COMP 3512
Assignment #1: Single-Page App

Version: 1.1 October 3, changes in yellow
Version: 1.2 October 4, changes in fushcia
Due Sunday October 16, 2022 at midnightish

Overview

This assignment provides an opportunity for you to demonstrate your ability to generate a PHP-based data-generated web site. **You can work in pairs or by yourself (I recommend you do work in pairs if at all possible).** Please don't ask me to be in a group of 3.

The data you have been provided with is Spotify data about hit songs from 2016 to 2019 (sorry about living in the past but that's the data I was able to find). Your site will provide the user with the ability to browse and filter songs, examine single songs, and add songs to a favorite list.

Beginning

I feel foolish saying this in a third-year university course, but it is your responsibility to read all the assignment instructions thoroughly and carefully. If you are unclear about something, ask me. But before you do, read the material in question again!

Grading

The grade for this assignment will be broken down as follows:

Visual Design and Styling	20%
Programming Design	15%
Functionality (follows requirements)	65%

Files

You will be able (eventually) to find the most recent version of the database scripts at the GitHub repo for the assignment:

`https://github.com/mru-comp3512-archive/f2022-assign1`

Periodically, I may upload a revised version of the database if errors are found in the data.

GitHub

You need to have "a single source of truth" when it comes to your source code. That is, there must be a "master" location that contains the definitive version of your source code. If you are working as a pair, then one of you will have to decide whose github account has the master location. But even as a pair, each group member will need their own Github account.

Your repo on Github can be either private or public. The free GitHub account doesn't allow private repos; if you sign up for the Student Developer Pack (https://education.github.com/pack) you can have free private repos while a student. The other way is to email me, and I can create a private repo under our department's github organization (https://github.com/MountRoyalCSIS) for your group. You would need to supply the github names or emails for each member. You can also decide to use a public repo.

You will want to push out updates fairly frequently (1-2 times a day when working on it, or more frequently if lots of work is being done). I will be examining the commits when marking. You can push your content to GitHub via the terminal, using the following commands (not the stuff in square brackets though, as those are comments):

```
git init     [only need to do this one command once for your assignment]
git add *
git commit -m "Fixed the rocket launcher"     [alter message and name as appropriate]
git remote add origin https:... your-repo-url.git   [do this just once]
git push -u origin master   [login using your own individual github credentials]
```

For more information about Git and GitHub, read pages 571-577 of textbook (2nd Edition). There are many online guides to git (for instance, https://guides.github.com/introduction/git-handbook/).

~~Submitting and Hosting~~

~~You will be using PHP and either SQLite or MySQL in this assignment. Eventually this will mean your assignment will need to reside on a working host server. In the labs, you made use of XAMPP on your local development machine as both host server and as a development environment. While easy, it means no one but you (or your group members and myself) can ever see your work. If you ever want to use your assignment as a portfolio piece or have it marked by me, it needs to be on a working host server. Static file hosts such as GitHub Pages will not work for this one.~~

~~For this assignment, these are your hosting/submitting options:~~

- ~~Heroku. It has a free tier (but only until Nov 1) and is/was a popular hosting option that integrates nicely with github. It requires that at least one person register with heroku and install its CLI software on their computer. That person then uses a few command line instructions to copy software from github repo to the heroku servers. Some additional commands are needed to add mysql (or MariaDB which is the same thing) via third-party marketplace. Using sqlite is much easier, but it does require enabling it via the composer.json file for the project (google it). I will provide a lab that illustrates this option.~~
- ~~Inexpensive Hosting. These usually won't be free (often about $5/month), but some are free. Once setup, your site can live forever. Possibilities include epizy, infinityfree and bluehost.~~
- ~~Google Cloud Platform. In this case, you will setup a virtual LAMP stack on a Compute Engine (a virtualized server). Will be likely too expensive over time, but you can get free credits that will last for a few months. I will provide a lab that illustrates this option.~~
- ~~Amazon Web Services (AWS). Similar to Google's offerings. Will be likely too expensive over time, but you can get free credits that will last for a few months. Probably easiest approach is launching LAMP stack on AWS Lightsail (first month for lightsail is free).~~
- ~~Make use of a Docker LAMP container and then deploy container on any host environment that supports Docker. This will provide experience in the most important DevOps platform and will thus be excellent to put on your resume.~~

