CPSC 462/662

Project Description

1. Overview

- Metube Project is to build a database-driven media-file sharing website similar to Youtube. But unlike YouTube system in which video is the only media type hosted, the content of MeTube system includes graphics objects, video, audio, images, and animation clips.

2. Requirement

- Graduate students should implement all the features. Undergraduate students can implement only the basic functions. Undergraduate students will get extra score if you implement the advance feature correctly. (Details are in the grading spreadsheet)

1) User account

- ***Basic functions:***
- A user can register for an account to use the Metube system.
- Password needs to be confirmed during registration.
- If you use email to register, the email format needs to be checked during the registration.
- A user can sign-in and update his/her personal profile, such as changing the password and other personal information.
- After logging in, a user can log out at any time.

2. Requirement

1) User account

- ***Advanced functions:***

A user has a contact list, a friend list and can manage these lists.

- A contact list is a list of users who you can contact with. You can add users to the contact list or remove users from this list.
- A friend list is a part of the contact list. Some users might in your contact list, but these users are not in your friends list because they are not your friends. You can add users to your friends list or remove users from your friends list.
- You can block some users so as to prevent them from adding you to their friend/contact lists. You can also remove the block.

2. Requirement

2) Data sharing

- ***Basic functions:***

- A signed-in user should be able to upload multimedia files into the Metube system. During uploading, the user should be able to input the meta-information about the multimedia file, including the title and description of the media file, and keywords used for searching the media file.
- Internet user should be able to download and view media files available in Metube system through a media player embedded in the web interface.

2. Requirement

2) Data sharing

- ***Advanced functions:***
- Sharing media files: a user can specify how to share the media file with others (for instance, share with everybody or just friends, allow discussion or not, allow rating or not, etc.)
- Blocking user: a user can block other users from viewing/downloading the media files he/she uploaded. A user can also remove the block.

2. Requirement

3) Media Organization

- ***Basic functions:*** all users should be able to browse the media files by categories. Signed-in users should be able to organize their uploaded media files and their interested media files in different ways, including channel, playlists, favorite lists.
- All media files uploaded by a user are organized into a broadcasting channel that other users can subscribe to. A user can subscribe to any channel created by another user or cancel a subscription.
- Users can also organize media files they viewed into playlists. A user can create many playlists. He can add medias to the playlist and remove medias from the playlists.
- A user can create a favorite list of media files. He/she can add his/her favorite medias to the favorite lists and remove medias from the favorite lists.

2. Requirement

3) Media Organization

- ***Advanced functions:***
- Show the most-viewed media files: ranking the medias by the number of times viewed by users.
- Show the most-recently uploaded files: ranking the medias by the upload date.

2. Requirement

4) User interaction

- ***Basic functions:***
- Send message: a registered user can send messages to other users and receive messages from others users.
- Comment: a user can comment a media file if the user who uploaded the media file enabled the discussion option for the media file. He/she can also remove the comment he/she made before.
- ***Advanced functions:***
- Media rating: a user can rate the media file he/she viewed by giving a score.
- Group discussion: a user can build a group and join in a group for making discussions. Once a user joins a group, he/she can start a discussion topic or post comments on a discussion topic.

2. Requirement

5) Search

- ***Basic functions:***
- Keyword-based search: a user can search the media file by keyword.
- ***Advanced functions:***
- Word cloud: there is a word cloud to show the keywords of the media files in the system.
- Media recommendation: when a user selects a media file to view, links to other related media files should be provided.
- Feature-based media search: a user can search the media through other features, such as media category, file size, upload date, etc.

3. Project resources

- **1. Web service:**
  - Your system should be posted and tested on the School of Computing lab machines. You are required to provide the link to your system (e.g. people.cs.clemson.edu/~USERID/u1/index.php) and can be accessed via web browser.
  - To access the web pages via terminal from ANY of our School of Computing lab machines:

    cd /web/home/$USERID/public_html

3. Project resources

- **1. Web service:**
  - Permissions for files with in the public_html directories are as follows:
    - PHP files just need to be readable by user (chmod 600 (rw-------)) in order for the web server to read them.
    - All other files (html, jpg, gif, etc) need to be readable by everyone (chmod 644 (rw-r--r--)) in order for the web server to read them. **In order to play or download the uploaded medias, the mode of these files should be 644.** Run the chmod command in your PHP script after the upload:

      chmod($file, 0644)
    - Directory listings are on by default. If you'd like the web server to generate directory listings, you need to make sure the desired directory is readable and searchable by all (chmod 755 (rwxr-xr-x)).

3. Project resources

- **1. Web service:**
  - The School of Computing lab machines also have Python available for use for development for the web pages. Put the following information on the files used for web pages, which are also placed in the /web/home/$USERID/public_html directory:
    - name the file with a .cgi extension
    - include the header: #!/usr/bin/env python
    - make sure permissions are chmod 711 (rwx--x--x)

3. Project resources

- **2. Database server:**
  - You can use https://buffet.cs.clemson.edu to create your own databases (MySQL or Postgres).
  - You can access the databases vim command line from any of the School of Computing lab machines.

4. Environment setting locally

- **Windows Installation:**
  - Install the latest version of **WAMP** (http://www.wampserver.com/en/) in Windows. Open your browser and type http://localhost/ to make sure that your apache runs. It will show "It works!" if your installation works.
  - After the installation, using one of the icons you created, or *Start –> All Programs –> WampServer –> start WampServer*, you can launch the management console of WAMP.

4. Environment setting locally

- **Linux Installation:**
  - Install the latest version of ***APACHE, MYSQL, PHP*** and ***phpmyadmin***.

4. Environment setting locally

- **Test and Run :**
  - Use any text editor to create a *hello.php* including the content *<?php echo "hello world";?>*

    Then put the *hello.php* file under path *WAMP_INSTALL_PATH/www/* in Windows, */var/www/* in Debian/Ubuntu. Next, in the browser, type http://localhost/hello.php to check if your PHP runs. If you read "*hello world*", this proves PHP works with apache.

    Test your other php files in the similar way.
  - Install and use *phpmyadmin* to check whether your MySQL runs well. You should activate your *phpmyadmin* under URL http://localhost/phpmyadmin in your browser.

Thank you!