

Clemson University

MeTube Project Technical Report

Group U4

Savannah Lawson and Rachael Austin

CPSC 4620 - Database Management Systems

Dr. James Z. Wang

20 April 2017

Table of Contents

System Design 3

ER Diagram 4

Database Schema 5

Function Design 6

Implementation Details8

Test Cases 20

Testing Results
21

User Instruction 23

System Design

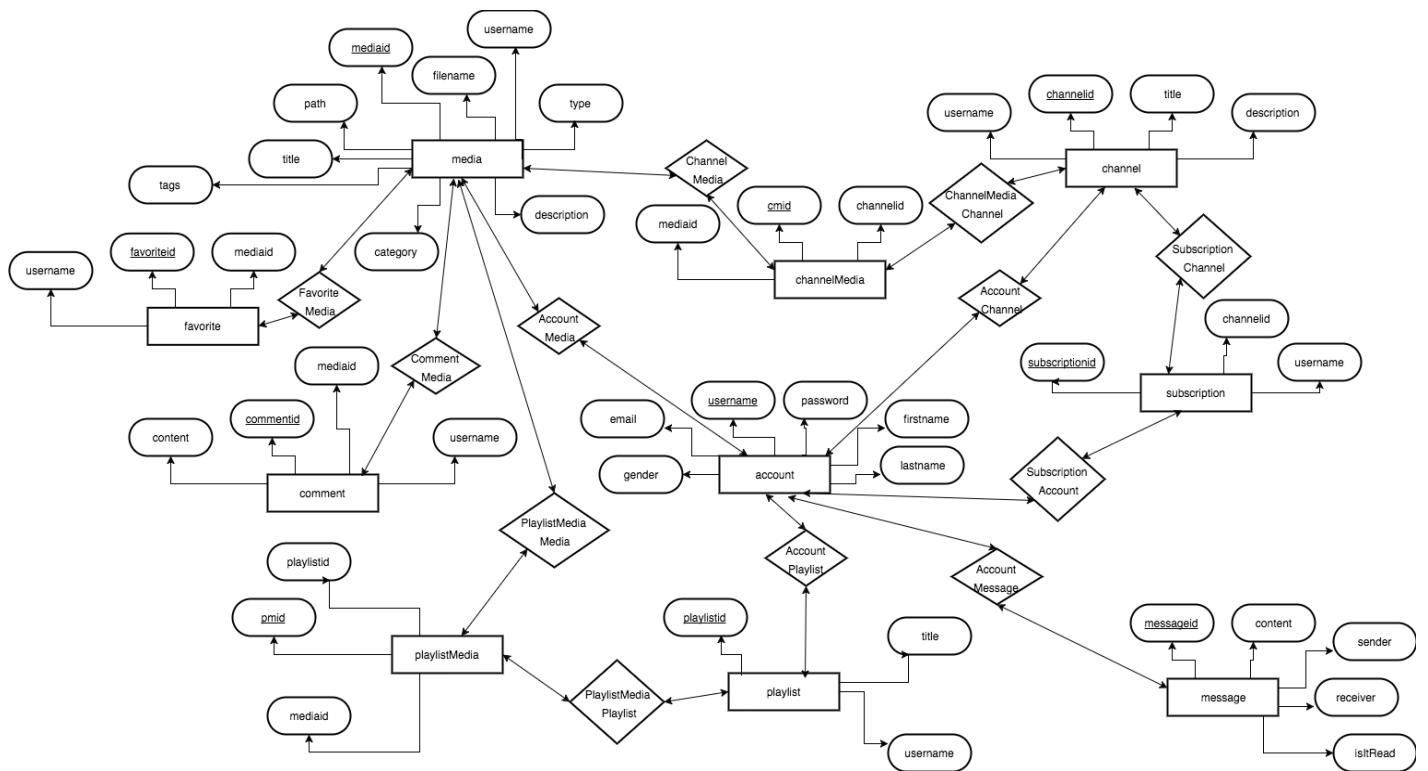
Our MeTube system uses three different components: MySQL, PHP, and HTML. MySQL is used to create, maintain, and search our database. This database consists of ten tables: account, channel, channelMedia, comment, favorite, media, message, playlist, playlistMedia, and subscription. Each table has a primary key that can uniquely identify a row in that table and they also have other fields that contain important information. For example, the account table has the primary key of username. No two users can have the same username, that way each user can be found in the table without confusion. The account table also has five other fields that contain information about that account: password, firstname, lastname, gender, and email.

To access the MySQL database we use MySQL and PHP queries. PHP is a server side scripting language that can create dynamic and interactive websites. We use PHP script to access the MySQL database. PHP is used throughout our system but is very concentrated in the files that end in “_process.php”. These files are mostly PHP script that query the MySQL database. The other files can include PHP and MySQL queries but they also have HTML.