~~This step is going to take some time, so don't leave it till the very end. The hosting should be arranged and tested (i.e., verify PHP and MYSQL work) a few days before the assignment is completed!!!~~

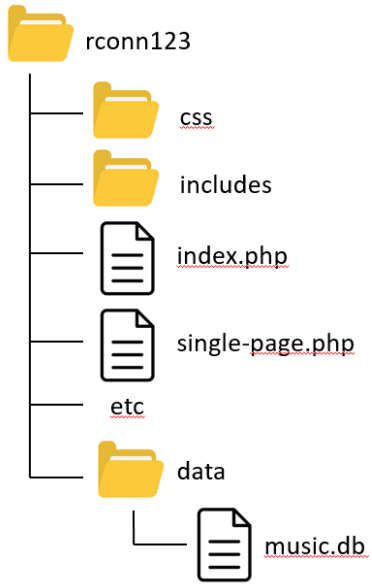
~~When everything is ready, send me an email with the URL of the site and the github repo link (if private, add me as a collaborator).~~

Submitting

Unfortunately, the removal of Heroku's free tier has made hosting PHP a lot more work and/or money. So this year, there is no hosting requirement for your PHP assignment.

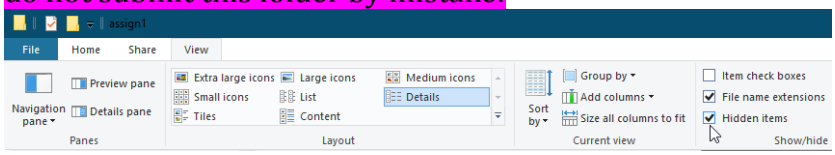
Instead you will submit using the traditional MRU submit drive approach:

1. Your assignment **must** be in a folder whose name is the same as your user id (e.g., rconn123). If you are in a group, pick one your user ids for the folder name. This folder **must** contain your index.php pages and your other files. They must **not** be nested within some other folder! For instance, your folder must look something similar to the following:



NOTE: DO NOT SUBMIT THE .git subfolder!!!

2. If you submit a zip file instead (i.e., you use the mymru web portal to upload your assignment), then you are forcing me to do extra steps (unzip then copy), which will make me disappointed and unhappy, which is not how you want me to be when marking your assignment. So, only use the web portal submit approach as a last resort.
3. Make sure you use the SQLite version of the database rather than the MySQL version. That is, make sure your connection string is correct for the SQLite file which must be in a folder named data.
4. Do NOT submit the .git subfolder, if you have one!!! This folder contains literally hundreds of extra files, which means copying your submitted folder to my machine for marking takes 10 minutes instead of 30 seconds. If you do this, I will simply give you a mark of zero. You may have system files hidden, so be sure to make them visible so you do not submit this folder by mistake:



5. When you are ready to submit, I recommend making a fresh folder with the appropriate name in your htdocs folder. Copy the relevant files. Test pages (still work with the SQLite connection?). Check folder structure. Check there is no .git folder. Copy/Submit it.

Data Files

Data has been provided: a SQLite database, an Excel file (for quick examination), and (eventually) SQL Server script files. The database consists of four tables: songs, artists, genres, and types (artist types). The key table is songs, which contains key information about the songs: title, artist id, genre id, and year. It also contains a variety of Spotify music analysis metrics, which are:

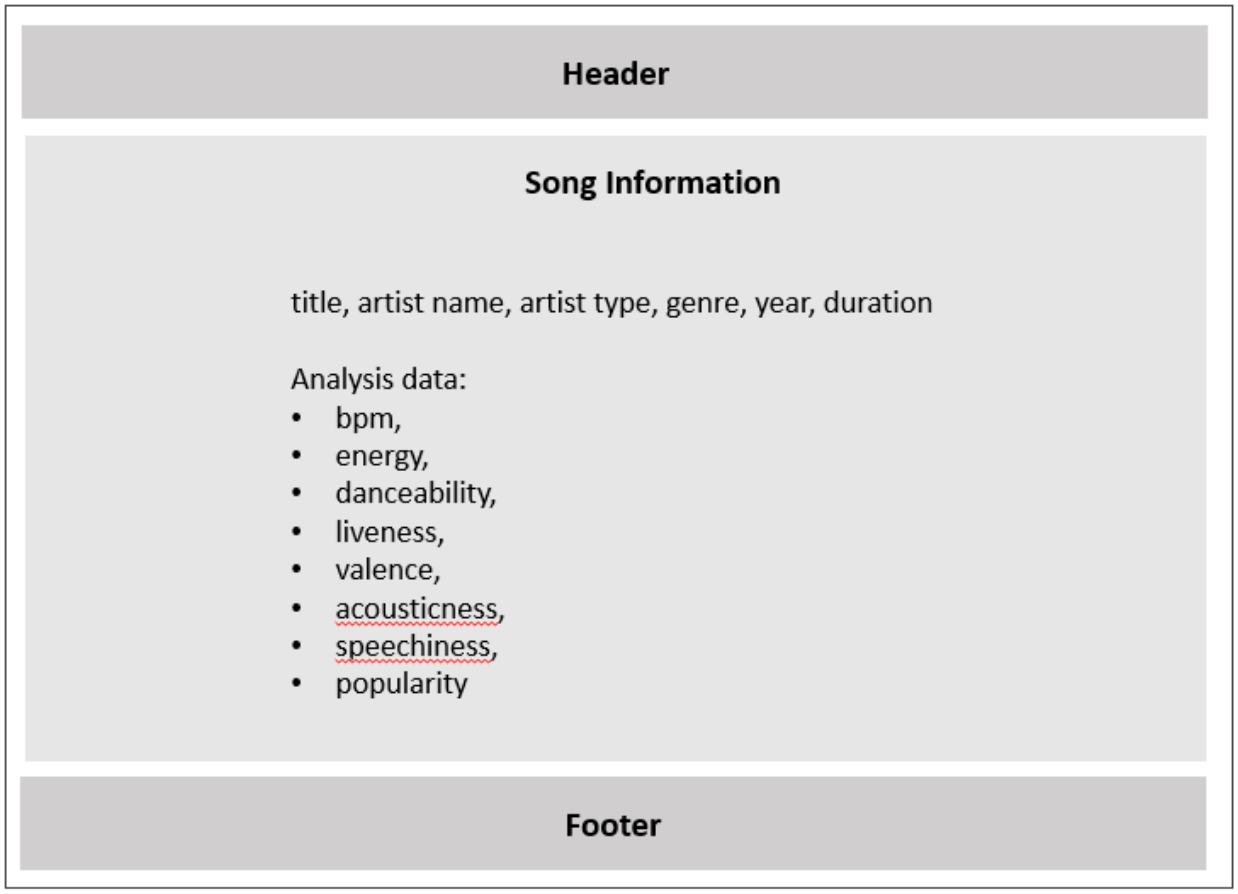
- **bpm**: The overall estimated tempo of a track in beats per minute (BPM).
- **energy**: Represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Value between 1 and 100.
- **danceability**: Describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. Value between 1 and 100.
- **valence**: Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). Value between 1 and 100.
- **liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. Value between 1 and 100.
- **acousticness**: A confidence measure of whether the track is acoustic. Value between 1 and 100.
- **speechiness**: This detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 100 the attribute value. Value between 1 and 100.
- **loudness**: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Not used in this assignment.
- **duration**: The duration of the track in seconds.
- **popularity**: The higher the number (closer to 100), the more popular the track.

Requirements

1. Your assignment will consist of five PHP pages. I have provided you with simple screen captures to show you roughly the functionality required, but needless to say I would expect your assignment to be a lot more visually styled than that shown in the screen captures.
2. If you make use of CSS recipes you found online, you must provide references (i.e., specify the URL where you found it) via comments within your CSS file. **Failure to properly credit other people's work in your CSS could result in a zero grade.** You can make use of a third-party CSS library; if you do, be sure to indicate this on the home page. Attractive styling with your own CSS will result in a higher style mark than if you use a third-party library; poor styling with your own CSS will likely result in a lower style mark than if you use a third-party library.
3. No JavaScript on this assignment. In the second assignment, you will be making a variety of improvements to this assignment using JavaScript.
4. Most of the functionality in the app can be found in the sketches shown on the next few pages. The sketches provided are used to indicate functionality not the layout. You can completely change the layout. Each of these is described in more details below.

