MeTube Technical Report

Undergraduate

Mark Todd

Table of Contents

System Design

MeTube is a functional replica of YouTube.com. While significantly stripped down in presentation and lacks a few of the features, it provides a good example of a system based around user created content. The design of the project gives a fair overview of a common use-case for the LAMP software bundle (Linux OS, Apache http, MySQL, PHP). This involves: A mySQL database backend that holds all long-term information. PHP scripts to provide the web server access to the database and distribute parsable information to Apache. And HTTP sources using a number of tools to the give the user an experience they can participate in.

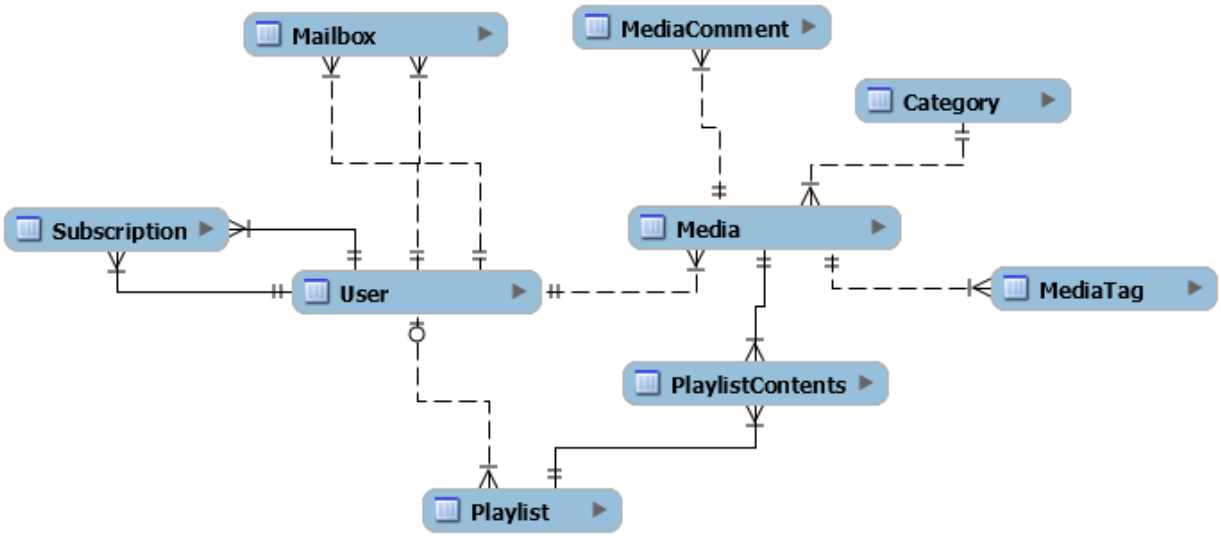
The experience for a user has a certain flow baked into the design. Every part of their use is regulated and passed through the mySQL database, holding everything from their user information and messaging to what media they've uploaded and when. To ensure a consistent experience, the database fits at least 3rd Normal Form requirements, while also using SQL's ability to manage foreign key contraints, many-to-many relationships through an intersection key, auto management of primary key incrementation and uniqueness, and *Views* to make the PHP scripts a lot cleaner in access. The database is as concise as possible while attempting to be usefully robust.

Every part of the database is used by PHP. The PHP uses a tool called MeekroDB to make mySQL database calls significantly cleaner and properly escaped (to ensure no dangerous user-form interaction). By using both *Views* where the same selection contexted is required in multiple places and specific select calls elsewhere, the code is able to provide useful errors when the backend database is altered without fixing the scripts as well. To help with maintenance, the files and functions are organized by purpose and common-ness of use.

The frontend HTTP is interwoven with the PHP where necessary and separated wherever possible to provide readability and problem tracking. The HTTP uses both javascript and more helpfully jQuery to give a bit of dynamism yet still keep the experience light. To keep the user wanting to view more content, the media display page uses a tool called Fancybox that creates a "lightbox" for selected items. This makes it easier to quickly switch out of one piece of media and into another without headache. Further, each page has the tools directly related to their backend functionality available (Media pages provide access to subscribing, adding to playlists, leaving comments…)

Finally, the client player is a JWPlayer, a javascript solution that uses either HTML5 or Flash to provide as compatibility wherever possible. Using JWPlayer also allows for built-in playlist support and gives the web designer many options for how and when to display the content.

ER Diagram



Solid lines indicate relationships where the foreign key is a member of the primary key. All relationships are one-to-many relationships where the carrot is the many connection and the barre is the one.

