CPSC 462: MeTube Technical Report

Team Number: U5

Team Members: Thomas Cannon

Date: April 25, 2014

CPSC 462, U5: Metube Technical Report

Team Number: U5
Name: Thomas Cannon
Username: tcannon
Assignment: CPSC 462 Project
Date: April 25, 2014

An important note about this group

There is no Team Evaluation in this submission because the team lost members due to circumstances with the class.

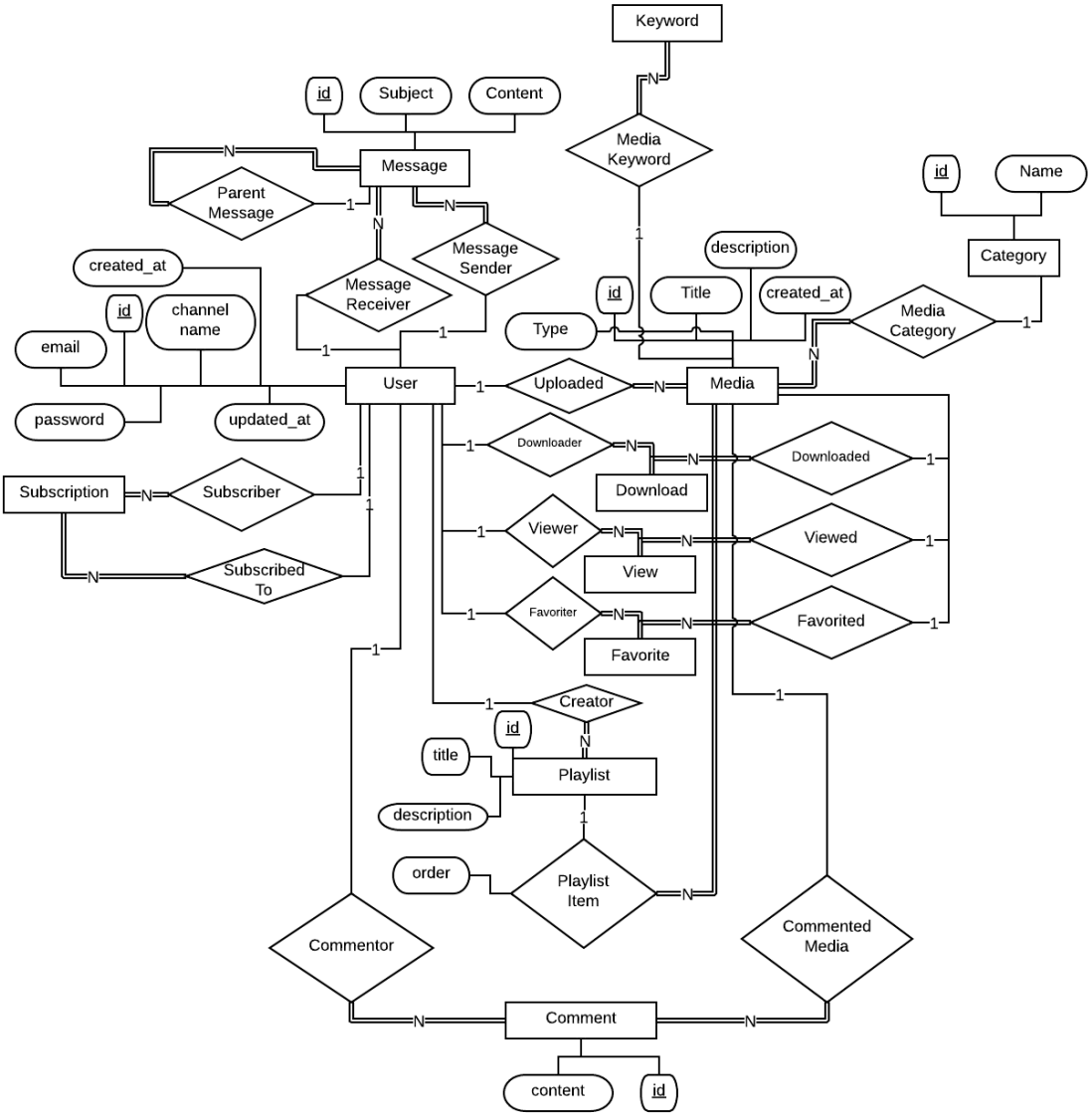
System design

MeTube is an implementation of a media sharing site that's similar to YouTube. Unlike YouTube, it allows photos and audio files to be uploaded alongside video. Anyone can view or download the media posted on MeTube. Anyone can also find media using keyword searches, categories, or by viewing a Channel's activity.

In MeTube, registered users are able to upload media and are considered "Channels". MeTube also allows users to favorite and leave comments on media, message other users in an email-like system, and subscribe to each other's "Channel". Users are also able to create and order playlists for their media, which are only viewable by the user that created them. After registering, users can update their email address, password, or channel name.