5. The first shown below is the **Single Song Page**. Your page doesn't have to say "Song Information"; it, and the other bold labels on these pages, are there to explain what functionality must be on the page, not necessarily the actual labels that must appear on your page. I would expect you to group and present this data in an intelligent way. For instance, no user would like to see the duration listed as it is in the database (number of seconds); instead they would expect to see number of minutes and seconds. Likewise, don't show the foreign key, show the value from the associated table (e.g., don't show artist id of 110, show the name Miley Cyrus). Don't be afraid of boxes, font contrast, color contrast, icons, progress bars.

   How will your page "know" which song to display? You will pass the song_id field value via query string parameter.



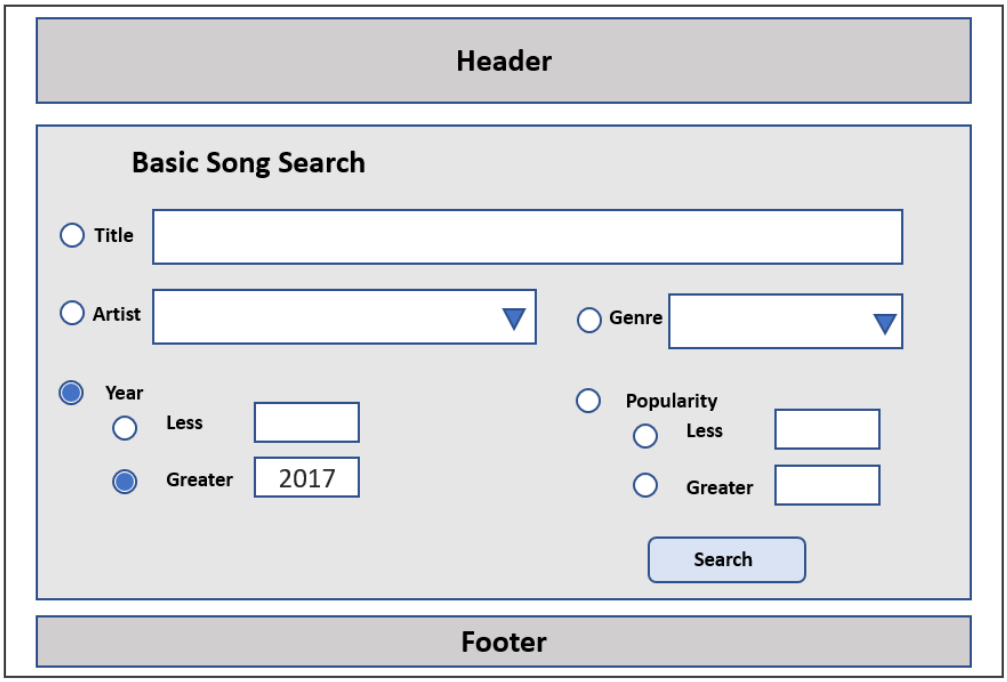
6. **Header**. The page title should be COMP 3512 Assign1. The subtitle should be your name. The Header will also need to include a navigation system for accessing the other pages in the site.
7. **Footer** The Footer should contain course name, copyright with your name, and the github repo link for the site as well as the github links for each group member.

8. The next page is the **Search Page**. The user will arrive at this page via a navigation link. This page allows the user to filter the songs by a variety of search criteria. ~~It has a basic search and an advanced search. I would recommend beginning with the basic search.~~

   Notice that the basic search terms are OR (hence the use of radio buttons): the user can search by title (the entered text has to appear somewhere in the song title), or artist, or genre, or year, or popularity. ~~For the advanced search, the user can filter by just one of the song analysis categories (energy, valence, etc).~~ In the second assignment we will be making UI improvements using JavaScript, such as disabling elements based on the status of the radio buttons.

   For the number inputs, I would recommend making use of HTML form elements for inputting numbers. The artist and genre select lists should be populated from the relevant tables sorted alphabetically.

