Project Name: **SPOTYFI**

Authors: Charlito Piao & Cedez Gulane

About: Store your favorite artist with their albums and songs to let you remember what to listen.

Framework used:

1. Express Js
2. Mocha & Chai (For Testing)
3. Nodejs

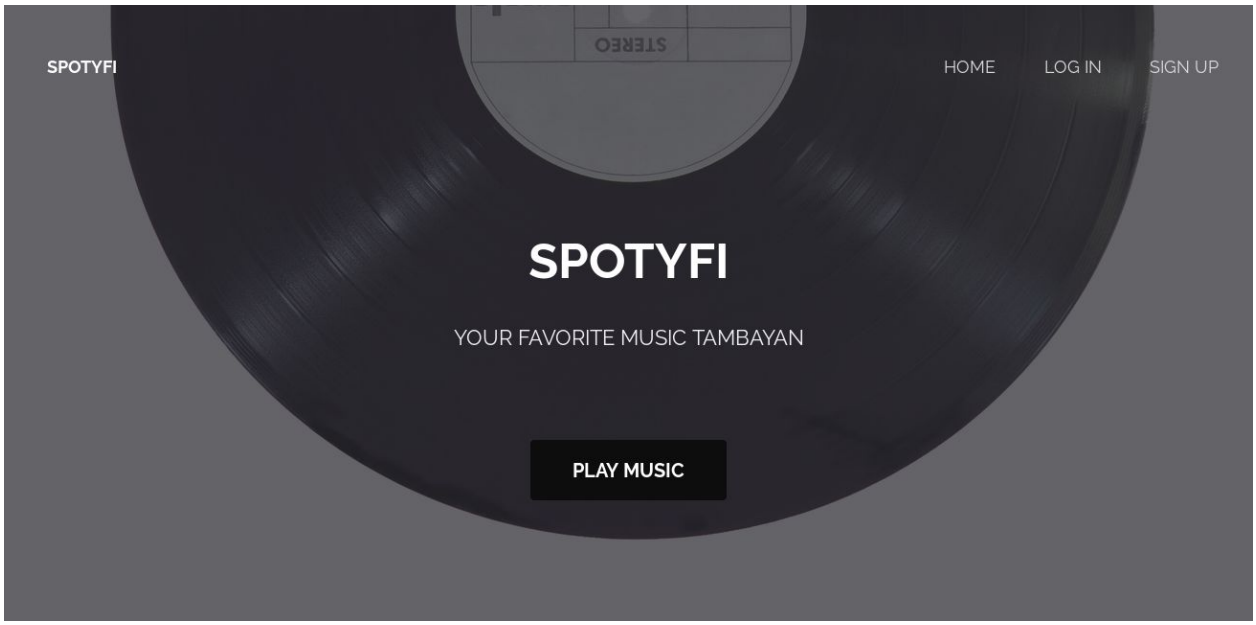
Instructions to run:

1. Install nodejs at: https://nodejs.org/en/download/ or follow https://docs.npmjs.com/getting-started/installing-node
2. Install express and follow instruction at: https://expressjs.com/en/starter/installing.html
3. Download project at: https://github.com/Sedce/music_storage
4. Open terminal and run: node server.js or Install nodemon and run: nodemon server.js

To run tests:

Open browser and put the link address: http://localhost:3000/tests

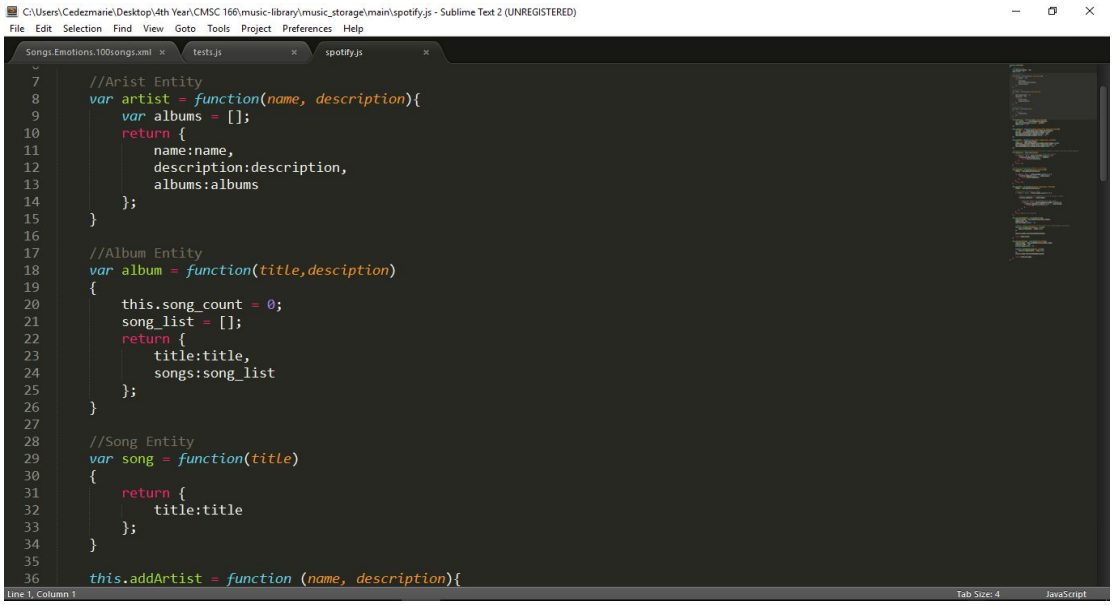
A look at the main page:



Self-Documenting Code

Classes

- Classes are well named and can be treated as black boxes with comments that doesn't repeat the use of the function but tell what it is.



Routines

- The same with classes, functions are clearly named and understandable. Each routine have specific task and each part of the routine is important to run those specific task and unnecessary comments are deleted.

C:\Users\Cedezmarie\Desktop\4th Year\CMSC 166\music-library\music_storage\main\spotify.js • - Sublime Text 2 (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

Songs.Emotions.100songs.xml | tests.js | spotify.js

```javascript
    }
    this.addArtist = function (name, description){
        var _artist = new artist(name,description);
        list_of_artists[this.length] = _artist;
        this.length += 1;
    };

    this.addAlbum = function (singer,album_title,album_description){
        var alb = new album(album_title,album_description);
        alb_len = this.getArtist(singer).albums.length;
        this.getArtist(singer).albums[alb_len] = alb;
        this.getArtist(singer).albums.length += 1;
    };

    this.addSong = function (song_title, album_title, artist){
        var s = new song(song_title);
        songs_len = this.getAlbum(artist,album_title).songs.length;
        this.getAlbum(artist,album_title).songs[songs_len] = s;
        this.getAlbum(artist,album_title).songs.length += 1;
    };

    //Instead of sending the index in the array of singers it sends the Artist entity instead
    this.getArtist = function(singer){
        //finds the singer in the array and returns the singer;
        for (var i  = 0; i < list_of_artists.length; i++) {
            if((list_of_artists[i].name) === singer){
                return list_of_artists[i];
            }
        }
        return -1;
```

Line 34, Column 6 | Tab Size: 4 | JavaScript

Data Names

- Data type names are descriptive enough and named well. Each variables used have specific purposes. We used different convention for local variables, class variables and global variables.

The class variable names starts with a capital letter while our function variable names starts with small letter. The Artist, Album and Song, in our case, are considered as classes since we follow functional programming using javascript.

```javascript
function Spotify()
{
  //Renamed Array
  var list_of_artists = [];
  this.length = 0;

  //Arist Entity
  var Artist = function(name, description){
    var albums = [];
    return {
      name:name,
      description:description,
      albums:albums
    };
  }

  //Album Entity
  var Album = function(title,desciption)
  {
    this.song_count = 0;
    song_list = [];
    return {
      title:title,
      songs:song_list
    };
  }

  //Song Entity
  var Song = function(title)
  {
    return {
      title:title
    };
  }
```

The getAlbum, getSong, and displayAllAlbums are functions/routines. Also, we renamed the loop counters to ‘count’ instead of the usual i,j,k.