Database Schema (mySQL DUMP)

user

```
CREATE TABLE `user`
  (
     `uid`       INT(11) NOT NULL auto_increment,
     `email`     VARCHAR(45) NOT NULL,
     `username` VARCHAR(45) NOT NULL,
     `password` VARCHAR(45) NOT NULL,
     PRIMARY KEY (`uid`)
  )
```

category

```
CREATE TABLE `category`
  (
     `cid`  INT(11) NOT NULL auto_increment,
     `name` VARCHAR(25) DEFAULT NULL,
     PRIMARY KEY (`cid`)
  )
```

media

```
CREATE TABLE `media`
  (
     `mid`           INT(11) NOT NULL auto_increment,
     `title`         VARCHAR(45) DEFAULT NULL,
     `description`   VARCHAR(300) DEFAULT NULL,
     `type`          VARCHAR(45) DEFAULT NULL,
     `date_created` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
     `viewcount`     INT(11) DEFAULT '0',
     `path`          VARCHAR(150) DEFAULT 'media/default.mp3',
     `uid`           INT(11) DEFAULT NULL,
     `cid`           INT(11) DEFAULT NULL,
     `imagepath`     VARCHAR(100) NOT NULL DEFAULT
'media/default.png',
     PRIMARY KEY (`mid`),
     KEY `uid_idx` (`uid`),
     KEY `category_idx` (`cid`),
     CONSTRAINT `category` FOREIGN KEY (`cid`) REFERENCES
`category` (`cid`) ON
     DELETE no action ON UPDATE no action,
     CONSTRAINT `mediauser` FOREIGN KEY (`uid`) REFERENCES `user`
(`uid`) ON
     DELETE no action ON UPDATE no action
  )
```

mediatag

```
CREATE TABLE `mediatag`
  (
```

```
    `tid`  INT(11) NOT NULL auto_increment,
    `name` VARCHAR(125) DEFAULT NULL,
    `mid`  INT(11) DEFAULT NULL,
    PRIMARY KEY (`tid`),
    KEY `media_idx` (`mid`),
    CONSTRAINT `tagmedia` FOREIGN KEY (`mid`) REFERENCES `media`
(`mid`) ON
    DELETE CASCADE ON UPDATE CASCADE
  )
```

playlist

```
CREATE TABLE `playlist`
  (
    `pid`   INT(11) NOT NULL auto_increment,
    `uid`   INT(11) DEFAULT NULL,
    `title` VARCHAR(60) DEFAULT NULL,
    `share` INT(11) DEFAULT NULL,
    PRIMARY KEY (`pid`),
    UNIQUE KEY `title` (`title`),
    KEY `playlistuser` (`uid`),
    CONSTRAINT `playlistuser` FOREIGN KEY (`uid`) REFERENCES
`user` (`uid`) ON
    DELETE no action ON UPDATE no action
  )
```

mediacomment

```
CREATE TABLE `mediacomment`
  (
    `cid`       INT(11) NOT NULL auto_increment,
    `mid`       INT(11) DEFAULT NULL,
    `uid`       INT(11) DEFAULT NULL,
    `date_sent` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
    `message`   VARCHAR(500) DEFAULT NULL,
    PRIMARY KEY (`cid`),
    KEY `media_idx` (`mid`),
    KEY `user_idx` (`uid`),
    CONSTRAINT `commedia` FOREIGN KEY (`mid`) REFERENCES `media`
(`mid`) ON
    DELETE no action ON UPDATE no action,
    CONSTRAINT `comuser` FOREIGN KEY (`uid`) REFERENCES `user`
(`uid`) ON
    DELETE no action ON UPDATE no action
  )
```

mailbox

```
CREATE TABLE `mailbox`
  (
```

```
    `meid`       INT(11) NOT NULL auto_increment,
    `uid`        INT(11) DEFAULT NULL,
    `suid`       INT(11) DEFAULT NULL,
    `title`      VARCHAR(45) DEFAULT NULL,
    `date_sent` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
    `message`    VARCHAR(1000) DEFAULT NULL,
    `unread`     INT(11) DEFAULT '1',
    PRIMARY KEY (`meid`),
    KEY `user_idx` (`uid`),
    KEY `sender_idx` (`suid`),
    CONSTRAINT `mailuser` FOREIGN KEY (`uid`) REFERENCES `user`
(`uid`) ON
    DELETE no action ON UPDATE no action,
    CONSTRAINT `sender` FOREIGN KEY (`suid`) REFERENCES `user`
(`uid`) ON
    DELETE no action ON UPDATE no action
  )
```

subscription

```
CREATE TABLE `subscription`
  (
    `uid`    INT(11) NOT NULL,
    `subid` INT(11) NOT NULL,
    PRIMARY KEY (`uid`, `subid`),
    KEY `subscription_idx` (`subid`),
    CONSTRAINT `subuser` FOREIGN KEY (`uid`) REFERENCES `user`
(`uid`) ON
    DELETE no action ON UPDATE no action,
    CONSTRAINT `subscription` FOREIGN KEY (`subid`) REFERENCES
`user` (`uid`)
    ON DELETE no action ON UPDATE no action
  )
```

