Web Application development

Design and implement a web based application

Richard Gent

[Year]

Contents

[Table of figures 4](#_Toc481325887)

[Acknowledgements 5](#_Toc481325888)

[Introduction 6](#_Toc481325889)

[Background reading 7](#_Toc481325890)

[Common themes in online video services 7](#_Toc481325891)

[List of features to include 9](#_Toc481325892)

[Core Features 9](#_Toc481325893)

[Optional features 9](#_Toc481325894)

[User Feedback on current applications 10](#_Toc481325895)

[Table Designs 12](#_Toc481325896)

[Data Dictionary 12](#_Toc481325897)

[First Normal form 14](#_Toc481325898)

[Video Table 14](#_Toc481325899)

[User Table 14](#_Toc481325900)

[Second normal form 16](#_Toc481325901)

[Video table 16](#_Toc481325902)

[Comments table 16](#_Toc481325903)

[Rating Table 17](#_Toc481325904)

[User table 17](#_Toc481325905)

[Favourites table 17](#_Toc481325906)

[History table 17](#_Toc481325907)

[Subscribed to table 18](#_Toc481325908)

[Flagging table 18](#_Toc481325909)

[Logical level 19](#_Toc481325910)

[Database folder diagram 19](#_Toc481325911)

[Site Topology 20](#_Toc481325912)

[User mind map 21](#_Toc481325913)

[Entity relationship diagram 22](#_Toc481325914)

[Storyboards 23](#_Toc481325915)

[Homepage 1 23](#_Toc481325916)

[Homepage 2 24](#_Toc481325917)

[User page 25](#_Toc481325918)

[Video Page 26](#_Toc481325919)

[Subfolder name ideas 27](#_Toc481325920)

[Security features 28](#_Toc481325921)

[Pseudo code 29](#_Toc481325922)

[Webpages: 29](#_Toc481325923)

[Server Side Scripts 31](#_Toc481325924)

[Login Script 31](#_Toc481325925)

[File upload script 32](#_Toc481325926)

[Registration script 33](#_Toc481325927)

[Functions Script 34](#_Toc481325928)

[Application and User testing evaluation 35](#_Toc481325929)

[Self-testing review 35](#_Toc481325930)

[User testing/feedback review 38](#_Toc481325931)

[Self-review of application 40](#_Toc481325932)

[Core Features 40](#_Toc481325933)

[Additional features 40](#_Toc481325934)

[Core features 41](#_Toc481325935)

[Additional features 46](#_Toc481325936)

[Table designs 49](#_Toc481325937)

[Front end design 50](#_Toc481325938)

[Security Design 51](#_Toc481325939)

[Final Evaluation 52](#_Toc481325940)

[Admin manual and Maintenance guide 53](#_Toc481325941)

[Front end website tools and functions 53](#_Toc481325942)

[Homepage 53](#_Toc481325943)

[View page 54](#_Toc481325944)

[User page 55](#_Toc481325945)

[Sub page 56](#_Toc481325946)

[User Settings page 57](#_Toc481325947)

[Video settings page 58](#_Toc481325948)

[History and favourites page 59](#_Toc481325949)

[Video upload page 60](#_Toc481325950)

[Register page 61](#_Toc481325951)

[Forgotten password page 62](#_Toc481325952)

[Admin page 63](#_Toc481325953)

[Back end maintenance 66](#_Toc481325954)

[DBConnect Script 66](#_Toc481325955)

[Functions script 67](#_Toc481325956)

[Search function 69](#_Toc481325957)

[User registration 70](#_Toc481325958)

[File uploading script. 71](#_Toc481325959)

[Login script. 73](#_Toc481325960)

[New password scripts 74](#_Toc481325961)

[File host setup 76](#_Toc481325962)

[phpMyAdmin database 78](#_Toc481325963)

[CRON jobs 80](#_Toc481325964)

[Bibliography 81](#_Toc481325965)

[Appendix 82](#_Toc481325966)

[Testing log 82](#_Toc481325967)

[User testing and feedback 100](#_Toc481325968)

# Table of figures

* Figure 1 provided by the author
* Figure 2 provided by the author
* Figure 3 provided by the author
* Figure 4 provided by the author
* Figure 5 provided by the author
* Figure 6 provided by the author
* Figure 7 provided by the author
* Figure 8 provided by the author
* Figure 9 provided by the author
* Figure 10 provided by the author
* Figure 11 provided by the author
* Figure 12 provided by the author
* Figure 13 provided by the author
* Figure 14 provided by the author
* Figure 15 provided by the author
* Figure 16 provided by the author
* Figure 17 provided by the author
* Figure 18 provided by the author
* Figure 19 provided by the author
* Figure 20 provided by the author
* Figure 21 provided by the author
* Figure 22 provided by the author
* Figure 23 provided by the author
* Figure 24 provided by the author

# Acknowledgements

* I would like to acknowledge John Myers who explained with certain functions of PHP scripting
* John Myers, Alex Colley and Zack Friel for user testing and feedback
* Terry Wilding and Douglas Parr for further testing and feedback
* Rotherham College of Arts and Technology for anonymously testing and giving feedback for the application.

# Introduction

This document is to show the design and thought process behind the web application that we are developing. The web application in this case is an online video playing/streaming service, akin to other web applications such as Netflix, YouTube or Daily Motion. The idea here is to make a service that can improve on the formula that makes a good video streaming service. A brief overview of the application we are creating will be included along with logical and physical representations of what the application should do and will display when being used.

# Background reading

## Common themes in online video services

When creating an online video service, it is imperative that the experience is as easy as possible for a viewer to find and watch a video within a few seconds. From personal experience I found that free online video services such as YouTube or Twitch will highlight current “trending” videos that are currently being watched or favored by users. From there we can select these videos and instantly start to watch them. In some cases there may be advertisements before you can view the video, depending on what the video is and how popular it is. This will be taken into account when designing this application, which videos are popular and having the ability to instantly start viewing them from the front page. Paid for services such as Netflix or Amazon Prime use this same technique. However they will show you a preview of the video before asking for your membership sign in. Since we are potentially creating a service where the user has to sign in to access and upload content, this type of approach may also be used.

The video players in most video streaming services offer the ability to resize the video player and be able to alter the quality at which the video streams at. YouTube, Twitch, Netflix, DailyMotion, Amazon prime video and Sky Go all offer the ability to alter these settings. Some services however such as Netflix or Amazon, use an automatic detector to determine what the video should be streamed at. On a personal note, I prefer the ability to choose a streaming option, due to either poor internet or the device you are using. Therefore I propose that the option to be able to alter the quality of a video would be more appropriate for a video player.

All of the web applications previously mentioned, also have the ability to resume a video from where a user last left it. Of course to be able to do this, the user must have an account, otherwise the data will not be saved. For the free video service providers, this is a clever way of getting users to create accounts with them, so for example, they can be used for targeted advertising when watching a video. For pay to watch subscription services, this comes as standard, along with a possibility of removing advertisements as part of the subscription. Accounts within the web application we are designing could have features such as this, in order to make the experience of using the application better.

As for video uploads by users, this can vary from service to service. For example, YouTube does allow for users to upload their own content, add annotations to the videos, add descriptions, thumbnails, comments, ratings etc. However it does have community guidelines and policies in place to avoid sensitive content. No nudity, graphic, dangerous or hateful content can be posted (YouTube, 2017). Any spam or outright advertisement videos are also banned along with tags or descriptions that can fool users to watching “click bait” material (YouTube, 2017). These obviously cannot be monitored by the system. Instead it is upto users and moderators to filter content such as this and either remove the videos or warn the users to take it down. Therefore a reporting system could be put into place to stop any sensitive content from being posted on the proposed video service we are creating.

“Gamification” is giving an application or service some elements of game playing to encourage engagement with a product or service according to the OED. In a loose sense, Netflix, YouTube, Twitch and Amazon Prime video, have this feature within the Xbox one home console. The services offer “Achievements” for doing certain actions within the application. For example a YouTube achievement is for a user to watch one hour of news content. This can then be seen by other users and shared between friends. Although this is not present on the web applications, this could be implemented into this application to engage users into watching different content or potentially rate videos. This could also serve to show people the extent of what they can do with the application

# List of features to include

Here we will list core features that would be needed within the web application for a video player service. An optional list will also be compiled and talked about realistically to whether or not it should be included within the application.

## Core Features

* Watching or streaming videos from a database
* Account registration/sign in
* Video uploading
* User level access
* Search function
* Video sizing function
* Video quality function
* Sorting options, both user and logical level based
* Administration tools for removing, updating or uploading videos
* Reporting system

## Optional features

* User comments and ratings
* Video recommendations for users
* User history and video resume features
* Gamification such as achievements
* User favorite lists
* User subscription to other users
* Menu panel for users and/or admins

The optional features here will be added if it is possible and time permitting. User rating and comments for example maybe easier to implement than that of Gamification features. Gamification features may take longer or may not be possible without specialist online tools. This also applies to video resume features. User subscriptions and a menu panel should be able to be implemented fairly easily, again providing that time can be put aside to build the system.

# User Feedback on current applications

While looking through current design choices for video services, I thought it would be a good idea to gather information from people that use these services on a regular basis. One person commented that they prefer the sleeker design that Netflix offers rather than the bright white design that YouTube offers. They also explained that Netflix provides a simple user interface where a video series can be watched straight from the homepage, whereas YouTube you have to search for a specific list. A series can also be resumed from Netflix’s homepage whereas YouTube you cannot. I would feel this would be a welcome addition to a web application service, being able to return to a series or playlist upon revisiting the app.

3 people have confirmed out of the 4 that I interviewed, that the comments should not be a part of the video viewing experience. The interviewees explained that it can sometimes ruin the experience of watching a video and had very little information or enjoyment out of them. Personally I also believe that comment sections can make the experience of watching a video less enjoyable. However if we were to go down a Netflix or Amazon Prime route of video service, comment sections would be irrelevant in this case.

One person mentioned an idea that they referred to whilst playing old video games. Ridge Racer for the PlayStation one uses a technique whereby a secondary game can be played while the main game was loading. For people with poor internet, a simple small game would probably be beneficial to users as a video loads. This would also open up opportunities to implement an achievement based system as discussed earlier and possibly have an edge on what current web applications offer for video streaming services.

Much like what was discussed earlier, all of the interviewees like the idea of popular or trending videos being shown to them. They explained that they could watch new content they had not seen before or new users or programs that appealed to them without having to search for it. Personally I also believe that this is now an essential feature to have for a video streaming service, as users get to watch varying and new content and the service can get a better understanding of what its user base wants from it.

One person did mention that they wanted YouTube to be more like a service than a video sharing platform. They explained that it would be preferable to see users make content that they want to see, rather than YouTube personalities for example. I feel that this could be an opportunity to create a web application that can cater to this need. This would certainly help the application feel more like a service that will be a different experience to that of Netflix or YouTube, but combine what is good about both services.

# Table Designs

## Data Dictionary

Note that size is the maximum size that will be given to an attribute. In case of variable fields this will be applied if absolutely necessary.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Size(B)** | **Validation** | **Description** |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| Video Reference | Varchar | 1000 | NOT NULL | Reference to where the video is kept. |
| Video Name | Varchar | 100 | NOT NULL | Name given by the uploader to the video. Can be edited by Admin |
| Video Description | Varchar | 4000 | NOT NULL | Description of the video given by the user. Can be edited by Admin |
| Video Date and Time (Upload) | Timestamp | N/A | N/A | Date of when the video was uploaded. |
| Video Thumbnail | Varchar | 1000 | Revert to default location if none provided | Directory to the image that will be used to depict what the video is |
| Tags | Varchar | 1000 | N/A | These will be used for defining the videos and help with the searching process |
| Comments | Varchar | 4000 | N/A | Comments of other users. Can be edited by admin |
| Comment Timestamp | Timestamp | N/A | N/A | Time of when the comment was made |
| Rating | Tinyiny | 1 | Must be 1, 2, 3, 4 or 5 | Rating system for videos are defined here |
| Genre | Varchar | 1000 | N/A | Genre of video, will be used to separate out videos into different folders possibly. |
| Length of video | Float | 7 | N/A | Length of the video. Will be used to stop uploading any content over a certain length and sorting purposes |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| User Access level | Tinyint | 1 | Must be 1, 2 or 3 | User access level for either watching videos only, watching and uploading only or Admin access |
| Username | Varchar | 20 | Must be larger than 5 char | Username of the user |
| Email | Varchar | 50 | Must be larger than 6 char | Email of the user |
| Password | Varchar | 254 | Store as a hash | The password will be converted to a hash string In the database so that passwords are hidden |
| User Image | Varchar | 200 | Revert to default image if none applied | Image that can be assigned to a user |
| About | Varchar | 8000 | N/A | A description of the user, made by the user |
| FavouritesID | Unique Identifier | N/A | NOT NULL | This will be a list to define favourite videos that a user likes. |
| HistoryID | Unique Identifier | N/A | NOT NULL | This will be a list of videos that a user has watched and can access if need be |
| SubscribedID | Unique Identifier | N/A | NOT NULL | A list of users that a particular user can follow if requested. |
| Flagged | Boolean | N/A | Default false | This will be used for if a video is flagged as inappropriate |
| Reason for flagging | Varchar | 4000 | NOT NULL, More than 10 Characters | The description as to why the video or user has been flagged |

## First Normal form

### Video Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| Video Reference | Varchar | 1000 | NOT NULL | Reference to where the video is kept. |
| Video Name | Varchar | 100 | NOT NULL | Name given by the uploader to the video. Can be edited by Admin |
| Video Description | Varchar | 4000 | NOT NULL | Description of the video given by the user. Can be edited by Admin |
| Video Date and Time (Upload) | Timestamp | N/A | N/A | Date of when the video was uploaded. |
| Video Thumbnail | Varchar | 1000 | Revert to default location if none provided | Directory to the image that will be used to depict what the video is |
| Tags | Varchar | 1000 | N/A | These will be used for defining the videos and help with the searching process |
| Comments | Varchar | 4000 | N/A | Comments of other users. Can be edited by admin |
| Comment Timestamp | Timestamp | N/A | N/A | Time of when the comment was made |
| Rating | Tinyint | 1 | Must be 1, 2, 3, 4 or 5 | Rating system for videos are defined here |
| Genre | Varchar | 1000 | N/A | Genre of video, will be used to separate out videos into different folders possibly. |
| Length of video | Float | 7 | N/A | Length of the video. Will be used to stop uploading any content over a certain length and sorting purposes |

### User Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| User Access level | Tinyint | 1 | Must be 1, 2 or 3 | User access level for either watching videos only, watching and uploading only or Admin access |
| Username | Varchar | 20 | Must be larger than 5 char | Username of the user |
| Email | Varchar | 50 | Must be larger than 6 char | Email of the user |
| Password | Varchar | 254 | Store as a hash | The password will be converted to a hash string In the database so that passwords are hidden |
| User Image | Varchar | 200 | Revert to default image if none applied | Image that can be assigned to a user |
| About | Varchar | 8000 | N/A | A description of the user, made by the user |
| FavouritesID | Unique Identifier | N/A | NOT NULL | This will be a list to define favourite videos that a user likes. |
| HistoryID | Unique Identifier | N/A | NOT NULL | This will be a list of videos that a user has watched and can access if need be |
| SubscribedID | Unique Identifier | N/A | NOT NULL | A list of users that a particular user can follow if requested. |
| Flagged | Boolean | N/A | Default false | This will be used for if a video is flagged as inappropriate |
| Reason for flagging | Varchar | 4000 | NOT NULL, More than 10 Characters | The description as to why the video or user has been flagged |

## Second normal form

### Video table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| UserID | Unique Identifier | N/A | NOT NOLL | Used for defining who uploaded this video |
| Video Reference | Varchar | 1000 | NOT NULL | Reference to where the video is kept. |
| Video Name | Varchar | 100 | NOT NULL | Name given by the uploader to the video. Can be edited by Admin |
| Video Description | Varchar | 4000 | NOT NULL | Description of the video given by the user. Can be edited by Admin |
| Video Date and Time (Upload) | Timestamp | N/A | N/A | Date of when the video was uploaded. |
| Video Thumbnail | Varchar | 1000 | Revert to default location if none provided | Directory to the image that will be used to depict what the video is |
| Tags | Varchar | 1000 | N/A | These will be used for defining the videos and help with the searching process |
| Genre | Varchar | 1000 | N/A | Genre of video, will be used to separate out videos into different folders possibly. |
| Length of video | Float | 7 | N/A | Length of the video. Will be used to stop uploading any content over a certain length and sorting purposes |
| Flagged | Boolean | N/A | Default false | This will be used for if a video is flagged as inappropriate |

### Comments table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CommentID | Unique Identifier | N/A | NOT NULL | Key field to define each comment. |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| Comments | Varchar | 4000 | N/A | Comments of other users. Can be edited by admin |
| Comment Timestamp | Timestamp | N/A | N/A | Time of when the comment was made |

### Rating Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rating ID | Unique Identifier | N/A | NOT NULL | Key field to define each rating. |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| Rating | Tinyint | 1 | Must be 1, 2, 3, 4 or 5 | Rating system for videos are defined here |

### User table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| User Access level | Tinyint | 1 | Must be 1, 2 or 3 | User access level for either watching videos only, watching and uploading only or Admin access |
| Username | Varchar | 20 | Must be larger than 5 char | Username of the user |
| Email | Varchar | 50 | Must be larger than 6 char | Email of the user |
| Password | Varchar | 254 | Store as a hash | The password will be converted to a hash string In the database so that passwords are hidden |
| User Image | Varchar | 200 | Revert to default image if none applied | Image that can be assigned to a user |
| About | Varchar | 8000 | N/A | A description of the user, made by the user |
| Flagged | Boolean | N/A | Default false | This will be used for if a video is flagged as inappropriate |

### Favourites table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FavouritesID | Unique Identifier | N/A | NOT NULL | This will be a list to define favourite videos that a user likes. |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |

### History table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HistoryID | Unique Identifier | N/A | NOT NULL | This will be a list of videos that a user has watched and can access if need be |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |

### Subscribed to table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SubscribedID | Unique Identifier | N/A | NOT NULL | A list of users that a particular user can follow if requested. |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| Usersub ID number | Unique Identifier | N/A | NOT NULL | Key field to define which user is subscribed to who |

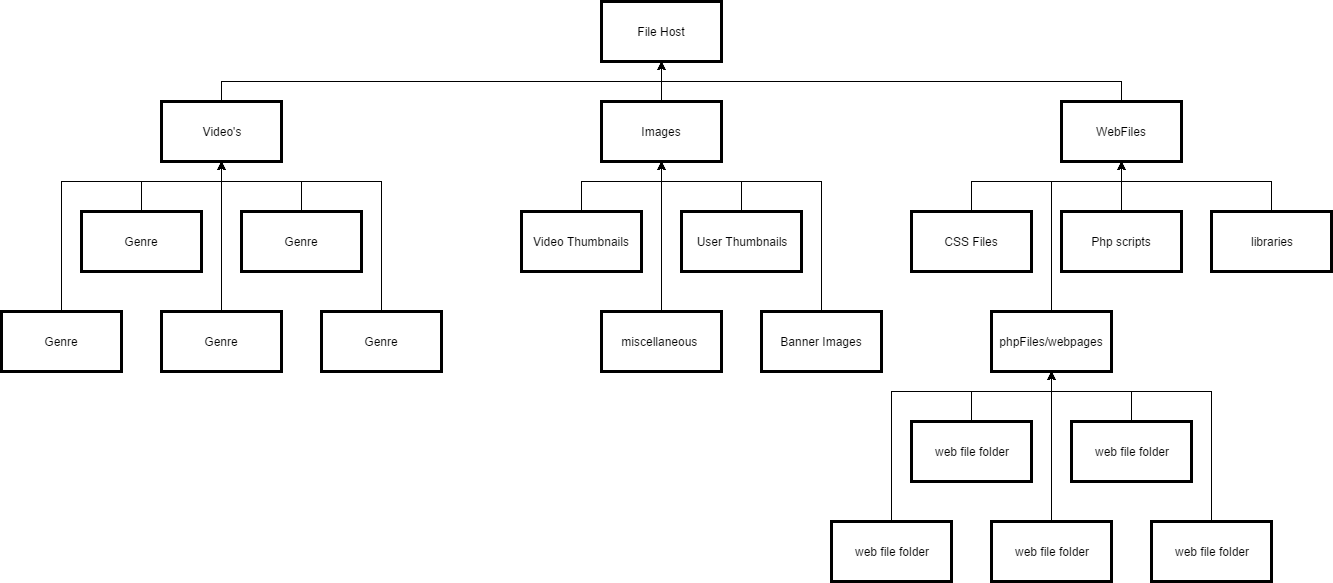
### Flagging table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FlaggedID | Unique Identifier | N/A | NOT NULL | The list of flagged videos or users. |
| Video ID number | Unique Identifier | N/A | NOT NULL | Key field to define videos from one another |
| User ID number | Unique Identifier | N/A | NOT NULL | Key field to define users from one another |
| Reason for flagging | Varchar | 4000 | NOT NULL, More than 10 Characters | The description as to why the video or user has been flagged |

# Logical level

Here a few diagrams showing what the website will look like on a logical level. This will include features such as how the tables will be organized within the database, the topology of the site and a mind map of how the user should see and use the web application. A brief overview of each diagram will also be included.