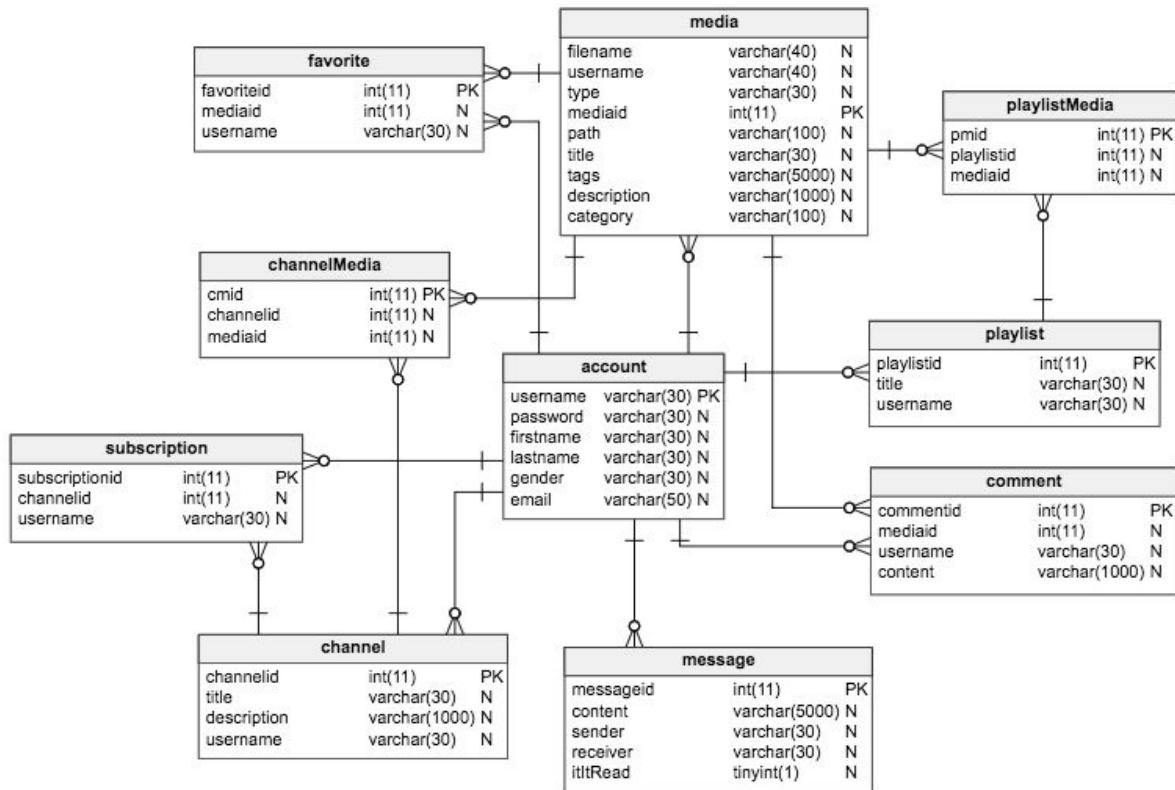
HTML is used to make our webpages more interactive and easy to use for everyone, not just people with a programming background. HTML is a markup language, this means that it allows us to display information. To link PHP and HTML, we use HTML forms to send information to PHP. So when a user clicks on an HTML button, corresponding information gets sent to PHP which then allows us to query the MySQL database. HTML provides an interface that the users can easily follow to accomplish a task. This means that there is an obvious path of clicks the user can execute to complete a task or find the information they were looking for.

In summary, to complete our MeTube system, we combine MySQL, PHP, and HTML to make an interactive web application.

ER Diagram



Database Schema



Function Design

The first set of functionality that needed to be implemented was the functionality dealing with a user account: registration, sign-in, and profile update. Registration is exactly what you would expect: a user needs to be able to register for an account. After a user registers, they need to be able to use the credentials they just created to sign back into the system. Once they have successfully registered and signed into the system, they need to be able to update their profile. Without this feature a user is stuck with the original information they put in. So, for example, if a user changes email addresses they would like to be able to inform their friends. If they can't update their profile there is no way for them to do so.

The second set of functionality that was implemented was the data sharing functionality: upload, metadata input, and download/view. Once a user has an account they are able to upload media. When a user uploads the media they can input the metadata information such as title, description, category, and tags. Once the media has been uploaded, any user, registered or not, can download and view it. An additional functionality that we implemented is the ability to update the metadata for media that a specific user owns after it has been uploaded.