The interactions both users and visitors have with the site are recorded for future analytical purposes. In the initial implementation, the analytics are basic counts of views, favorites, and downloads of each media. However, this data should be collected in such a way to allow better analytical tools to be developed in the future.

ER Diagram



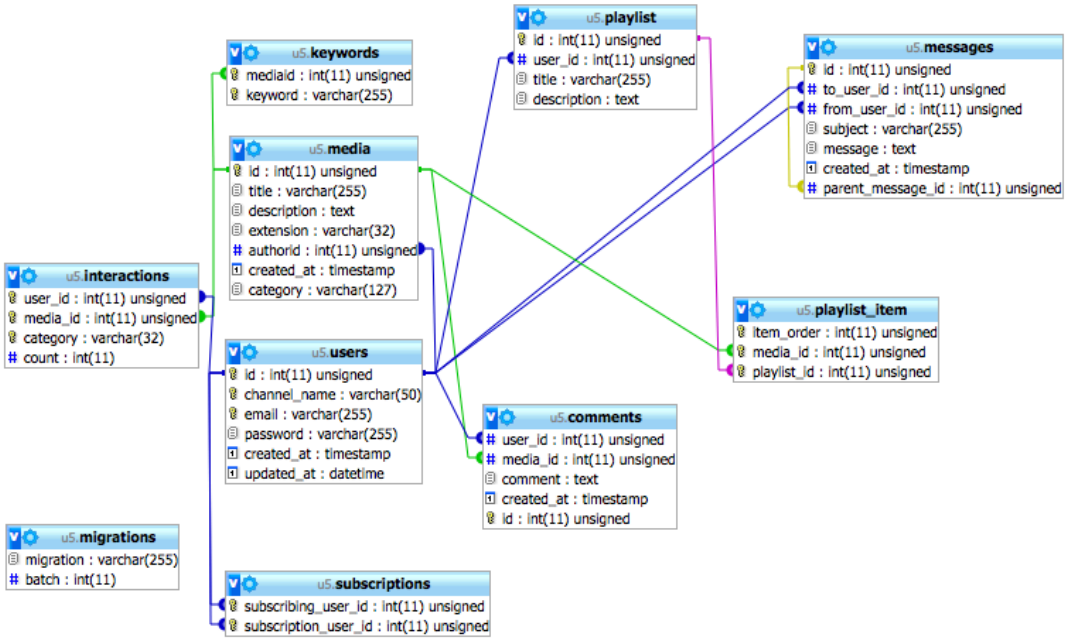
The ER Diagram for this implementation of MeTube

Database schema

The `migrations` table is created by the Database Migrations feature which is part of the Laravel Framework. This feature allowed me to build out my database with a series of scripts, which could be run at any time and managed through source control. The `migrations` table acts as metadata for this feature and is not used in the application itself.

It is also important to note that the `interactions` table is the implementation of the Favorite, View, and Download entities in the ER Diagram. I will discuss this further in the “Implementation Details” Section.

Below is a graphical representation of the database schema. The actual schema file is `database_schema.sql`, which is located in the root of the submission.



The Database Schema for this implementation of MeTube

Function design

In my experience, I've found the MVC (Model View Controller) design pattern to be particularly effective for designing web applications. It allows for a better separation of concerns between interacting with the database, rendering a user interface, and manipulating the application through GET and POST requests.

Because of this pattern the functionality of the application was divided into the following categories:

- **Models**: PHP Classes that would be used to interact with the database to generate "objects" to be displayed and manipulated throughout the application. Their responsibilities were:
    - Sanitize and validate input before the corresponding record is saved or updated in the database.
    - Generate human-readable error messages describing why the object could not be saved, which would be displayed as part of the user interface.
    - Provide methods to easily retrieve objects (or object properties) as needed. This includes tasks such as "getting all media for a category", or "counting all the downloads for a media".
- **Views**: Files built for a specific templating language that would render the HTML for the user interface. The views needed to satisfy the following requirements:
    - Involve as little "code" as possible (e.g: inserting, updating, collecting data from the application). The data required for the user interface should be collected by the Controller, or retrieved using very basic Model functions.
    - Be Reusable. Many parts of MeTube's functionality involve the same UI components. Rather than re-create each one, the templating language should allow developers to create a single "template" that can be used across the application as necessary. This also extends to "layouts" for the application, which consolidate the basic structure of the user interface into a single template.
    - Allow for simple control structures such as `if`, `elseif`, and `foreeach`. These are critical for conditionally rendering the view based on the current user and state of the application.
- **Controllers**: PHP Classes that are used to manipulate the application's state through Models based on user input and render the associated Views. Controllers should satisfy the following requirements:
    - There should be no direct interaction with the database. All database interactions should be handled through the Models.