## Database folder diagram



**Figure 1**

Based upon what tables were derived from the attributes list earlier in the document, this is how I predict how the tables and content should be stored within the database. Here we have 3 main folders, Videos, images and web files. Videos will house all content associated with videos. Each video will be assigned a genre and stored within the relevant folder to help keep the videos organised. Images will be kept within a different folder and also separated out into other subfolders. These will be folders such as video thumbnails, user thumbnails, banner images and other miscellaneous images that the website uses. Finally the web files folder will hold all of the necessary scripts, web pages, and files associated with making the web application function. This will house subfolders, one for css files, one for php scripts, one for php libraries and one for php files/html webpages. The last folder in the list will again contain various subfolders for webpages, which will remove the need for users to type in the name of the web file. This type of file structure should make it easier for administrators and the site to handle

.

## Site Topology

**Figure 2**



Here is a proposed topology that will be the basis of what the website will be built around. Here we can see there is several different pages a user can go to from the home page. If a user picks a video link for example, the user should be redirected to the video straight away where it can be loaded and played. Each choice above describes either a process that a user will go through or links that a user can choose.

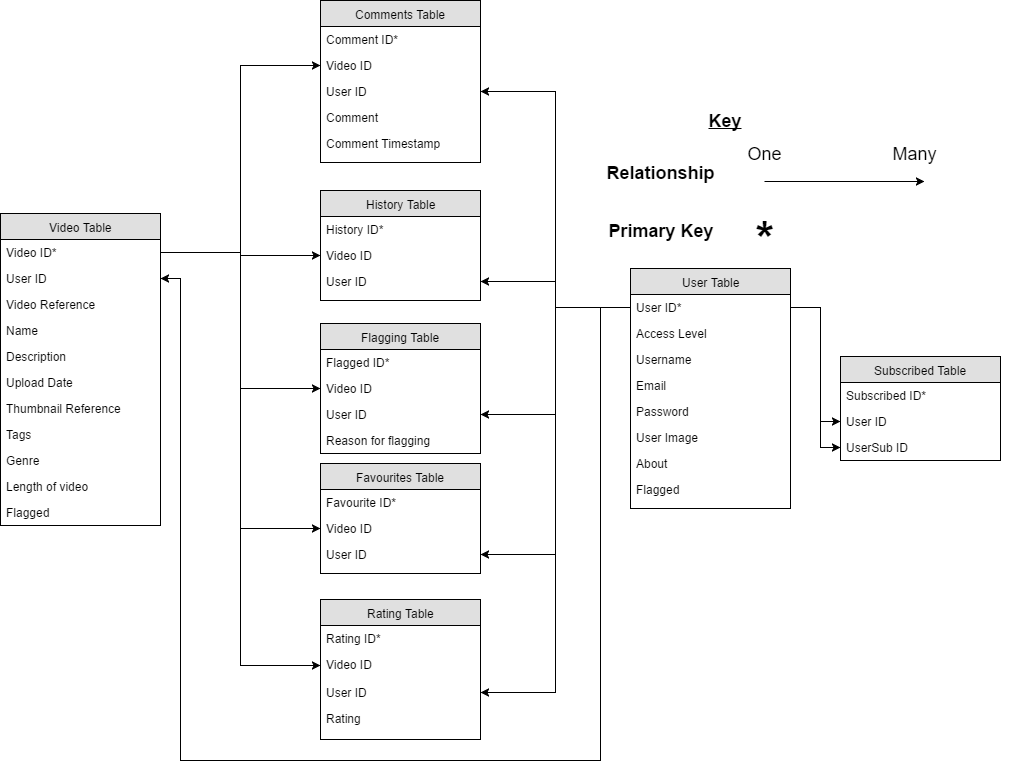
## User mind map



**Figure 3**

This mind map represents what a user can do from the homepage of the website. This describes in detail what each action will do and what the consequences of those actions will be. Again this will help the backend of the application and what the code should do, when given any of the parameters as seen in figure 3

## Entity relationship diagram

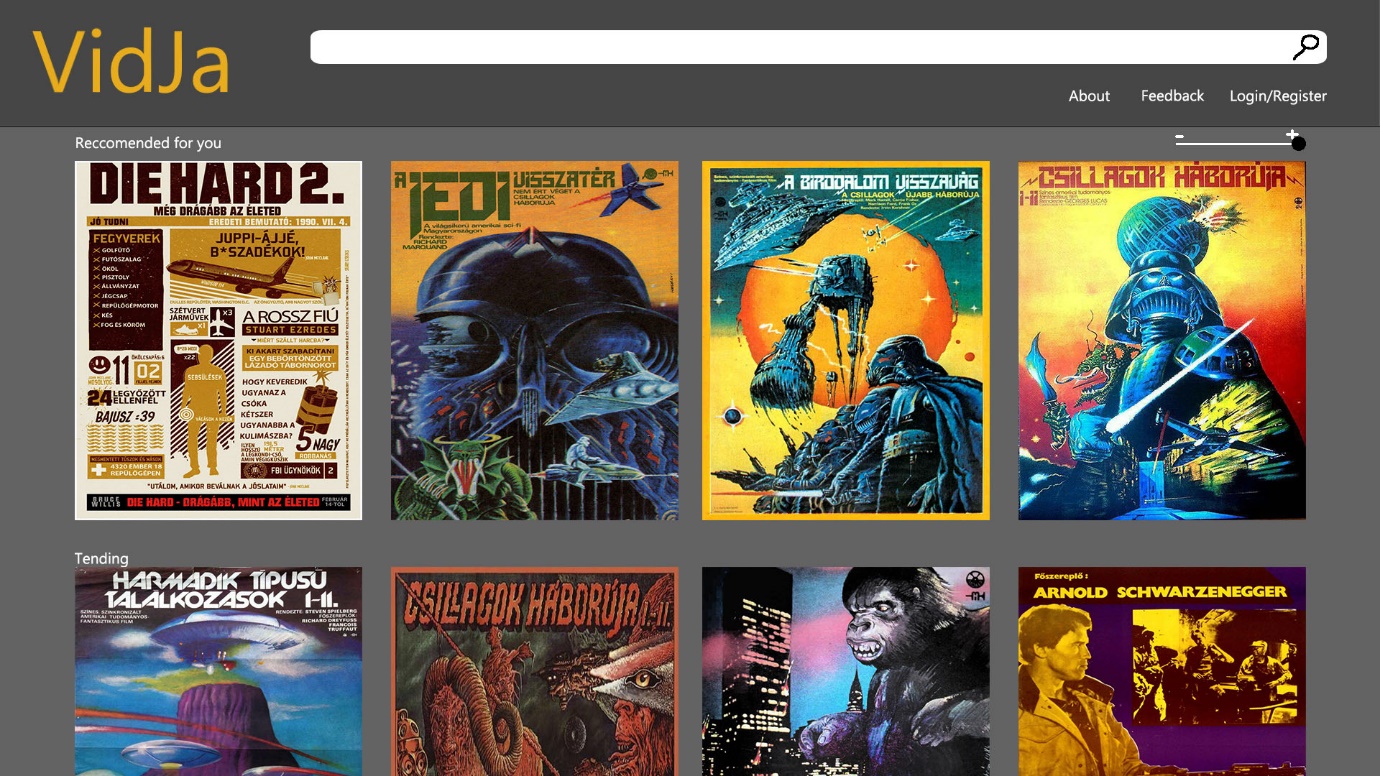


**Figure 4**

This is the entity relationship diagram for the web applications database tables. As we can see both the user and video table will play a major part in keeping relationships together. The user table in this case will be referenced within every single table. This is due to the web application being heavily based around the user and the way they can view the videos. By allowing tables such as history, favourites or ratings to be related to the video and user tables, users can access these records easily and will simple the database queries needed to fetch, edit or input data into them.

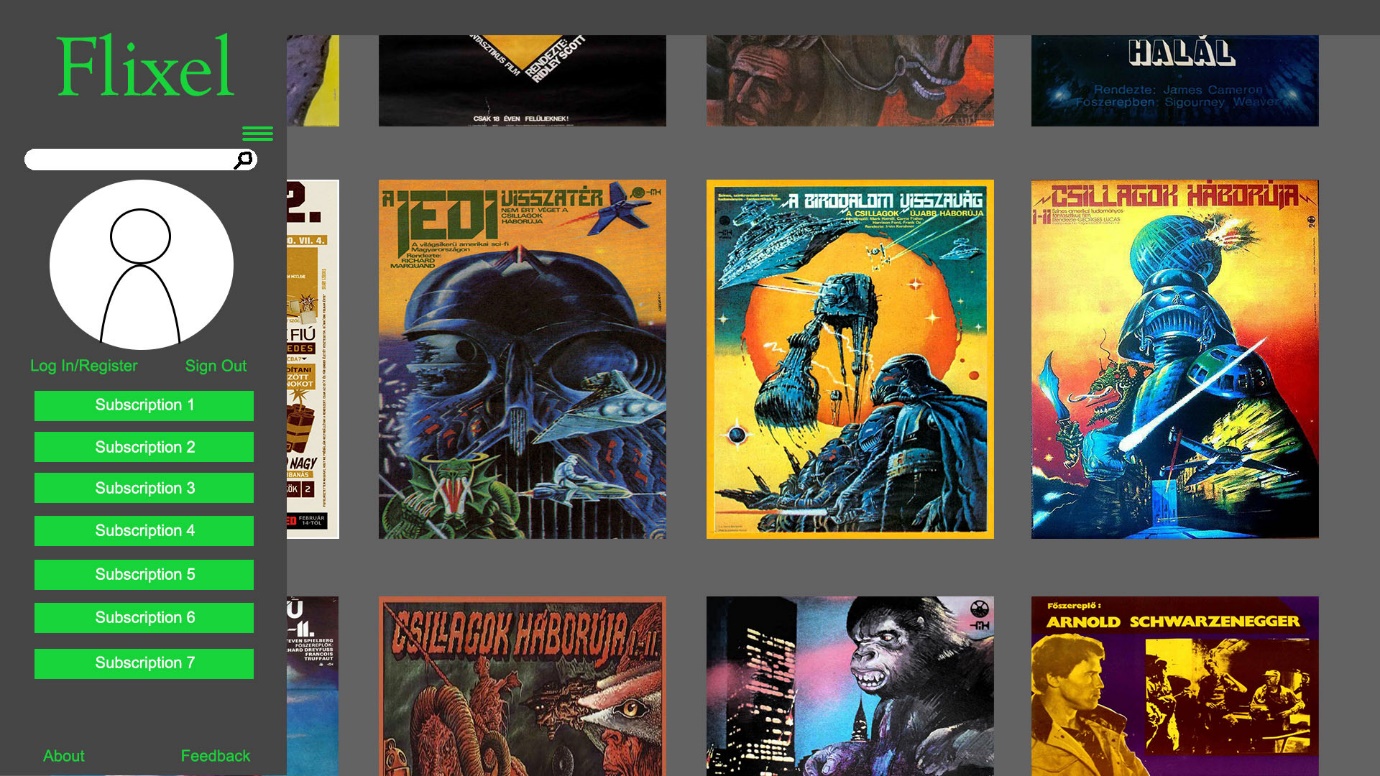
# Storyboards

## Homepage 1



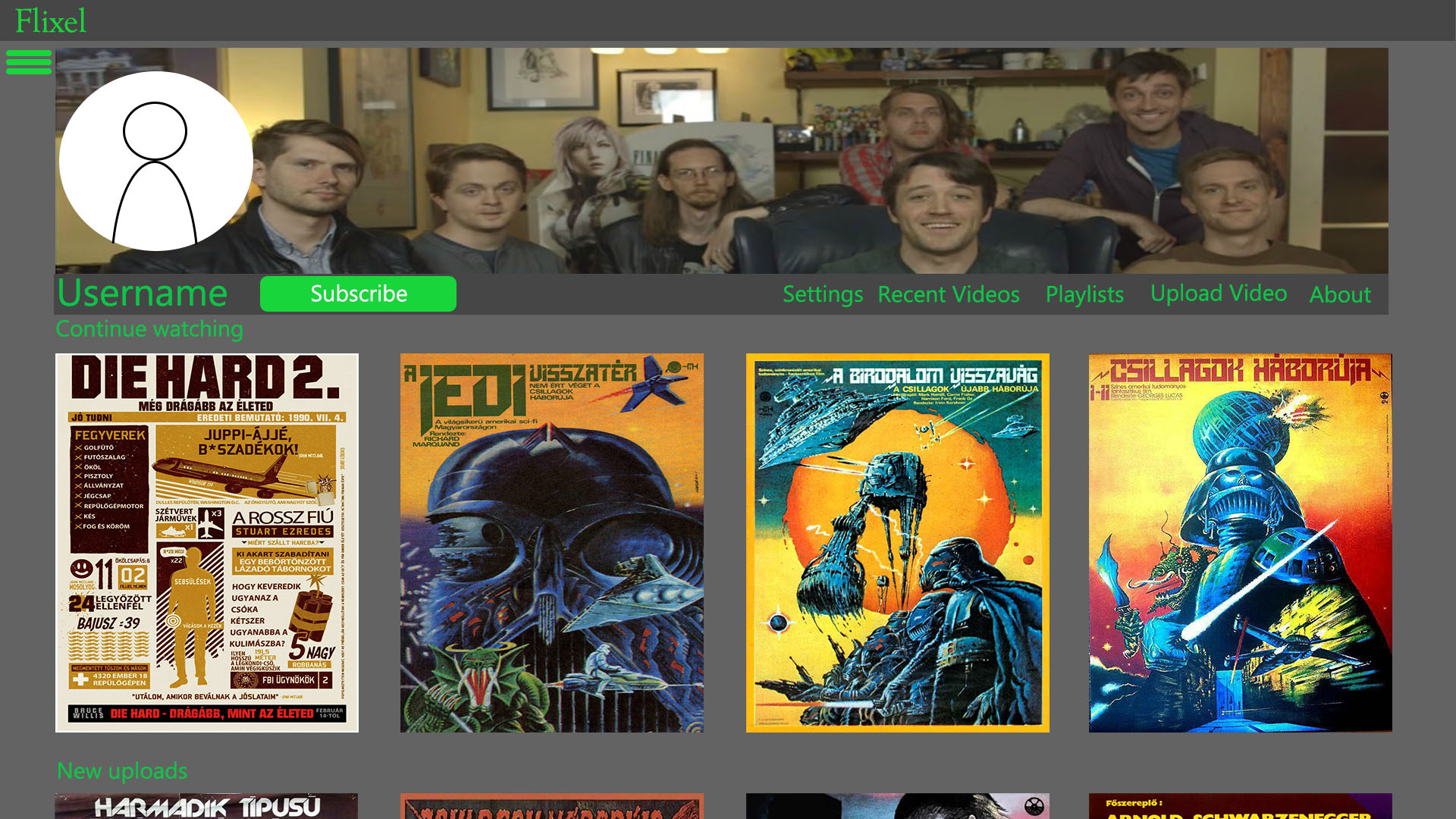
Here is a first look as to what I believe the front page should be similar to. Here we can see that we have thumbnails that will lead to various films or series. By selecting these, the user should then be taken either straight to the video or to a series list page. There is also a slider to change the size of the thumbnails shown if the user needs to do so. Users can also sign in or register using the link at the top of the screen. This may take the user to a log in page where they can sign up to the service or simply log into the website. Alternatively the sign in feature may work as a small pop-up UI window so that users will not have to be redirected to a new page. The search bar at the top of the page will be used to update the thumbnails to whatever the user types into it. This algorithm will work if it can find the title to the requested video or a tag associated with a video. Once relevant videos have been found the home page will then automatically update and refresh to display them.

## Homepage 2



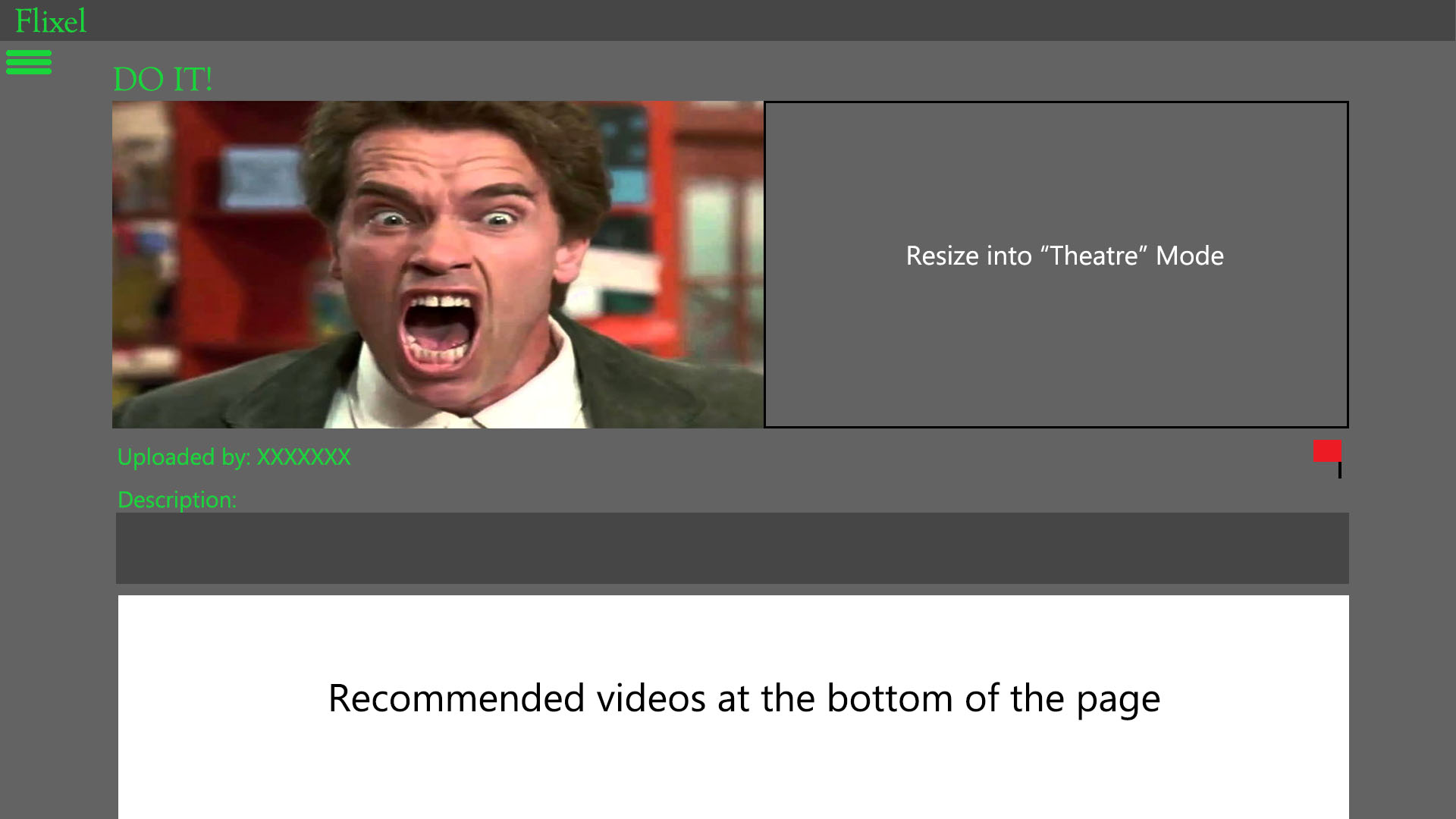
This is another idea for a homepage that could be used. It is similar to the previous one in the way the thumbnails work. However this time there is only the thumbnails on screen. I believe a minimalistic approach for a video streaming application will make it more appealing to use than that of a more clustered one such as YouTube. I have proposed using a side menu which can be accessed using the 3 lines in the top left corner of the page. The side menu will hold the search function along with the ability to sign in or register a user into the application. The user’s subscriptions will be listed underneath their profile. By selecting a subscription, the page should then be redirected to the selected subscription page.