After data sharing, the next functionality that was implemented was the media organization: browse by category, channel, playlists, and favorite lists. Browse by category is exactly what you would expect: a user can see all media sorted into their appropriate category that is assigned by the original owner. A user can organize media into their own channels and other users can subscribe to that channel. A user can also organize media into their own playlist, but that playlist is private to that user; no one else can see it. Any media that a user likes, that user can favorite it and it is added to their favorites list. The user can add and remove

media to channels, playlists, and favorites lists as they please. They can also subscribe and unsubscribe to channels as they wish.

After data sharing, user interaction was implemented: messaging and commenting. A user can send another user a message as long as they know their username. The receiver will get a notification that they have a message in their inbox and can reply. When looking at a specific media, a user can leave a comment. Any user can click on the username of any other user that has commented to view their profile.

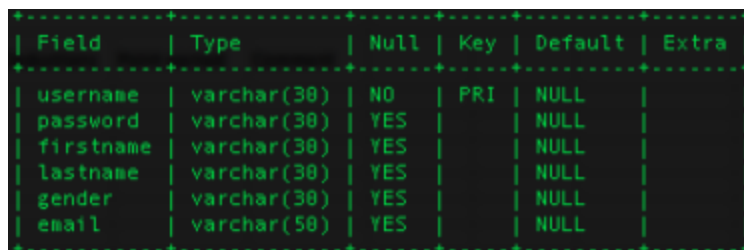
The last feature that was implemented is the keywords-based search feature. When uploading media the user can add tags to that image. The keywords-based search searches the database for media that have the specified tag and displays it back to the user.

Implementation Details

To begin our implementation, we first had to decide what information needed to be stored in the database. For each functionality we had to determine if the current tables could be used to implement the functionality or if we needed to create a new table. If we needed to create a new table then we had to come up with the columns that would be in that table.

We started with the user account functionalities. These included registration, sign-in, and profile update. To start this block of functionality we first had to decide what information we needed in the account table. We ended up deciding that we needed 6 columns: username (primary key), password, firstname, lastname, gender, and email. Figure 1 is a description of this MySQL table. The first functionality we implemented was registration. Registration is implemented in the file register.php. In register.php, we have the user inputting the relevant information into text fields and radio buttons (used for gender). We check to make sure that the two passwords fields that are given are the same and we check to see if another user already has the username that was given. Once those two checks pass, then a function called `check_user_exists` in the file function.php inserts those values into the accounts table. Once registration was complete, we moved on to sign-in. For sign in, the user is taken to a login page which is implemented in the file login.php. In this file we have the user fill in their username and password in text fields. We first check to see if the user completed the required fields. Next, we look to see if the username given is in the account table and we also look to see if the password given is correct based on the username given. These two checks are done in function.php with a function called `user_pass_check`. If the username and password both check out then the `$_SESSION['username']` is set to the username given. Finally we worked on the update profile functionality. This functionality is implemented in the file update_profile.php. In this file, we have

text fields and radio buttons that are pre-populated with the information from the account table under the `$_SESSION` username. If the user would like to change any of these fields they can make those changes in the text fields and radio buttons. We first check to see if the new passwords that are entered are the same, if they are then we call a function named `update_profile_info` (located in `function.php`) that checks to make sure the fields are not empty and then updates the account table for that username. Since username is the primary key we do not allow it to be changed.



Field	Type	Null	Key	Default	Extra
username	varchar(30)	NO	PRI	NULL	
password	varchar(30)	YES		NULL	
firstname	varchar(30)	YES		NULL	
lastname	varchar(30)	YES		NULL	
gender	varchar(30)	YES		NULL	
email	varchar(50)	YES		NULL	

Figure 1: `mysql> DESCRIBE account;`

The next block of functionality we decided to implement was data sharing, which included metadata input, upload, and download/view. For this set of functionalities we decided we needed a media table that has the columns filename, username, type, mediaid (primary key auto increment), path, title, tags, description, and category. The description of this table is shown in Figure 2. The filename is the name of the file that is uploaded, including the file type. The username in this table is the username of the user that uploaded the media. Type describes if the media is a video, image, audio or something else and the file type. So for example, a JPEG image is `image/jpeg`. Next, mediaid is the primary key. Each media is assigned a mediaid. The path is where the file will be placed once it is uploaded. This was included for viewing and downloading. Title, tags, description, and category are the metadata that we added for the metadata functionality. We added title so the user can give their media a descriptive title. Tags are used for keyword searching. Description was added to allow the user to describe the media.