    - Common code in a controller should not be repeated. Controllers often provide “before filters”, which allow for common functionality like “checking if the logged-in user has access to this page”. This makes it easier to update the application as time goes on.
    - Controller methods should only do one thing. This means there might be two controller methods for actions like form submissions. One method would be used in a GET request to render the form (with any validation errors), the other method would be accessed through a POST request and would attempt to use the Models to update the application (or redirect back to the GET action if something went wrong).
    - Controller actions shouldn’t be complex. They simply act as the intermediary between Views and Models, calling methods to change the application’s state.
- **Router**: A special file used to map URLs in the application to Controller actions. This is how the application knows which actions to perform when certain URLs are called. A route specifies the HTTP method required, the pattern for the URL, the ‘name’ of the route (which used in views and controllers for redirection and URL generation), and any filters on the route. These filters allow for basic actions to be performed before the controller is called, such as checking if the user is logged in before accessing a protected part of the site.

It is crucial that the media on MeTube be accessible from as many browsers or devices as possible. Given the complexity of displaying video and audio on the web due to codec issues, a lot of research was necessary to find the right process for displaying audio and video.

Given the scale of the project, development needed to be manageable. Therefore a common “platform” for development and source control was established to prevent last-minute errors or lost work.

Implementation details

This version of MeTube is implemented with a modified version of the Laravel Framework (http://laravel.com/). These modifications were necessary to make the site work on the mmlab servers.

Models and database interaction

The Model classes created to interact with the database are stored in `www/app/models`. Most of the classes in this directory are directly mapped to database tables, however there are two exceptions:

- Category.php: Categories are not defined in the database. Instead, this class acts as a wrapper for the constant strings that make up the “id” and human-readable name for each category.

- FileConverter.php: This class is solely responsible for managing the process of converting files to display correctly on MeTube, checking that the files have valid extensions, and returning the media type (video, audio, image) for a certain file extension.

The Database is interacted with through Laravel's Database class (which is a nice wrapper for PHP's default PDO library). The queries to interact with the database were hand-written, and use database parameters to avoid SQL injection.

Views and Controllers

The Views for the application are located in `www/app/views`. Generally they are organized according to the entities they represent. The views are rendered using the Blade Templating Engine, which is included in Laravel by default.

The Controllers for the application are located in `www/app/controllers`. They follow the standard rules set in the Function Design section. Special "before filters" are used for actions that require an entity to exist or that the logged-in user has access to the entity. Redirects and Laravel's built in support for rendering "404" pages are used to manage the request flow through the application.

Routing (mapping URLs to Controller actions)

The Router for the application is located in `www/app/routes.php`. The Router file is well documented in Laravel's documentation, but the general idea is that each line represents a GET or POST request, tying a URL in the application to a specific controller action. The URL can contain "variables", which can be used in the Controller actions. This is how a GET request for `/media/1` is mapped to `MediaController@show`, passing the "1" as the "ID".

The routes are named, and some include the `'before' => 'auth'` option. This indicates that the route is protected, meaning that only logged in users can access the resource. This filter is defined in `www/app/filters.php`, and is part of the larger interface Laravel provides for Authentication.

Some of the routes use anonymous functions, which are a new feature in PHP 5.3, to indicate which view should be rendered. However, most of the routes are directly mapped to Controller actions.

Data validation and integrity

Validations on user input and the application state were all implemented on the server side to ensure the application would always work as expected, even if the user has Javascript disabled. If there was an issue creating a new entity, the associated Model would generate a human-readable error message to be passed back to the form through a redirect.

In order to prevent SQL injections, Laravel's built-in database library was used. This is a nice wrapper to PDO, which is a built-in PHP library that replaces the `mysql_*` functions previously provided. It allows for the use of SQL parameters, which reduce the possibility of SQL injection by passing the user input through the library. This automatically escapes any malicious characters and explicitly tells MySQL not to interpret the text as SQL commands.

Before filters on the Controller were used to ensure that users did not attempt to access other user's information. This prevents one user from reading another's messages, updating their profile information, modifying their playlists, etc.

User-provided data for an entity is also escaped in the View code. This is to prevent XSS attacks, where the user could include malicious Javascript in the input fields to modify the site's behavior.

The Interactions table

In the ER Diagram, there are 3 distinct entities that can be described as "Interactions" between users and media: Views, Downloads, and Favorites. The data structure for these 3 entities are extremely similar. To keep the database size small and reduce complexity, I designed the `interactions` table to encompass all 3 types of interactions (with support for more interactions in the future). The table has the following structure:

- **user_id**: a foreign key relation with a record in the `users` table. Represents a single user. Note that this field can be `NULL`, indicating that the "user" in this case is a visitor (not logged in)
- **media_id**: a foreign key relation with a record in the `media` table. Represents a single media
- **category**: the type of interaction recorded between the user and media. The current values are: "downloaded", "viewed", and "favorited"
- **count**: an integer that is incremented by the Model, indicating how many times the interaction has been performed.

