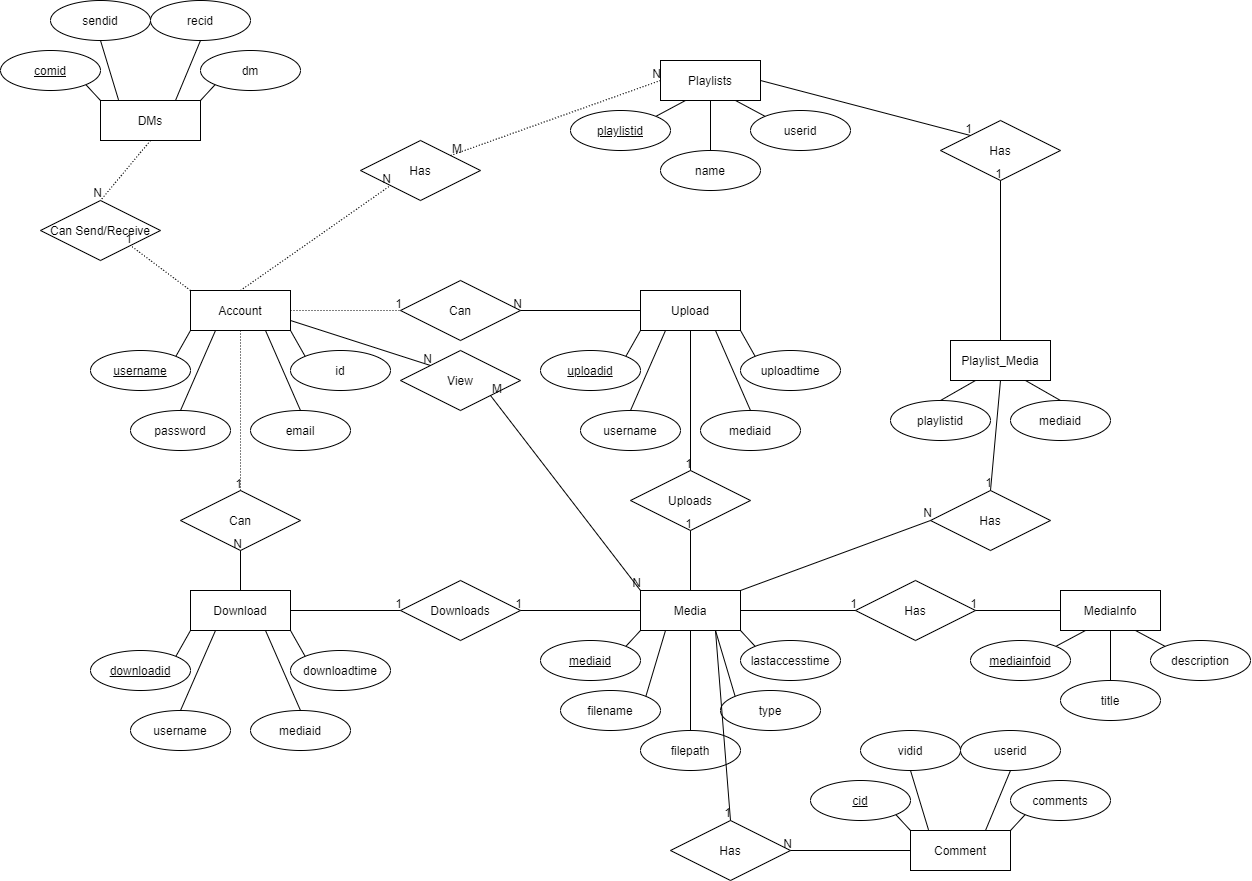
<http://webapp.computing.clemson.edu/~cmearl/databaseproject/metube/index.php>