## User page



This is a proposed user page that could be used. Here we have a user banner, user image, name and links to other various other pages that they can use. For example, selecting the user image will allow a user to alter their image to something new. The settings option will take the user to another page where they can alter their password or update their banner image etc. Other links may be added to the navigation bar if necessary. Users can also resume watching videos or see other videos that they have loaded from this page. The menu can still be accessed from this page via the menu button in the top left corner of the screen. This will be used likewise for users looking at other users profiles. Some features such as the settings option will also be removed in this case.

## Video Page



The video page is also meant to be as simplistic as possible. Users can rearrange the size of the video along with the quality as discussed earlier. The video will also include who it was uploaded by, the description of the video along with time stamp and recommendations to other videos that are similar to it. Again the user’s personal menu bar will be available using the button in the top left corner of the page.

# Subfolder name ideas

Here are some popular categories that could be considered to use for the video file folders. These would also be seen by the client as different types of categories to choose from.

* ActionAndAdventure
* Animation
* BeautyAndFashion
* Comedy
* Documentary
* Entertainment
* Family
* Food
* Gaming
* HeatlhAndFitness
* HomeAndGarden
* LearningAndEducation
* Nature
* News
* ScienceAndTech
* ScienceFiction
* Sports
* Travel

Other categories could be added if the web application was to grow in scope.

# Security features

Security is going to be a high priority when implementing this system. Firstly the system must not hold passwords as plain text whatsoever. Instead the site could use some form of encryption like sha1 or PHP’s password hashing function using a salt. Most web hosting sites come with their own default salt, which may be used in order to hash the passwords and store them within the database.

SQL or any sort of code injection must also be protected against. This can be a major problem in an application that will require text box entry. There are numerous ways to prevent attacks like this such as adding validation to the text boxes to not allow special characters. This could prevent users entering <script> tags within the coding. Other measures within PHP code could also be added, such as using different techniques like procedural or object oriented methods. By utilizing some methods over others in key areas, SQL code injection may be combated against.

Protecting who can sign into the site is also a cause for concern. The site must be registered by people with real email accounts. Upon entering their email into the site, the user must be added to the applications database and verified using an email link of some description. This will ensure that users must enter a valid email, thus removing potential spamming of the system using bots for example.

# Pseudo code

This section will focus around how the code should run for a few select specific webpages and server side scripting. Some of these areas will be talked about as to why these specific coding designs were chosen.

## Webpages:



Here is proposed coding for a generic web page. This includes functions for fetching all videos within the database in server side scripting and functions for opening and closing the side menus within the web page. A search function will be implemented in some way using either synchronous or asynchronous ajax functions. Synchronous may be more beneficial for this search function as it will have to wait for the script to return data before updating. The login function will also work similarly to this. However on a failed attempt, the function will then write the error message onto the webpage to show the user. If the user is signed in the PHP server code should automatically load a user’s name, user image and any necessary links that will be associated with them.

## Server Side Scripts

### Login Script

This is what the basic login script should perform when a user wants to log into the site. Various session variables will be stored upon a “true” completion of the script so that important pieces of information can be kept throughout their usage of the site.

### File upload script

This file upload script has 2 major parts to it. One to upload a video thumbnail image and one to upload the video file itself. The various checks that are in place ensure that both the image and video files are usable on an online video platform. The posted variables and locations of these files will also be uploaded to the database to be used later.

### Registration script

This script will be used to register a user to the database and send an email to the registered user. The script will check to see if the email or username exists in the database. If this is true the script will terminate. The password given will be hashed and stored into the database to make it more secure. A token will be generated using the current timestamp plus the user id to make it unique.

### Functions Script

Multiple different functions will be used within this script. The most important of these will be the database connect function. This will be used throughout all scripts and PHP web files in order to connect and send requests to the database. There are also other functions for retrieving multiple different videos with different genres or tags for example. There will also be a function for fetching a user’s subscriptions depending on who the user is. This script may be subject to change during development.

# Application and User testing evaluation

## Self-testing review

Testing was carried out for every HTML, JavaScript and PHP scripting. This testing was done within the Google chrome browser as the testing browser. The procedural method used was done to try and remove any potential errors that scripting could run into. This included testing input boxes with maximum values, no values, wrong file types and expelled characters. Most scripting within the site works on user based input, therefore testing for this was made essential. From the test results given from almost all tests. All input boxes function properly, even if erroneous data is given to them. PHP scripting also works side by side with the input boxes to stop any errors and maintain data integrity for the database. The system could be improved to use the anti-scripting technique for all input boxes. Currently this only applies to 2 within the view page.

Tests 24-31 and 36-40 cover the registration system. Although this had a few teething issues at first, this feature now allows users to register to the database using their own personal email. The scripting used is supposed to make multiple checks to make sure that the user is genuine and trying to register with the application as intended. These tests also show that the coding used is indeed fit for its intended purpose. A user will be provisionally added to the database, if an incorrect link is given to the user or the user tries to use this link with different parameters, the script will automatically delete the user from the database. This is necessary to stop other pieces of data being corrupted in the database or the record it is meant to register. Users cannot use the same link twice as it will just ignore the user, explaining that the user already exists. The link will also expire after 10 minutes as seen within test 39. This feature was also implemented for memory reasons. If a user is provisionally added to the database and does not register, the user will be automatically deleted by the CRON Job shown in test 100. These features still perform correctly and even after implementation, user feedback has yielded no problems with this system. Gmail accounts however will not work with the PHP scripting. It is unknown why the mailing system does not accept Gmail accounts and will lockout a large portion of users. Ideally an actual mailing server would be used in the implementation to combat this issue.

Tests 41-52 showed the implementation of the admin page, which contains multiple tools for administrators to use. Each tool was created around the flagging system that was added to the view page. Many of these tests cover the running of Ajax scripts that run various pieces of PHP scripting in order to ban videos, un-ban videos and accept user upgrade requests. The rigorous testing cycle shows that both scripting languages ran into problems. Now ideally scripting such as this should not be done using Ajax, due to some users having JavaScript disabled in their browsers. Showing the names of the PHP scripts also displays where these files are being retrieved from. Again this is not a preferred way to code as users could potentially get into the folder directory and retrieve the scripting. Since this is on the admin page, this has been ignored as these methods are prominent on this page. Although many of these functions that are used call on different and varying PHP scripting, the page still works correctly as a whole. If one script fails, it still allows the admin user to carry on using the other tools available to them.

2 settings scripts for users and videos were added and tested within tests 61-69 and 83-88 respectively. Now the scripting used for these pages use one large PHP script file to handle all of the data. Due to the numerous tests shown here it is clear that this was not effective method to use. Tests constantly showed that the scripting could cause multiple issues such as incorrect data being inserted into the database, wrong files being placed or not placed within the file host or the scripts just failing altogether. Although these 2 large scripts now work with their respective forms, separate functions and scripts should have been created to handle each piece of data. By using one large script this has opened up the possibility for multiple failures to occur if one section fails whatsoever. Conversely the large scripts do account for almost all possible errors and will handle the data from one form. The system would benefit from each field within the form to have its own script, to reduce errors and maximise efficiency as smaller scripts would mean less server runtime.

Tests 89, 90 and 91 followed the implementation of the applications search feature. It was recorded that the Ajax, used to display the data sent back to it, was showing multiple results. However it would show these results in multiple successions, meaning that the same video would show 4 or 5 times upon the PHP script returning the results. This was ultimately fixed by only printing out the results once from the PHP script. Again however the scripting will still show multiple fields if the user types in quick succession. This is due to the key-up event which calls the Ajax. Using the Ajax method for search results is useful as it allows users to stay on the current page, say a video, while being able to look for another video or user. The key-up event does cause issues which will display multiple results due to how fast a user can type. The Ajax method would still be preferred as the page does not need to be reloaded and the search results still display and work, even if there are multiple results given to them. Instead better Ajax scripting or a different key event such as having to hit ENTER would remove this problem entirely and would make for a good improvement.

The reset password script used works in a very similar way to that of the registration system. The user can request a password, whereupon a default password will be set and a link will be sent to their corresponding email. Tests 94-96 show this. No errors occurred for this system whatsoever. The system performs as it should, no further improvements would be needed to be made based off of the testing. Better methods could be incorporated to produce a safer, more unique default password and user link but the system serves its purpose.

Briefly mentioned earlier, a CRON job has been added to the application. This feature was set to remove unregistered users from the database twice a day using an external PHP script. The testing showed positive and the feature does remove the correct records every 12 hours. No further changes would need to be made here, apart from more CRON jobs which could clean both the file host and the database.

## User testing/feedback review

The feedback for the application has been extremely mixed as can be seen from the results seen in the appendix. Majority of the results show that the system is easy to navigate and easy to read. The application was designed with this in mind to try and make this website very easy to use and get around. The colour scheme was decided upon to be green and grey from the designs as these would be vastly contrasting colours. The font used “Righteous” was decided upon as it was naturally bold and suited the aesthetic of the application.

Overall rating for the application was 3 out of 5. The reasons given as to why this is are very mixed. For example multiple users explain that the application lime green and grey colours do not suit the application whatsoever. They feel that the application suffers because of this and actively puts them off from using it. Again it was explained that these colours were used to contrast against each other. All users explain that the text is readable but the colour scheme is not to their liking. This application is not catered for front end design. The original purpose of this application is for users to be able to easily read and navigate the site. This has been achieved and would not need to be a major change for this application. One or 2 users did mention that they had difficulty reading the font. Different font could be used in order to make it easier to read. This in turn could potentially increase the user base.

A few users did explain that the searching system was not working correctly. Now it was explained in the self-testing section, that the system is prone to displaying more results than there actually is. Apart from this one issue, the search function has worked correctly during the self-testing phase and for most other users. If the script fails or the database connection fails, then no results or errors will be shown. This could be an explanation as to why this feature did not work for these users. The feature does need improving within the Ajax. The current method of calling the search script constantly could also cause the feature to fail. A more traditional posting form method or update upon the ENTER key press could also irradiate these potential issues.

Some users also pointed out that some of the text boxes and some other elements within application did not scale properly on mobile platforms or other browsers such as internet explorer. This application has not been created to cater for multiple platform deployment yet. Some elements have had features included to resize upon the screen size altering. By applying better CSS rules, the system could be designed to respond to the change in screen size or resolution.

One astute user tested the registration system by using the same email twice. Now although there is a function that checks if a user is registered into the database, the system does not check for what case their email is set to. The user was able to register to the site twice. This is a big issue as it would conflict the system if the user was to sign in with their email. To combat this, the str\_lower() function has been implemented into the registration coding. This now lowers the case of all emails. This now means that emails cannot be inputted that have any uppercase letters within them which removes the problem.

Again mentioned within the self-testing phase, the web host will not email gmail accounts with a registration link. 2 users mentioned this within their feedback, explaining that it alienates people including one of these users. This is not a fault of the application but the web host. In an ideal situation, a reputable emailing service or server would come as standard, thereby allowing all emails to be contacted.

Did the site meet the expectations of users? Nearly all of the users involved with the project answered yes with a few changes that they would have liked to have seen. For example one mentioned not being able to upload .wav files. This was simply due to the application being designed to use mp4 videos only. The HTML5 video player tag, supports this format on all browsers, which is why it was chosen. Adding other videos is possible by simply adding new tags to the player with different file formats. This again could be upgraded to support multiple formats in later possible iterations. Most other comments explain about a user’s personal preference, i.e. the colour scheme, the types of currently uploaded videos, alignment issues etc. These would be addressed, however for this project these features were deemed negligible. Overall the system works, users can watch videos, sign into the application and upload videos if they wish, which is the basic function of the system as a whole.

# Self-review of application

Looking at the web application it is clear that it operates at a decent working standard. The application allows users to view video content, register and log into the application and upload and edit videos as they see fit to name some of the basic features. Initially the main goals here were to make a system that would replicate that of common video web applications such as YouTube or Twitch. The features that were mentioned in the design stage were as follows:

## Core Features

* Watching or streaming videos from a file host/database
* Account registration/sign in
* Video uploading
* User level access
* Search function
* Video sizing function
* Video quality function
* Sorting options, both user and logical level based
* Administration tools for removing, updating or uploading videos
* Reporting system

## Additional features

* User comments and ratings
* Video recommendations for users
* User history and video resume features
* Gamification such as achievements
* User favorite lists
* User subscription to other users
* Menu panel for users and/or admins

## Core features

Starting with the core features, the web application has been built so that any user can open the system and start watching video content. Extra features have been added in order to entice users to register, such as uploading videos themselves or commenting on videos for example. All videos load quickly and efficiently due to the video files all being one video type. This does however mean that the application is limited to MP4 formats which can actively put off new users and does not adhere to other sites such as YouTube which allows multiple file formats. The positive to using MP4 files however is that it is accessible across most platforms such as Google Chrome, Apple Safari, Firefox, Internet Explorer and most mobile devices, making it the most ideal format for compatibility.

User registration has been added to the application which allows users to add themselves to the database so that they can sign in. The system used for this will add the user to the database conditionally until they use a custom made link which is unique to them. This system is ideal as it means that users cannot make multiple accounts which would cause database storage space issues or be subject to attacks such as DOS. A CRON job has also been put into place to delete users that are not fully registered to further optimise the database. Passwords are stored as hashes within the database to further add a layer of security for both the benefit of users and the administrators of the application. The hosts default password hash has been used in this case. This is a safer way of password storage but a salting method would have been more preferred to further encrypt them. Due to constraints again by the hosting site being used, this was not possible.

The video uploading code works to combat most possible errors a user can possibly cause. The file upload limit has been limited to 8MB, the video file cannot be empty and the video file must be an mp4 format. Although these are just a few ways to stop any potential errors both database wise and system wise the code could still be improved upon. If the file size is ludicrously large, the current web host will not post the data to the script. This can cause problems as a user can be stuck on the video upload page unable to progress. The use of session and local messages has been implemented to tell the user of any errors that occur, but this will unfortunately not cover errors such as failure to connect to the database for example. The input boxes have a maximum character count within them and are required to be filled out before the user can upload the video. Again this will mean that the database will not be left with any null or incorrect values when this record is called upon elsewhere. The image file also has the same parameter checks as that of the video input field. Yet the user can leave this blank to upload a thumbnail for the video at a later date. Again this means that the video can be uploaded quicker which will make the experience easier for users.

User level access has been implemented into the web application. A user that is not signed in is classed as a level 4. This means they have basic functions of watching and searching for videos. Users who register however have access to a number of features at level 3. These include:

* Customisable profile page
* Ability to comment on videos
* Report videos
* Subscribe to other users
* Have their viewing history recorded
* Making a list of their own personal favourite videos

These features can be extremely beneficial to users that will operate the web application in great deals. It gives them more freedom to find more content or users and enables them to customize the experience for them or other people who find them. If a user requests to be a level 2 user, which allows video uploading and editing, they have to request an administrator for these rights in order to do so. This has been decided to stop any users uploading videos as now they have to be verified before they can upload. Thus reducing the likelihood of any bad content being uploaded, such as graphic or sexual content. Finally Level 1 users are administrators. Administrators have the same abilities as those listed above, however now the user has access to an admin hub page and the ability to ban users or videos and edit any video of their choosing. This levelling system ensures that users can be moderated, ensuring that the web application will not be overburdened with multiple video uploads to it at any given moment. Administrators also benefit as they can stop users or videos from being public, which will make the entire platform safer and easier to navigate and control.

A search function has also been included into the system. Currently the system uses Ajax scripting in order to instantaneously fetch results for users to select from. The system has been designed using a LIKE query within the server side scripting. Although this method works, it will only display results where the search query is exactly within the database tags column. This means that a user cannot type multiple different words in succession unless they are separated as such within the database field. This could be more optimised to search for multiple key words rather than just one or an exact phrase or sentence. Because the Ajax scripting calls upon the server side scripting multiple times this has led to a few issues. One example is that because the server side script is called upon on every “key up” event, the server will be dealing with multiple queries in a small given time. This could lead to connection issues or even causing the server to crash. To remove this problem the search system could be designed around just calling the script upon enter. But again this would sacrifice instant results and was why this original method has been used instead.

A video sizing function has not been included in the application. Because the application uses embedded HTML 5 video players, the code will allow the user to full screen the video. CSS has been applied to reduce the size of the image until a set dimension. Again this will allow the video to run in different browsers or platforms as the video size will automatically change. A temporary video sizing function would not serve any beneficial purposes on top of the ones described, simply because the video can be full screened, which will auto fit to the devices screen resolution.

Because the video uploads have been limited to 8MB per video, due to the hosting service that is being used, this has not been implemented into the system. This would be unnecessary in this case. Although this could be applied to short, detailed videos, the benefits of choosing the quality would be negligible, hence why it was chosen to leave this feature out.

Sorting options have been included into the site, however this has only been applied to the logical level. The functions.php script has been designed so that in PHP coding a certain genre of video or videos defined by tags is an option. A function has also been added to define these sorting options to one particular user. This makes it so administrators, for example, have the ability to fetch defined video lists to put into other parts of the application. User defined sorting for videos would have been preferred for this system. Instead the only method that is close to this is the searching feature. If the application were to be upgraded, a sorting feature would be extremely beneficial to users wanting to find specific content.

Administration tools have been included into the application. There are a plethora of different actions that an administrator can perform in the front end of the website. For example, an administrator has the ability to ban multiple videos or users as they see fit. The code used to do this will automatically update the database so that any banned users or videos will be removed from public viewing. The only users that can see these videos will now be the administrators. Again this is beneficial as administrators have control of the site and can stop user’s intent on causing problems. Not only will a user ban remove them from being viewed publically, this will also ban all videos associated with them. This therefore acts as a ban all button for multiple videos. Conversely however this means that videos that are not in breach of any of the applications rules will be banned. Moreover if the user is made available for viewing once again, all videos associated with them will also be made available for viewing. This can be incredibly problematic, as videos previously banned will be made available publically. The system could be improved upon to make sure that the videos that were banned before will stay banned once a user is made available once again.

Administrators also have the ability to edit user videos. This has been included so that if a video needs to have a name changed, tags added or removed from it, or moved its current directory, the administrator can change this without needing to change it manually in phpMyAdmin. Currently this is the only way to delete videos as well which again is less than ideal. A feature to delete videos within the administrator’s menu page could be added.

The admin page also has user requests listed within it. This gives administrators the ability to confirm or deny video upload requests, without the need to change the user’s access level manually within the database.

Videos that have been flagged have also been added to admin page. Admin can look at the reasons as to why the video has been flagged and then decide whether to ignore these flags or ban the video.

Looking above we see that multiple different admin tools are available. Admin have the same abilities of a level 2 user, with added extras of banning, editing all videos and verifying users. However some of the tools are lacklustre in features, such as the banning of users which in turn bans all videos associated with them. The tools could be more refined and added to, to make this application more suitable for a full scale deployment with multiple people using the platform.

