**Final Project Report**

Groups for Project 14

Ilana Wasserman, Walker Mayercheck

**Abstract**

This is the report for our final database project. The database stores files in a way that links files to their components in order to create a “family history” for the file.

**Table of Contents**

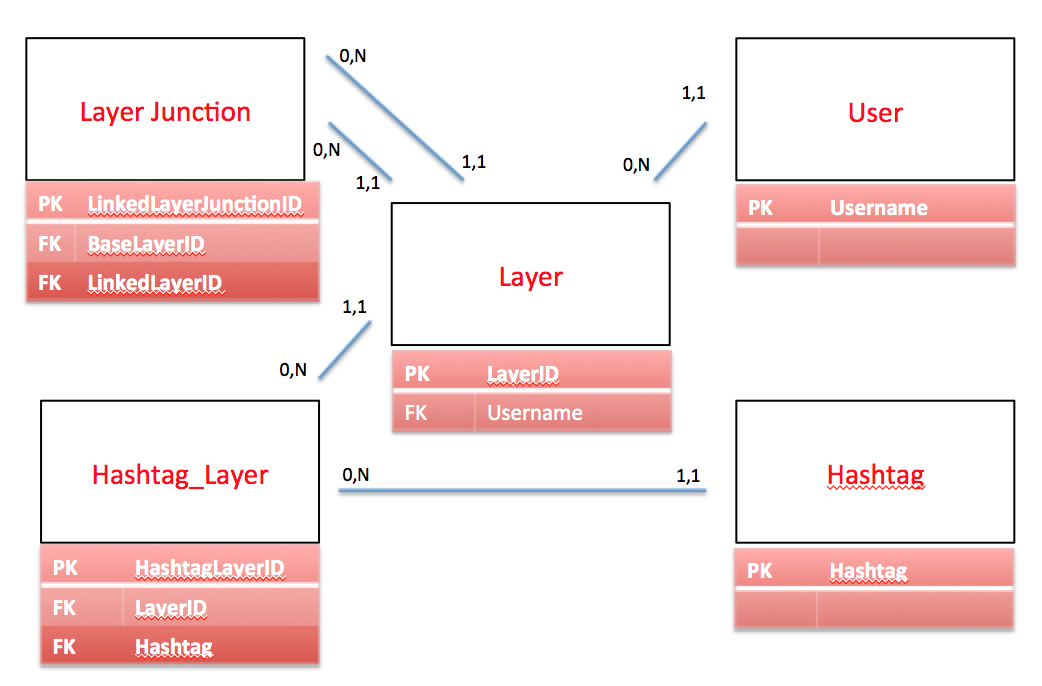
|  |  |
| --- | --- |
| Table of Contents | 1 |
| README | 2 |
| ERD | 3-5 |
| * Simple ERD | 3 |
| * Detailed ERD | 4 |
| * EER Diagram | 5 |
| User Flow | 6-8 |
| Lessons Learned | 9 |
| Future Work | 11-12 |
| Additional Information | 13 |