The primary key for this table is the unique combination of the `user_id`, `media_id`, and `category`.

With this structure it is very easy to determine how many times an action has been performed, and querying for overall analytics on the interaction is simple. For example, this is a query to get the count of all the downloads for the Media with `id = 123`:

```
SELECT SUM(count) as count FROM interactions WHERE category='downloaded' AND media_id = 
```

Displaying Media on as many browsers as possible

Because different browsers support different video and audio filetypes and codecs, a generalized solution needed to be created. This involved a two-pronged approach:

- Use FFMPEG to convert uploaded audio and video files to web-compliant MP3s and MP4s (respectively)
- Find a player that would provide the greatest browser support for MP3 and MP4 files

The `FileConverter` Model provides the link to FFMPEG as part of its goal to abstract processing and moving uploaded files. It checks the filetype to ensure it's valid, and if the file is an audio or video file it calls the `ffmpeg` command to convert the file to the proper format with the correct codecs.

MediaElement.js proved to be the best video and audio player for this project. It hooks into the HTML5 `<audio>` and `<video>` tags to create a player that is easy to stylize. If these tags are not supported or the browser cannot play the audio or video, MediaElement.js provides a Flash player fallback that attempts to use the same styling. The Flash player fallback is a little buggy but it allows almost every browser to view the media on MeTube.

Development environment and workflow

There were 3 overall goals when it came to the development environment used to create MeTube:

- Ensure the code would work in production by making the development environment match the "production" (mmlab) environment 1:1
- Prevent accidents with the the database schema by formalizing its structure and modifications
- Stop people from stepping on each other's toes when working on the codebase.

The first goal was accomplished by using Vagrant. Vagrant allows developers to create a headless VM based on a series of scripts, which is used as to "execute" the application's code. This allows the application to run in an isolated environment while allowing developers to edit code in their preferred environment. It also allowed me to setup a server that almost exactly matched the "mmlab" server, which helped me catch a number of bugs during the development process.

The database tables were exclusively created and updated using Migrations. This allows developers to write simple scripts to modify the database. These scripts are checked into source control. This ensures the scripts are run in the correct order (since each script is timestamped), and that the changes made to the database are easily traceable. Furthermore, migrations can usually be "rolled back", giving developers the ability to "undo" changes to the database. You can read more about Migrations in Laravel's Documentation. The migration scripts used hand-written queries.

Finally, the code was maintained using the Git version control system. The repository was hosted on GitHub, and dploy.io was used to automatically deploy the application from the GitHub repository.

Additional dependencies

The application is currently hard coded to exist in `/spring14/u5/` on the server. You can change the site to work in another location by finding and replacing all instances of this string.

The application uses MediaElement.js to provide a Flash player fallback for audio and video files (http://mediaelementjs.com/). Your server should be able to serve Flash files.

The application originally used the Composer package manager (https://getcomposer.org/) to manage the packages required for the site. However, for the interest of making the site easier to grade and test, the package files have been "vendorized" (stored in the `www/vendor/` directory). If the site is acting up however, you can follow the instructions for installing Composer (available on the Composer site). After Composer is installed, simply run `composer install` from the `www/` directory to install the missing packages.

Additionally, I used Vagrant (http://www.vagrantup.com/) to create a headless VM to run the application in an isolated environment. This was to make sure the development environment matched the mmlab server as closely as possible. If you have any issues running the project I'd be more than happy to give you access to the full Github Repository (which includes the code necessary to create the Vagrant VM).

Test Cases and Results

Test Case 1

Users should be able to register in MeTube and update their profile information

Steps:

1. Click "Sign Up" from the Home Page
2. Fill out the fields and click "Register"
3. Confirm that the User has been signed into MeTube by checking that the "Sign Up" button has changed to "Sign Out"
4. Sign out of MeTube, and Sign back in with the User's credentials
5. Click on "Edit Profile" under the "User" section of the navigation
6. Change the Channel Name
7. Click on "Update Profile"
8. Click on "Profile" to confirm the Channel Name has been changed

Status: Passed

Test Case 2

The registration and profile update forms should validate the data provided

Steps:

1. Click “Sign Up” from the Home Page
2. Only fill out some of the fields and click “Register”
3. Confirm that the form has not been submitted and that validation errors have been printed
4. Sign into MeTube using a valid account
5. Click “Edit Profile” and empty out the Channel Name
6. Click “Update Profile” and confirm the form has not been submitted and that validation errors have been printed
7. Attempt to change the password but use an incorrect password for the “Current Password”
8. Click “Update Profile” and confirm the form has not been submitted and that validation errors have been printed
9. Attempt to change the password but use different passwords for the password and confirmation fields
10. Click “Update Profile” and confirm the form has not been submitted and that validation errors have been printed
11. Attempt to change the password but make the password and/or confirmation fields blank
12. Click “Update Profile” and confirm the form has not been submitted and that validation errors have been printed

