset":0,
    "artists":
    [
        {
            "id":"a3cb23fc-acd3-4ce0-8f36-1e5aa6a18432",
            "type":"Group",
            "type-id":"e431f5f6-b5d2-343d-8b36-72607fffb74b",
            "score":100,
            "name":"U2",
            "sort-name":"U2",
            "country":"IE",
            "area":
            {
                "id":"390b05d4-11ec-3bce-a343-703a366b34a5",
                "type":"Country",
                ...
            },
            ...
        }
        ...
    ]
    ...
}
```

Description of the Return Result

[Document]

For example, as to artist entity, the fields are:

Field	Description
alias	(part of) any alias attached to the artist (diacritics are ignored)
primary_alias	(part of) any primary alias attached to the artist (diacritics are ignored)
area	(part of) the name of the artist's main associated area
arid	the artist's MBID
artist	(part of) the artist's name (diacritics are ignored)
artistaccent	(part of) the artist's name (with the specified diacritics)
begin	the artist's begin date (e.g. "1980-01-22")
beginarea	(part of) the name of the artist's begin area
comment	(part of) the artist's disambiguation comment
country	the 2-letter code (ISO 3166-1 alpha-2) for the artist's main associated country
end	the artist's end date (e.g. "1980-01-22")

Field	Description
endarea	(part of) the name of the artist's end area
ended	a boolean flag (true/false) indicating whether or not the artist has ended (is dissolved/deceased)
gender	the artist's gender (“male”, “female”, “other” or “not applicable”)
ipi	an IPI code associated with the artist
isni	an ISNI code associated with the artist
sortname	(part of) the artist's sort name
tag	(part of) a tag attached to the artist
type	the artist's type (“person”, “group”, etc.)

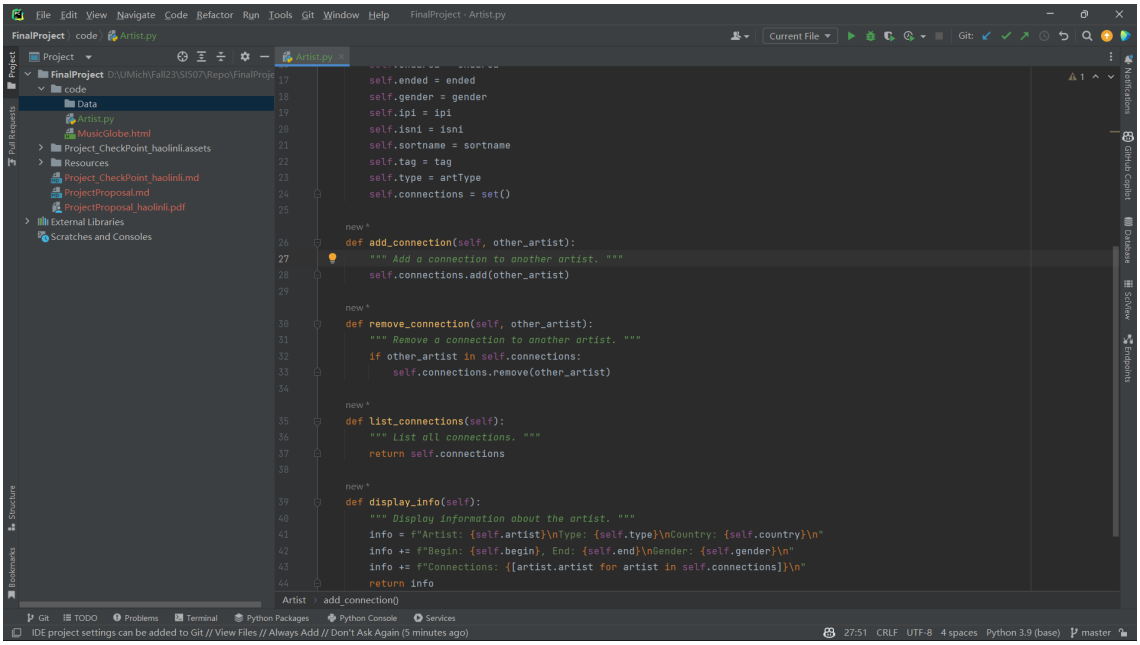
Data Structure

For this project, I plan to:

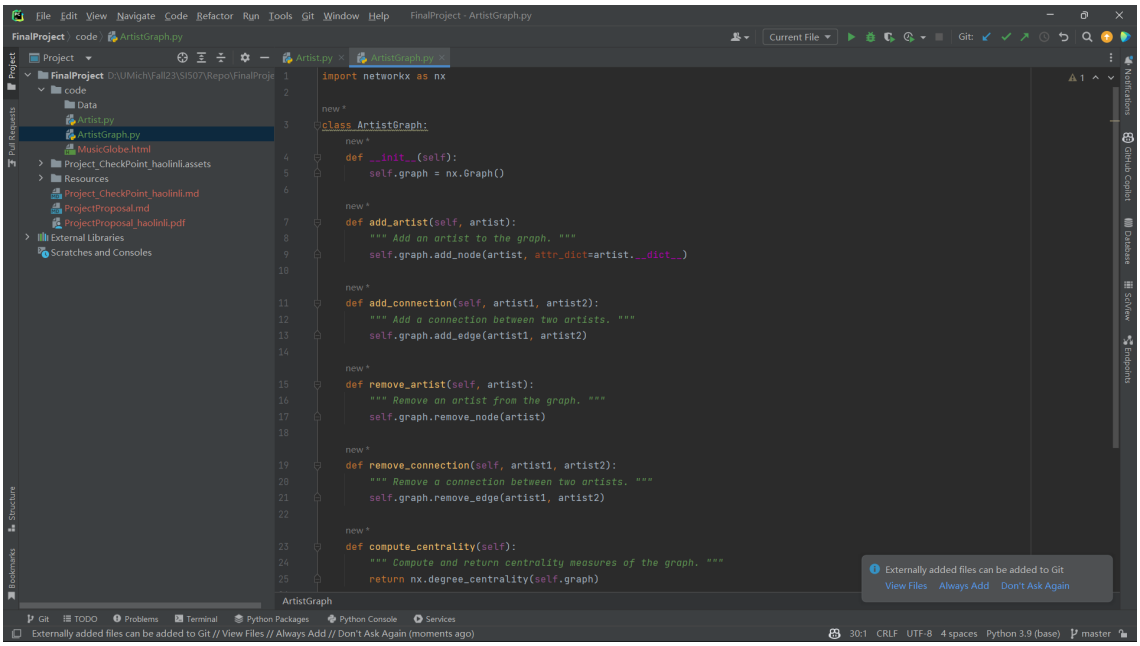
- Utilize a **graph** to represent relationships between musicians.