Unfortunately only one reporting system made it into the application due to time constraints. A reporting system has been added to videos but not for users. Although this system does inform administrators of the site that a video has been reported, this could be for multiple reasons. For example a video could be reported to report a specific user within the comment section. Alternatively the video could be reported to report something within a user biography to name another example. The application would really take advantage of a reporting system for users or comments. This would also mean the administrator menu page would also need to implement these features, which would also benefit from this.

## Additional features

User comments have been added as a feature. The way this works is by using Ajax scripting to send a comment to the database to be stored. The server side scripting will then respond with the comment to the Ajax scripting, in turn adding it underneath the comments. A user must also be signed in to use this feature. If any previous comments have been posted, the server side scripting will fetch the data and write it before the page is loaded. This system is very effective and simple and has been designed to limit the amount of data that can be stored. Comments would take advantage of a deleting system however. The only way currently to delete comments is by deleting the record within the table it is housed in in the database. Again the system could be further improved for users and administrators if this system was added. This was finally decided upon against the people who were interviewed within the design documentation. It adds the ability for users to talk to each other and give feedback to the user who uploaded the video. This makes it a welcome addition in this case.

Ratings were considered during development. However it was decided not to be implemented due to the fact that the only benefit this would give users is a top rated section within the homepage. Due to this being a small feature that would store multiple different records within one table to do so and would require calculations to fetch an average, it was ultimately decided that it would not be an efficient use of resources.

Video recommendations have not been included within the application. This would have been a welcome feature within the application and could have been implemented given the right amount of time put into it. Recommended videos would require retrieving history of a given user, comparing the genres of those videos and performing a search query based upon the results. This would have been preferable, however it would have been impossible to determine which videos would be more preferable to a user by genre alone. Instead a similar video listing function has been added to each videos viewing page, which shows videos that are the same genre as the one within the page at the time.

As mentioned above, user history has been added as a feature. Users can view their video history, delete specific videos from their history or clear the list entirely. This was added so that users could find a specific video that they had watched before. This was also was going to form part of the recommended videos function as mentioned. The feature still serves useful for users and could be used to be implemented into a recommendation system in future.

Video resuming features has not been possible using an HTML 5 video player. This would only be doable by implementing a custom made video player.

Gamification was considered for this project. Achievements would not have been a part of this however, due to the complexity and confusing nature it would have been for both users and administrators. Instead a game was considered to be implemented into the site using an embedded java applet to make it easier for users to use and to implement on the application. Using a game however was found to drastically reduce the loading time of the web page it was considered to be embedded into. First and foremost the application is a video hosting platform. Adding this to the video viewing page would have distracted the user and would have taken the video longer to load. Therefore this was left out as a direct result.

Favourite lists were included in the system. Users can add a video to their favourites by simply selecting this option on a video page. The user and others can view this favourites list and select the video to watch it again. Much like the users viewing history, the user can delete a video from their list if they so desire. Although this feature is very useful, it is also pretty irrelevant. Most other platforms offer this feature, but the feature is limited in scope. User playlists would have been more preferable or even a sorting or searching feature for the favourite area would have made this more beneficial for users to make use of it.

User subscriptions were added as a feature within the application. This allows users to subscribe to users they prefer and will display them constantly within the user’s navigation menu. These will link to the user profiles, making it easier for the current user with the list to get back to the user they have subscribed to. At any time on the other users profile that the user has subscribed to, they can use the same button to unsubscribe from them. This will then remove them from the list. This feature is again useful, as it helps users find their favourite uploaders. Upgrades could be added to it once again, such as locally removing a subscription, without having to visit an uploader’s page.

Mentioned multiple times before, the users and administrators have a side navigation menu which holds the ability to search for other users or videos, login, register, access a user’s personal page, or list a user’s subscriptions. Administrators also have the option to go to the administrator’s menu page as well. By implementing this side navigation menu in this way, it has allowed a user to have direct access to travel to any page they want to, sign into the site without having to reload the current page and go back to their personal page. Certainly this is very advantageous due to these reasons, however upon sign in, a user would not be able to make a comment on a video for example. The user would have to reload the webpage in order to do this. Meaning that the side menu is useful but once again could be improved upon in some areas.

## Table designs

The table designs have mainly been implemented into the system with little changes. The main changes within implementation was the Videos and Users tables. All other tables have not been altered from the design.

The video table was updated to include the number of people who have viewed the video and if the video was banned. The views field in this case is not used for any data manipulation or functions within the system itself. The inclusion of the field named banned however now plays an integral role for the system. This now determines whether the record that it’s associated to is visible to users within the site. Administrators will still have the ability to see it and restate it or delete it if necessary. This has been extremely beneficial for keeping the system under control. With the scripts that have been put in place to ban these videos, they can be micromanaged relatively easily. Conversely however, keeping this value to one field could be problematic given that if the database was subject to an SQL injection attack, all the banned videos could be made visible. Therefore although the table is now fit for purpose there are still other failsafe’s that could be put into place to stop users changing the values manually. Hashing could be a good use of storing the true or false values. No other changes have been made to field lengths, default values or data types from the original table designs.

Many changes have been made to the Users table. Again some of these changes were for non-computational/more aesthetical reasons rather than computational/system reasons. The Token and Token\_Validity fields have been included for a registration system that was put in place. Token times and Validity are used to ensure that users registered to the database are genuine. If a false email is given the user will not be registered to the system for long. Instead the database will wipe this user if the registered field stays at 0. This system is integral for any web application that requires a log in. This removes the ability for users to create fake accounts to access exclusive features that they wouldn’t be able to access otherwise. Therefore this is why these extra fields were ultimately added as it was deemed more necessary to do so. Much like the Videos table, the Users table has also been updated with a banned field. This again is to ban the user from accessing their account or other users being able to access their profile page or content. This was deemed necessary to stop users who were abusing the system from being able to carry on using it. Administrators now have the ability to ban individual users, by the admin tools on the system, making this added field a necessary addition to the table. The upgrade field was also added for users to request to upgrade their account to a level 2 user. Again this was added for admin tools which allow them to see user’s requests without having to access the database directly.

Other tables have not been altered from the original design specification. The tables have still served their purpose for the system without the need for alteration. The ratings table was created for the application but, during development, was not implemented as mentioned within the additional features section.

## Front end design

The initial storyboard designs shown have been adhered to. The website front page has tried to mimic that of the original designs shown within the design section of the document. Homepage design 2 was ultimately chosen to give the application a similar look to that of Netflix and YouTube combined. YouTube uses a side menu system whereas Netflix opts for a thumbnail only system as soon as the homepage is loaded. All navigation was kept to the side bar to maximise this Netflix type layout while still being able to access other webpages from one area. Although this is a good idea to keep all navigation contained onscreen at any one point, the side menu itself is not plainly obvious at a first glance. It was decided that the side navigation bar should be kept closed until a user activates it. This could be made more obvious to users due to the fact that users will not know how to navigate around the site. In hindsight it may have been more pertinent to direct users who aren’t signed in to the about page upon entry of the site. This would then tell them how to navigate the site and get better accustomed to it before use.

The user profile page that was also discussed within the design section has also been replicated to a good degree. Users have the same layout which we feel is well laid out and easy to navigate. All links are displayed underneath the banner and user image, with the option to subscribe to the user if on the sub page. Thumbnails have changed once again to a smaller size with written information. The reason the thumbnails were changed, were to let users know what video they would be watching, rather than basing their assumption on a thumbnail image. This was also changed due to the fact that users did not have to upload their own images for the thumbnail. This could lead to thumbnails that would use the same default image with nothing to distinguish them, hence the change during implementation.

The design for the video viewing page has stayed somewhat similar to the original design. The idea of resizing into theatre mode was scrapped due to the video source becoming stretched and removing other aspects from the screen such as the comments or related videos. The flagging system remained in place along with a similar layout in the implementation. Recommended videos were added to the side of the video at certain widths and then moved underneath the video if the screen size was smaller. The design has been kept as simple as possible to keep users from being overwhelmed when using the site.

## Security Design

Some features have been included into the application in order to try and keep users and the applications data safe. Password hashing has been introduced to store passwords safely within the database. Salting passwords would have been a more preferred method of storing user passwords within the database. Due to the file host being used, salting unfortunately was not an option that could be supported. However passwords were stored as a hash using the web hosts default password. The password verify function within PHP has then been used to decrypt and match passwords. This method of password protection is very standard on most web applications and websites and does serve its purpose here. However it would have been more beneficial to use a complex method of password storage, such as using salting.

Some text box security has been applied to the application. The video viewing page has had its comments text area and flag text area upgraded to protect against code injection. Any comments or flags that contain any “<” or “>” symbols will be automatically ignored and the user will be banned. Now although this does stop simple script injection, it does remove the ability for users to use these characters altogether. This is not a major sacrifice as these are not required for users to experience watching videos. The code could be improved better to stop profane words from also being entered into the text area.

User account registration was mentioned within the security design earlier in the document and earlier within this section. This has been implemented successfully and stops threats like bots or throwaway accounts to upload harmful content for example.

# Final Evaluation

The application has adhered to most of the features that were initially intended to be implemented into it. Most core features have been included and been utilized to a decent degree. For example the search function works as expected but could be better optimised to reduce connection issues or any other server side errors due to constant requests. Features such as video quality options that were ultimately left out, were done so due to limitations of the hardware being used and not being feasible for a full scale deployment. The system can be used as a video hosting platform and will provide the same features that most common internet users come to expect from a system such as this. However it has been noted that multiple changes and improvements to the current system would make the platform more optimised for a large number of users, and make the system more user-friendly both user and administratively speaking.

The application is functional. It completes its initial tasks and more with other features such as user settings, video settings, favourites and history pages, subscription features etc. From all the evidence that has been shown it can be safely said that the application is more than fit for purpose, but would need tweaks in certain areas to make the application appealing to all users and make it more manageable for administrators who would run it.

# Admin manual and Maintenance guide

This manual is to show administrators of flixel how to use the basic functions within the site and how to maintain it from a back end perspective. This will also include screenshots to various sections if required.

## Front end website tools and functions

The website has been designed around making navigation and functions easy to use. An administrator will have the same abilities as a regular user, however they will have access to admin tools and other functions which would be unique to one specific user.

### Homepage

**Figure 5**

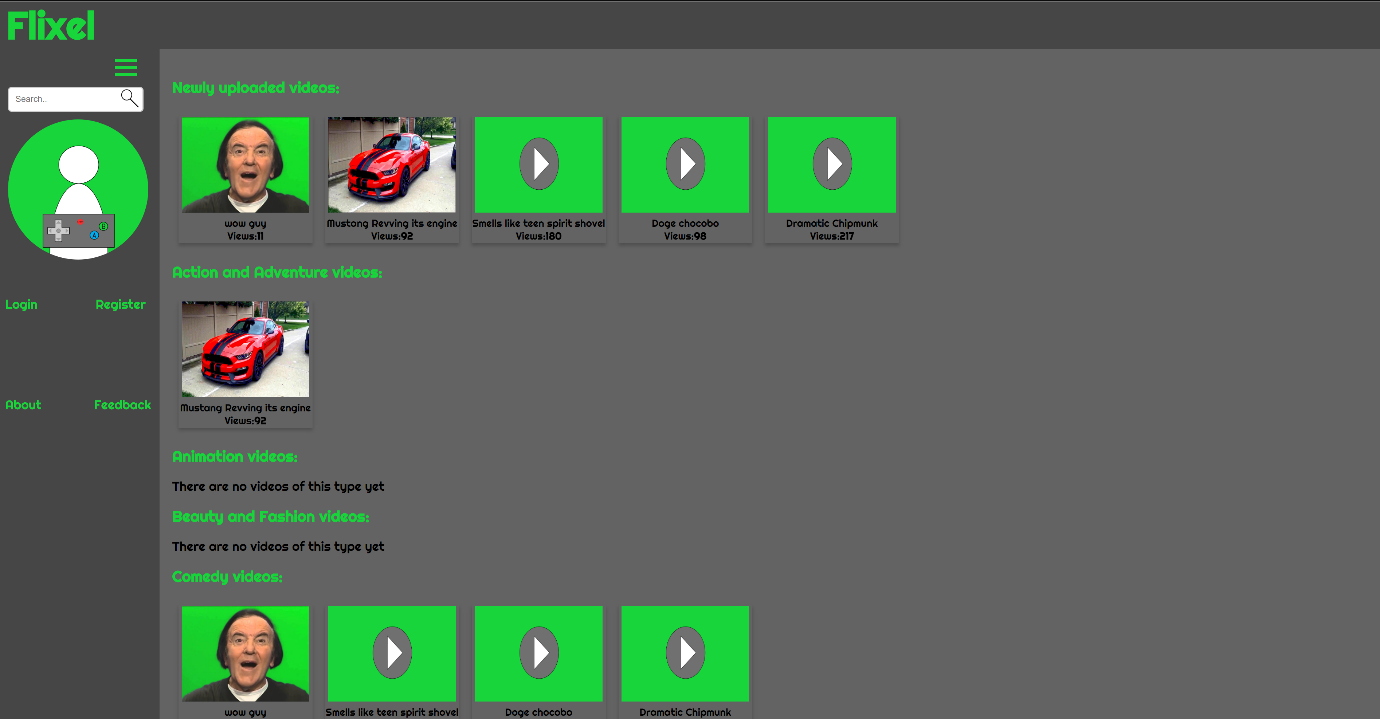
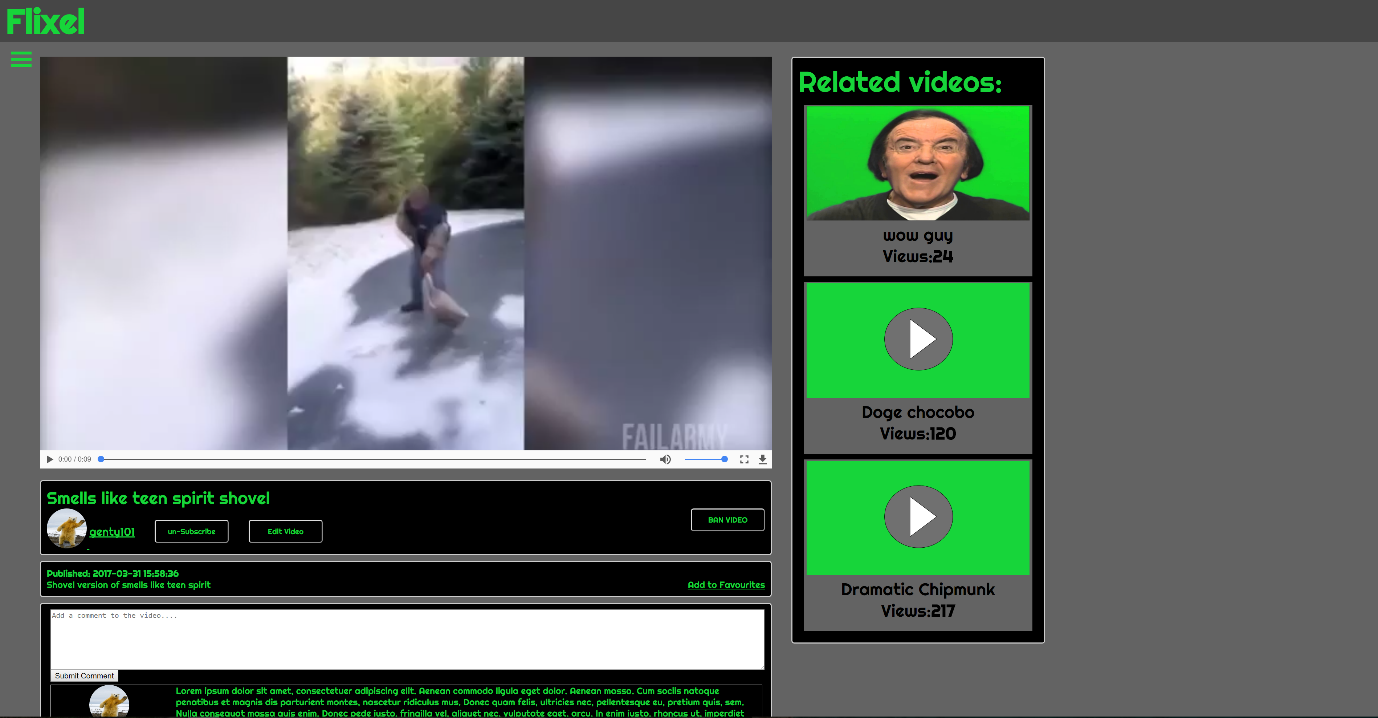


Figure 5 shows the homepage when a user enters the site. From here the user can either pick a thumbnail which will take them straight to a video. The side menu can also be used, seen on the left here. They can either use the search function, which will search for videos and users alike, login/register if the user needs to login or register to the site respectively or leave feedback or see the sites about page if they so wish. If an administrator signs into this side menu, the option to go to the administrator’s page will also appear. The search function will also now show omitted results to other users.

### View page

**Figure 6**



This is the viewing page for the videos stored on the website when an administrator is logged into the website. Both users and admins can make comments on the video, subscribe to the user, flag the video if necessary and add the video to their favourites list (if the user is logged in). If the user uploaded the video or an administrator is logged in, both can choose the edit video if they so wish. Currently this is the only way to delete the video entirely from the website. Administrators also have the opportunity to ban the video outright, using the present ban button seen here. This will lock the video for both the user who posted it and other users from viewing it on the website. An administrator can also un-ban the video using the same button if required. Users will also be able to flag this video, however this has been removed for the administrators due to the ability to ban the video outright. There are barriers put in place so that users that are not signed in cannot make comments or flag the video for example. This means that non signed in users can only view the video.

