Front End Draft URL: flip2.engr.oregonstate.edu:3306/

Marissa Castillo and Scott Hudson
Rhythm Finder

HTML Interface Reviews/Discussion

a) Feedback by the peer reviewer

Reviews 1 and 2 are from the same person. As such five reviews were received.

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.
Yes, it looks like there will be a SELECT statement for each table in the schema.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?
I don't see the ability to search or filter. There is a drop-down menu in the add music page that may be dynamically populated from Genres but it does not search or filter anything.

Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.
Yes, it looks like there is a login/sign-up function that I would assume would insert into the users table and the Add Music page would insert into all the other tables

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).
Yes, from what I see the Add Music page would insert the corresponding FK attributes including their M:M relationship.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.
Yes, the Music page that has the users saved songs has the ability to delete and that would also make a DELETE to the intersect tables for artists and genres

Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?
Yes, the Music page has the ability to UPDATE which would impact at least one entity

Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.
I may be missing this but I don't see this explicitly anywhere.

Do you have any other suggestions for the team to help with their HTML UI?
First I want to say that your homepage is great, I love the image as it fits the theme and creates an interest for the user as soon as they land on your site. As long as you pull through the styling throughout the rest of the pages (which I am sure you were planning on) then I think you have done a really good job here. Way to go!

Does the UI utilize a SELECT for every table in the schema? **In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.** Yes, there are options listed for each table in the Table views drop-down menu. However, see note at the bottom of this about a potentially missing table for AlbumsArtists in the overall ERD/schema.

Does at least one SELECT utilize a search/filter with a dynamically populated list of properties? ***Initially*** I had written that No, I don't see a way to search or filter here. I think there are many possibilities for how you could do that, though. For example, on the Music page, you could add a search box for searching your saved songs by song title, artist, album, and/or Genre pretty easily. Similarly, on the Table views, you could have searches on each one (or, some of them if you don't want to do all of them) to be able to search each table by either song title, album name, etc. to see if what they're looking for exists in the DB. ***HOWEVER***, after looking through this again, I realized that the Music page shows the information from the UsersSongs intersection table where the user id = the current user id. So technically there might be code for it, but it's not a dynamic choice that can be made in the UI. I think that adding something along the lines of what I had initially written here will make it more clear, and provide a way to choose a certain search/filter. Another idea for searching might be on the User Table. You could make the table that appears on page load only have their own information (being a search for the current user's information and no one else's), but you could have a search option on that page to search for other users (like if they wanted to see if their friend uses the site, or something). Those search results could display the table of the User information where the name is what they entered as the search criteria, but you could limit it to only display basic information about the other users and keep sensitive information hidden.

Does the UI implement an INSERT for every table in the schema? **In other words, there should be UI input fields that correspond to each table and attribute in that table.** No, I'm only seeing an insert that would work for the individual entities (Artist, Album, Song, Genre), and one for new users (Sign Up), but nothing for inserting new relationships. However, There are some nuances here that I think need to be fleshed out a bit. For example, what would happen if you wanted to add a song in a genre that isn't available in the drop-down menu? How do you add a new Genre to the DB? Similarly, if an Artist already exists and you want to just add a new song they've released, how do you add the song without adding redundant Artist information? Another example might be if a new "greatest hits" album releases - how do you add the album and tell it which songs (that might already exist in your DB) belong to that new album? I think these kinds of situations might be made more clear if you consider how to add an instance of just one of the entities to the DB without needing to rely on the others, and then try to link them up through relationships.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? **In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).** This is hard to know from just the html UI on the client side without seeing any code, but since the add form has all four entities listed in it, there's likely room for the code on the back end to use that information to add the information to the M:M relationship table as well as the individual entities.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? **In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.** Yes, there is a delete option when the user is viewing their own list of saved songs - this makes sense for the context, considering that it looks like the user here is not the one managing the DB. It seems like that delete button would only remove the information from the M:M relationship table between Users and Songs, and would leave the individual entity User and Song intact.