Category is used to implement the browse by category functionality. For the upload functionality, the user first has to input the relevant information. This is done in `media_upload.php`. In this file, we ask for the title, description, and tags in text fields. We also ask for the category through a selection field. This allows the user to pick from a preset list of categories that include Animals, Cars, Children, History, Home, Humor, Music, News, Outdoors, Photography, Science, Sports, Travel, Weather, and Other. Next, we ask them to select a file from their computer to upload. The filename they have selected will appear and then they can click the submit button to upload the media. The submit button will `$_POST` the information to the file we use to implement media upload, `media_upload_process.php`. We first had to create the directory where the media will be stored, if it does not already exist. Next, we put the file in the directory we created. Then, we insert the other information to the media table. For download, whenever the user clicks on a “Download” button, a new window will be brought up in the web browser and the user can use the browser functionality to download the media. Most browsers allow this with a right click and then selecting “Save Image”. To view the media, the user can click anywhere the filename is printed and that will take them to the media page implemented in `media.php`. In this file we display the media with the allowed information. If a user is logged in, the media will be displayed at the top with the title followed by the description, category, whether or not it is on the user’s favorites list, adding it to channels, adding it to playlists, and any comments related to the media. If a user is not logged in then only the title, actual media, description, and category will be displayed. We also included an update media feature. This is implemented the same as update profile. Only the user who uploaded the media can update it. The media table is updated in the `update_media.php` file.

Field	Type	Null	Key	Default	Extra
filename	varchar(40)	YES		NULL	
username	varchar(40)	YES		NULL	
type	varchar(30)	YES		NULL	
mediaid	int(11)	NO	PRI	NULL	auto_increment
path	varchar(100)	YES		NULL	
title	varchar(30)	YES		NULL	
tags	varchar(5000)	YES		NULL	
description	varchar(1000)	YES		NULL	
category	varchar(100)	YES		NULL	

Figure 2: mysql> DESCRIBE media;

The next block of features we implemented was media organization. This included the functions browse by category, channel, playlist, and favorite list. The only table needed to implement browse by category was media, seen in Figure 2. To browse by category the user has to click on the “Browse by Category” button located in browse.php (this you can get to by clicking the media tab on the navigation bar). By clicking “Browse by Category”, the user is taken to browse_category.php. In this file, for every one of the categories (listed above) we list the media that is associated with that category. This is done for each category; we query the media table where the category is equal to the category we are listing. When we list the media we include, the mediaid, title, filename, and a “Download” button.

Next we implemented channel. For channel we decided that we needed three separate tables: channel, channelMedia, and subscription. In channel we have four columns: channelid (primary key auto increment), title, description, and username. The channel table is described in Figure 3. The channelid is the primary key that uniquely identifies the channel. Title allows the user to give the channel a meaningful title. They can also give a description of the channel. Username refers to the username of the user that created the channel (who owns it). This also allows us to access the user information in the account table since it is that table’s primary key. In the channelMedia table there are three columns: cmid (primary key auto increment),

channelid, and mediaid. The channelMedia table is described in Figure 4. The cmid is the primary key that uniquely identifies the separate media in a channel. Channelid is used to identify the channel in the channel table since it is that table's primary key. Mediaid is used to identify which media the user is currently dealing with since mediaid is the primary key in the media table. In the subscription table we have three columns: subscriptionid (primary key auto increment), channelid, and username. The primary key subscriptionid is used to uniquely identify the subscription. The channelid is used to identify the channel since it is the channel table's primary key. The username is used to identify the user who did the subscribing. This is useful because username is the primary key for the account table. The subscription table is described in Figure 5. The channel functionality is implemented in nine files: add_media_to_channel_process.php, all_channels.php, create_channel.php, create_channel_process.php, delete_channel_process.php, delete_media_from_channel_process.php, individual_channel.php, subscribe_process.php, and unsubscribe_process.php. The file add_media_to_channel_process.php inserts into the channelMedia table. The all_channels.php file lists all of the channels that a user owns and their subscriptions. The file create_channel.php is where the user inputs the information to create a new channel. Next, the create_channel_process.php inserts the information taken in create_channel.php into the channel table. The delete_channel_process.php file deletes the specified channel from the channel, deletes all media in that channel from the channelMedia table and deletes all subscriptions to that channel in the subscription table. The delete_media_from_channel_process.php file deletes a specified media from a specified channel from the channelMedia table so it is deleted from that channel. The file individual_channel.php lists all of the media that is in a channel. The subscribe_process.php file is used to insert the relevant information into the subscription table whenever a logged in user

clicks a subscribe button. The unsubscribe_process.php file is used to delete the subscription from the subscription table whenever a logged in user clicks an unsubscribe button for a channel.