   Clicking on ~~either of~~ the search button~~s~~ will take the user to the **Browse / Search Page**. Hint: this page doesn't actually do the filtering (i.e., it doesn't run a fancy sql statement). It merely is a data entry form that passes on the form data to the browse / search page.



NOTE 1: This sketch only meant to show functionality. Please don't make your page look exactly like this.

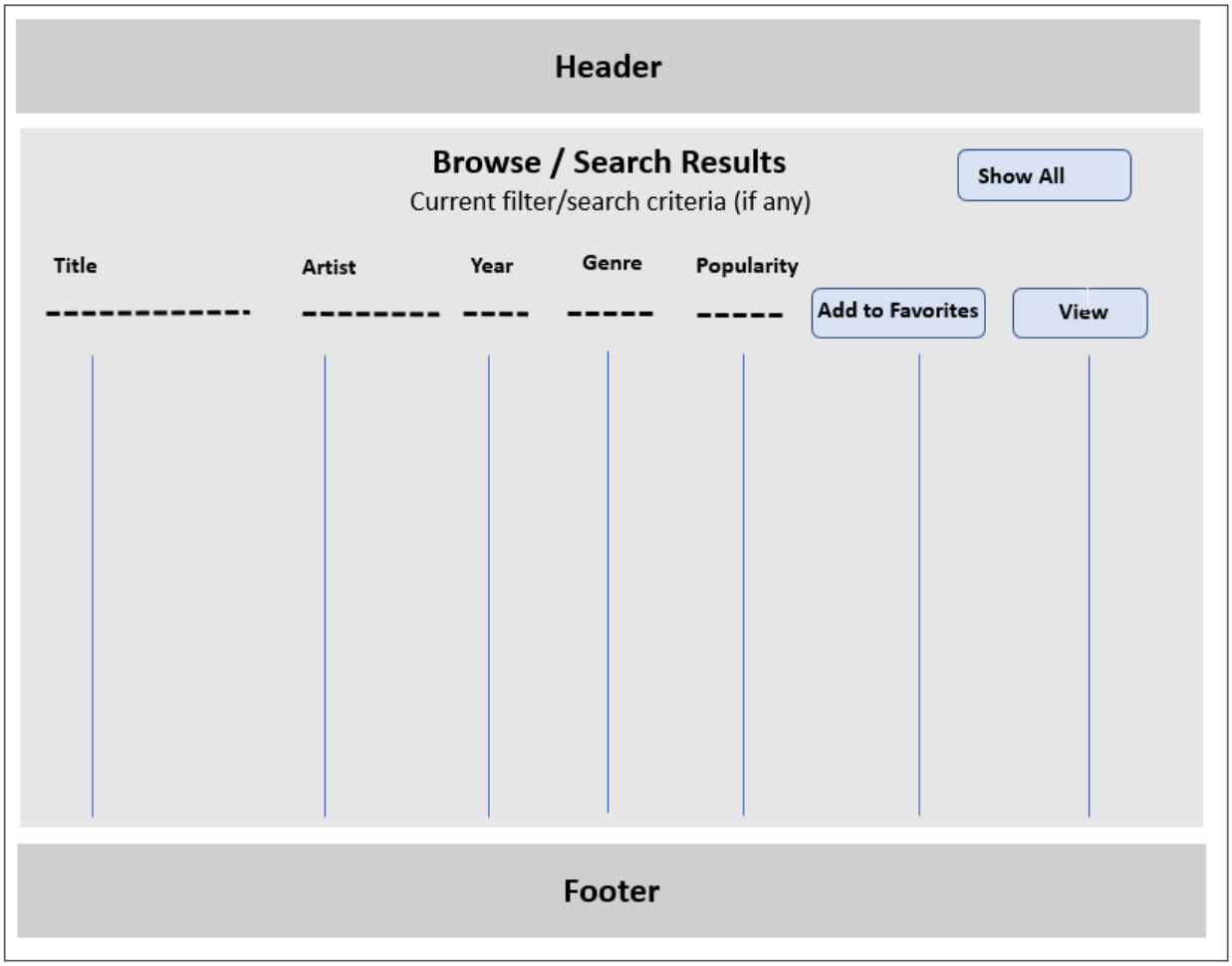
NOTE 2: the radio buttons in this sketch are meant to show that the search is mutually exclusive (i.e., you can search by title OR by artist OR by year etc). The idea for now is that even if user selects an artist, unless they have chosen the artist radio button, it will be ignored. Once you learn JavaScript for assign 2, you will be able to disable/enable items based on the radio button.