Is there at least one UPDATE for any one entity? **In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?** Yes, there is an update button, but I'm not sure it makes sense in the context. If the user wants to update something that's showing up in the table on the Music page, will it go in and update that instance of the entity itself? For example, the delete functionality there would delete that user/song combo from the UsersSongs intersection table. But since the visible fields are Title, Artist, Album, and Genre, if they update the Album name - where does it update? It wouldn't be updating the information in the same M:M relationship table as the delete button effects, so would it update that name on the Album entity that is associated with the Song entity that is associated with that intersection table? It might make sense for the update button to be

in the Songs table in Table views so the user can add that song to their saved songs (i.e., adding it to the UsersSongs intersection table with their user id). You may also want to have update fields somewhere else where you can edit the data related to a user, song, artist, album, and/or genre, because even my most recent suggestion is really more of an insert than an update. A simple solution might be for a user to update their information in their User entity. For example, if they want to update their email address, name, or phone number. Those are all reasonable things that someone might want or need to update at some point. Similarly, an artist might change their name, so that could be a place where you could have update functionality.

Is at least one relationship NULLable? **In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.** There's no indication on the pages that any of the information can be null, though according to the relationships in the PDF there are optional relationships (such as Songs and Genres). It's unclear whether the deletion of a song, for example, would null the song id in the relationship table with the genre(s) or if it would delete those instances in the relationship table.

Do you have any other suggestions for the team to help with their HTML UI? See below

Overall Feedback / other things I noticed:

Love the picture on the home page! Made me laugh :)
On the table view for Songs, the title says "here you can view artists". Probably just a copy/paste in the code that forgot to be updated :)
On the table view for AlbumGenre, the title says Albums and Artists, and the table shows the Album and Artist, but not the Genre.
Aside from the feedback above on the website/UI:
In your Draft PDF, I noticed that you have Artists listed as having a M:M relationship with Albums, but then Albums has the artistId as an FK attribute (pg 5). That should probably be updated, and a M:M intersection table added for ArtistsAlbums to make sure it's listed properly. If it's a M:M relationship, then there are multiple FK ids that would be listed there, so it wont' work to have the artistId in the Albums table. It would also help to list the relationship/intersection tables in your ERD as a way to double check that the information and relationships you're listing in the outline match up with the way it's being implemented. The ERD and Schema don't have the Artist/Albums M:M relationship that's mentioned in the outline (pg 5, under the details about the Artist entity), and the Albums entity in the ERD and Schema doesn't have the artistId that's mentioned in the outline, so that could be cleared up a bit.
Overall, I like this a lot! I think there are some nuances that can be worked out, but I think you've got a great start here!

- Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.
  > Yes, each table can be listed out.
- Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?
  > Yes, there is a search function.
- Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.
  > Mostly, genres cannot be created.
- Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).
  > Yes, inserting a song will insert an Album.
- Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.
  > Yes, music can be deleted.
- Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?
  > Yes, music can be updated.
- Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.
  > Not sure, I didn't see any optional relationships.
- Do you have any other suggestions for the team to help with their HTML UI?
  > More responsive styling, use styling to increase distinction between different text elements.

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.
Yes. The utilize a SELECT for every table in the schema.
Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?
Yes. I can see the table views are used to filter the music based on pre-set condition.
Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.

Yes. I think when insert a new song, it will insert the necessary artist, album etc.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

Yes. Each INSERT also add the corresponding FK attributes, including at least one M:M relationship.

Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes. There is a delete function. But can't tell will it remove things from M:M relationship.

Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Yes. There is a update.

Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

Can't tell whether there is one relationship Nullable.

Do you have any other suggestions for the team to help with their HTML UI?

The website is clean and easy to follow. Some pictures of music will be nicer.

- **Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.**
  - Yes all the data is there.
- **Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?**
  - I see no search feature but that could be easily implemented
- **Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.**
  - Yes you can add to each entity in the schema.
- **Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).**

- They properly add a M:M relationship and you can insert rows into it.

- **Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.**
  - There is a delete function.
- **Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?**
  - There is an update function.
- **Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.**
  - No indication that any relationships are optional
- **Do you have any other suggestions for the team to help with their HTML UI?**
  - I think their UI is the most clean part of this project..

b) Actions based on the feedback

- **Feedback:** We got mixed reviews on having a search function, some people mentioned that we had it others mentioned that it was missing. There were a few suggestions about how to implement it, the most notable of which was this comment, “on the Music page, you could add a search box for searching your saved songs by song title, artist, album, and/or Genre pretty easily. Similarly, on the Table views, you could have searches on each one (or, some of them if you don't want to do all of them) to be able to search each table by either song title, album name, etc.”
  - **Response:** A search function was added to the music page and home page where a user can search by the song title.