Status: Passed

Test Case 3

Confirm that Users cannot sign in with bogus credentials

Steps:

1. Click “Sign In” from the Home Page
2. Use a bogus email and password
3. Confirm that the form has not been submitted and that the User has not been signed into MeTube

Status: Passed

Test Case 4

Users can upload and edit Media

Steps:

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. Confirm that the “Upload” button does not exist in the navigation
3. Confirm that `/media/new` redirects to the Sign In page
4. Sign into MeTube and click “Upload”
5. Fill out the form, making sure to choose a valid image file that is less than 20 MB in size
6. Click “Upload Media”
7. Confirm that the Media show page is displayed, with the correct Media uploaded and available
8. Click “Edit Media”
9. Modify the title, description, and keywords for the Media
10. Click “Update Media”
11. Confirm the Media has been updated

Status:Passed

Test Case 5

Only supported file types are allowed to be uploaded when uploading Media

Steps

1. Sign into MeTube and click “Upload”
2. Fill out the form, choosing a file that is less than 20 MB in size and is not an image, video, or audio file
3. Click “Upload Media”
4. Confirm that the form was not submitted, and that the error message “Filetype not supported” appears at the top of the form

Status: Passed

Test Case 6

Users cannot edit Media they did not upload

Steps

1. Sign into MeTube
2. View a Media that has been uploaded by another User
3. Confirm that the “Edit Media” button is not present
4. Append `/edit` to the URL for viewing the Media and attempt to navigate to the page
    - e.g: `http://mmlab.cs.clemson.edu/spring14/u5/media/1/edit`
5. Confirm that the request has been redirected to the home page

Status: Passed

Test Case 7

Visitors and Users can download and view Media

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. Click on one of the thumbnails for a Media on the homepage
3. Confirm that the Media is being displayed correctly
4. Confirm that you can read the title, description, Comments, and how many views/favorites/downloads the Media has
5. Confirm that the file correctly downloads on your system when you click “Download”
6. Sign into MeTube
7. Repeat Steps 2–5

Status: Passed

Test Case 8

Only Users can favorite Media, comment on it, or add it to a Playlist

Steps

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. Click on one of the thumbnails for a Media on the homepage
3. Confirm that the “Favorite” and “Add Playlist” buttons are not present
4. Confirm that the Comment Box and “Add Comment” button are not present
5. Sign into MeTube
6. Repeat Step 2
7. Confirm that the “Favorite” and “Add Playlist” buttons are present
8. Confirm that the Comment Box and “Add Comment” button are present
9. Confirm that clicking “Favorite” increases the favorite count for the Media
10. Fill in the Comment box and click “Add Comment”
11. Confirm the Comment has been added to the Media
12. Confirm that choosing a Playlist and clicking “Add To Playlist” adds the item to the Playlist
    - Note: You may need to create a Playlist before running this test (See Test Case 9)

Status: Passed

Test Case 9

Playlists can be created, edited, and deleted

Steps

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. Confirm that the “Create Playlist” link does not exist in the navigation
3. Confirm that `/playlists/new` redirects to the Sign in page
4. Sign into MeTube
5. Click on the “Create Playlist” link in the navigation
6. Fill out the form and click “Create Playlist”
7. Confirm that the Playlist has been created, and the Playlist title is shown in the navigation
8. Click on the “Edit Playlist” button when viewing the Playlist
9. Change the values in the form, then click “Update Playlist”
10. Confirm the Playlist title and description have been changed
11. Record the URL for the Playlist
12. Click “Delete Playlist”
13. Confirm the Playlist no longer exists, that the URL for the Playlist returns a 404 page, and that the Playlist title is no longer in the navigation

Status: Passed

Test Case 10

Playlists Items can be ordered and deleted

Steps

1. Sign into MeTube
2. Create a Playlist if one does not already exist (See Test Case 9)
3. Add a Media to the Playlist (See Step 12 in Test Case 8)
4. Repeat Step 3 for 2 different Media, adding 3 total items to the Playlist
5. View the Playlist
6. Confirm that all the Media in the Playlist have the up, down, and delete buttons in their preview box
7. Confirm that clicking the up button on the topmost item does not change the Playlist ordering
8. Confirm that clicking the down button on the bottommost item does not change the Playlist ordering
9. Confirm that clicking the up button on the middle and bottommost items moves them up the Playlist order
10. Confirm that clicking the down button on the middle and topmost items moves them down the Playlist order
11. Confirm that clicking the delete button on each item in the Playlist removes it from the Playlist and updates the Playlist order