Field	Type	Null	Key	Default	Extra
channelid	int(11)	NO	PRI	NULL	auto_increment
title	varchar(30)	YES		NULL	
description	varchar(1000)	YES		NULL	
username	varchar(30)	YES		NULL	

Figure 3: mysql> DESCRIBE channel;

Field	Type	Null	Key	Default	Extra
cmid	int(11)	NO	PRI	NULL	auto_increment
channelid	int(11)	YES		NULL	
mediaid	int(11)	YES		NULL	

Figure 4: mysql> DESCRIBE channelMedia;

Field	Type	Null	Key	Default	Extra
subscriptionid	int(11)	NO	PRI	NULL	auto_increment
channelid	int(11)	YES		NULL	
username	varchar(30)	YES		NULL	

Figure 5: mysql> DESCRIBE subscription;

Next we worked on implementing the playlist functionality. To do this we needed two tables: playlist and playlistMedia. The table playlist has three columns: playlistid (primary key auto increment), title, and username. The playlistid field is used to uniquely identify a playlist in the table. The title field is used so the user can give it a meaningful title. Username is used to identify the user that owns the playlist and it is useful to access the account table since username is its primary key. The playlist table is described in Figure 6. The playlistMedia table also has three columns: pmid (primary key auto increment), playlistid and mediaid. PlaylistMedia

is very similar to channelMedia. The primary key, pmid, is used to uniquely identify media that are in a playlist. The playlistid is used to identify the playlist that the media is associated to. This is helpful since playlistid is the primary key for the playlist table. Finally, the field mediaid is used to connect the media in a playlist to the media in the media table since mediaid is the primary key for it. To implement the functionality for playlist we created five files:

add_playlist_process.php, all_playlists.php, delete_media_from_playlist_process.php, delete_playlist_process.php, and individual_playlist.php. In the file add_playlist_process.php, we have three different \$_POST ids that are sent: 'createPlaylist', 'createAndAddToPlaylist', and 'addToPlaylist'. We first check to see if the ids 'createPlaylist' or 'createAndAddToPlaylist' are set and if they are we insert into the playlist table with all the information needed (title is sent through \$_POST). Next we check to see if 'createAndAddToPlaylist' or 'addToPlaylist' are set. If they are then we insert the relevant information into the playlistMedia table. This information is found by sending the playlist title and mediaid through \$_POST and then selecting the other information from the playlist table. The file all_playlist.php lists all of the playlist that a user has. This is only done for the \$_SESSION username because a user can only look at their playlists. The file delete_media_from_playlist_process.php deletes a specified media from a playlist. This is done by removing it from the playlistMedia table where the mediaid and playlistid are equal to the ones that were \$_POSTed. The next file, delete_playlist_process.php will delete from the playlist table where the playlistid is equal to the one that was \$_POSTed as well as all the entries in the playlistMedia with the same playlistid. Finally, the individual_playlist.php will print all the media that is in a playlist. This is done by selecting the media from the playlistMedia table where the playlistid is the same as the one that was \$_POSTed.

Field	Type	Null	Key	Default	Extra
playlistid	int(11)	NO	PRI	NULL	auto_increment
title	varchar(30)	YES		NULL	
username	varchar(30)	YES		NULL	

Figure 6: mysql> DESCRIBE playlist;

Field	Type	Null	Key	Default	Extra
pmid	int(11)	NO	PRI	NULL	auto_increment
playlistid	int(11)	YES		NULL	
mediaid	int(11)	YES		NULL	

Figure 7: mysql> DESCRIBE playlistMedia;

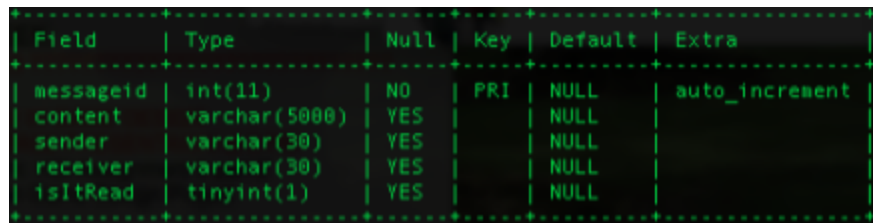
To implement favorites list we came up with one table: favorite. The favorite table has three columns: favoriteid (primary key auto increment), mediaid, and username. It is described in Figure 8. The primary key favoriteid is used to identify a unique instance of this table. The field mediaid is used to identify the media that is being favorited. This is useful since mediaid is the primary key for the media table. Finally, username is used to identify the user that is doing the favoriting. This also is useful since it is the primary key for the account table. There are three files that implement the favoriting functionality: favorite_process.php, favorites.php, and unfavorite_process.php. In favorite_process.php, we insert into the favorite table using the mediaid and username that are \$_POSTed. The file favorites.php is clicked when a logged in user goes to their profile and clicks "My Favorites". This lists all the media that this user has favorited. This is done by selecting from the favorites table where the username is equal to the \$_SESSION username. Finally, in unfavorite_process.php, we delete from the favorite table where the mediaid is equal to the \$_POSTed mediaid and where the username is equal to the \$_SESSION username.