**README**

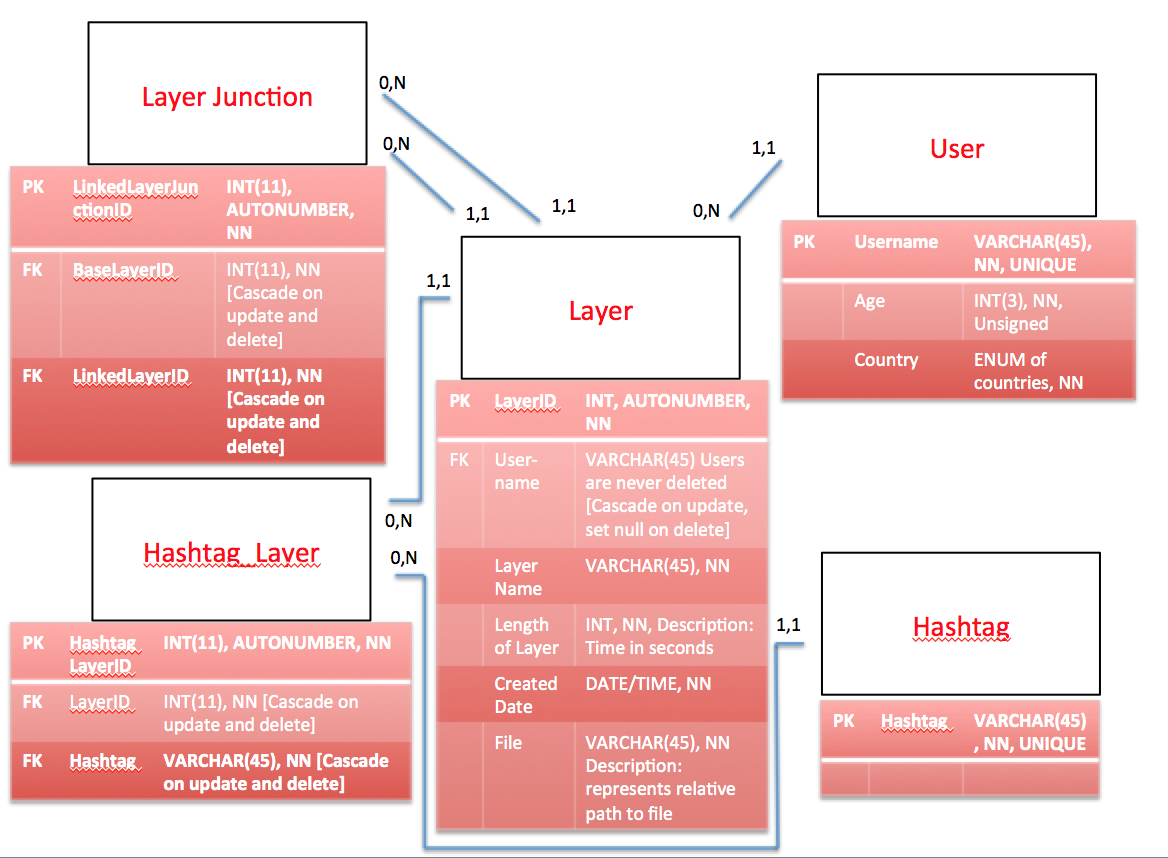
* Initial Setup
  1. Run the MySQL server.
  2. Download the provided zip file.
  3. Open the zip file, and then open the folder.
     + One of the first files is called “server.js”.
       - Open this file, and change “password” on line 13 to your password for the root user in MySQL.
  4. Open and run the provided MySQL code in the MySQL Workbench to create the schema.
  5. Open a new terminal window, go to where the folder is stored (cd to where it is stored and cd sp-node-mysql) and then type in “node server” in the terminal.
     + You should see “Server running on port 3000” and “Connection established”.
  6. Go to Google Chrome browser and type in “localhost:3000”.
* Functionality
  1. When the page loads, it will show all of the layers in the database with basic information (**READ**).
  2. If your user does not already exist in the database, click “Create Account”, and type in a username, age, and country, and then save the information.
  3. On the main page, enter information into any field to filter the layers.
  4. Enter information into all fields (with your username) and press “Upload” to create a new layer (**CREATE**).
  5. Click “Edit” to edit a layer, and that layer’s information will populate in the edit fields (**UPDATE**).
     + Only Layer Name should be editable (due to collaborative nature of the site).
     + Once edit is complete, click “Update” to save update in the database.
  6. Click on the layer name link in order to view more information about the specific layer, such as created date, hashtags, and links.
  7. On the layer detail page, click “Add” under Hashtags to add a new hashtag to describe the layer.
     + If hashtags exist for that layer, click “Remove” to delete the reference between the hashtag and layer (**DELETE**).
  8. On the layer detail page, click “Add” under Sublayers to create a new linked layer in the database.
     + A new sublayer should show in the section as a link that can be clicked on to view the details of the linked layer.
     + User should not be able to delete any links.
  9. Click on the “Layers” link at the top of the page at any time to go back to the main page.