### User page

**Figure 7**

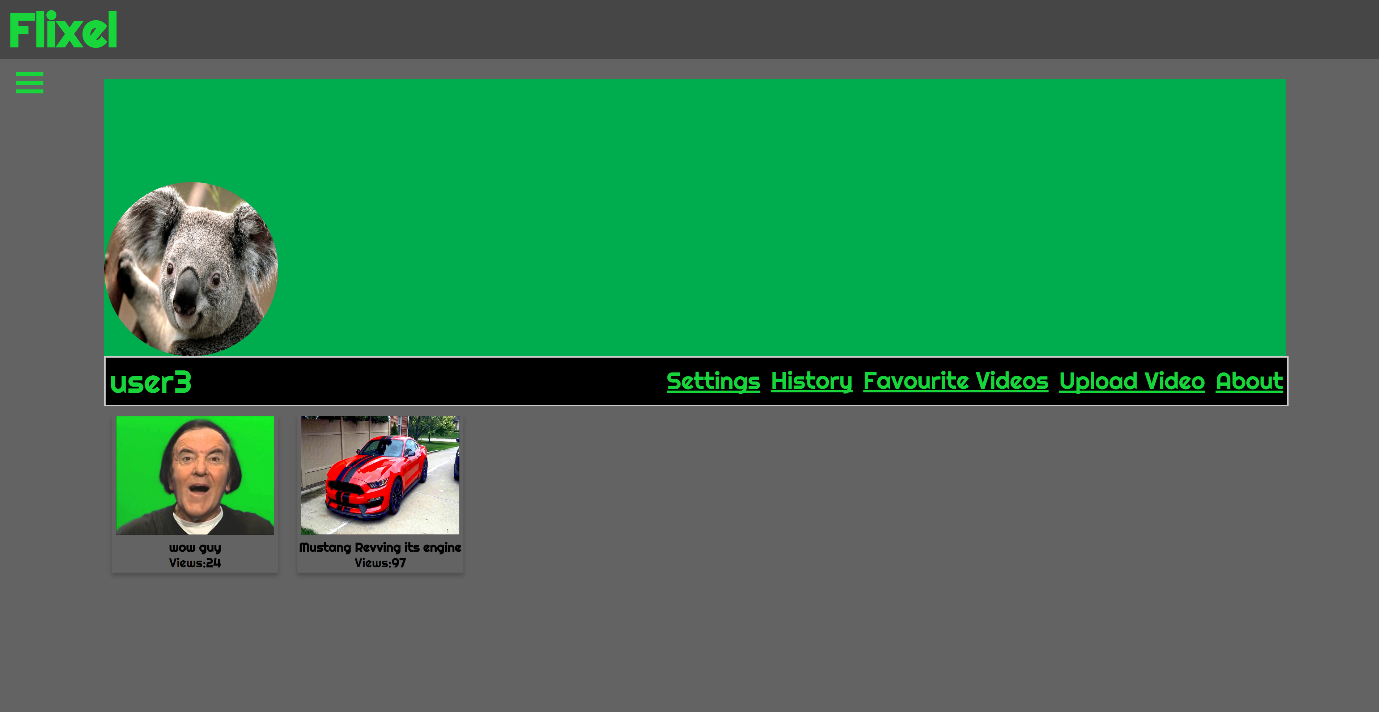
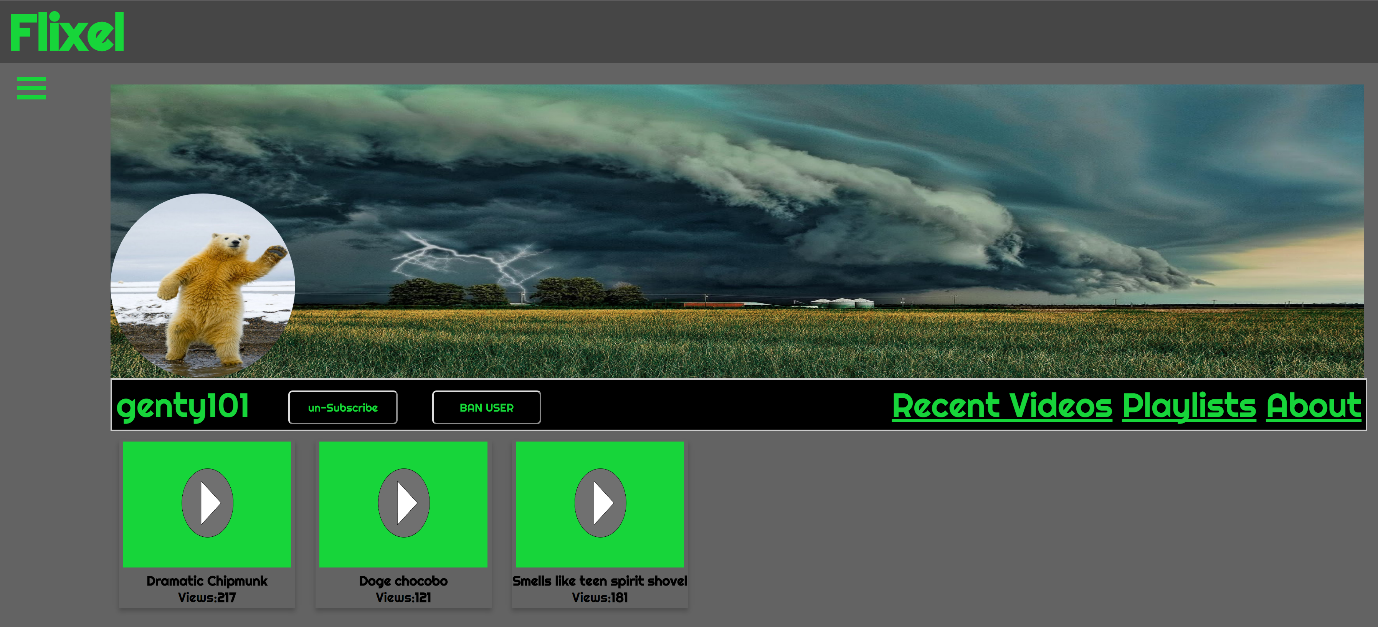


Figure 7 shows a user’s personal profile page. The user’s uploaded videos will be shown underneath the top navigation area. From here the user can move towards their settings page, video viewing history, their favourite videos, choose to upload a video or view their bio page. The settings option is currently the only way a user can alter their profile page. There is also no difference between a user’s profile page and an admins profile page.

### Sub page

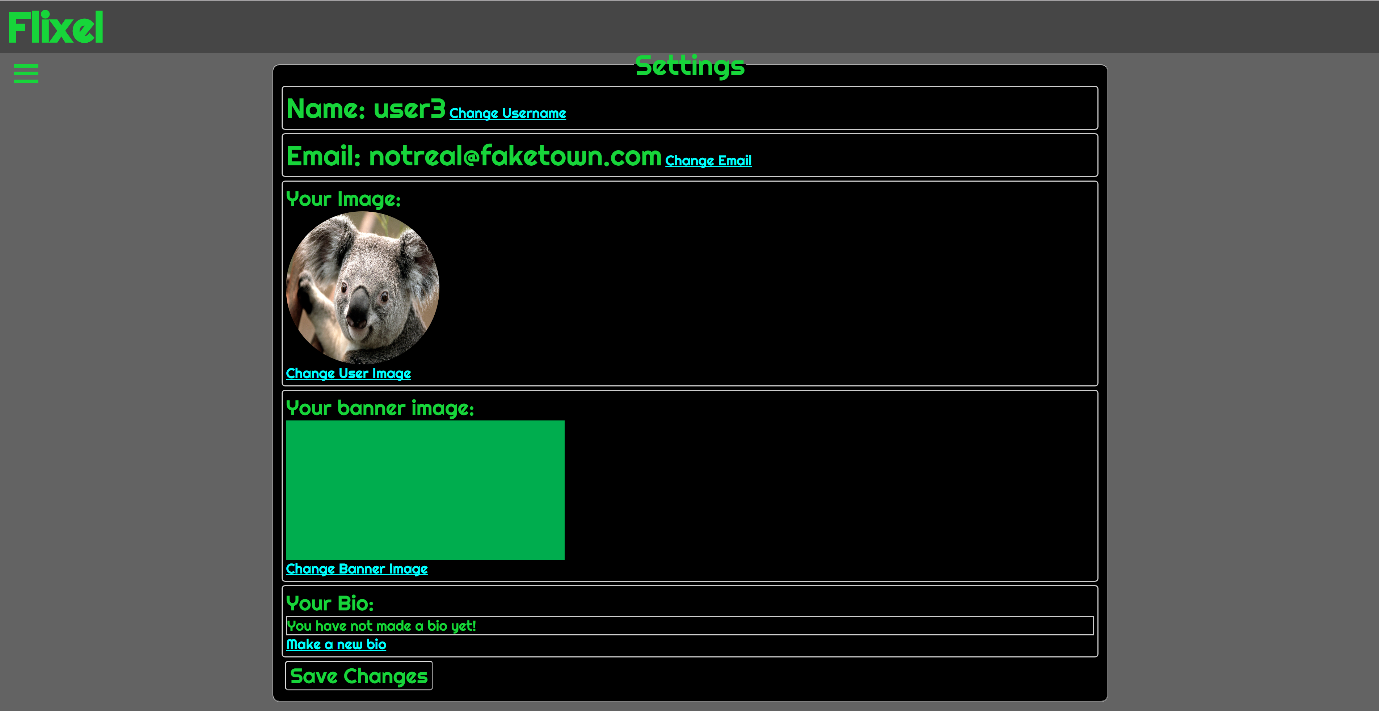


**Figure 8**

When a user visits another user’s profile page, this is the screen they will be greeted with. Again much like a user’s personal page, the layout is the same, however different options are available. The user can visit the most recently uploaded videos posted by the user via the “Recent Videos” link shown here. The user can also see what videos the user has put into their personal playlists by selecting the playlists option. Lastly users can use the “About” link to see the users bio if they want to. Users can also subscribe and un-subscribe to this user by selecting the button next to the user’s username. Administrators also have the ability to ban the said user on this page. This will remove the user from being viewed publically, and ban the user from logging into their account. This will also remove all videos from the site from being viewed publically.

### User Settings page

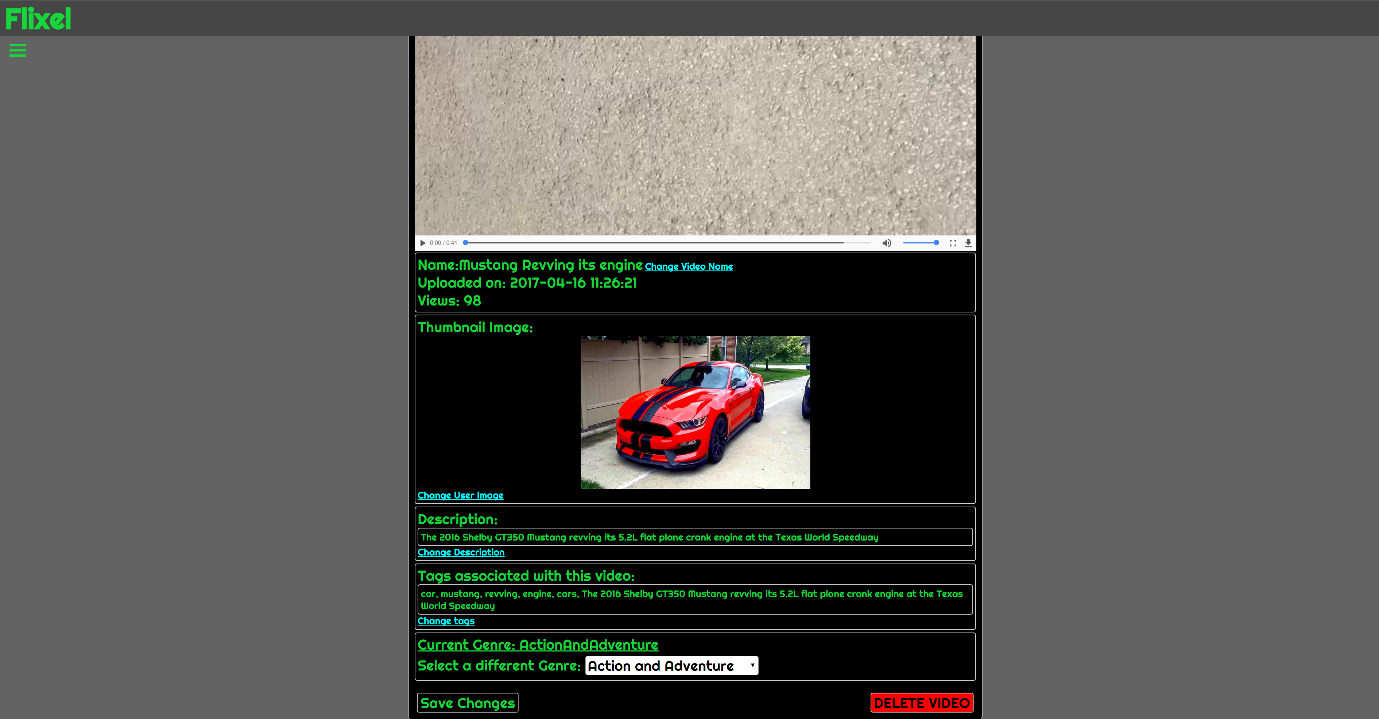
**Figure 9**



Upon entering the user’s settings page, a number of options are available for the user to choose from. The user can alter their username, user image, banner image for their profile page or their biography for their personal section for people to view. Another option is also available to level 3 users which will allow them to submit a request to upload videos to the administrators. Once this has been approved, the user will be able to upload videos. By selecting the save changes option, the user can update their information in the database. If any values are left blank within the form, the database will keep the current settings upon saving the changes. If any errors occur in the process, such as another user having the same name for example, the script will display an error message showing that this change has not been saved.

### Video settings page

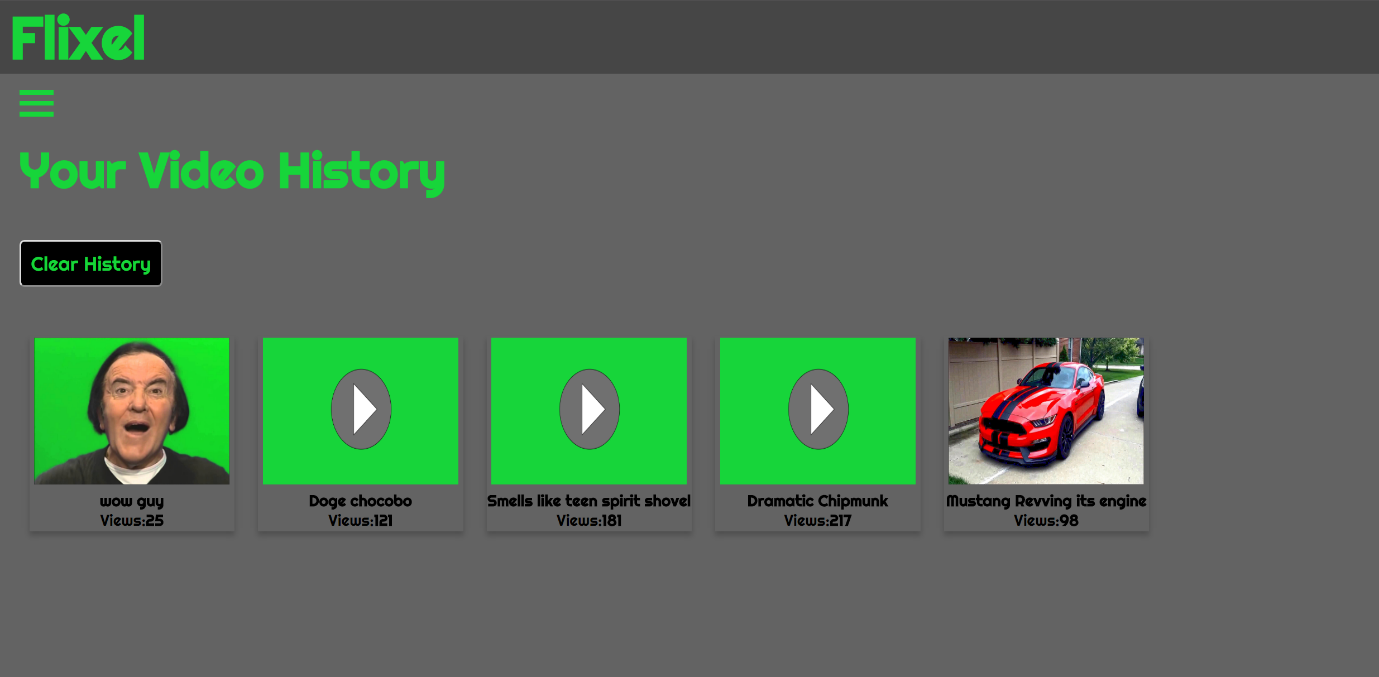
**Figure 10**



Like the user settings page the video settings page has a number of different options to choose from. Users can change the name of the video, thumbnail, description, tags and the genre of the video from this page. An option to delete the video is also available if the user wishes to. However if this option is chosen, all references to the video within the database and the video itself will be deleted from the file host. Administrators also have the ability to edit user videos, if the video does need to be altered. This can still be accessed by administrators if the video is banned. Failsafe’s are in place in case the user cannot change the name to a video that already exists. Instead the script will save all other changes apart from the changes that have caused the error.

### History and favourites page

**Figure 11**



Both the history and favourites page use the same layout and system as each other. A user’s previously viewed videos or favourite videos will be displayed like this format. The user can delete each individual video from their lists by selecting the small cross that appears within each thumbnail. Users can also clear these lists by selecting the option to do so at the top of the list.

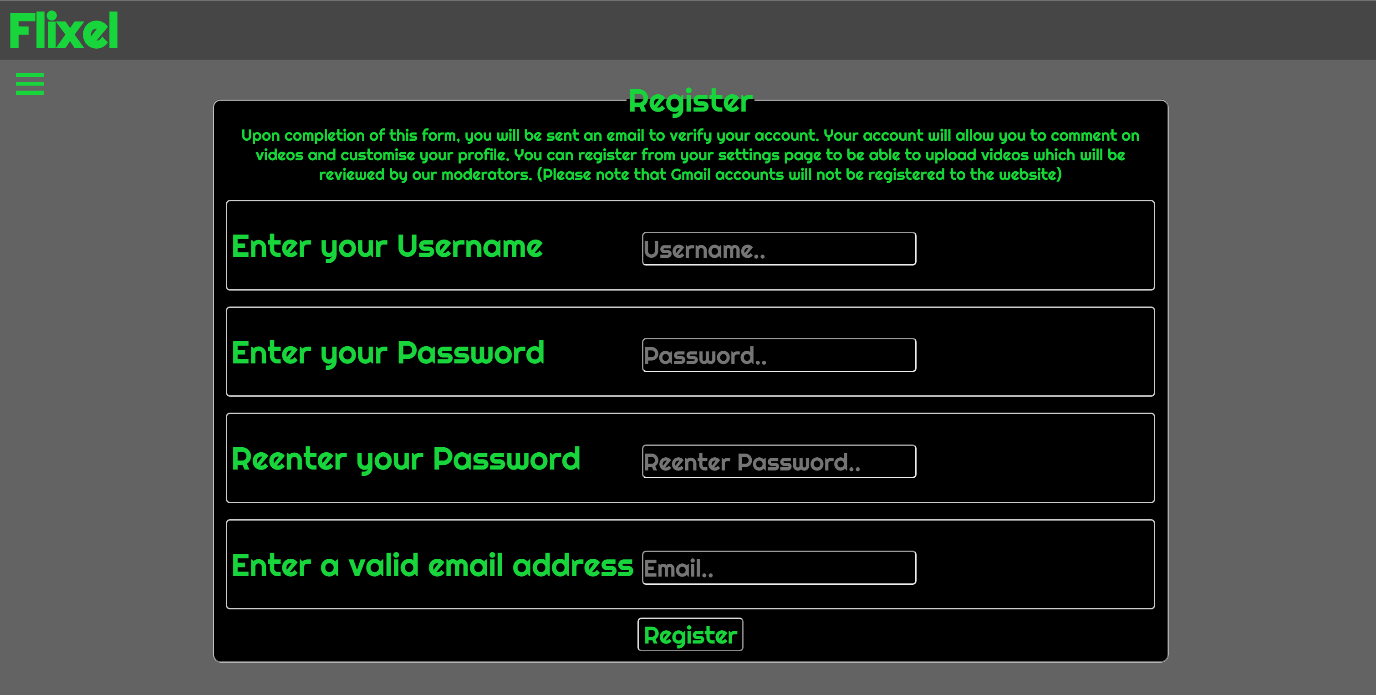
### Video upload page

**Figure 12**



Figure 12 illustrates what the video upload page looks like to all users. Every one of these fields have to be filled out in order to upload a video, besides the thumbnail. If the thumbnail is left blank, the scripting will automatically use a default video image. On upload the page will display a loading circle, indicating that the video is being uploaded. Upon any errors the page will inform the user using a custom alert message. All parameters must be met here however before the video is uploaded to the file host and updated within the database.

### Register page

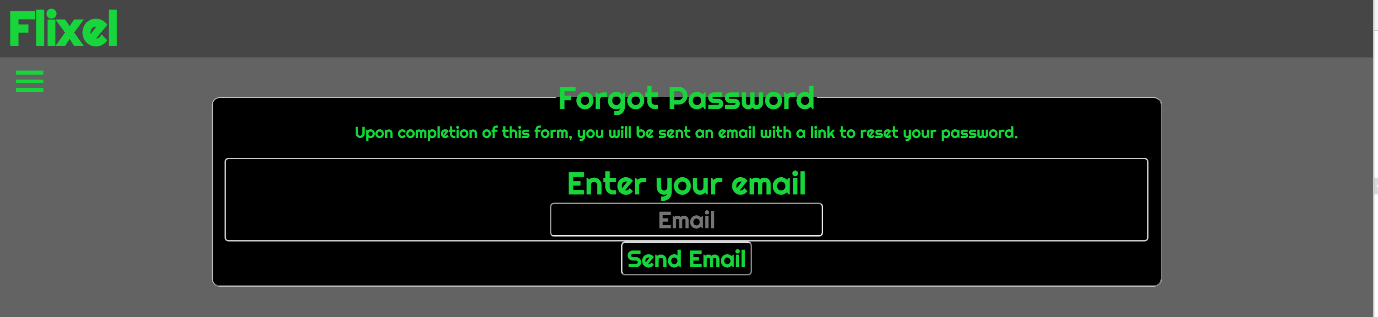


**Figure 13**

Users can register for an account using this webpage. A user can only have a name that is not currently used within the website. Users must also re-enter their password to make sure that their password is correct and memorable. A valid email must also be used so that a registration link can be sent to them. Upon a successful registration the user can then log into the site and use features such as commenting on videos etc. If when the user uses the register link sent to them does not match the database records or fails in any way, the user will be deleted from the database and be told to start the process again. Signed in users cannot access this page and will be redirected if they manage to find this page. For more information on the scripting of this feature please refer to the “Back end maintenance” section of this document.