Status: Passed

Test Case 11

Visitors and Users can view the Channel and activity of other Users

Steps

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. View a Media and click on the author's name
3. Confirm the Channel page for the Media's Author is displayed
4. Confirm the 4 most recently uploaded Media are displayed on the Channel
5. Confirm that the "View More" button shows all the Media uploaded by the User
6. Confirm that the "Uploaded", "Downloaded", "Viewed", and "Favorited" links all work as expected
7. Sign into MeTube
8. Repeat Steps 2–6

Status: Passed

Test Case 12

Users can subscribe to another Channel, view their 4 most recently uploaded Media from "My Subscriptions", and unsubscribe from a Channel

Steps

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. View a Media and click on the author's name
3. Confirm that the "Subscribe" button is not present
4. Sign into MeTube
5. Repeat Step 2
6. Click the "Subscribe" button
7. Confirm that the "Subscribe" button has been replaced with the "Unsubscribe" button
8. Click on "My Subscriptions"
9. Confirm that the Channel you have just subscribed to is present in the list of Subscriptions
10. Confirm the Channel's 4 most recently uploaded Media are present in the Subscription view
11. Confirm that the "View More" button shows all the Media uploaded by the Channel
12. Confirm that the "Unsubscribe" button is present in the Subscription view
13. Click "Unsubscribe"
14. Confirm the User has unsubscribed from the Channel by clicking on "My Subscriptions" and confirming the Channel is no longer present

Status: Passed

Test Case 13

Users can send a Message to another User, view the Messages sent to them, and view the Messages they have sent.

Steps

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. Confirm that the “Messages” link does not exist in the navigation
3. Confirm that `/messages` redirects to the Sign in page
4. Confirm that `/messages/new` redirects to the Sign in page
5. Confirm that `/messages/sent` redirects to the Sign in page
6. Sign into MeTube on two separate browsers using two separate Users
7. Click on “Messages” on the first browser
8. Click on “New Message” on the first browser
9. Fill out the form and click “Send Message” on the first browser
10. Click on “Sent Messages” on the first browser
11. Confirm that the Message you just wrote is in the list of Sent Messages on the first browser
12. Click on the envelope to view the Message on the first browser
13. Confirm the Message is correctly displayed on the first browser
14. Click on “Messages” on the second browser
15. Confirm that the Message you wrote in the first browser is in the Inbox on the second browser
16. Click on the envelope to view the Message on the second browser
17. Confirm the Message is correctly displayed on the second browser

Status: Passed

Test Case 14

Users can reply to a Message, and viewing the reply shows the Message thread

Steps

1. Follow Steps 6–9 in Test Cast 13 to sign for two different Users and create a Message
2. Click on “Messages” on the second browser
3. Confirm that the Message you wrote in the first browser is in the Inbox on the second browser
4. Click on the envelope to view the Message on the second browser
5. Click “Reply” on the second browser
6. Fill out the form and click “Send Reply” on the second browser
7. Confirm that the Message was created on the second browser, and that the previous Message is shown above the reply you just created

8. Click on “Messages” on the first browser
9. Confirm that the reply is in the Inbox of the first browser
10. Click on the envelope to view the reply on the first browser
11. Confirm that the reply is displayed correctly on the first browser, and that the previous Message is shown above the reply you just created
12. Confirm that the “Reply” button exists when viewing the reply

Status: Passed

Test Case 15

Users can delete Media, which deletes it from any Playlist the Media was included in.

Steps

1. Sign into MeTube
2. Create a Media (see Test Case 4)
3. Add the Media to a Playlist (See Step 12 in Test Case 8)
4. View the Media
5. Record the URL of the Media
6. Click on “Delete Media”
7. Confirm that the Media is no longer present on the site and that the URL for the Media returns a 404 page
8. View the Playlist
9. Confirm the Media has been removed from the Playlist

Status: Passed

Test Case 16

Users can quickly view the Media they have uploaded, downloaded, viewed, and favorited

Steps

1. Sign into MeTube
2. Confirm that the “Uploaded”, “Downloaded”, “Viewed”, and “Favorited” links exist in the navigation
3. Confirm that the “Uploaded” link only shows Media the currently signed in User has uploaded
4. Confirm that the “Downloaded” link only shows Media the currently signed in User has downloaded
5. Confirm that the “Viewed” link only shows Media the currently signed in User has viewed
6. Confirm that the “Favorited” link only shows Media the currently signed in User has favorited

Status: Passed

Test Case 17

Visitors and Users can browse Media by category, both from the navigation and the home page.