- **Feedback:** Also received mixed reviews on insert statements, some people were unhappy that they couldn’t insert relationships into M:M table, or other tables.
  - **Response:** These inserts occur with FK ids and to implement them in a way that makes sense to users might be beyond the scope of this project. It’s also worth noting that when a user adds a song the song is added to every table where

needed-- in other words, the backend will automatically take care of M:M relationships based on user interaction with the site that may not obviously look like adding a M:M relationship in a database. For example, a user may “Favorite” a song to add to their account, which looks like adding a row in the M:M table in the backend, but does not appear as such by the user.

- **Feedback:** A simple solution might be for a user to update their information in their User entity. For example, if they want to update their email address, name, or phone number.
    - **Response:** This is a great suggestion that we implemented.

- **Feedback:** All reviews noted that we are missing Nullable relationships.
    - **Response:** This is correct and a way to implement this could be to allow users to include incomplete information on song entries. For example a user could add a song with its name, artist, and genre without having the album, thus making the album relationships null/optional.

- **Feedback:** “In your Draft PDF, I noticed that you have Artists listed as having a M:M relationship with Albums, but then Albums has the artistId as an FK attribute (pg 5). That should probably be updated, and a M:M intersection table added for ArtistsAlbums to make sure it's listed properly.”
    - **Response:** This is a previous discussion we’ve had and to limit the scope of the project we decided not to implement this as a M:M table. The idea being that you can still see an artist albums and songs off of the artists entity without causing a full implementation. Ideally, this would be a M:M table.

- **Feedback**: “On the table view for Songs, the title says "here you can view artists". ”
    - Table names and their headers have been fixed.

c) Upgrades to the Draft version

- Added a search function to the Music and home page where users can search for a song by title.
- Added a page where a users can view and edit their information
- Adjusted the Add Song page to allow users to add as much or little information as they want. A song must have a name and artist name to be added but album and genre are optional.
- The Songs page allows users to ‘Favorite’ a song
- Typos have been fixed

Database Reviews/Discussion

a) Feedback by the peer reviewers

Three responses were received:

1.

- **Does the overview describe what problem is to be solved by a website with DB back end?**
    - Yes, it is a music library that intends to grow from a small company to one which could rival Spotify!
- **Does the overview list specific facts?**
    - Yes it lists it could potentially handle over 100 million users after significant growth.
- **Are at least four entities described and does each one represent a single idea to be stored as a list?**
    - Yes, each entity is a singular idea and is organized well.
- **Does the outline of entity details describe the purpose of each, list attribute data types and constraints and describe relationships between entities?**
    - Yes, it does. You could maybe also list the 1:M relationship between song and album under the album entity as well, same for the song M:M genre relationship.
- **Does the outline clearly indicate which entities (tables) will be implemented and which team member is primarily assigned to the associated page(s)?**
    - Yes, it indicates that all entities will be implemented by both group members, with Marissa working on the main entity tables, and Scott working on the M:M tables.
- **Are 1:M relationships correctly formulated? Is there at least one M:M relationship?**
    - There is at least one M:M relationship. The song/album M:1 relationship looks good. Maybe I'm reading this wrong, but is artist/album supposed to be a 1:M relationship? Under the artist entity, it is listed as 1:M, but on the ERD it looks M:M. An artist could have many albums and an album could have many artists.
- **Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?**
    - Yes there is consistency in naming overall.

Awesome idea!

2.