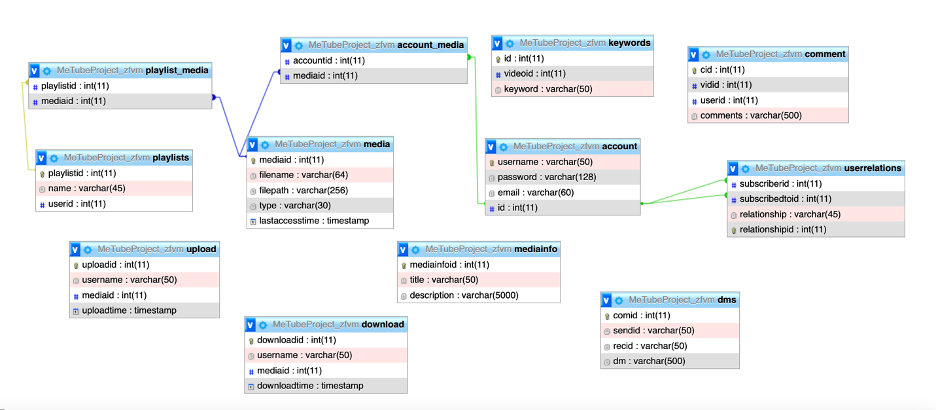
**System Design**

Our metube website is built off the preexisting framework provided to us. It uses apache, php, JavaScript, and MySQL. Most interactions between the front end and the server are facilitated using MySQL queries to the databases using php. A restriction would be MySQL’s inability to store array values and create unique structures on demand. This would be better for comments on videos or direct messages between users as they are structurally simple and would be easier to manage the content of in an admin program.

**ER Diagram**

****

**Database Schema**

****

**Function Design**

Logout: Php sessions is cleared via a quick redirect

Registration: Registration checks that users have filled out every box and the uses a query to check if the username is unique. If the username is unique and the 2nd password matches the first then the user information is added.

Profile: The form checks if the user has filled any of the fields. If they 1 or more is then it updates the field. Once again the two passwords are checked to see if they are identical and if they aren’t the password won’t be updated. Username is also checked again to see if it’s unique.

Dms: The page first will fill a list of all users and the user must select one to receive the comment and fill out the comment. Upon doing so the comment will be added to the file and the receiving and sending user will be able to see it and any other pervious comments they made or received.

Media (Comments): Comments allows a user to make a comment and see any previously made comments and who made them.

Media (viewing): Viewing a video allows a user to see the video or an image if it’s an image. There is also a message if the media type they attempt to view isn’t supported.

Upload: A user who uploads a file to the database can add a title, description, and keywords for the video. Keywords is hidden from viewing in the media as it is intended to be used for searching.

Search: A user can type in words that may appear in the title or description of an uploaded media to filter only those results in their view.

Channels: Channels groups videos by who uploaded them. When you click on a channel you can see all the videos uploaded by that user.

Playlists: Playlists are customizable by the user. They can make different playlists and add any video to them. This allows them to group videos however they want.

Favorites: Favorites allows users to like a video. When a user likes a video it will be added to their favorites list.

Categories: Categories allow users to view videos based on a category they belong in. When uploading a video a user can choose a category for it and then users can browse by category.

**Implementation details**

Logout: If user is logged in, clear session information and move them to index.html

Registration: If the user filled out all forums, the 2 passwords match, and the username is unique add them to the database and return them to the main page.

Profile: If a user fills out a field push it to the database unless it’s username or password. If username was changed make sure it’s unique, and if it’s the password both passwords match. Reload the page on completion.

Dms: Query the database to pull all users, loop thru and display in dropdown. User must have set the user to be sent to for a message to be sent. (Since both are filled automatically no changes need to be made to send a message). The page will then display the message. The page also displays all DMs sent and received by the user in the order which they were added FIFO via querying the db and looping thru any that have the currently logged in user as the sender or receiver.

Media (Comments): Comments checks if the user has added a comment and then pushes it to the comment table with their username. Any comments made on the media will be queried by the media’s mediaid and any found will be displayed with the user who added the comment to the video.

Media (Viewing): The database will pull the requested media and mediainfo displaying the media, title, and description. The function will attempt to allow the user to view a video if its mp4, ogg, or webm. Pictures are supported, but if it finds a file type that isn’t the function will output that the media isn’t viewable and should be downloaded.

Upload: The database only checks that the file is unique nothing else as the videoid assigned in upload is used to match the video information to the video. A user doesn’t need to fill out any of the information for the video for it to be uploaded. It will be inserted into the DB as long as the file is unique to the user.

Search: Takes the search parameters and queries the database for media titles or descriptions that are like the parameters, calling the media and displaying it with its link should search results exist.

Channels: Uses a relationship between account and media. Accounts and media are connected in the upload table. To get the content of a channel, we queried that table to get the media that was uploaded by that user.