Steps

1. Confirm you are using MeTube as a Visitor (Not signed in)
2. Confirm that the links for each category exist in the navigation
3. Confirm that the 4 most recently uploaded Media for the category are shown on the home page, along with a “View More” button
4. Confirm that the link on the navigation and the “View More” button for each category present a list of all the Media that is under that category
5. Sign into MeTube
6. Repeat Steps 2–4

Status: Passed

Test Case 18

Visitors and Users can search for Media by keyword using the search bar at the top of the site

Steps

1. Create 2 Media, sharing a keyword between them (See Test Case 4)
2. Confirm you are using MeTube as a Visitor (Not signed in)
3. Confirm that the search bar is present in the navigation
4. Type in the Keyword that is shared between the 2 Media you created
5. Click on the Search Button
6. Confirm that only the 2 Media that share the keyword are displayed
7. Sign into Metube
8. Repeat Steps 3–6

Status: Passed

Test Case 19

The “Upload” and “Edit Media” forms should validate the data provided

Steps

1. Sign into MeTube
2. Click “Upload”
3. Do not fill in the title or choose a file to upload

4. Click “Upload Media”
5. Confirm that the form did not submit and that error messages are present on the form
6. Create a Media (See Test Case 4)
7. View the Media and click “Edit Media”
8. Empty out the title of the Media
9. Click “Update Media”
10. Confirm that the form did not submit and that error messages are present on the form

Status: Passed

Test Case 20

The “Create Playlist” and “Edit Playlist” forms should validate the data provided

Steps

1. Sign into MeTube
2. Click “Create Playlist”
3. Do not fill in the title or description
4. Click “Create Playlist”
5. Confirm that the form did not submit and that error messages are present on the form
6. Create a Playlist (See Test Case 9)
7. View the Playlist and click “Edit Playlist”
8. Empty out the title and description of the Playlist
9. Click “Update Playlist”
10. Confirm that the form did not submit and that error messages are present on the form

Status: Passed

Test Case 21

The “New Message” and “Reply” forms should validate the data provided

Steps

1. Sign into MeTube
2. Click “Messages”
3. Click “New Message”
4. Do not fill out the subject or message fields
5. Click “Send Message”
6. Confirm that the form did not submit and error messages are present on the form
7. Send a Message to the currently signed-in User (See Test Cast 14)
8. View the Message and click “Reply”
9. Do not fill out the subject or message fields

10. Click “Send Reply”
11. Confirm that the form did not submit and error messages are present on the form

Status: Passed

Test Case 22

The Comment Box must require a comment

1. Sign into MeTube
2. Create a Media (See Test Case 4)
3. View the Media
4. Do not fill in the Comment Box
5. Click “Add Comment”
6. Confirm that the Comment was not added and error messages are present above the Comment Box

Status: Passed

Test Case 23

Users should be able to sign out of MeTube at any time

Steps

1. Sign into MeTube
2. Confirm that the “Sign in” button has changed to “Sign out”
3. Click “Sign out”
4. Confirm you are using MeTube as a Visitor (Not signed in)

Status: Passed

Test Case 24

Users should be able to upload video, audio, and images

Steps

1. Sign into MeTube
2. Create 3 different Media: 1 video file, 1 audio file, and 1 image (See Test Case 4)
3. Each time, confirm that the Media has been successfully created
4. Each time, confirm that the Media can be correctly viewed on a number of browsers

Status: Passed

Test Case 25

Users should not be able to upload a file over 20 MB

Steps

1. Sign into MeTube
2. Click on “Upload”
3. Fill out the form, choosing an image, audio file, or video file that is over 20 MB
4. Click “Upload”
5. Confirm that the form is not submitted, and that an error message explaining the file size is too big is displayed above the form

Status: Passed

Test Case 26

Users should not be able to view the Playlists of other Users

Steps

1. Sign into Metube on two different browsers for two different Users
2. Create a Playlist on the first browser (See Test Case 9)
3. View the Playlist on the first browser
4. Copy the URL for the Playlist from the first browser to the second browser
5. Confirm that the second browser cannot access the Playlist, and is redirected to the home page

Status: Passed

Test Case 27

Users should not be able to view Messages they have not sent or received

Steps

1. Create 3 different Users (see Test Case 1)
2. Sign into MeTube on two different browsers for Users #1 and #2
3. Send a Message from User #2 to User #1 (See Test Cast 14)
4. Sign out as User #2 and Sign in as User #3 (the second browser)
5. View the Message as User #1 (The first browser)
6. Copy the URL for the Message from the first browser to the second browser
7. Confirm that the second browser cannot access the Message, and is redirected to the home page

Status: Passed

User Manual

Visitors to the site

Visitors to the site are able to:

- View any Media on the Home Page
- Search for Media using the search bar at the top
- Browse Media by category by clicking on the desired category in the “Categories” section of the navigation
- View Media by clicking on the preview thumbnail for the Media
- View the Comments on a particular piece of Media
- View a User’s “Channel” (The Media they have Uploaded), along with the Media they have viewed, downloaded, and favorited
- Register for a MeTube account by clicking “Register” and filling in the form
- Sign in to their MeTube account by click “Sign in” and filling in their User credentials

MeTube Users

MeTube Users are allowed to upload, edit, favorite, and comment on Media. They can also manage Playlists and Subscriptions, Message other Users, and update their profile information. Users are allowed to perform any of the actions a visitor can, except for Signing in and Registering. Users are also able to sign out of MeTube at any time by clicking “Sign out”.

