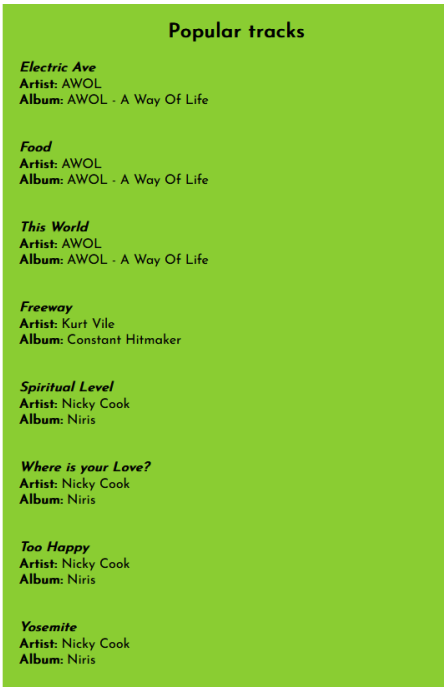
New Feature Report

A ‘cool feature’ we have implemented into the Music Web Application is a sidebar (on the right) with the most popular (top 15) tracks on the Music Web Application based on the number of Reviews. If there are no Reviews and/or multiple tracks have the same amount of Reviews then the tracks with the same amount of Reviews will be in order of ID number.

The Tracks are presented in a format that consists of the track title (in bold), underneath the title is ‘Artist:’ (in bold) followed by the artist's name, underneath the Artist is ‘Album:’ (in bold) followed by the Artist name ‘-’ Album name.

If the Track name is clicked, the page will redirect you to another page consisting of information about the track, such as name, artist, album, genres, duration, URL and a list of the track reviews. In this page you are also able to leave a review about the chosen track.



To implement this new feature, we defined a function (in the Service Layer) within utilities/services.py to sort the tracks in the Abstract Repository by Reviews called sort_tracks_by_reviews(repo).

```
def sort_tracks_by_reviews(repo: AbstractRepository):

    sorted_by_reviews = sorted(repo.get_tracks(), key=lambda x: len(x.track_reviews), reverse=True)
    return sorted_by_reviews
```

Within the utilities/utilities.py We also had to create a Flask Blueprint in order to organise application functionality into different units. Registering the utilities blueprint was done in the __init__.py file, registering the blueprint was crucial in order to extend the application with its given contents.

```
from .utilities import utilities
app.register_blueprint(utilities.utilities_blueprint)
```

To get the top 15 most popular tracks based on the number of Reviews, we had to define a function called get_top_tracks(). get_top_tracks() would allow us to retrieve the top 15 tracks with the highest number of Reviews.

This function was implemented by first calling the sort_tracks_by_reviews method (mentioned above, in the Service Layer, ie. utilities/services.py) on the list of tracks within the Repository.

After retrieving the list of tracks in sorted order, an if else statement was implemented. If the quantity (quantity is 15, because quantity of most popular tracks displayed is at most 15) is greater than the number of tracks in the list of sorted tracks retrieved above, we would assign the entire sorted list of tracks to the variable top_tracks, meaning the most popular track list would consist of the entire sorted tracks list in order. Else, ie. the number of tracks in the list of sorted tracks retrieved above is greater than 15, we would retrieve the top 15 tracks within the sorted list of tracks and assign it to the variable top_tracks.

Afterwards we would return the variable top_tracks in order to present the top 15 tracks within the Music Web Application based on Reviews.

```
def get_top_tracks(quantity=15):

    sorted_tracks = services.sort_tracks_by_reviews(repo.repo_instance)
    top_tracks = []
    if quantity > len(sorted_tracks):
        top_tracks = sorted_tracks[:]
    else:
        top_tracks = sorted_tracks[:quantity]
    return top_tracks
```

To present the Tracks in the format we wanted we had to structure our HTML file for the most popular tracks in a specific way.

- The HTML file starts with declaring the position/id as a “sidebar”

- Using Header 1 to present “Popular Tracks” at the top of the sidebar
- Using jinja to loop through the selected tracks using a For Loop
- Retrieving the track by using the unique track ID and displaying the track title. The track title is displayed in bold by using the HTML element.
- ‘Artist:’ is also put in bold, followed by retrieving the tracks artists full name using a jinja statement
- ‘Album’ is also put in bold, followed by retrieving the tracks albums title using a jinja statement

```
<aside id="sidebar">
    <header>
        <h1>Popular tracks</h1>
    </header>
        {% for track in selected_tracks %}
            <div id="link_bar">
                <br>
                <a href="{{ url_for('tracks_bp.display_track_info', track_id = track.track_id) }}"><em><strong>{{ track.title }}</strong></em></a><br>
                <strong>Artist:</strong> {{ track.artist.full_name }}<br>
                <strong>Album:</strong> {{ track.album.title }}
                <br>
                <br>
            </div>
        {% endfor %}
</aside>
```