**ERD Diagrams**

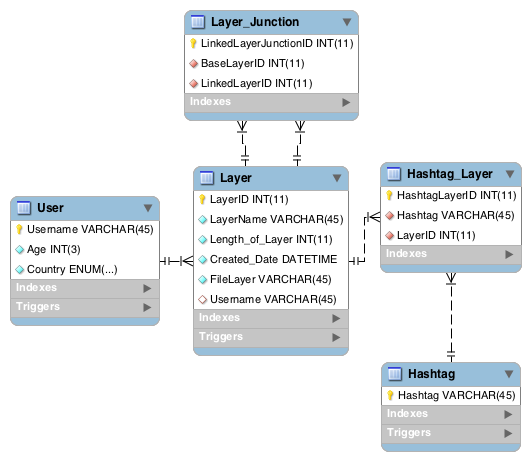
**Simple ERD**

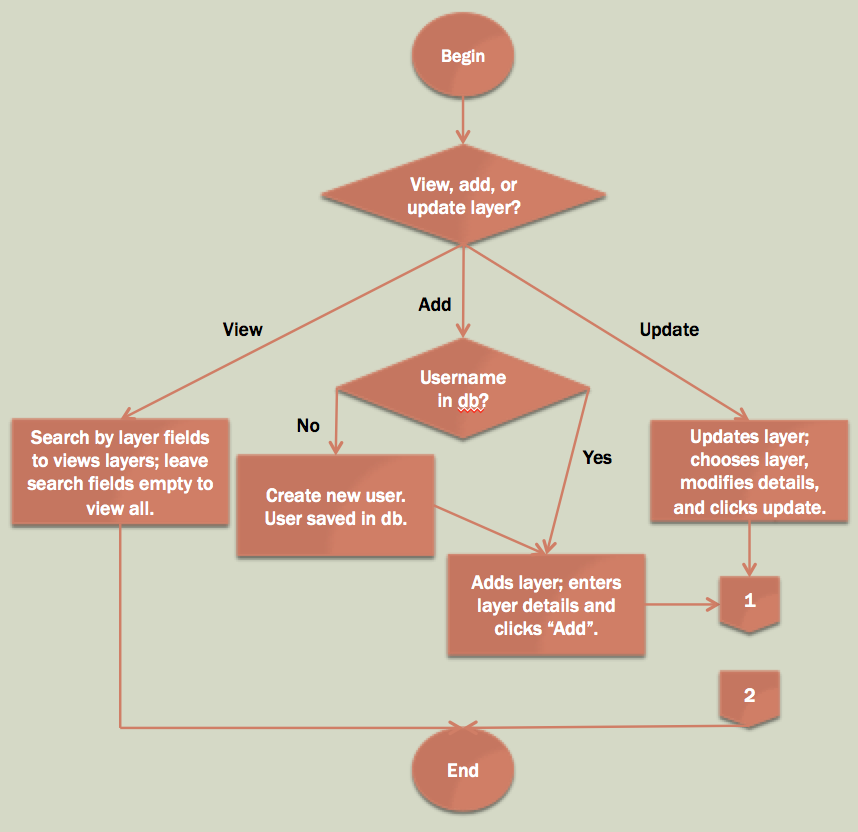
****

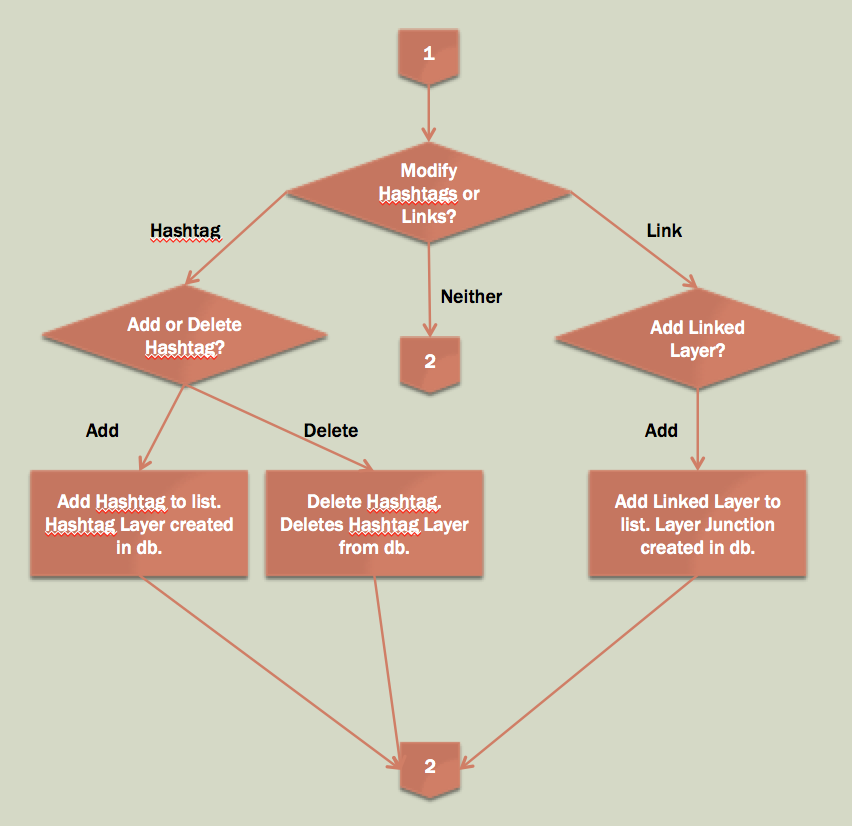
**Detailed ERD**



**EER Diagram**

**User Flow**





**Upload a Layer (Create)**

* The user can create a new “Layer” record.
  + The result in the database:
    - A new record is inserted in the “Layer” table in the database.
      * Autonumber is the primary key
      * User inputs the layer name, username, length of layer in seconds, and puts in a relative path to the file as a string
      * User creates hashtags
      * Today’s date is automatically stored as Created Date
      * User can attach the layer that they are uploading to multiple layers already in the database by entering the existing layer ids
        + In the database:

BaseLayerID = current layer

LinkedLayerID = existing layer that current layer is being attached to (becoming sub-layer to current layer)

A new record is inserted in the “Layer\_Junction” table in the database using BaseLayerID and LinkedLayerID.

The user cannot modify this relationship after it is created due to the fact that is it now part of the family tree for that layer.

**Edit a Layer (Update)**

* The user can change the “Layer Name”
  + The result in the database:
    - The “Layer” record in the database is updated.
* Delete/Add hashtag
  + Add
    - The result in the database:
      * A new record is inserted in the “Hashtag” table in the database.
        + Occurs if the hashtag does not already exist in the table.
      * A new record is inserted in the “Hashtag\_Layer” table in the database.
        + Always occurs.
  + Delete
    - The result in the database:
      * The record is deleted in the “Hashtag\_Layer” table in the database.
      * No records are deleted in the “Hashtag” table, as other records may be referencing the hashtag as well.

**Delete a Layer (Delete)**

* The user cannot delete a layer because other layers may refer to the layer.
  + The integrity of the family tree of layers cannot be compromised.
* Since the user cannot delete layers, the user can instead delete hashtags on a layer.
  + The result in the database:
    - The “Hashtag/Layer” record in the database is deleted.

**View a Layer (Read)**

* The user can view all of the layers in the database, or specific layers through filters.
  + The result in the database:
    - No data is modified; a SELECT statement is used on the tables in the database.

**Create a User**

* The user creates an account if the user does not already have an account.
  + The result in the database:
    - A new record is inserted in the “User” table in the database with all of the provided information by the user:
      * Username, Age, Country

**Views**

* User can see searched layers (blank fields to search for all layers)
* User can see hashtags and links to the linked layers for a particular layer

**Lessons Learned**

**Upload/Download**

The “Download” button does not work. We need to implement a way to actually store an mp3 file using a “Browse” button to search files on the computer and upload the desired file. During all of our research, we found that the recommended strategy was to store a relative path to the file in the MySQL database and then match that path to the file stored on a mainframe computer. As such, the data should be accurate in our database, but the frontend functionality needs to be implemented. We did not anticipate the complexities in storing a media file as opposed to simple text, but the download capability does not change the interactions with our database nor the CRUD functionality of our project. It does, however, affect the usage of our website as a collaborative music workspace for users.

**Future Work**

**Planned Uses**

A collaborative workspace for music producers of any experience level and status to share music (beats, vocals, etc.) and find the individual components of the tracks in order to simplify the remix and create process for new tracks. The music would need to be edited on a separate application, but would be re-uploaded in its new form to our website and be linked to the tracks that contributed to its creation.

Origins and components of uploaded music can now be more easily tracked, and as such, music enthusiasts can more easily interact with and remix their favorite music.

**Areas for Added Functionality:**

**User Table**

The focus of our project is how users create and interact with Layers and the functionality around the Layers (i.e. Hashtags). As such, the User table is not as robust as it would need to be in practice if we were designing a database to be used on a website with full functionality.

The User table would in practice need to have more complexity, such as an Active/Inactive field, to identify which users are currently interacting with the website and the database. In addition, the User table would need to have a Created Date field, to track when each User is created and therefore gain an understanding of how marketing efforts for the website are panning out and find trends based on seasons or specific events that trigger more users to join.

The User table would also have more functionality, and Users would be able to update their provided information, and delete a User account [create an inactive user rather than actually deleting the user from the database].

**Search Features**

Users would be able to search for specific hashtags.

**Statistics**

Users would be able to see which layers are the most popular (forked the most).

**Hierarchy**

The linked layers would be formatted as a “tree” structure to better understand the relationships between layers uploaded, forked, and re-uploaded.

**Browser Compatibility**

Make the website compatible across all major browsers, both computer and mobile.

**Upload/Download Files**

The user would be able to upload and download an mp3 file, rather than simply storing the relative path of the file.

**Additional Information**

Please note that we needed to learn all aspects of MEAN stack for this project; neither partner came into this project with any MEAN stack programming experience.