  (Each artist is a node, and collaborations form the edges.)
- Apply network analysis techniques like **centrality, community detection,** etc., to identify the most influential musicians or the largest collaboration clusters.

Screenshot of Progress

1. `Artist` class to store the entity



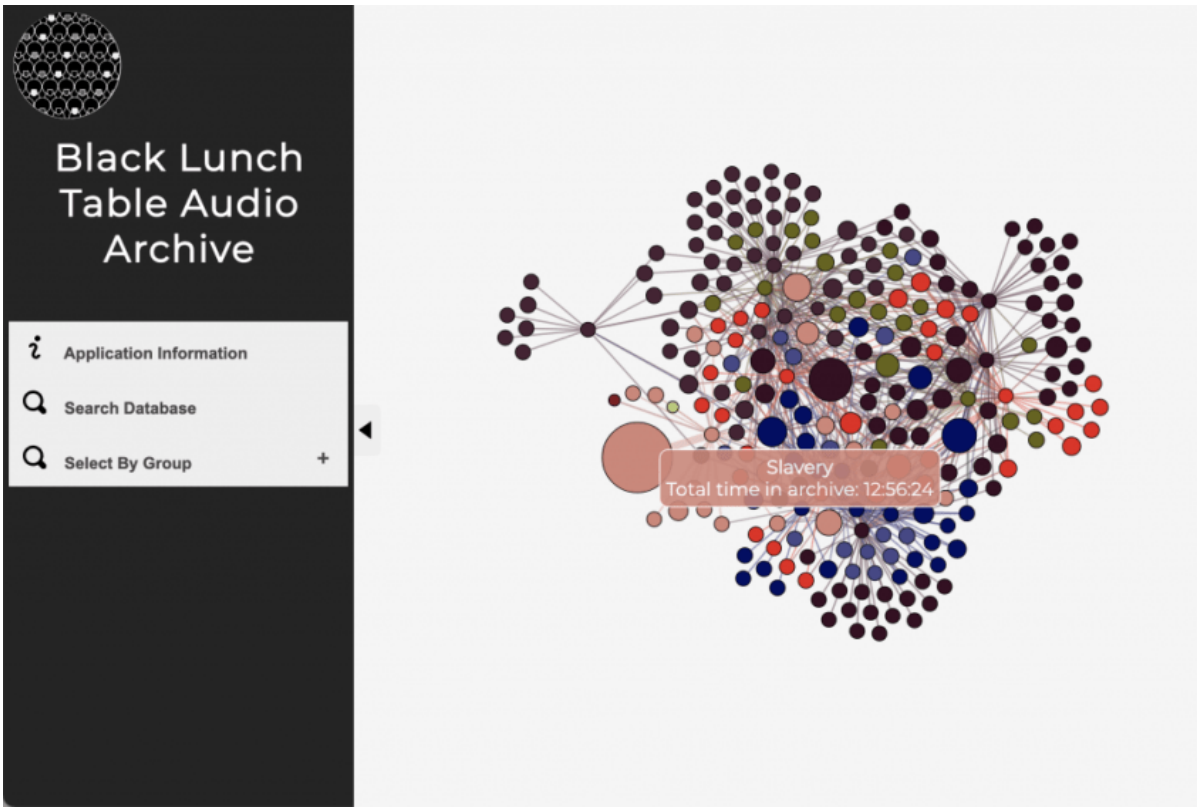
2. `ArtistGraph` class to analyse the network



Interaction and Presentation Plans

- Integrate graphical libraries, like D3.js or Cytoscape.js, to **display the network of musicians** and allow users to visually perceive connections between artists (possibly interact with the networks and navigate through artists etc.)
- Allow users to **search for specific artists** and highlight their position in the network.
- Permit users to opt for viewing specific types of relationships, such as collaborations from a particular decade or within a specific music genre.
- Provide data filtering and sorting functionalities, such as by the number of collaborations, popularity, etc.

The general plan for the ultimate UI and presentation is to show a interactive network of artists. The side bar



Visualization of the artists network using Gephi (reference: https://studentwork.prattsi.org/infovis/visualization/visualizing-artists-connections-with-gephi/)

Extra works if possible:

Below is a image of the previously constructed webpage, here I borrow most of the layout elements from the example provided by ESRI. If possible, integrating the network into the pop-up window would be great (which is to create a network for each selected country)