Field	Type	Null	Key	Default	Extra
favoriteid	int(11)	NO	PRI	NULL	auto_increment
mediaid	int(11)	YES		NULL	
username	varchar(30)	YES		NULL	

Figure 8: mysql> DESCRIBE favorite;

The next block of functionality we implemented was user interaction. User interaction includes messaging and commenting. To implement messaging we created the table message which is described in Figure 9. In message we have messageid (primary key auto increment), content, sender, receiver, and isItRead. The messageid is the primary key that uniquely identifies a message. The content field is what the message says. The sender is the username of the person sending the message. The receiver is the username of who the message is being sent to. These are useful because they correspond to a username which is the primary key in the account table. The field isItRead is used to see if the message has been read by the receiver. There are two files used to implement the message functionality: inbox.php and message_process.php. In inbox.php we first print all of the usernames that the current user has messages with. On that list the user can click on the username they want to message and it will take them to the message_process.page. If a message is sent or received then the username printed will show up red. We use the isItRead field in the table to do this. If the receiver sees the message then it gets set to true, otherwise it stays to what it was set to automatically which is false. Next on the inbox.php file, we give the user the option to message a user they don't currently have a message chain with. They can do this by entering the username of the person they want to message and hit the send button which takes them to message_process.php. If they input a username that is not in the account table or their own username then they are asked to input a different username. In message_process.php, we print every message (in order

of being sent) that the two users have sent each other. Next, we have a textarea where the \$_SESSION user can send a new message. We do this by inserting the corresponding information to the message table where the \$_SESSION username is the sender and the \$_POSTed username (or the \$_GET username depending on how it was sent) is the receiver. Users can also send a message to a user when they view their profile and click “Send Message To” button.

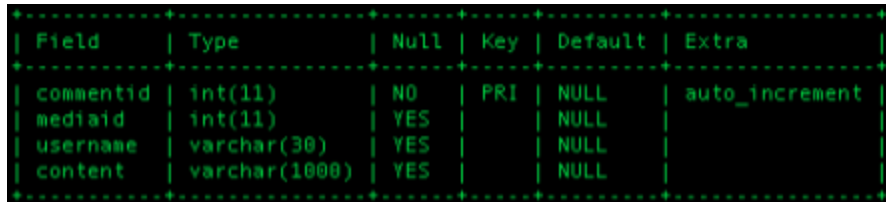


Field	Type	Null	Key	Default	Extra
messageid	int(11)	NO	PRI	NULL	auto_increment
content	varchar(5000)	YES		NULL	
sender	varchar(30)	YES		NULL	
receiver	varchar(30)	YES		NULL	
isItRead	tinyint(1)	YES		NULL	

Figure 9: mysql> DESCRIBE message;

The next functionality we did was commenting. Commenting is similar to messaging. To implement commenting we created a table called comment which is described in Figure 10. It has four columns: commentid (primary key auto increment), mediaid, username, and content. The primary key commentid is used to uniquely identify each comment. The mediaid field is used to identify the media that a user is commenting on. This is useful because mediaid is the primary key for the media table. Username is used to identify the user that is leaving the comment. This is also useful because username is the primary key for the account table. Content is used to store the actual comment the user wants to leave. We created comment_process.php and used media.php to implement commenting. The file comment_process.php is sent through \$_POST the content and the mediaid. The username is the \$_SESSION username. Next we insert into the comment table with the username, mediaid, and content. In the media file, we check to see if a user is logged in, if they are then they can see all of the comments that have been left on this media. This is done by selecting everything

from the comment table where the mediaid is equal to the mediaid we are currently looking at which is sent through \$_GET. A logged in user is also given the option to leave a comment in a textarea. After they enter their comment and then they hit the “Post” button the relevant information is sent to message_process.php and then the media page refreshes which makes the new comment appear with the rest of the comments that go with that media.



Field	Type	Null	Key	Default	Extra
commentid	int(11)	NO	PRI	NULL	auto_increment
mediaid	int(11)	YES		NULL	
username	varchar(30)	YES		NULL	
content	varchar(1000)	YES		NULL	

Figure 10: mysql> DESCRIBE comment;

The next feature we worked on was keyword based search. This was implemented by using the existing media table. To search, the user has to input a word or words they want to search on in the search bar in the navigation header (discussed in the next paragraph) and then click the “Submit” button. This brings up the file search_process.php. In this file we break down the search words using the explode function and then loop through those words. Each time through the loop we select the media where a search word is like one of the tags that media has. If a media has a tag we are searching for then it gets printed to the user. At the end of the loop, all media that has the key word as a part of their tags is printed to the user.