9. Next is the **Browse/Search Results Page**. The user can arrive at this page via a navigation link or via the Search button from the Search Page. It should display the song title, artist name, year, genre name, and popularity score. If the user comes to this page from search page, then the request will have query string parameter values that will be used by this page to construct a SQL query. If the user comes to this page from a navigation link or they click the show all button, then the request will have no query string parameters, and thus all the songs must be displayed.

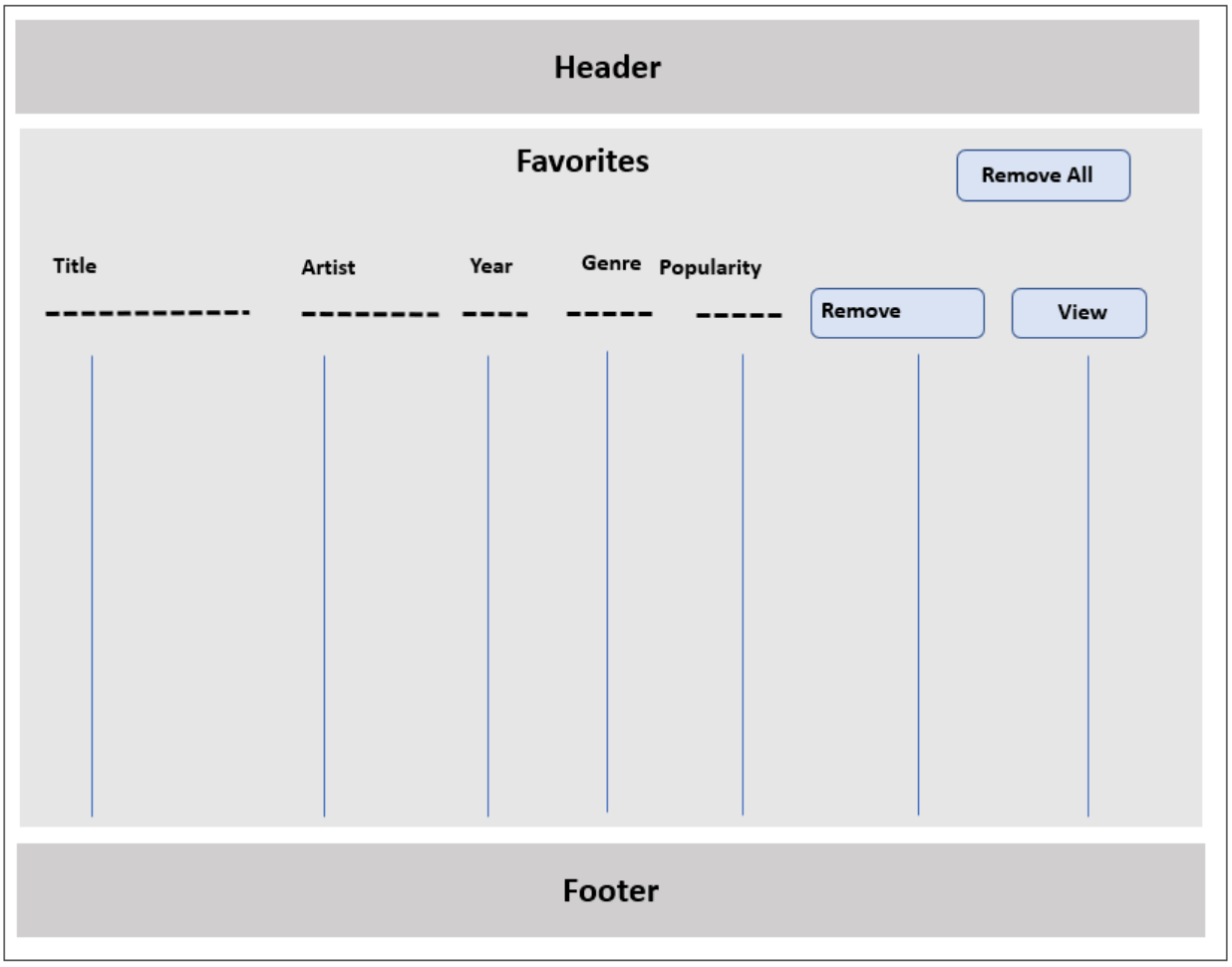
   This page will mainly involve constructing the relevant SQL WHERE clause based on the query string parameters it receives, and then displaying the matching song data.