- **Does the overview describe what problem is to be solved by a website with DB back end?**
    - Yes, the outline gives great detail of the problem and how a the db/website will be used as a solution. I like this idea, it's easy for everyone to understand.
- **Does the overview list specific facts?**
    - Yes the overview quantifies number of users. Approximate number of songs/artists/etc may also be helpful.
- **Are at least four entities described and does each one represent a single idea to be stored as a list?**

    - Yes, this is done in a clear way. Each entity is unique and there are at least four entities.
- **Does the outline of entity details describe the purpose of each, list attribute data types and constraints and describe relationships between entities?**
    - This is done, but I agree with the review above. It would be much more clear to list out the relationships on the corresponding tables. That way it's hard to miss.
- **Does the outline clearly indicate which entities (tables) will be implemented and which team member is primarily assigned to the associated page(s)?**
    - Yes, this is done after the first paragraph.
- **Are 1:M relationships correctly formulated? Is there at least one M:M relationship?**
    - I agree with the reviewer above, album/artist relationship should be changed in either the table or the ERD diagram (similar to song/album relationship). Diagrams look really good and very easy to read.
- **Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?**
    - ArtistId is in the Albums entity but not in any of the diagrams. Everything else seems consistent.

3. **Great job overall!**

- **Does the overview describe what problem is to be solved by a website with DB back end?**
    - Yes, the problem and solution is laid out clearly. Given this is a song related app, it's probably implied, but would users be able to actually play songs? It's not mentioned, but may be obvious enough.
- **Does the overview list specific facts?**
    - Yes, it does list out facts regarding a competitor's user base and how the app is expected to grow incrementally towards an expected number of users. It might be nice to list some kind of expectation of an initial upper limit on songs, artists, albums, etc as even if it's very high, it's something to plan around.
- **Are at least four entities described and does each one represent a single idea to be stored a s a list?**
    - Yes, each entity represents a unique idea and is expressed clearly!
- **Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?**
    - Yes, the entities and attributes are clearly listed. The relationships seem clear to me with one exception. In the Songs entity, there is an attribute called songAlbum, and it seems this possibly relates directly to the albumID in the Albums table. In the relationship description, it's mentioned that songs relate to albums, but it's not explicitly stated that these two similar attributes relate directly so it's not clear (also because the attribute names don't exactly match). Also, the schema with arrows does not show a relation between songs and albums.
- **Does the outline clearly indicate which entities (tables) will be implemented and which team member is primarily assigned to the associated page(s)?**

    - Yes, this is listed in the project outline clearly.
- **Are 1:M relationships correctly formulated? Is there at least one M:M relationship?**
    - Outside of the above on songs and albums, the above reviews mentioning the ER diagram relationship between albums and artists are correct. An album is showing on the ER diagram as having at least one artist, but the relationship description in the outline mentions that an album should only have one artist.
- **Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?**
    - Everything looks great generally! It's probably just a typo, but in the entity list just above the ER, the very last attribute is listed as "UserID" but I think should be "userId" to match the rest.

b) Actions based on the feedback

- **Feedback:** “Maybe I'm reading this wrong, but is artist/album supposed to be a 1:M relationship? Under the artist entity, it is listed as 1:M, but on the ERD it looks M:M. An artist could have many albums and an album could have many artists.”
  “I agree with the reviewer above, album/artist relationship should be changed in either the table or the ERD diagram (similar to song/album relationship).
  “An album is showing on the ER diagram as having at least one artist, but the relationship description in the outline mentions that an album should only have one artist.”
    - **Response:** Removed the album FK from the artist entity so it isn’t duplicated in the database and will better support a M:M relationship. Outline has been updated to clarify the M:M relationship rather than a 1:M.

- **Feedback**: “Given this is a song related app, it's probably implied, but would users be able to actually play songs? It's not mentioned, but may be obvious enough.”
  “It might be nice to list some kind of expectation of an initial upper limit on songs, artists, albums, etc as even if it's very high, it's something to plan around.”
    - **Response:** Added additional details to clarify the project does not play songs and add upper limits on songs, artists, albums, and users’ saved songs.

- **Feedback**: “The schema with arrows does not show a relation between songs and albums.”
    - **Response**: We added an arrow showing that albumId is a FK in the Songs entity in the Schema

- **Feedback**: “You could maybe also list the 1:M relationship between song and album under the album entity as well, same for the song M:M genre relationship.

