README.md

CPSC 449 - Fall 2019 - Project 1

Names

Oscar Olazabal
Shivam Shah
Kevin Chen

Dependencies

This project depends on Foreman and all of the Python modules listed in `requirements.txt`. To install foreman and the modules and run:

```
sudo apt install ruby-foreman
pip3 install -r requirements.txt
```

If you run into dependency errors, ensure all of the modules in `requirements.txt` are install properly. You may have to use `apt install` to install some.

Running the Application

To run the application, you have to initialize the database and run each service on their own port. To do so, run the following:

```
./init
foreman start
```

Testing the Application

To test the application you can use the provided `curlfile` executable.

```
./curlfile
```

The executable creates, retrieves, updates, and deletes from the database using the endpoints of the application

Flask Output

```
15:53:56 tracks.1       | 127.0.0.1 - - [16/Oct/2019 15:53:56] "POST /tracks HTTP/1.1" 201 -
15:53:56 tracks.1       | 127.0.0.1 - - [16/Oct/2019 15:53:56] "POST /tracks HTTP/1.1" 201 -
15:53:56 tracks.1       | 127.0.0.1 - - [16/Oct/2019 15:53:56] "POST /tracks HTTP/1.1" 201 -
15:53:56 tracks.1       | 127.0.0.1 - - [16/Oct/2019 15:53:56] "PATCH /tracks/3 HTTP/1.1" 200 -
15:53:56 tracks.1       | 127.0.0.1 - - [16/Oct/2019 15:53:56] "GET /tracks/3 HTTP/1.1" 200 -
15:53:56 tracks.1       | 127.0.0.1 - - [16/Oct/2019 15:53:56] "DELETE /tracks/3 HTTP/1.1" 200 -
15:53:56 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:56] "POST /users HTTP/1.1" 201 -
15:53:56 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:56] "POST /users HTTP/1.1" 201 -
15:53:57 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:57] "POST /users HTTP/1.1" 201 -
15:53:57 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /users/username1 HTTP/1.1" 200 -
15:53:57 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /users/username2 HTTP/1.1" 200 -
15:53:57 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:57] "DELETE /users/username3 HTTP/1.1" 200 -
```

```
15:53:57 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:57] "PATCH /users/username1 HTTP/1.1" 200 -
15:53:57 users.1        | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /users/authenticate HTTP/1.1" 200 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "POST /playlists HTTP/1.1" 201 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "POST /playlists HTTP/1.1" 201 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "POST /playlists HTTP/1.1" 201 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /playlists/2 HTTP/1.1" 200 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /playlists/all HTTP/1.1" 200 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /playlists?creator=username2
HTTP/1.1" 200 -
15:53:57 playlist.1     | 127.0.0.1 - - [16/Oct/2019 15:53:57] "DELETE /playlists/3 HTTP/1.1" 200 -
15:53:57 descriptions.1 | 127.0.0.1 - - [16/Oct/2019 15:53:57] "POST /descriptions HTTP/1.1" 201 -
15:53:57 descriptions.1 | 127.0.0.1 - - [16/Oct/2019 15:53:57] "POST /descriptions HTTP/1.1" 201 -
15:53:57 descriptions.1 | 127.0.0.1 - - [16/Oct/2019 15:53:57] "GET /descriptions/1 HTTP/1.1" 200 -
```

Endpoints

Tracks `PORT 5000`

- `/tracks/<int:id>`
  - `GET` Retrieves a track given its ID
  - `DELETE` Deletes a track given its ID
  - `PATCH` Updates a track given its ID and the JSON data of the fields to be modified
- `/tracks`
  - `POST` Inserts a track given the JSON data of a new track

Playlists `PORT 5001`

- `/playlists/<int:id>`
  - `GET` Retrieves a playlist given its ID
  - `DELETE` Deletes a playlist given its ID
- `/playlists`
  - `GET` Retrieves a playlist of a user given the argument `creator=`
  - `POST` Inserts a playlist given the JSON data of a new playlist
- `/playlists/all`
  - `GET` Retrieves all playlists

Users `PORT 5002`

- `/users/<username>`
  - `GET` Retrieves user's profile (does not include password) given their username
  - `PATCH` Updates a user's password, given an updated password in JSON format
  - `DELETE` Deletes a user given their username
- `/users`
  - `POST` Inserts a new user given their data in JSON format
- `/users/authenticate`
  - `GET` Validates a user and password given JSON data containing a username and password

Descriptions `PORT 5003`

- `/descriptions/<int:id>`
  - `GET` Retrieves user descriptions given the ID of a track
- `/descriptions`
  - `POST` Inserts a user description of a track given a username, a track id, and a description

Data Structure

Below are sample JSONs of the entities.

Tracks

```
{
    "id" : 1
    "track_title": "Track Title",
    "album_title": "Album Title",
    "artist": "Artist's Name",
    "track_length": 2.6,
    "media_file_url": "file:///home/student/Music/mediafile1",
    "album_art_url": "file:///home/student/Music/albumart1"
}
```

Playlists

```
{
    "title": "Playlist Title",
    "creator": "Username",
    "tracks" : [
        "file:///home/student/Music/media_file1",
        "file:///home/student/Music/media_file2"
    ],
    "description": "Description of playlist"
}
```

Users

```
{
    "username" : "User name",
    "user_pass" : "Password of user",
    "disp_name" : "Display name",
    "email" : "useremail@email.com",
    "url_homepage" : "http://userhomepage.com"
}
```

Descriptions

```
{
    "track_id": 1,
    "user_name": "User name",
    "description": "This is a track description"
}
```