Playlists: Playlists uses a relationship between the playlist table and the media table. When playlists are created by users, the user’s id is also recorded. This is so that user’s can only see their own playlists. When a user enters a video into a playlist, it is entered with a connection to the playlist.

Favorites: When a user likes a video, it is added to their favorites lists. This relationship is captured by the account\_media table. This connects the user id to the id of the video they liked so that when a user clicks his favorites he can see all the videos he’s liked.

Categories: We implemented three categories in our system: entertainment, music, and gaming. When a user uploads a video they have the option to add it to a category. This adds the name of that category to the keywords table. When you click on the category in browse mode, those videos are queried based on the category.

**Test Cases & Results**

***Logout***:

**Input**: Set \_Sessions[‘username’]

**Result**: \_Sessions[‘username’] = Null

***Registration***:

**Input**: Username = testacc Password=pass Password2= pass Email = a@a

**Result**: User added to database

**Input**: Username = metube Password=pass Password2= pass Email = a@a

**Result**: Fails because username already exists

**Input**: Username = test2 Password=pass Password2= pass2 Email = a@a

**Result**: Fails because passwords don’t match

***Profile***:

**Input**: Username = metube

**Result**: Fails because username already exists

**Input**: Username = test3

**Result**: Success, username is changed

**Input**: Password = test Password2= wrong

**Result**: Fails because passwords don’t match

**Input**: Password = test Password2= test

**Result**: Success, password is updated in the database

**Input**: Email = a@a.com

**Result**: Success, email is updated in the database

***Dms***:

**Input** User: test Comment: test comment

**Result**: Logged in user & user test are able to see Comment

***Media(Commenting)***:

**Input**: Comment = BLAH

**Result**: Comment will be visible

**Input**: Comment1= BLAH Comment2 = Hello

**Result**: Comment1 then Comment2 should be visible

***Media(Viewing)***:

**Input**: User chose a mp4 in browse

**Result**: The user can play the mp4 video

**Input**: User chose a wmv in browse

**Result**: Site tells the user that wmvs aren’t supported

***Uploading***:

**Input**: File = one user has already uploaded

**Result**: Fail, because the user has already uploaded said file

**Input**: Title = test, Description = description, Keywords: video

**Result**: Information is added to the database

***Searching***:

**Input**: “SAMP”

**Result**: *“There are 4 results:”* sample 2 (this is sample 2), sample 3 (sample 3 infp), sample 1 (this is sample 1), sampa (train)

**Input**: “cool”

**Result**: *“There are 3 results:”* fortnite video (cool fortnite video), fortnite video (cool fortnite gaming video), fortnite (cool)

**Input**: “bruh”

**Result**: *“No result matching ‘bruh’”*

**Input**: “test”

**Result**: *“There are 1 result:”* test video (test video testing the size limit of uploads)

***Channels:***

**Input: User uploads a video.**

**Result: Video will appear under their channel**

**Input: User clicks on another users channel**

**Result: All videos from that user will be displayed**

***Playlists:***

**Input: User types a playlist name and hits create.**

**Result: Playlist will appear under their playlists button**

**Input: User clicks add media and uploads video to playlist**

**Result: Video shows up under playlist**

***Favorites:***

**Input: User likes a video.**

**Result: Video will appear in their favorites.**

**Input: User likes a video they’ve already liked**

**Result: Video will only be in their favorites once**

***Categories:***

**Input: User clicks a category when uploading a video.**

**Result: Video will be able to be browsed by that category**

**Input: User browses by category that hasn’t had a video uploaded in it yet**

**Result: No videos will display under category**

**User Guide/Manual**

All users, whether logged in or not, can search for media, and can furthermore browse all existing media. All users can similarly view and download media. A user not logged in can log in if they have an existing account, or can register to create a new account. All users are able to view comments posted on media.

A logged in user can update their profile information, should they choose to. They can similarly view their channels, playlists, and media they’ve designated as favorites. They can also send and receive direct messages (DMs), and save contacts to view their uploaded media. Logged in users can view media categories, and are likewise able to comment on media. In order to subscribe to a channel, the user must click the video under the certain channel, and hit subscribe underneath the video. Finally, a logged in user can upload media.

In order to view media, a user need only browse until they find something of interest to them, or otherwise search if they have interesting ideas in mind. Upon finding media that interests them, they can click the link to pull up the media page, allowing them to view it directly. They can also download media, either by clicking the download button associated with the media in question, or by going to the media page and downloading from there.