   The View button here can be a hyperlink, button, or image: clicking on it will take the user to the Single Song Page with the song_id as a querystring. As well, make the song title in the list a link that does the same thing.

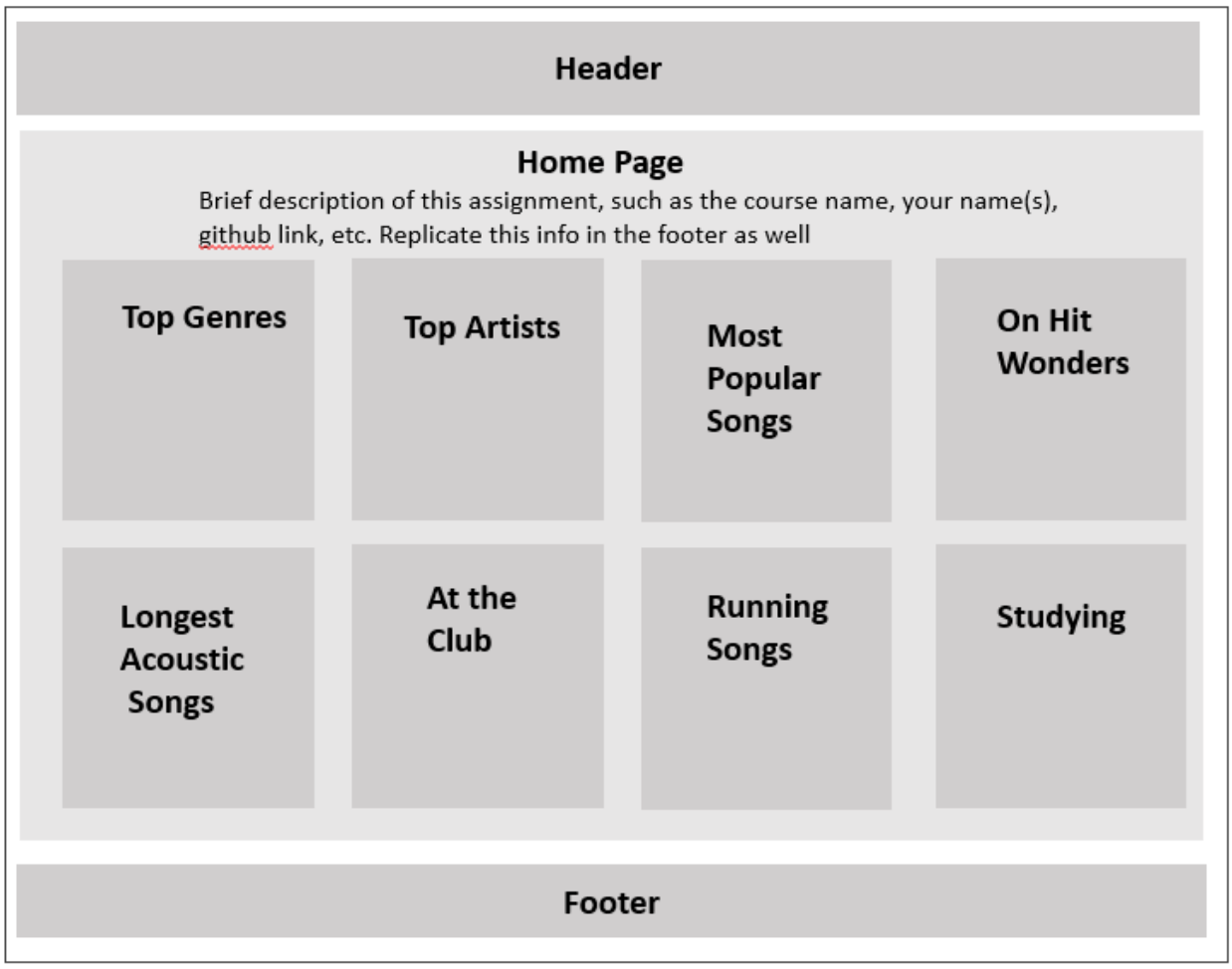
   The Add to Favorites button here can be a hyperlink, button, or image: clicking on it will add the song to a session-based favorites list and then navigate to the **View Favorites Page**.