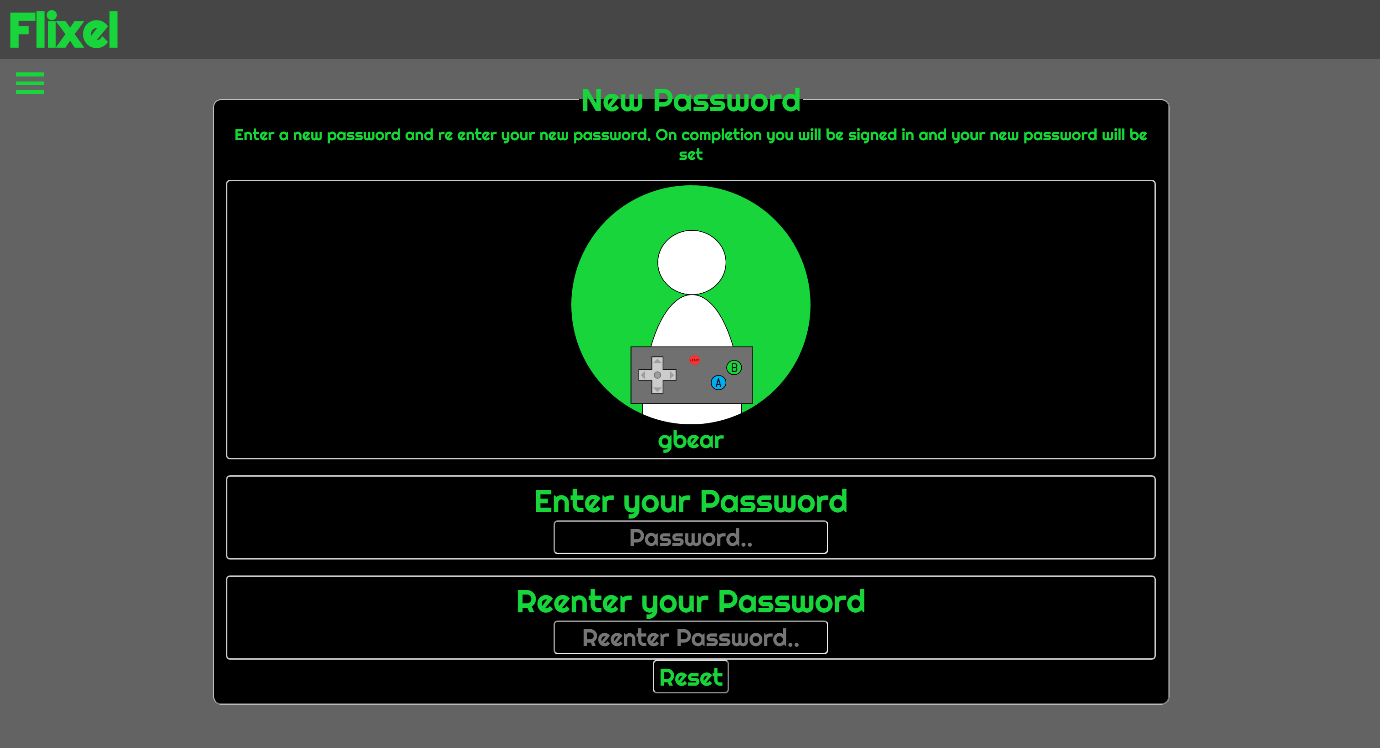
### Forgotten password page

**Figure 14**



If a user has forgotten their password, they can request a new password using this form here seen in figure 14. The user must present their email to this field in order to reset their password. This will make a unique default password for each user and will differ every time that they do this action. Upon completing this form the user will receive an email with the link to reset their password. Here is the page they will be taken to upon selecting that link:

**Figure 15**



Now the user will be able to reset their password to a new one. If the link was to fail in any way the user would be told to start the process again. This is due to an unknown default password being used for their account. By going through the process again the user will be given another randomly generated password, thereby rectifying this issue.

### Admin page

**Figure 16**

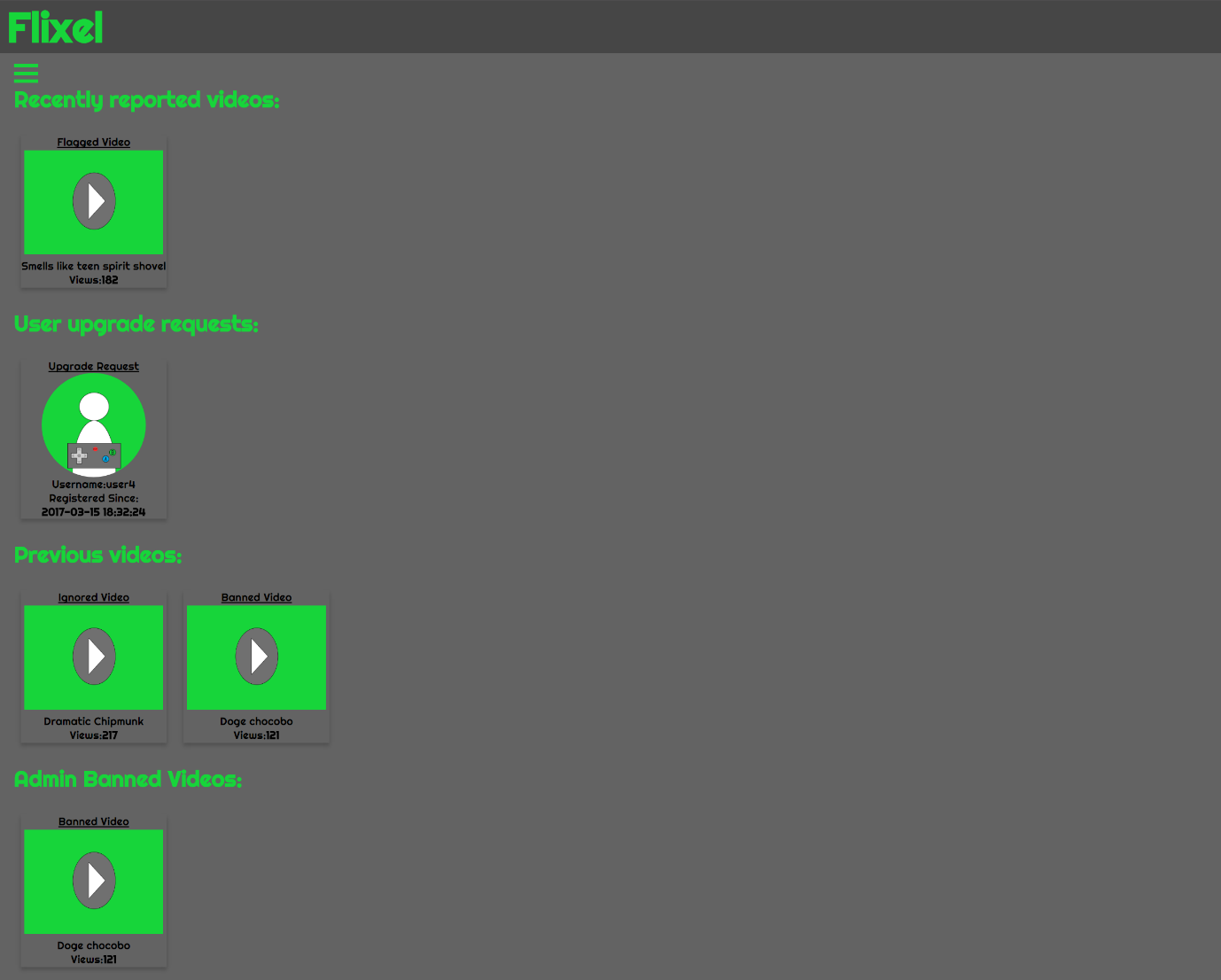
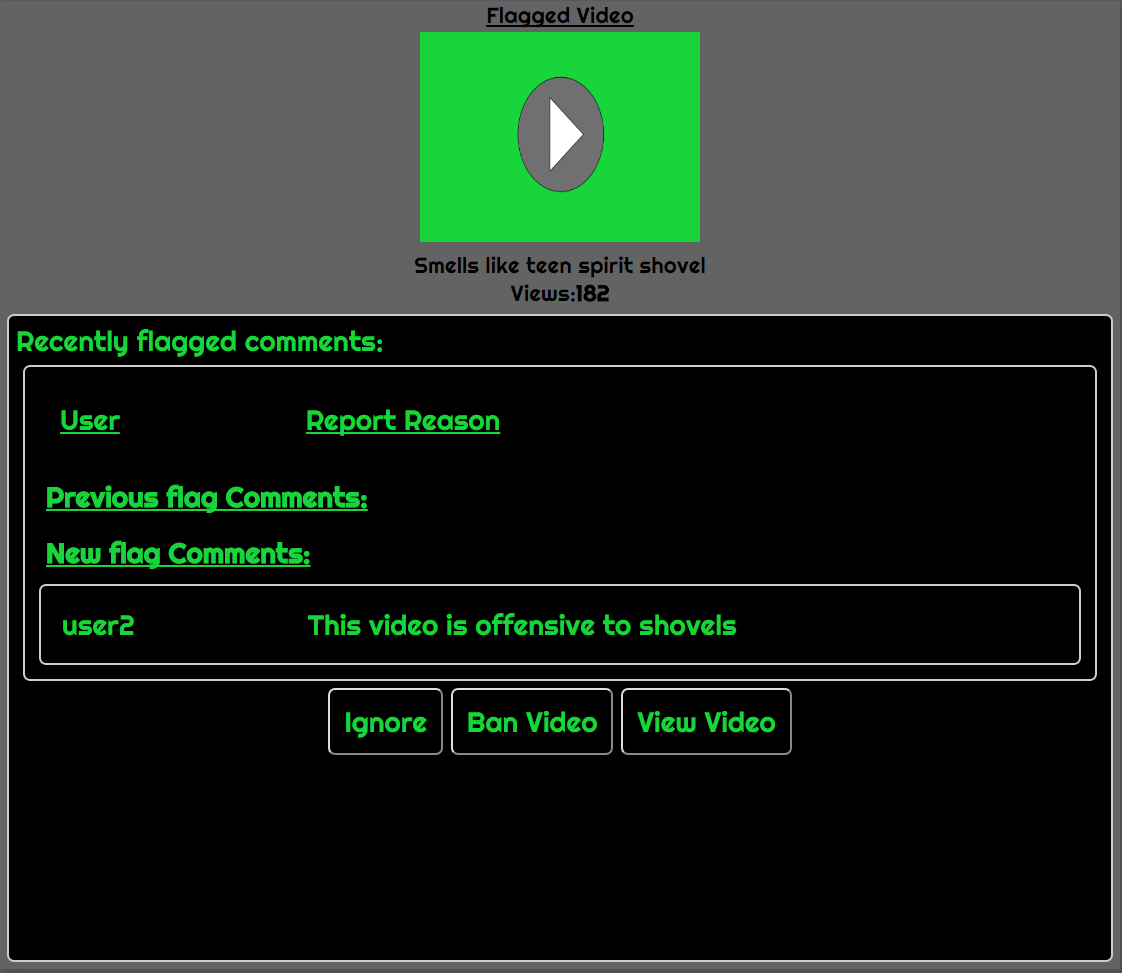


Figure 16 shows the administrators page. From here there are a number of options that the admin can choose from. Firstly at the top of the page there are recently reported videos. These are thumbnails which will appear when a video has been flagged by one or more users. If a video has been ignored or banned before, the old flagged comments will also be shown here. Figure 17 below shows the thumbnail when it chosen:

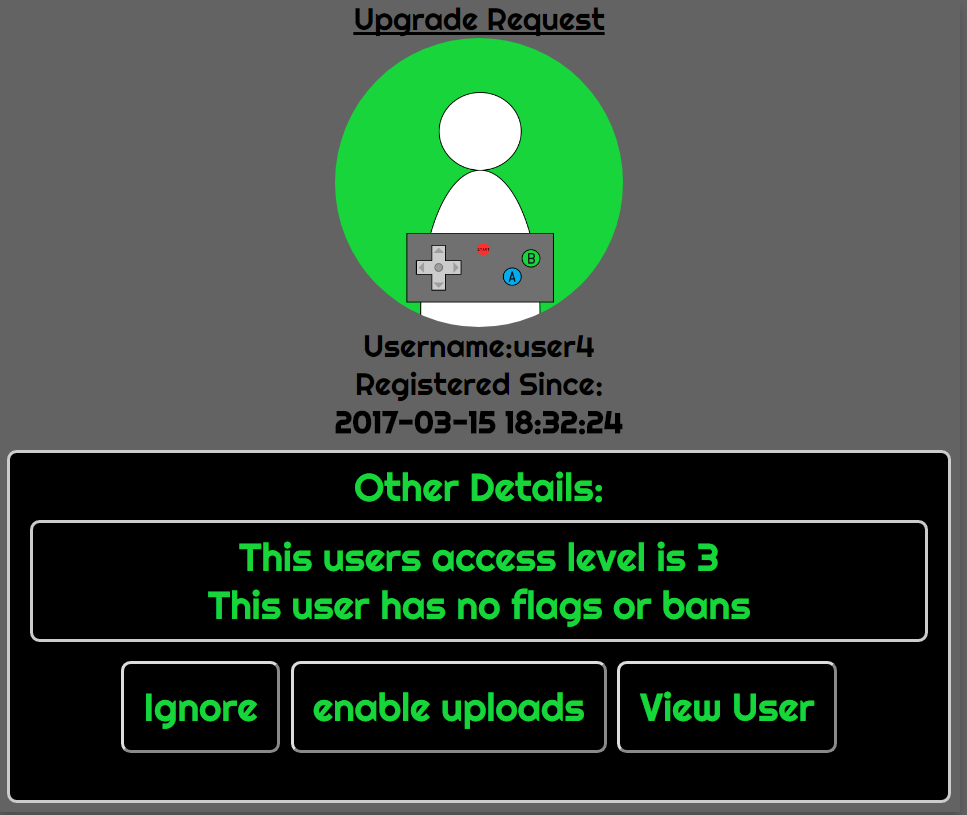
**Figure 17**



The administrator then has 3 options. The video can be ignored which will move it to the previous videos section to show other admin that this video has had flags but another administrator has ignored it. The administrator can outright ban the video on the spot depending on the flagged comments given. This will then move the video to the admin banned section on the page. Or finally the admin can view the video for their selves and then either ban it or ignore it while on the video page.

Figure 18 below now shows the upgrade request for a user that wants to upload videos:

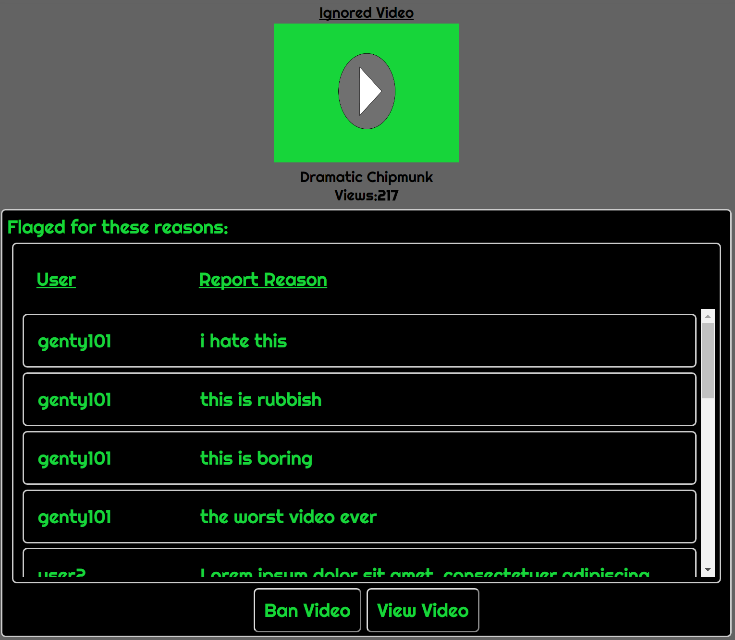
**Figure 18**



This thumbnail shows basic details of the user and if they have been banned or flagged before. Again the administrator has the option to ignore this request, enable the user to upload videos or view the users profile to then make a decision.

The next figure shows a previous video that has been ignored by an administrator but shows the reason as to why it was flagged in the first place:

**Figure 19**



The administrator can again ban the video if they think that these reasons are valid or if they mistakenly chose the “ignore video” button. The administrator again has the opportunity to look at the video once again to before making another judgment.

**Figure 20**

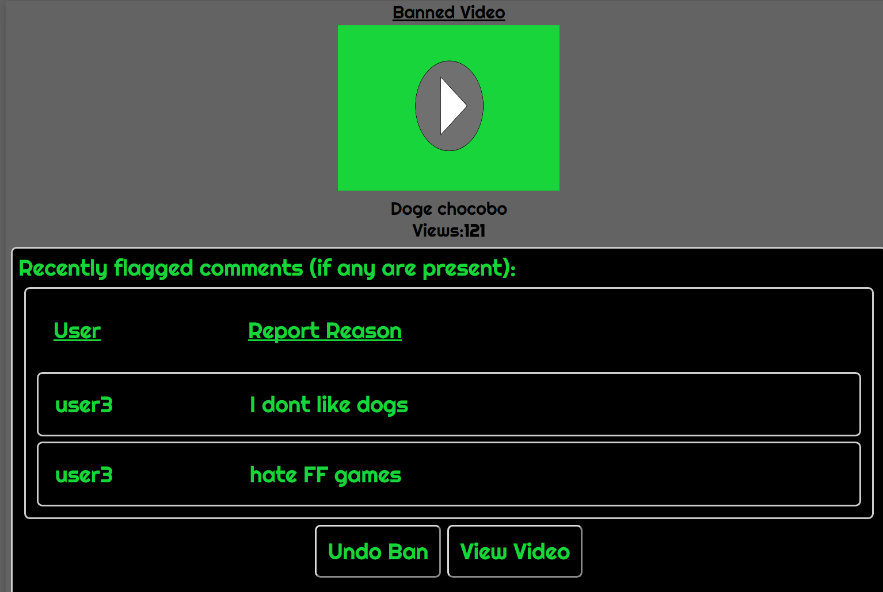


Figure 21 now shows a thumbnail from the admin banned videos section. This will once again show the reasons for why the video was banned if any are present. If the administrator banned the video with no flags beforehand then this section will be empty. The administrator can then make the video suitable for public viewing again or review the video by selecting the view video button.

## Back end maintenance

This section of the document explains certain key scripts and general maintenance of the website without using any of the tools within the website. Snippets of code may also be shown to better explain better how some of the features within the website work.

### DBConnect Script

This script is literally to connect to the database to send requests to it and receive replies from it.

The reason that this has been kept separate is to make it easier to just call it every time it is needed, rather than having to type out this section of code out in every single script or page. This code also calls on another script which is the config file. This file stores the host, username, password and database name for the connection. This again keeps this data behind another layer of scripts.