Finally, we implemented what we call our navigation header. This header’s purpose is to make navigating the MeTube site easier. We implement the header in header.php. This file is included in every one of our viewable files. In this file we check to see if the \$_SESSION user is set and if it is we create the buttons: ME TUBE, Media, Users, Inbox, Profile, Logout, and the search bar. This is shown in Figure 11. METUBE and Media will take the user to the browse.php file which lists all of the media in the media table. The Users button will take the user to the accounts.php page where all of the user accounts are listed. Inbox will take the user to their

inbox (inbox.php) where they can see their messages. Profile will take the user to their profile where they can view it on profile.php. The Logout button will log the user out and take them back to index.php. If a user is not logged in then they see: ME TUBE, Media, Users, Login, Register, and the search bar. This is shown in Figure 12. ME TUBE, Media, and Users are the same functionality as if someone is logged in except they will have different options when taken to browse.php and accounts.php. The Login button will take the user to the login.php page and the Register button will take the user to the register.php page.



Figure 11: Navigation Header view when user is logged in.



Figure 12: Navigation Header when user is not logged in.

Test Cases

User Account:

Registration:

- Making sure the user can register
- Making sure the data shows up in the account table

Login:

- Checking username and password entered with what is in account table
- Making sure the user can logout

Update Profile:

- Updating account table with information gathered from user using the primary key username

Data Sharing:

Upload Media:

- Users can upload media
- Making sure the data shows up in the media table

Update Media:

- Updating media table with information gathered from user using the primary key mediaid

Metadata Input:

- Making sure that users can add metadata such as title, tags, description and category

Download and View:

- Any user can download and view a media in the media table

Media Organization:

Browse By Category:

- By clicking on the Browse By Category button, a user can see what media is in what category

Channel:

- Logged in users can create channels and add media to channels
- Logged in users can subscribe and unsubscribe to other user's channels

Playlists:

- Logged in users can create playlists and add media to playlist

Favorites List:

- Logged in users can favorite and unfavorite media

User Interaction:

Messaging:

- Logged in users can message one another in a message chain

Commenting:

- Logged in users can leave comments on media

Search:

Keyword Search:

- Users can search the media based on keywords called tags

Testing Results

User Account:

Registration:

- User registration is successful.
- The system checks to see if the username that was given isn't already in the account table and then it adds all of the information given to the account table.

Login:

- User login is successful. The system checks to see if the username and password entered is correct based on what is in the account table.
- User logout is successful.

Update Profile:

- Profile update is successful. The system populates the logged in user's information and the user can change that information. The screen will refresh and the account table will be update based on the information given.

Data Sharing:

Upload Media:

- Upload media is successful. Users can upload many different types of media, including photos, videos and audio files. Only a logged in user can upload media.
- The system successfully adds the information to the media table.

Update Media:

- Update media is successful. The system populates the media information and only the user who uploaded that media can change that information. The screen will refresh and the media table will be update based on the information given.

Metadata Input:

- Attaching metadata is successful. A user can add a title, tags, description and category to the media that they have uploaded.

Download and View:

- Download media is successful. Any user can view and download any media file.

Media Organization:

Browse By Category:

- Browse by category is successful. Any user can browse by category. The system searches the media table based on the set categories. Each category will be shown and the associated media will be listed below.

Channel:

- Channel is successful. A logged in user can create multiple channels and can add any media to each channel. For each channel created, it is added to the channel table and each media that is added to a channel, gets added to the channelMedia table.
- Logged in users can also subscribe and unsubscribe from channels by adding to or deleting from the subscription table. Any user can view a user's channels. The user who created a channel can delete it and any media associated with a channel that they have.

Playlists:

- Playlist is successful. A logged in user can create a playlist and add any media to the playlist. For each playlist created, it is added to the playlist table and each

media added to that playlist, gets added to the playlistMedia table. The user who created the playlist can delete the playlist and the media attached. Only the user who created the playlist can view it.

Favorites List:

- Favorites list is successful. A logged in user can favorite or unfavorite any media. By favoriting a media, that information gets added to the favorite table and by unfavoriting a media it gets removed from the favorite table.

User Interaction:

Messaging:

- Messaging is successful. A logged in user can message any user. The system does this by adding a message to the message table.

Commenting:

- Commenting is successful. A logged in user can comment on any media. The system does this by adding a comment to the comment table.