    - **Response**: Our team opted to reject this feedback as we do not want to have duplicate data in our entities.
- **Feedback**: “Everything looks great generally! It's probably just a typo, but in the entity list just above the ER, the very last attribute is listed as "UserID" but I think should be "userId" to match the rest.”
    - **Response**: Adjusted the name to remove the typo.

c) Upgrades to the Draft version

Project Outline and Database Outline - Updated Version:

Project Title: Rhythm Finder

Rhythm Finder is working to create a music service which would allow its Users to find new music, keep track of their music library, and search for music based on features such as Song Title, Artist, Album, and Genre. The company expects that use will be small at first but growth will occur with user numbers potentially approaching 100 million, which is less than Spotify but still a very significant number of users. That said, this will be an incremental development and for this initial start up the database will support up to 15 million songs, artists, albums, and users’ saved songs up to 15,000.

All entities will be implemented with Marissa Castillo working on the main entity tables and Scott Hudson working on the M:M tables.

Entities

- **Users:**
  Users using the website with an account to keep their saved songs
    - userId: INT() or BIGINT(), auto increment, unique, Not NULL, PK
    - userName: VARCHAR(20), user input, Not NULL
    - userEmail: VARCHAR(30), user input, Not NULL
    - userPassword: VARCHAR(20), user input, Not NULL
    - Relationships:
        - M:M Users can have multiple Songs, and Songs can have multiple Users. Users can have no Songs, and Songs can have no Users. In this implementation a relationship table with the userId as a foreign key links the songId as a foreign key. See UsersSongs table.

- **Songs:**
  Unique songs of music
    - songId: INT() or BIGINT(), auto increment, unique, Not NULL, PK
    - songName: VARCHAR(30), user input, Not NULL

    - songAlbum: INT() or BIGINT(), Can be NULL, FK
    - Relationships:
        - M:1 A Song can have one Album, and an Album can have many Songs. A Song can exist without an Album. The albumId exists as a foreign key in the song entity.
        - M:M Songs can have multiple Artists, and Artists can have multiple Songs. A Song can't exist without an Artist, an Artist can exist without a Song. In this implementation a relationship table with the artistId as a foreign key links the songId as a foreign key. See SongsArtists table.
        - M:M Songs can have multiple Genres, and Genres can have multiple songs. A song can exist without a Genre, and a Genre can exist without a song. See SongsGenres table.

- **Artists**:
  Unique Artists that create music
    - artistId: INT() or BIGINT(), auto increment, unique, Not NULL, PK
    - artistName: VARCHAR(30), Not NULL
    - Relationships
        - M:M Artists can have multiple Genres, and Genres can have multiple Artists. Artists can exist without Genres and Genres can exist without Artists. See ArtistsGenres table.

- **Albums**:
  Musical albums created by an artist
    - albumId: INT() or BIGINT(), auto increment, unique, Not NULL, PK
    - albumName: VARCHAR(30), Not NULL
    - artistId: INT() or BIGINT(), Not NULL, FK

- **Genres:**
  Categories which can be used to classify and describe themes in music.
    - genreId: INT() or BIGINT(), auto increment, unique, Not NULL, PK
    - genreName: VARCHAR(30), Not NULL

- **SongsArtists**
  Shows what Artists made which Songs
    - songId: FK
    - artistId: FK