playlistcontents

```
CREATE TABLE `playlistcontents`
  (
    `pid`     INT(11) NOT NULL,
    `mid`     INT(11) NOT NULL,
    `ordnum` INT(11) DEFAULT NULL,
    PRIMARY KEY (`pid`, `mid`),
    KEY `media_idx` (`mid`),
    CONSTRAINT `contplaylist` FOREIGN KEY (`pid`) REFERENCES
`playlist` (`pid`)
    ON DELETE CASCADE ON UPDATE no action,
    CONSTRAINT `playlistmedia` FOREIGN KEY (`mid`) REFERENCES
`media` (`mid`)
    ON DELETE CASCADE ON UPDATE no action
```

```
  )
```

mediauser

```
CREATE view `mediauser`
AS
  SELECT `media`.`mid`          AS `mid`,
         `media`.`title`        AS `title`,
         `media`.`type`         AS `type`,
         `media`.`date_created` AS `date_created`,
         `media`.`imagepath`    AS `imagepath`,
         `user`.`username`      AS `username`,
         `media`.`uid`          AS `uid`,
         `media`.`path`         AS `path`,
         `media`.`cid`          AS `cid`
  FROM   (`media`
          JOIN `user`
            ON(( `user`.`uid` = `media`.`uid` )));
```

popularTags

```
CREATE view `popularTags`
AS
  SELECT `mediatag`.`name`         AS `Name`,
         Count(`mediatag`.`name`) AS `Count`
  FROM   `mediatag`
  GROUP  BY `mediatag`.`name`
  ORDER  BY Count(`mediatag`.`name`) DESC;
```

Function Design

The projects functionality fits into a few key concept areas: *Users, User Interaction, Media Content, Organization*

Users:
The project is built around User posted content and interaction from site design down to the database schema. As such, the site requires a users be registered and logged in at all times. Users from any page should be able to quickly access the information they are looking for. To this end, the site interface provides a top menu for interacting on a user-base (messaging) and content searching, and a sidebar that provides access direct to media listings. This bar is constructed based on the choices the user has made (subscriptions, playlists) and the choices users as a whole have made (tag popularity).

User Interaction:
Given the focus on Users, a small amount of direct and indirect user interaction exists. Users may send and receive messages from other users. They may also leave comments on other users' content and add users to their "subscriptions". Subscriptions allows a User to view all content uploaded by a select group of users ordered by recency.

Media Content:
The site holds media of three types- Images, Audio, Video and makes no effective functional changes based on which you are viewing. What this means for the User is a consistent experience throughout the site. The Media holds "metadata" including a Title, Description, Category, Tags and Thumbnail that are used in the generation of the media player page as well as in how media is sorted.

Organization:
Media is organized by the server entirely based off its metadata. Users may search based on tags, view all files uploaded by another user, view by recency (the homepage default) and view those files based on placement into a static list of categories. Further, Users are able to organize media themselves through the use of Playlists, and play the entire playlist from one page.

Implementation Details

Website layout is dictated via the **style/style.css** and **index.php**. **index.php** holds the wrapper that holds the divs that dictate location of the menus and the iframe that holds the actual content.

Directory **ajax/** holds all of the scripts which call the database either from a new window (that closes after) or through AJAX calls. These scripts edit data that isn't directly retrieved by the site for the user experience. These include:

- **addmedia.php -** Handles the upload of files and metadate supplied by the form in **upload.php**, all files are chmod to 644 and any new created directories use 754
- **addplay.php -** Handles the addition of media to a playlist when a user select in **player.php**
- **register.php -** Handles user registration from **login.php**
- **remplay.php -** Handles removing a media file from a playlist while the user is watching a playlist (note, at the moment does not update the playlist during viewing)
- **sentcomm.php -** Handles comments submitted to videos by users on **player.php**
- **sentmess.php -** Handles message transmission between users
- **subscribe.ph -** Handles subscription to users (adds them to subscribed list and adds their videos to the My Subscriptions tab)

Directory **fancybox/** holds the fancybox tool discussed in the *System Design* that allows for "lightbox" iframes to display over the current page.

Directory **js/** holds the jwplayer tool discussed in the *System Design* that is used for playing media files (excepting images in nonplaylists) in the browser.

Directory **media/** is where all media uploaded to server is stored and the path the database directs to.