Uploading, editing, and destroying Media

- A User can upload Media by clicking “Upload” and filling out the form, choosing a Media file that is under 20 MB. If the Media file is not supported or there was a problem uploading the Media, the User is notified through an error message and is allowed to try again.
- A User can edit any Media that they have uploaded. This allows them to update the title, category, and keywords for the Media. To edit a particular Media, the User must view it by clicking on its preview thumbnail and then clicking “Edit Media”
- A User can delete any Media that they have uploaded. This will also delete any Comments, remove the Media from any Playlists in MeTube, and remove any interactions data. To delete a particular Media, the User must view it by clicking on its preview thumbnail and then clicking “Delete Media”
- Users can **only** edit or delete Media they have uploaded. If a User attempts to modify another User’s Media, they are redirected back to the Home Page.

Favoriting and commenting on Media

- Media can be favorited by clicking on the “Favorite” button when viewing a particular Media
- A User can Comment on a Media by scrolling to the bottom of the page and filling out the Comment form, then clicking “Add Comment”. Comments cannot be deleted, so be careful what you post!

Creating and managing Playlists

- Playlists can be created by clicking on the “Create Playlist” link at the top of the “Playlists” section of the navigation.
- A Playlist requires a title and a description to be created.
- Playlists can be viewed by clicking on the Playlist name in the “Playlists” section of the navigation.
- A Playlist’s title and description can be edited by clicking on “Edit Playlist” when viewing a Playlist. Similarly, a Playlist can be deleted by clicking on “Delete Playlist” when viewing a Playlist
- To add Media to the Playlist, just select the Media from the list of Playlists when viewing a Media and click “Add To Playlist”. This will redirect you to view the Playlist you added the Media to.
- The items in the Playlist can be ordered or deleted by using the buttons on the right side of each Media’s preview box. These buttons allow items to be moved up in the Playlist order, down in the Playlist order, or to remove this item from the Playlist altogether.
- A User can **only** view the Playlists they have created. They are not able to view or modify the Playlist of another User. If they attempt to view or modify another User’s Playlist, they are redirected to the Home Page.

Managing subscriptions

- A User can subscribe to other Users, which allows them to quickly view the last 4 uploaded Media by those Users by clicking “My Subscriptions” in the navigation.
- To subscribe to a User, the User must first view their “Channel” by clicking on their Channel name. Once the User is viewing the Channel, they can click the “Subscribe” or “Unsubscribe” button at the top of the Channel.
- While the “My Subscriptions” view only shows the 4 most most recently uploaded Media for each Channel the User has subscribed to, they can click “View More” to view the User’s Channel. Furthermore, the User can unsubscribe to a particular Channel by clicking “Unsubscribe” for a particular Subscription.
- A User can **only** view their Subscriptions. They do not have the ability to view other User’s Subscriptions, or who is subscribed to them.

Messaging Users

- A User can view their Inbox, read Messages, view their Sent Messages, and write a new Message through the Messaging section of the site. This is accessed by clicking on “Messages” in the “User” section of the navigation.
- The first thing a User sees in the Messages section is their Inbox. They can view a particular Message by clicking on the envelope or clicking on the Message’s subject. They can also view the Channel of the sender.
- A User can click on “Sent Messages” to view all the Messages they have sent. Similarly to the Inbox, they can view a particular Message by clicking on the envelope or clicking on the Message’s subject, along with the Channel of the receiver.
- When a User views a particular Message, its parent Messages are displayed as well to create a Message “thread”. This is to help Users make sense of replied Messages. Users can Create a reply to the Message being viewed by clicking on “Reply”.
- When replying to a Message, the subject and Message must be provided. The User is not able to specify who will receive the Message, since it will be sent to the sender of the Message they are replying to.
- To create a new Message, Users click on “New Message”. This form allows them to specify who will receive the Message, along with its subject and message. The subject and message must be provided.
- A User can **only** view the Messages they have sent or received. They are not able to view another User’s Messages, Inbox, or Sent Messages. If they attempt to view another User’s Messages, they are redirected to the Home Page.
- Messages cannot be deleted, so be careful what you write!

Updating profile information

- A User is able to update their profile information at any time. This allows them to change their Channel name, the email address associated with their account, and their password.
- In order for a User to change their password, they must first provide their current password.
- A Channel name and email address are required when a User is updating their profile information.
- A User can **only** update their own profile information. They are not allowed to update the profile information for another User.

Browsing the User’s activity

- The User is able to quickly browse the Media they have Uploaded, Downloaded, Viewed, and Favorited by using the links in the “User” section of the navigation.
- This information is also accessible by viewing a User’s Channel page.