### Functions script

The functions script is mainly used for gathering video thumbnails and user subscriptions. Initially the script was meant to house the search script and the DBConnect script for connecting to the database. However it was found to be more efficient to keep them separate, due to the DBConnect being used in every single script or page. The functions script was not able to complete ajax requests correctly, which also led to the search script being separate. Here is a snippet of the thumbnail function:

As we can see the function can be thrown a genre or tag to filter the amount of videos thumbnails it retrieves. If the script is thrown a \* then it will search for all genres or all tags. The results are bound to an array and then returned. The results are limited to 10 per query. This can be altered by changing the LIMIT function in the query if needed.

This function fetches a user’s subscriptions. This simply uses a user’s ID passed to it to fetch all of a user’s subscriptions. The query this time however uses an INNER JOIN to fetch other data such as the username in this case. The query uses the Subscriptions table and joins the user’s table query via the UserID fields being linked. Other data can be received via adding a Subscriptions or Users in the query field followed by a dot. After the dot another field can be chosen. For example Users.Email would return the users email in this case.

The third function is the same as first function. However this will also be thrown a user ID which will then only find videos that correspond with that user, along with a genre or tags given to the function.

### Search function

The search function uses an external script which incorporates the use of a WHERE LIKE query for the SQL statement. This is shown within the code snippet below:

This code will omit banned results for normal users but keep the banned results for administrators. This means that administrators can still search and go to banned content to further investigate it or un-ban the video. This script searches both users and videos and the ajax scripting will display them, one underneath the other. To alter how the search function works, the search key can be altered with the percentage signs. Changing these signs will change how the SQL query will search for records. For example the signs around the key will look for that word in any position in the selected column. A percentage before the key will look for words beginning with the key given. Setting the percentage after the key will look for words that end with the key.

### User registration

All users must register with the website in order to use its full features. The script used to register the users works by entering the user into the database and then assigning them a token and a token validity time. If the token validity time or the token is incorrect when the user is emailed the link, the user will be removed from the database entirely and will have to restart the process.

The code snippet above shows how the token it set and how the validity of the token is also set. The token is set by using the current timestamp as a string. The user ID is then added to the end of it to make it even more unique. The token validity time is set in seconds, which in this case is 10 minutes. To change this simply alter the time within the date function. The mail function seen here is used to mail the users with the link and details. The variables explain what each section does within this mail function. Again change these values to alter the contents of the message.

### File uploading script.

This script has a lot of variables and variable checks to ensure that the video is uploaded is not corrupted or incorrect in either the file host or the database.

This snippet above shows the variable storage and specified file paths for the script and database. The target directories in this case are for the code to insert the files into the file host. The DB\_dir variables are used for the database directories so that when these files are called within a web page, the page can find the files. The posted values are also gathered here to be used later within the script. Note there are variables that have been designated with a 0 or 1 value such as $imageUploadOK or $imageDefault. These are used for checking that the file uploads meet the requirements to be uploaded. Upon an unsuccessful upload these variables will be set to 0 apart from the $imageDefault variable which is used elsewhere. File paths could be determined using $\_SERVER [‘DOCUMENT\_ROOT’] variable, which will fetch the server route without the need to type it in manually. This would make the scripts be able to be ported to other file hosts without needing to alter the script.

These are some of the checks which are used for both the image upload and the video upload. Firstly the file will be checked to make sure that it is set. If this is not set then it will return $imageUploadOk as 0. The script will then exit with an error message telling the user that the script has failed because of one or multiple different checks. This also applies to the video upload file also. The only difference with the image checks is that this can be intentionally left blank. If it is then it bypasses all checks by using another variable called $imageDefault and set it to 1.

### Login script.

The login script runs by fetching a user’s information based upon a username or email that they post to it. To fully log in the password is decoded from its current hash and compared to the password it is given, using the password verify function. Numerous session variables are also set as a result which includes the user ID, username, access level and user image. The user’s subscriptions are also fetched as a direct result of a successful login and the database is updated to show the last login time of the user. The only time the script will not log a user in is as follows:

* If the username or password does not fetch any results
* The user is banned, which is defined in the database
* If the user has not fully registered yet
* The password given does not match the password given by the user

All SQL queries used here are self-explanatory. However the user subscription query is identical to the one used in the functions script. Please refer to this to learn how the INNER JOIN function works within this SQL query.

### New password scripts

The system for resetting a user’s password, uses the token that was required for the user to register beforehand. This time however the token is used, hashed and stored as a new temporary password within the database.



Shown in this code snippet, the script will hash the token and store it into a new variable. The token is then shuffled so the arrangement of the current token is now different. It is then re-entered into the database so that if it is needed again for this action it will be different. The newly hashed variable is then also stored into the database as a temporary password. The old token will then be sent in a link to the users email. The password hash could be changed here so that instead of using the hosting services password default encryption, a SALT could be used instead. This would require more resources to do this however so it could be taken into consideration, providing this doesn’t overburden the hosting systems.

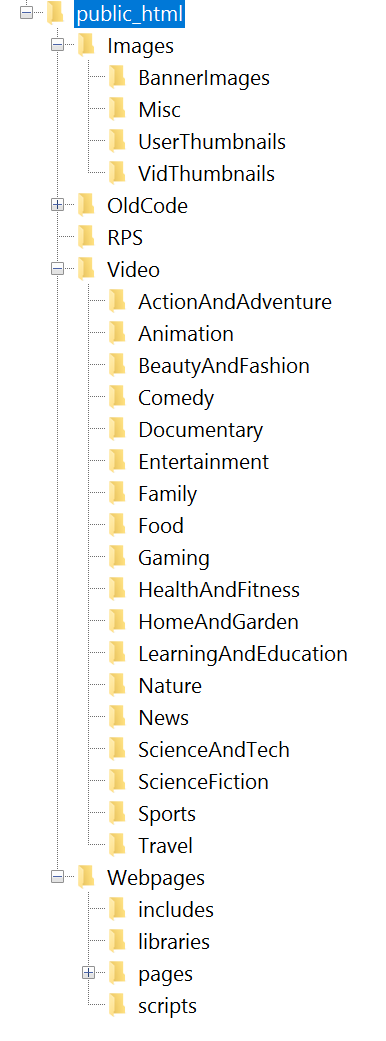
Once the user has selected the created link sent to their email that the previous script made. Here is the code snippet that checks that the link works correctly in the newPassword.php page.

The email and password are received from the link. A record is then fetched from the table based upon the email given. If the statement retrieves no rows than the user is sent to the homepage due to the email being incorrect. If a record is retrieved then the password will be verified from the one given in the URL. Again if they do not match the user will be redirected to the homepage to start again. This could be altered to accept usernames, however bear in mind that usernames would make it easier for an outside source to try and hack into the targeted users account. Emails are never shown to other users.

Assuming that the page has accepted this link. The user will then be made to input a new password. This password will then be hashed once again and stored within the database. Upon completion the script will then log the user in as normal. Another check is also added to make sure that if the user is banned, then the password will be updated but the message will still return that they are banned from accessing their account.

### File host setup

The file host is set out in this directory tree shown in Figure 21 below:



**Figure 21**

As we can see there are 3 major files that will be used, Images, Videos and Webpages. All of these are self-explanatory as to what they contain. Note that each of these contains subfolders. Images have subfolders for video thumbnails, user thumbnails, banner images for user profiles and any other miscellaneous images. Video thumbnails could be categorized the same as videos, however they are going to be updated more often than that of the video files which is why they have been intentionally left in one folder. This also makes it easier for the database to be changed if the images are updated. Videos are put into categorized folders to make it easier to define them both in the database and if the administrator needs to alter or delete a certain video. It will be much easier to find than having to search through a large folder with all videos within it. For any image or video files a CRON job could be used every 24 hours if the site was at maximum traffic. These CRON jobs can be set to remove files from the file host. This would simply require a normal PHP command to delete any files that do not match any results within the database.

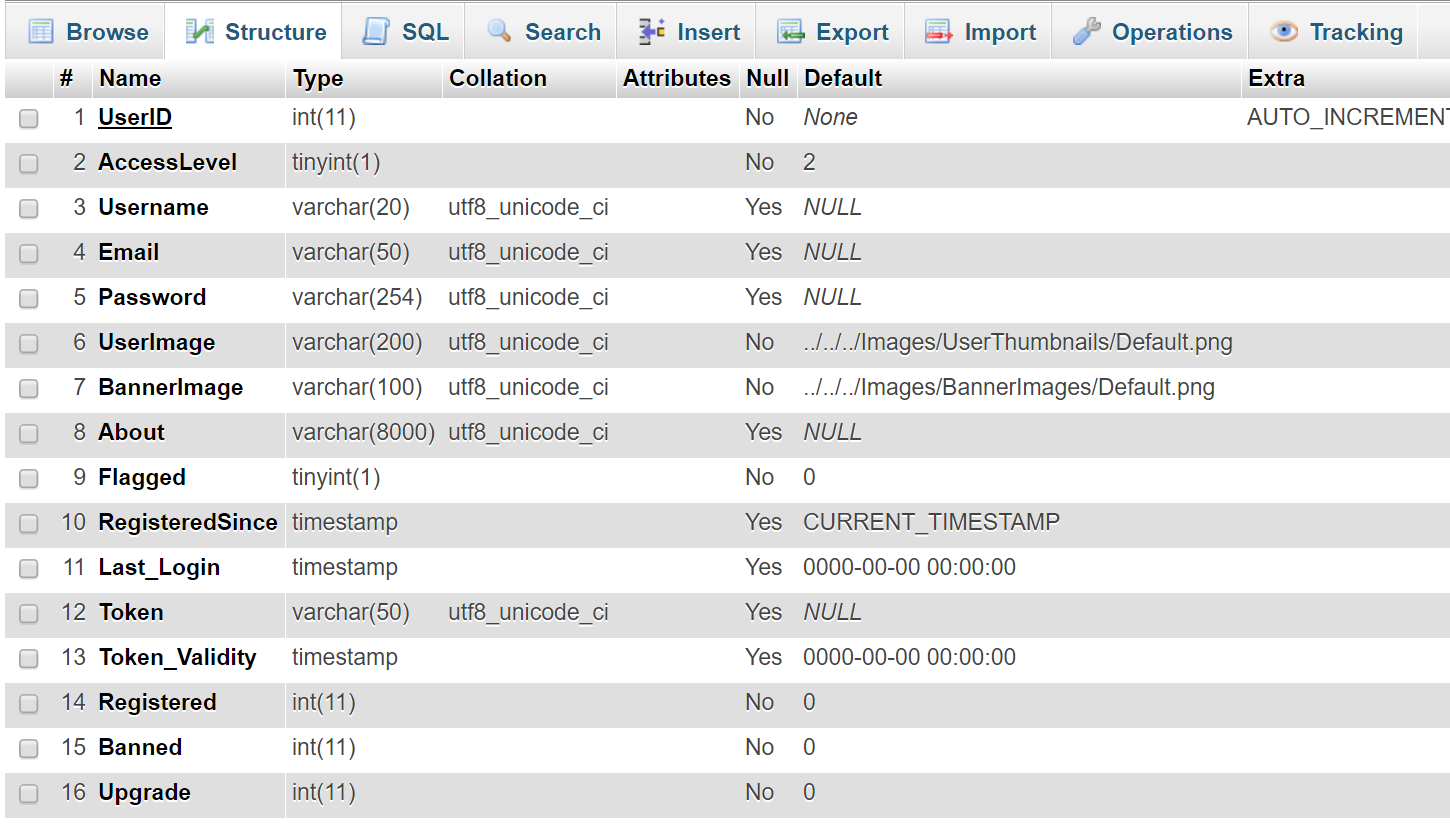
Finally the Webpages folder holds all of the scripts, webpages, library files and CSS files that the website needs to run. All users will be redirected to the pages subfolder upon entering the site. This also holds subfolders for each webpage. This is only used so that users only need to type the subfolder address into their browser and will hide the file type that they are viewing.

### phpMyAdmin database

The database has been written and maintained using the PHP my admin application. All of the tables and records can be accessed here and altered if need be. This manual will only cover the 2 biggest tables within the database which is the users table and the videos table.

#### Users Table

The Users table holds all of the user’s information within it. All passwords stored within this database are hashed, meaning that if the passwords were compromised in any way, then the user passwords would not be readable without it being decrypted.

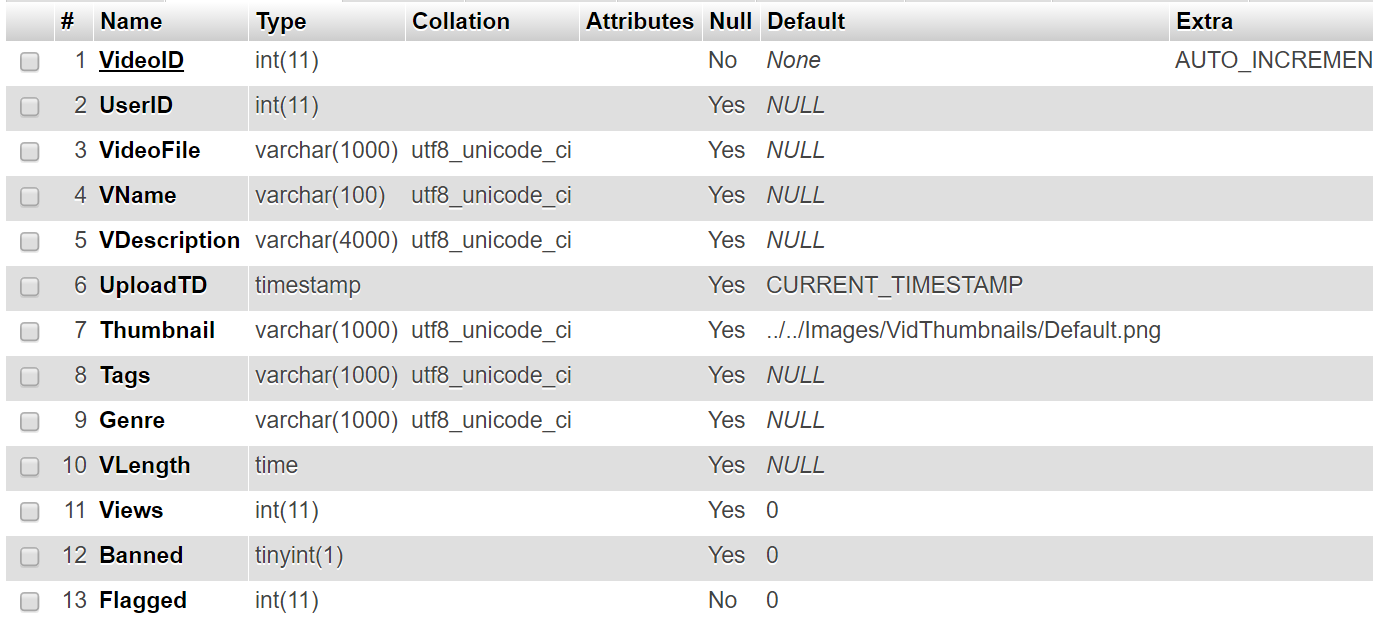


**Figure 22**

Figure 22 shows how the table structure is laid out. Most fields are defined a default value when a record is created. A default user image and banner image are set here when a user first registers with the database. This can be changed if required, the directories shown are only for browser sessions NOT physical file directories. Access level will be set as a default of 3. Here however it has been set to 2 for debugging purposes. All true or false fields such as registered will use either a 1 or a 0. Again as a default these will all be set to 0. Token validities may differ from current system time due to the fact that the server may be set to a different time. Currently this is the only way to permanently delete a user from the website. Simply delete the record as required, however note that all videos associated with them, subscription lists, favourites and history lists will still remain and will have to be deleted manually.

#### Videos table

The videos table uses much less columns than that of the user table. The video table only holds basic data such as its name, the user who uploaded it, the video file path etc.



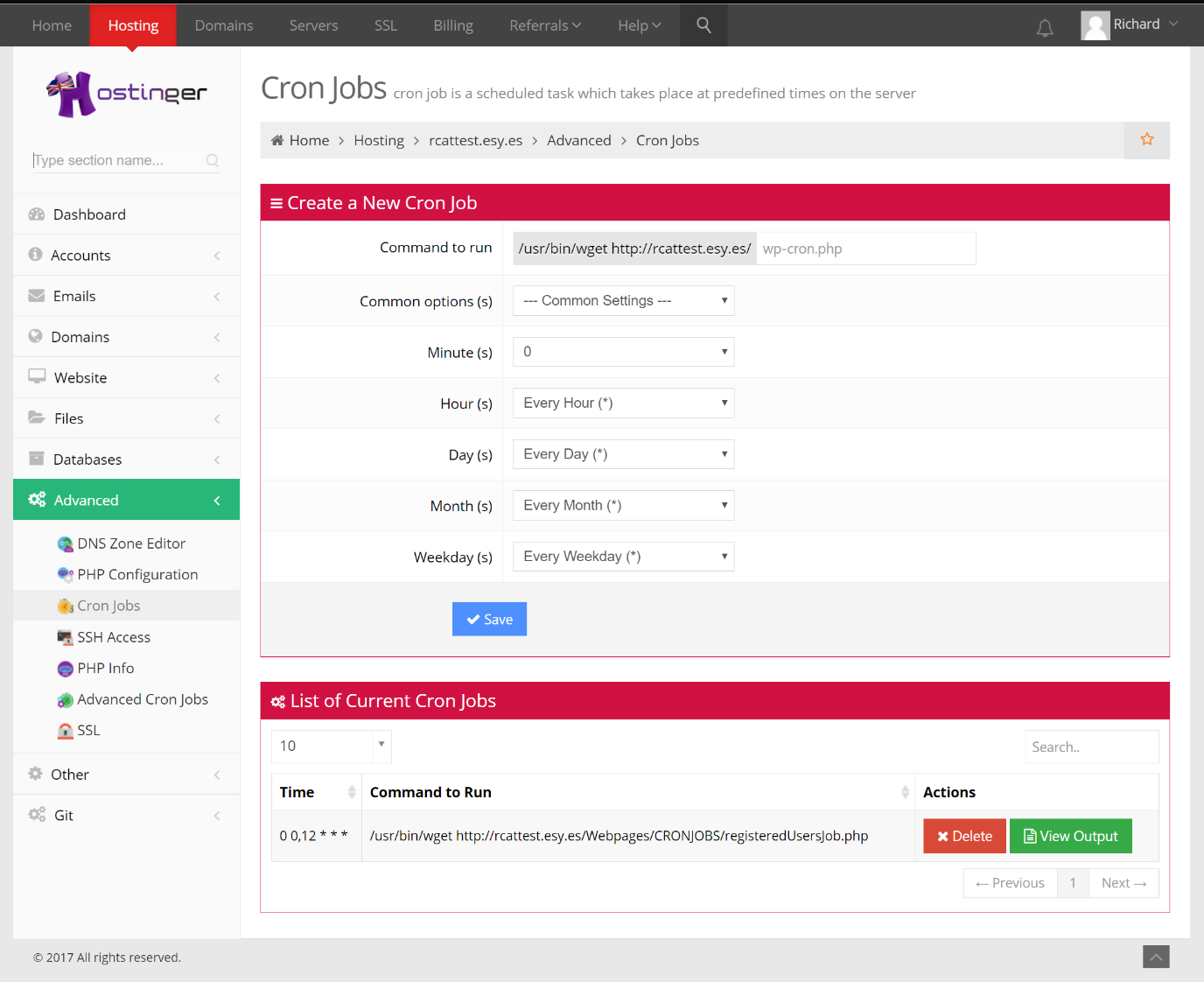
**Figure 23**

The only defaults here are for when the video was uploaded, i.e. when a record was created for it. The current timestamp is recorded for the upload date and a default thumbnail image is given if the video does not get uploaded with one. Views are also set to 0 by default and the video will also not be banned or flagged, so these values by default will be set to 0.

## CRON jobs

A CRON job is a service where selected files will be executed at certain times which is set within the CRON manager seen here:

**Figure 24**



The CRON job shown runs a registerUserJob.php script. This script is simply designed to drop any records in the users table of the database if a user has created an account but not registered fully. This script is ran twice daily and can be altered by editing the script it uses or by deleting the CRON job and adding a new CRON job. As seen in figure 24 any time periods can be set, including monthly jobs. Simply alter these values to make certain scripts run hourly, daily, every 10 minutes etc.