10. Next is the **View Favorites Page**. This should have the same format as the Browse / Search Page, except it will display all the songs currently in the favorite songs list. This page should also be able to clear to favorites via the remove all button. It should also be able to remove an individual song from the list.



11. The last page to implement should be the **Home Page**. It contains a variety of specialized lists. For these lists, limit their size to 10 items (use the SQL LIMIT statement). Each of these lists will require a unique SQL query. Many will require calculated columns in the SQL (I recommend solving each of these SQL queries within something like SQLite Studio first).



- **Top Genres**: the top genres based on the number of songs. This will require an INNER JOIN, a GROUP BY, and an ORDER BY. It will also require a calculated column.
- **Top Artists**: the top artists based on the number of songs. This will require an INNER JOIN, a GROUP BY, and an ORDER BY. It will also require a calculated column.
- **Most Popular Songs**: the top songs sorted by popularity. The list should include song title and artist name. This will require an INNER JOIN, a GROUP BY, and an ORDER BY.
- **One-hit wonders**: the most popular songs by artists with only a single song in the database. It will also require a calculated column. Sort them by popularity .
- **Longest Acoustic Song**: Select only those songs whose acousticness value is above 40. Sort them by duration .
- **At the Club**: Select only those songs whose danceability value is above 80. A song's suitability for the club is based on the calculation: danceability*1.6 + energy*1.4. The list should be sorted based on the calculation in descending order.

- **Running Songs**: Select only those songs whose bpm value is between 120-125. A song's suitability for running is based on the calculation: energy*1.3 + valence*1.6. The list should be sorted based on the calculation in descending order.
- **Studying**: select only those songs whose bpm value is between 100-115 and whose speechiness is between 1-20. A song's suitability for studying is based on the calculation: (acousticness*0.8)+(100-speechiness)+(100-valence). The list should be sorted based on the calculation in descending order.

For the lists displaying songs, the song title and artist name should be displayed. The song name should be a link to the appropriate **Single Song Page.**

Hints

1. Begin by getting your data access from the database working. Complete Lab14a as soon as possible!!
2. I would recommend you complete the **Single Song Page** first. It will be passed the song_id as a query string parameter. You can simply look at the provided excel file to see the song_id values for testing purposes. Don't worry about styling at this point, just get the data display working.
3. Complete the **Browse/Search Results Page** next. Initially, get it working on all the data. Don't worry about styling at this point, just get the data display and the hyperlinks working. Also don't worry about add to favorites yet.
4. Complete the **Search Page** next. Initially, get it working with the data (i.e., the genre and artist select lists are accurate). Make sure the form submit works and requests the **Browse/Search Page**.
5. Go back to the **Browse/Search Results Page** and verify it is getting the appropriate query string values from the **Search Page**. Start implementing at least some of the SQL query customization based on the query string data.
6. Most of the visual design mark will be determined by how much effort you took to make the pages look well designed. Simply throwing up the data with basic formatting will earn you minimal marks for this component. At this point, start improving the style of your existing pages.
7. If working in pairs, I would strongly recommend that each member is involved with retrieving and working with data. This will improve your midterm mark significantly.
8. You might break up the remaining functionality amongst the group members: one does the home page, while the other implements the favorites list. You will also need to add in the remaining search functionality.
9. Before submitting, test every single bit of functionality and cross-check with this document to make sure it has all the functionality I've requested. Carefully read the specifications for each bit of functionality. Every year, students lose all sorts of marks because they didn't read this document carefully enough!!