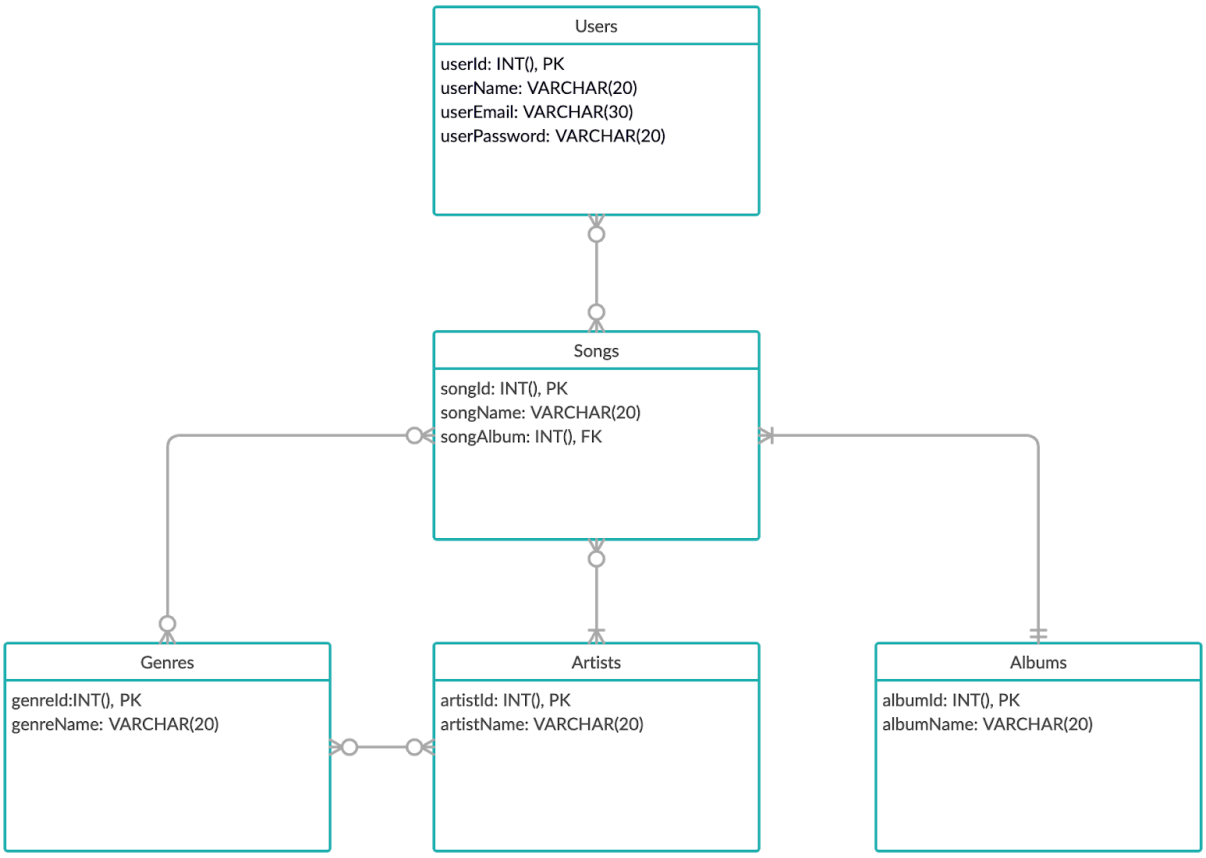
- **SongsGenres**

Shows what Genre(s) a Song is categorized in
    - songId: FK
    - genreId: FK

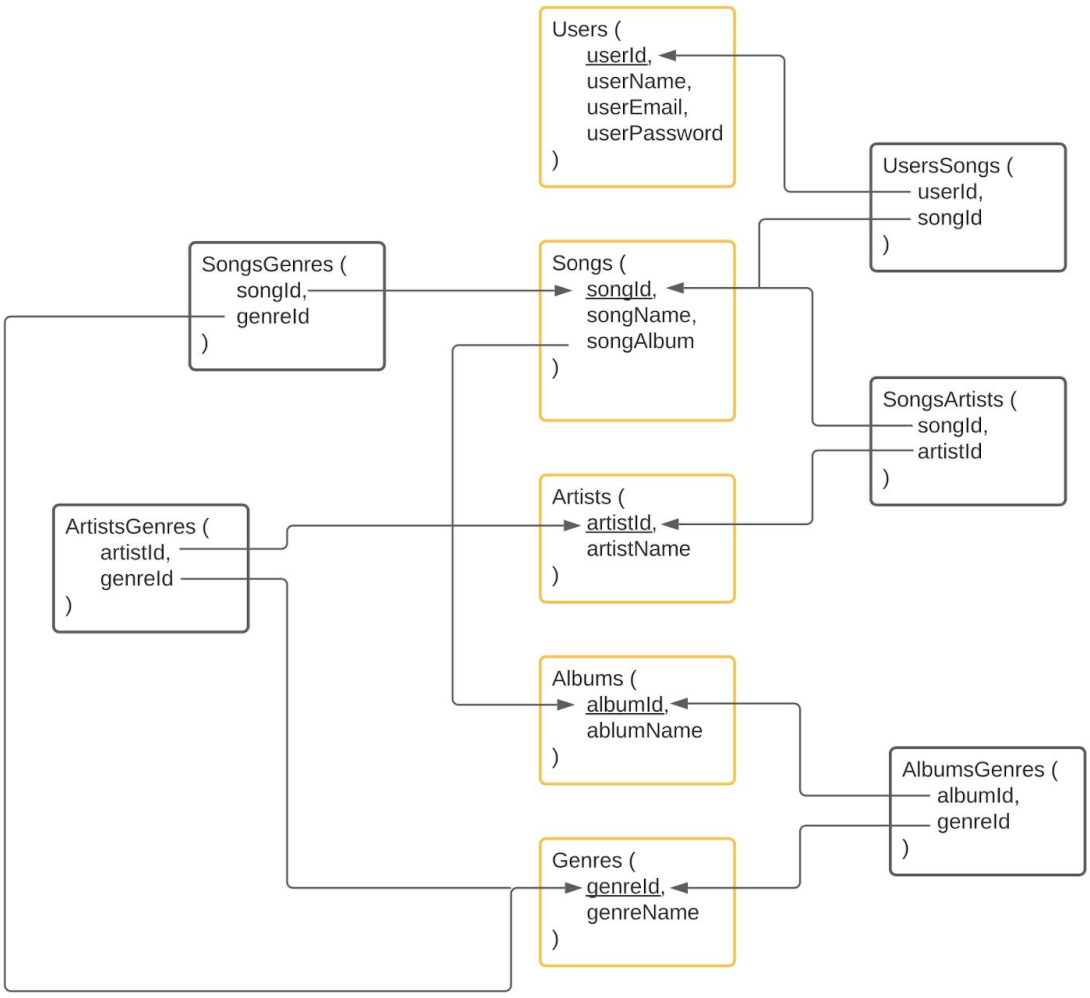
- **ArtistsGenres**
  Shows what Genre(s) an Artist is categorized in
    - artistId: FK
    - genreId: FK

- **UsersSongs:**
  Shows what Songs Users have saved to their account
    - songId: FK
    - UserId: FK

d) Entity-Relationship Diagram:



e) Schema:


Users (
userId,
userName,
userEmail,
userPassword
)
UsersSongs (
userId,
songId
)
SongsGenres (
songId,
genreId
)
Songs (
songId,
songName,
songAlbum
)
SongsArtists (
songId,
artistId
)
Artists (
artistId,
artistName
)
ArtistsGenres (
artistId,
genreId
)
Albums (
albumId,
ablumName
)
AlbumsGenres (
albumId,
genreId
)
Genres (
genreId,
genreName
)

CS 340 TEAM EVALUATION FORM

OCTOBER 19, 2020

RATE YOUR TEAMS PERFORMANCE USING THE SCALE BELOW.

1 = Strongly Disagree 2 = Disagree 3 = Agree 4 = Strongly Agree

GROUP NUMBER	Project Group 19	
NAME OF GROUP TEAM MEMBERS:	Marissa Castillo and Scott Hudson	
SCALE AND COMMENTS	RATING	ADDITIONAL COMMENTS
HOW PREPARED WAS YOUR TEAM? Research, reading, and assignment complete	4	
HOW RESPONSIVE & COMMUNICATIVE WERE YOU BOTH AS A TEAM? Responded to requests and assignment modifications needed. Initiated and responded appropriately via email, Slack etc.	3	MC: I noticed that I would update a document and forget to message Scott that I had done so until a couple days later. I will improve on this communication. Otherwise, everything has been great. SH: I do the same.
DID BOTH GROUP MEMBERS PARTICIPATE EQUALLY Contributed best academic ability	4	
DID YOU BOTH FOLLOW THE INITIAL TEAM CONTRACT?	4	

Were both team members both positive and productive?		

Are there any suggestions for improvement for your team and what are your goals moving forward?
(Better communication, follow the contract better, modify the initial team contract, more contribution, etc?)?

Marissa's comments: I noticed that I would update a document and forget to message Scott that I had done so until a couple days later. I will improve on this communication. Otherwise, everything has been great.