C:\Users\Cedezmarie\Desktop\4th Year\CMSC 166\music-library\music_storage\main\spotify.js - Sublime Text 2 (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

Songs.Emotions.100songs.xml | tests.js | spotify.js

```javascript

    //Return the Album entity instead of the index
    this.getAlbum = function(artists, album_name){
        artist = this.getArtist(artists);

        for (var count = 0; count <= artist.albums.length; count++) {
            if(artist.albums[count].title === album_name){
                return artist.albums[count];
            }
        }
        return -1;
    }

    this.getSong = function(song_title, album_name, artists){
        artist = this.getArtist(artists);

        for (var count = 0; count <= artist.albums.length; count++) {

            //Check if album we are looking for is in the singer's album
            if(artist.albums[count] === album_name){

                //Find the song we are looking inside the album
                for(var count_ = 0; j < artist.albums[count].songs.length;count_++){
                    if(artist.albums[count].songs[count].title === song_title){
                        return count_; //return the song
                    }
                }
            }
        }
        return -1; //if not existing
    }

    this.displayAllAlbums = function(artist){
        list_of_albums = this.getArtist(artist).albums;
        album_array= [];
        list_of_albums.length -= 1;

```

Line 87, Column 81 Tab Size: 4 JavaScript

Control

- Minimized nesting and grouped all similar statements in one function

```javascript
    this.displayAllAlbums = function(artist){
      list_of_albums = this.getArtist(artist).albums;
      album_array= [];
      list_of_albums.length -= 1;

      //Change complex algorithm to much simpler and understandable algorithm
      function checkAndInclude(album, index){
        album_array[index] = album.title;
      };

      list_of_albums.forEach(checkAndInclude);

      return album_array;
    }

    this.displayAllSongs = function(artist,album){
      list_of_songs = this.getAlbum(artist,album).songs;
      array_of_songs = [];
      list_of_songs.length -=1;

      function checkAndInclude(song, index){
        array_of_songs[index] = song.title;
      };
      list_of_songs.forEach(checkAndInclude);

      return array_of_songs;
    }
  }
```

Layout

- The programs layout shows its logical structure. Our program follows formatting in common programming languages.

In this snippet, this.getArtist is the function/routine name which has the highest hierarchy followed by the for loop and then the conditional if statement. Indent style is based on K&R and variants: 1TBS, Stroustrup, Linux kernel, BSD KNF.

```
  this.getArtist = function(singer){
    //finds the singer in the array and returns the singer;
    for (var count  = 0; count < list_of_artists.length; count++) {
      if((list_of_artists[count].name) === singer){
        return list_of_artists[count];
      }
    }
    return -1;
  }
```

Design

- For the design, we used bootstrap and changes are done according to the requirements of our project. The program is written according to the project's goal which is to make a music library.

```
<!DOCTYPE HTML>
  <!--
    Spatial by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license (templated.co/license)
  -->
  <html>
    <head>
      <title>Spotyfi</title>
      <meta charset="utf-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
      <link rel="stylesheet" href="css/main.css" />
    </head>
    <body class="landing">

      <!-- Header -->
        <header id="header" class="alt">
          <h1><strong><a href="index.html">Spotyfi</a></strong></h1>
          <nav id="nav">
            <ul>
              <li><a href="index.html">Home</a></li>
              <li><a href="login.html">Log in</a></li>
              <li><a href="signup.html">Sign up</a></li>
            </ul>
          </nav>
        </header>

        <a href="#menu" class="navPanelToggle"><span class="fa fa-bars"></span></a>

      <!-- Banner -->
        <section id="banner">
          <h2>Spotyfi</h2>
          <p>Your favorite music tambayan</p>
          <ul class="actions">
            <li><a href="#" class="button special big">Play Music</a></li>
```

Guide:
https://groundberry.github.io/development/2016/12/10/testing-express-with-mocha-and-chai.html