Directory **scripts/** holds the general and page specific function lists. These include:

- **core.php -** This is used by nearly the entire site, giving access a "check_logged" that ensures the user remains logged in on every page, and holds the database access information used by meekrodb
- **filefuncs.php -** Stores all functions used by the **addmedia.php** script to handle the php upload
- **meekrodb.php -** A LGPL tool that makes the mySQL database into a PHP class, providing functions that make queries significantly cleaner and safer by sanitizing input
- **playerfunc.php -** Stores all functions used by the **player.php** script including database access to playlists, media, and comments; as well as generating the jwplayer playlist object displayed to user

- **searchfunc.php -** Stores two functions used by **search.php** script that call the appropriate query based on url parameters and generate the table of media files

The main directory holds the page scripts “seen” by the user including:

- **login.php -** the first page users see, they may login or register
- **message.php -** Provides a list of messages received by other users, sorted by read and unread
- **player.php -** The page that provides the media player and metadata information (including access to commenting)
- **readmess.php -** Page supplied by **message.php** that displays the contents of a given message, marking it as read
- **sendmessage.php -** A form that allows users to send messages to another user
- **sidemenu.php -** Provides the links to the different accesspoints on the site for viewing media
- **upload.php -** A form that lets users upload files and metadata to the server

Test Cases

Registration:

Test Used	Result	Correctness
Pre-existing Username	Report username taken	Yes
Passwords Don't Match	Prevent form transmission, inform user	Yes
Empty Input	Prevent form transmission, inform user	Yes
Username available	Accept Registration	Yes

Login:

Test Used	Result	Correctness
Username/Password not in database	Report incorrect combination	Yes
Empty Input	Prevent form transmission, inform user	Yes
Valid Username/Password Combination	Accept login, move to index.php	Yes

Upload:

Test Used	Result	Correctness
Invalid file format	Fail to upload file, do not add to database	Yes
Empty form metadata	Prevent form transmission, inform user	Yes
Empty form file selection	Prevent form transmission, inform user	Yes
Filled metadata, including	Add to file to data with	Yes

valid file format, may include thumbnail	metadata	
Invalid thumbnail format	If above correct, ignore thumbnail, use default	Yes

View:

Test Used	**Result**	**Correctness**
Select media from **search.php**	*lightbox* the media, provide all metadata	Yes
Select playlist from sidebar	Open in right iframe, provide metadata for “current” media	Yes

Subscription:

Test Used	**Result**	**Correctness**
Subscribe to User not subscribed to	Add relationship to database, inform user	Yes
Subscribe to User already subscribed to	Do nothing, inform user	Yes
Unsubscribe to User subscribed to	Remove relationship from database	Yes

Playlist:

Test Used	**Result**	**Correctness**
Add to existing playlist	Add relationship to database	Yes
Remove from existing playlist	Remove relationship from database	Yes

Create new playlist and add media	Create database item, add new relationship	Yes
Add to existing playlist where media already in playlist	Do nothing	Yes

Messaging:

Test Used	**Result**	**Correctness**
Send message to invalid user	Report user doesn't exist	Yes
Empty form data	Prevent form transmission, inform user	Yes
Send message to valid user with content	Transmit form, add data to database	Yes
Attempt to read message	Alter message to indicate "read"	Yes

Search:

Test Used	**Result**	**Correctness**
Empty search box	Prevent form transmission, inform user	Yes
Search using space delimited tags	Present all media that contains at least one of the tags	Yes

User Manual

Signing In / Registration : A User is directed to the Sign-In / Registration page immediately upon entering the site. Immediately after sign-in they are popped into the homepage.

Home Page : This website uses an iframe for displaying all Media Content. What this means is that a page refresh will always lead the User back here. The Home Page displays a list of Media ordered exclusively by how recently it was uploaded. Users may at any time click any of the imaging to view the Media Content and its metadata.

Side Bar : As the site uses an iframe, the side bar content does not currently dynamically refresh. This requires the user to refresh to view changes. The Side Bar allows users to:

- View all Media Content sorted under a specific category
- View all Media Content by their subscription users (or choose a specific User's content to view)
- View all Media Content they themself have uploaded.
    - They may also elect to upload new content. This requires the user make a title, description and select a category for the content, then select the file they wish to upload. They may also choose to add a thumbnail to the file.
- View all Media Content based on current Popular Tags
- View their Playlist Content

Media Page: When a User selects a piece of media, they will be presented with a page with the media and its metadata. On this page, the user may choose to "subscribe" to the content uploaded or add the content to a new or existing playlist of theirs. Below this, they can also choose to download the file through the link and further down leave a comment and view the comments of others.

Messaging: Messaging is handled through links in the top bar "Check Messages" and "Send Message". The former allows the user to view all the messages, read and unread, sent to them by others. Selecting any one of them takes them to a page with the contents of the message. To send a message, a user need only fill out the title, receiving user's name, and the contents of the message through the "Send Message" form.

Playlist: This page provides an almost identical experience to that of the **Media Page** except it provides the User the ability to remove the currently select Media from the current list as well as quickly cycle between content within the list.

Logout: Lastly, a user can logout from the site via the link at the top left, sending them back to the Sign-in/Registration page.