Search:

Keyword Search:

- Keyword search is successful. Any user can search the media based on keywords in the search bar at the top of the screen. The system searches the media table looking for that keyword in the tags column.

User Instruction

To begin with, go to <http://webapp.cs.clemson.edu/~rdausti/metube462/index.php>. (rdausti is the only one who has access to the database. Using sllawso won't work because it was never added to Buffet.)

User Account:

Registration:

When you open MeTube, you will see a navigation bar on the top of the screen. On the far right hand side is a button labeled "Register." Click this button and it will take you to the registration screen. Fill in the fields for username, password, first name, last name, gender, and email. Click the submit button.

Login:

When you open MeTube, you will see a navigation bar on the top of the screen. On the far right hand side is a button labeled "Login." Click this button and it will take you to the login screen. Enter in your username and password. If you mess up, you can hit the reset button and it will clear the entry fields. Once you have entered your username and password correctly, click the login button.

Update Profile:

When on the Profile page, there is a blue bar, underneath the pink navigation bar. On the far right hand side there is a button labeled "Update Profile." Click this button and it will take you to the update profile screen. Change any data that you want, except your username, and click the update button. Your changes have been saved so feel free to navigate away from this screen.

Data Sharing:

Upload Media:

On the media page, on the far left hand side, there is a button labeled "Upload File." Click this button and it will take you to the upload media page. Fill in the fields for title, description, category, and tags. Choose the file you want to upload and click upload. This takes you back to the media page.

Update Media:

On the profile page, there is a blue bar labeled with [username]'s media. You will see all of the media that you have uploaded listed there. Next to each file, on the far right hand side is a button labeled "Update Media." Click this button and it will take you to the update media page. Update any of the fields except Filename, Username, Type of media, Media ID, and File Path. Click the update button. Your changes are saved so feel free to navigate away from this screen.

Download and View:

On any page that displays media, there will be a button labeled "Download" on the far right side of the screen. Click it to download the media.

When you are viewing a specific media, right underneath the media is a button labeled "Click to Download." Click this button to download the media.

Media Organization:

Browse By Category:

On the media page, on the far right hand side, there is a button labeled “Browse By Category.” Click this button and it will take you to the browse by category page. Scroll up and down to view all the categories. Click the filename to view the image. Click the “Download” button to download the media.

Channel:

On the profile page, there is a blue bar underneath the pink navigation bar. On the far left hand side there is a button labeled “My Channels.” Click this button and it takes you to the my channels page. Here you can see your channels and your subscriptions. You may also create and delete channels, unsubscribe from channels you subscribe to, and if you click on a channel, remove media from that channel.

To view another user’s channels, go to the Users page. Click on a user. There is a button labeled “[username]’s Channels.” Click this button to see their channels. Here you can subscribe to this user’s channels.

When viewing a specific media, scroll down and there is a blue bar labeled Channels:. The drop down bar lists the channels you created. You can add media to any channel that already exists.

Playlists:

On the profile page there is a blue bar underneath the pink navigation bar. Second from the right there is a button labeled “My Playlists.” Click this button and it will take you to the my playlists page. Here you can view your playlists, delete your playlists, or create a new playlist.

When viewing a specific media, scroll down and there is a blue bar labeled Select Playlist:. Here you can use the drop down to add the media to an existing playlist or you can create a new playlist and add it to that one.

Favorites List:

On the profile page there is a blue bar underneath the pink navigation bar. Second from the left there is a button labeled “My Favorites.” Click this button and it will take you to the my favorites page. Here you can view your favorite media, download it, or unfavorite it.

When viewing a specific media, scroll down and there is a blue bar labeled “Not On Favorites List” if you have not favorited the media, or “On Favorites List” if you have favorited the media. Here you can either favorite or unfavorite that specific media.

User Interaction:

Messaging:

On any page with the navigation bar there is a button labeled “Inbox.” This button will come up with a number beside it if you have a new message. Click the button to be taken to your inbox. Here you can see all messages that are in your inbox. If they are lit up red that means that either you have a message from them, or you sent them a message that they have not yet opened. You can also start a new message thread by putting in a username into the Send Message To: text field. If you click on a username of an existing message thread you can see all previous messages sent between yourself and this user. You can send a new message to this user from this screen.

Commenting:

When viewing a specific media, scroll down and there is a blue bar labeled Comments:. In the text area you may type a comment and press post to comment on this media. If a user has already commented on this media, you may click their username and view their profile.

Search:

Keyword Search:

On the navigation bar on the far right hand side there is a search bar. Enter a word into the search bar and press the submit button. If any of the media have this word as a tag you will see them in a list.