# Bibliography

YouTube. (2017, N/A N/A). *Community Guidlines*. Retrieved from YouTube.com: https://www.youtube.com/yt/policyandsafety/en-GB/communityguidelines.html

# Appendix

## Testing log

This is a log of varying self-tests that have been done for the web application. The way the tests will be set out will be what the test covers, the date of the test, the expected results of what the test should produce, the actual results of the test and finally any changes made if necessary.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **What the test covers** | **Date** | **Expected results** | **Actual results** | **Changes (if needed)** |
| 1 | This test covers the loginScript.php. This will test whether it will log the user into a web session | 22/02/17  13:53 | The script should accept an input from the user. If the database matches a username and a password, then session variables will be stored. In this case we will use valid details | Test was unsuccessful, bind param would not bind username to the statement | $\_POST cannot be used in a bind param function. This is now bind to a variable beforehand. |
| 2 | This test covers the loginScript.php. This will test whether it will log the user into a web session. With new fix in place | 22/02/17  14:02 | As above | Test was successful. All session variables were stored as needed. | N/A |
| 3 | This test will use the loginScript.php once again. However this time, invalid results will be thrown to the script | 22/02/17  14:06 | The script will not store any session variables. Instead the user will receive a message showing that their login was unsuccessful. | Test was successful. | N/A |
| 4 | These array of tests are to check that the uploadPage.php and fileUploadScript.php are working correctly. This test in particular will check that the video file will upload to the file host and update the database to its location | 26/02/17  12:17 | The fileUploadScript.php should receive a file from the uploadPage.php. From here the script should check that the file is set, the correct size, the correct type or if the file already exists. Upon going through these checks the file will then be uploaded to the file host and the file path should be stored within the database | Test was unsuccessful. The file type checker returned that the video was not the correct file type despite it being correct. | Checker was checking to see if the video file was correct instead of it being incorrect. This has now been rectified so if the function does not equal the correct format, the file upload will be aborted. |
| 5 | Another test of the one ran above, with new fixes in place. | 26/02/17  12:39 | As above | Test was successful. Video file was uploaded to the file host and file path was uploaded to the database. | N/A |
| 6 | These tests will now test the checkers within the fileUploadScript.php. First we will try to upload with no video file attached | 26/02/17  12:43 | The script should throw an error and display that no file has been passed to it. The database or the file host should not be updated. | Test was successful. Script found that the file was unset and stopped the script. | N/A |
| 7 | This test will check that if a video file already exists. For this we will attempt to upload the same video from tests 4 and 5 | 26/02/17  12:52 | The script will stop if the file upload if a path already exists with the same name. | Test was successful. The script found the file already existed and terminated the script. | N/A |
| 8 | The script will check that the file is not over 8MB. We will now attempt to upload a video file with a larger file size | 26/02/17  12:57 | The script should stop if the file size is too large | Test was successful. Script self-terminated appropriately | N/A |
| 9 | This test will now cover trying to upload a video that is not an mp4 format. For this a non-mp4 file will be uploaded to the script. | 26/02/17  13:12 | Script will again stop and terminate upon entering the if statement | Test was successful. Script performed as required | N/A |
| 10 | The fileUploadScript.php has now had more code added to it. An image file can be uploaded with the video file. This test will be to see if the file will be uploaded. | 26/02/17  14:46 | The fileUploadScript.php should receive a file from the uploadPage.php. From here the script should check that the file is set, the correct size, the correct type or if the file already exists. Upon going through these checks the file will then be uploaded to the file host and the file path should be stored within the database | Test was successful. Same checks were improved from earlier tests and were not hit upon submitting a valid image file | N/A |
| 11 | This test is to check that the new image system is working correctly. If no file is uploaded then the script should default to a pre-defined image. | 26/02/17  14:51 | The script should correct the file path to a default location and then only update the database, not the file host. | Test was unsuccessful. Other checks were still being hit despite an else if statement meant to bypass these checks | New parameter created. This parameter is used to bypass checks. If the parameter is true then all checks should be ignored |
| 12 | A new test to check that the newly improved defaulting system works correctly. | 26/02/17  15:17 | The script will now use a new parameter to skip checks entirely if no file is uploaded. Upon completion the database only should be updated with the default location of the default image | The test was now successful. All checks were bypassed as required and the database was updated with the default file-path. | N/A |
| 13 | This test will now cover the code that checks that the image file is an actual image file. An attempt to upload a non-image file will now be carried out. | 26/02/17  15:19 | The script should return out of itself and display an error message | Test was successful. File was not uploaded. | N/A |
| 14 | Another image file with the same name will now be attempted to be uploaded using the script. | 26/02/17  15:25 | The script will stop the upload and show an error message explaining that this file has already been uploaded. | Test was successful | N/A |
| 15 | Image files have been limited to a maximum of 5MB. An image file which is larger than this will now be attempted to be uploaded | 26/02/17  15:32 | The script will stop the upload and display an error message corresponding to this check. | Test was successful | N/A |
| 16 | A check is in place to only allow certain image types. These image types are jpg, gif and png. A “TIFF” file will be uploaded instead to test that this check is working. | 26/02/17  15:38 | The script should stop the upload and display the corresponding error message. | Test was unsuccessful. No matter what type of image file is uploaded the script still uploads the file regardless. | Problem was in the if statement. Statement used “OR” criteria instead of “AND” criteria. This is due to the fact that it is **not** looking for these values. Therefore the “AND” criteria rectifies this issue. |
| 17 | Same test as above but with the new fixes in place. | 26/02/17  16:14 | As above | Test is now successful. | N/A |
| 18 | Now that both file uploads work correctly, this test will make sure that the other data, such as the video name, description and tags will post to the fileUploadScript.php correctly | 26/02/17  16:31 | The script should receive the information posted by the form in the uploadPage.php. The data should then be uploaded to the database as a new entry. | Test was not successful. All data was entered into the database correctly. However the wrong file path for the thumbnail image was being entered into the database. This was unnoticed in earlier tests. | Wrong file path was used in the header of the script. This has now been rectified to correctly store the better file path |
| 19 | The test above will be run again with the new fixes in place | 26/02/17  17:04 | As above | Test is now successful. | N/A |
| 20 | This test is to check that the registrationScript.php is working correctly. A new user will be signed up to the site from scratch. The test is successful if the user is added to the database and the | 28/02/17  11:16 | The script should register a user, given that a valid email, username and password is used. Upon completing properly the script should then send an email to the registered user. | Test was unsuccessful. Timestamp field in the database would update on change. Meaning that the token created was technically valid for a shorter amount of time | Field was changed to default time upon record entry. Now the timestamp will not change if any field is updated. |
| 21 | Same test as the one above with the new fixes in place | 28/02/17  11:36 | As above | Still unsuccessful, valid token time cannot be converted correctly. | Incorrect string to date format was being used. This has now been converted to the correct timestamp format. |
| 22 | As above with new fixes in place again | 28/02/17  12:20 | As above | Again the test was not successful. No email was sent to the user as required. | There was a typo error in the mail function. An erroneous space caused it to stop working. This has now been recitifed. |
| 23 | As above with new fixes in place | 2/03/17  11:00 | As above | Test is now finally successful. Relevant data has been added to the database and an email has been sent as requested. | N/A |
| 24 | Once a user receives an email with the link to register their account. They will be sent to the verifyScript.php function. This test will make sure a successful run will work properly. | 2/03/17  11:12 | The script should receive a token and user id from the link given to the user. Upon entering the script via the link. The user should be updated to registered in the database | If statement that checks the date would not recognise the time as being valid. The statement would override the script and stop the user from being validated | Date function had an incorrect hour symbol. This meant that the hours were in a different format to the ones in the database. This has now been rectified |
| 27 | As above with the new fixes in place | 2/03/17  11:48 | As above | Test was now successful no other errors occurred and the user was validated in the database | N/A |
| 28 | This test covers the if statement that will stop the script if no user exists within the database. This will be done by giving the script an incorrect user ID | 2/03/17  12:23 | If no results of the user ID are found from the database, then the script will show an error message and return the user to the homepage | If statement was incorrect. The statement was mistakenly looking for results that were less than 0 which meant it would never enter it. | Statement now made equal to 0. |
| 29 | As above with new fixes in place | 2/03/17  12:34 | As above | New fixes made the test successful. | N/A |
| 30 | Test to check that the statement that checks the tokens will run if the tokens do not match. This will be done by throwing it an incorrect token from the local user | 2/03/17  12:45 | The statement should delete the temporary user from the database, redirect them to the homepage and display an error message | Test was successful. User was deleted from the database and redirected as required | N/A |
| 31 | This test is for the if statement that makes sure that the token is still valid. For this the test a user will be created and then left for 10 minutes | 2/03/17  13:22 | The statement should again delete the temporary user from the database and then redirect them to the homepage with an error message | Test was successful. User was deleted from the database and user was redirected. | N/A |
| 32 | This test covers the script that will display thumbnails within the homepage.php | 5/03/17  16:31 | The script should receive video thubnails based upon what function has been used. For example one of these is to request newer videos only. These will be stored within various arrays | The statement used to fetch the information about the videos throws an error for a non-object, therefore the script crashes. | Missing library needed to store the data of the arrays was not shown within the header of the file. This has now been added |
| 33 | This test covers the script that will display thumbnails within the homepage.php with new fixes in place | 5/03/17  16:37 | As above | SQL statement does not recognise the WHERE clause \*. This fetches null results. | The statement was changed to an “or” operator between WHERE clauses. The function will now find records if nothing is thrown to it. |
| 34 | This test covers the script that will display thumbnails within the homepage.php with new fixes in place | 5/03/17  17:22 | As above | The array will now not record into a new array. Meaning that the foreach loop later on will not display data as there is none passed to it. | Array did not equal the function when it was called. Therefore null values were being given to the array outside the thumbnail() function. |
| 35 | This test covers the script that will display thumbnails within the homepage.php with new fixes in place | 05/03/17  17:25 | As above | This test is now successful. The video thumbnails display as they should. | N/A |
| 36 | Run script verifyScript.php to see if it runs and registers a user with the database | 15/03/17  18:13 | The user id and registration token should pass data to the script. The script will then cross check the token and its expiry time. If everything meets those parameters the registered column will change to true. | Syntax error occurred on line 64. Bind parameter would not accept a number as a parameter. | Number had to be assigned to a variable which then could be used in bind parameters. Once this was done another test was run. This test was successful and the user was registered to the database. |
| 37 | Check that verifyScript.php will stop and delete the record if tokens do not match | 15/03/17  18:18 | Script should tell the database to drop the user immediately and return a message | Bind parameter did not use the id put towards it. | Sql statement was incorrect, used DELETE FROM instead of just DELETE. Script now performs as it should. |
| 38 | Check that the verifyScript.php will stop if sql returns no results when an invalid ID is put towards it. | 15/03/17  18:47 | If the database returns null or false then an if statement should display a message and stop the script | Script would skip the if statement despite giving it invalid numbers. | Changed the if statement to look for number of rows that are brought back from the database. If it is 0 then it runs the if statement |
| 39 | Check that the verifyScript.php token validity will expire as it should and drop the user from the database if so | 15/03/17  19:35 | The user should be shown a message displaying that their link has expired and be dropped from the database. | Would not go run the if statement due to a current date error. | Current date did not include time. Time was subsequently added and the if statement ran exactly as hypothesised. |
| 40 | Now check that the verifyScript.php works after the changes for tests 37, 38 and 39 have been altered | 15/03/17  19:43 | The script should now run as normal and make the user registered within the database. | Test was successful. User was registered with the system as predicted. | N/A |
| 41 | Run php code within the adminPage.php for displaying videos that have been flagged by a user | 28/03/17  15:46 | The script should produce 3 filled arrays, one with all the flagged comments, one with the users associated with these comments and the video information that the comments apply to. | First array was successful, stored data as it should have. Second and third array within a for each loop for the previous array failed, threw exception bind\_param() as null | The statement was not closed within the foreach loop, therefore the data stream was crashing the statement. Closed the statement after each iteration and fixed the issue. |
| 42 | Run php code within the adminPage.php again with the fix in place | 28/03/17  16:30 | As above. | First and second array was now successful but third array is still not storing data. | Upon using the SQL code directly in the database it was found that the problem was with the fetch function. This was missing from the code and therefore was missing out storing the results. Fetch statement added and the data stores as it should |
| 43 | Run php script for the adminPage.php with new fixes in place | 28/03/17  16:53 | As above | All 3 arrays have now stored the necessary data and will display the correct data when called upon | N/A |
| 44 | Run php script in the HTML of the adminPage.php. | 28/03/17  17:00 | This code will use the 3 arrays seen in tests 5-9. These should print out a new div class where a video thumbnail and information relating to it should be printed. A nested foreach loop has been used to print out comments associated with the thumbnail | Thumbnails have printed out properly along with the comments underneath them. However due to incorrect loop statements in the previous script, the thumbnails have printed numerous times. | Problem was solved by adding an if statement in a previous for loop. If the previous iteration of the loop was duplicate in a video ID then the iteration was skipped. |
| 45 | Run php script in the HTML of the adminPage.php with new fixes in place | 28/03/17  18:17 | As above | Correct number of videos along with the relevant comments have now displayed properly. | N/A |
| 46 | Implemented banning and ignoring scripts which have been assigned to 2 buttons within the thumbnail of the adminPage.php. This will first look at the ignore button | 29/03/17  09:45 | The ignore button should run an ajax javascript function which will post a response to a script which in turn will update the flagging table to being resolved. On a successful run the javascript should then confirm this by changing the thumbnail div | Error occurred within the ignoreScirpt.php. A null variable was being used within the bind param function. | A number was replaced by declaring a variable using said number. The script now runs independently when a number is posted to it. |
| 47 | Run the previous test with the new fixes in place | 29/03/17  10:11 | As above | The ignoreScript.php worked as expected, however upon return to the javascript, the script to change the styles threw an undefined error type | The issue was that the class variable was not declared properly. The thumbnail now gives the function the class number. |
| 48 | Run the previous test with the new fixes in place | 29/03/17  10:15 | As above | Function now performs properly. The table in the database updates and the html changes appropriately | N/A |
| 49 | The ban button with script will now be used. **Note this test is after the previous one was changed and fixed. So changes were made which were apparent in the same places as the previous test.** | 29/03/17  10:26 | Same as the ignore button, this button should run an ajax javascript which will then post data to an external script which updates the flagging table as resolved and the video table to banned. Upon success the javascript will then let the user know by changing the thumbnail div | Function works properly, both tables update and the thumbnail div displays the relevant information. | N/A |
| 50 | adminPage.php has been updated to now not display thumbnails if they have been resolved. | 29/03/17  10:40 | The new updated php code should now only display videos that are flagged which are not resolved | Change has worked as expected. Upon banning video with the id of 6 the second video with an id of 7 still remains | N/A |
| 51 | Continuation of previous test but now one video will remain banned and the other will be ignored | 29/03/17  10:43 | The php will now produce no video thumbnails and show a message saying there is no reported videos at present | Upon reloading the page the web browser now displays a white screen with no content | The issue here was that the exit(); function was used within the php script if no results were returned. This has now been removed. |
| 52 | Previous test with fixes in place | 29/03/17  11:14 | As above | The page now displays and a message displays to the user showing them that there are no videos that are currently reported | N/A |
| 53 | Code has been added to the viewPage.php to now redirect the user if they try to access a blocked video and display an error message | 29/03/17  12:17 | When trying to access a banned video, the user should be redirected back to the homepage with an error message displaying on screen | Test was successful. User was redirected and the display message appeared as it should have done | N/A |
| 54 | The userPage.php has been updated so that if a video has been banned, the thumbnail will print but display a banned message | 29/03/17  12:46 | The video that has been banned will be replaced by a message explaining to the uploader that the video has been banned. | Upon entering the user page, the video has instantly shown that it has been banned and cannot be clicked anymore | N/A |
| 55 | A comment system has been added to viewPage.php. This check will make sure that a user who is not signed in will not be able to leave a comment | 29/03/17  13:28 | If the user is not signed in, a warning should display, explaining that they need to sign in to do use this feature | No error message was displayed. | The if statement used quotes to surround the false. These were removed and the code now displays the error message. |
| 56 | This test will check that the ajax and commentScirpt.php are working together properly. A small sentence of 4 words will be used. | 29/03/17  13:37 | The ajax should post the data to the commentScript.php. The script should then store the comment and reply with the comment. The ajax will then display the comment underneath the other comments | The test did prove successful however on return the comment was printed twice underneath the video. | The ajax has been configured to run the “on success” function when the server returns a ready state of 4 (request is finish and there is a response) and a status of 200 (status is ok) |
| 57 | The text box for the comments will now be used without any text in it. | 29/03/17  13:52 | The javascript will check to see if the box is null. If so an alert message will be shown and the operation will cancel | The javascript did not stop or display an alert message. | Javascript function did not return a value of false for another function to use. This has now been rectified and works as it should |
| 58 | The text box for the comments will be used to have the max characters within it, which is 1000 | 29/03/17  13:57 | The comment box should stop the user from entering any more than 1000 characters. The database has been limited to 1000 characters so the test should succeed | Test was successful. The comment was stored within the database and the text area did not allow anymore characters to be inputted into the field | N/A |
| 59 | This is a final test to make sure that the comment system still works after the changes that have been made in tests 21-24 | 29/03/17  13:59 | The comment should be added to the database and updated within the comments section of the viewPage.php | The test was successful, no new errors have occurred | N/A |
| 60 | This test is to check that the statement fetching the comments is working correctly. Here we have used an inner join function to fetch data from 2 different tables. | 29/03/17  14:42 | The script should fetch all comments associated with a video along with the username of the user that posted the comment. | Test was successful. Inner Join fetched the data requested from the second table. | N/A |
| 61 | This test checks that the settingsScript.php will fetch the old image paths and replace the old paths with the “true” file hosting image path | 1/04/17  09:43 | The script should fetch both results from the database for a specified user and then cut the strings so that the image name remains. | The data was fetched from the database. However the strings were not cut | Missing a parameter in the str\_replace function. The strings now cut as intended |
| 62 | This test will run the settingsScript.php along with the uSettings.php. | 1/04/17  19:45 | Both scripts should send and handle a plethora of different information such as images or text variables. The user should be able to make changes without needing to fill out all of the required boxes. The test is successful if the database updates the required fields. For this test all fields will be changed. | The settingsScript.php crashed on line 229. Was missing a semicolon | Added semicolon to the line as needed. |
| 63 | Previous test with new fixes in place | 1/04/17  19:55 | As above | The script returned a value of 0 and changed the username to 0 of the user. | Upon echoing out the replies not all of the text boxes were replying with results. Name was missing from the textarea which was causing it to throw a null value. This also applied for both. |
| 64 | As above with new fixes in place | 1/04/17  20:19 | As above | Although all of the files and data is being passed to it. Nothing is being stored within the database. | The script was working correctly. However the script was throwing an error for a file with lowercase jpg being used. This has now been rectified. |
| 65 | As above with the new fixes in place | 1/04/17  20:32 | As above | Script again ran as normal but did not update the database records that it should have done. | Syntax for the sql statement was incorrect. Therefore the script was ignoring this without throwing any errors. |
| 66 | As above with new fixes in place | 1/04/17  21:47 | As above | Script now runs as expected. Files were uploaded and the database has changed as required | N/A |
| 67 | This test is to check that the settingsScript.php can ignore the image uploading scripts if no files are posted to it. | 1/04/17  22:12 | The script is programmed to ignore the checks if the files are not set. From here the file directory will revert to its current one and not update the database | Script is throwing errors for the file paths. However it is ignoring the surrounding if statements which means the user will receive php server errors. | The if statement could not use the isset function for a file. Instead we now check for if the file is not null. |
| 68 | As above with the new fixes in place | 1/04/17  22:35 | As above | The test is now successful. The script throws no errors and the database does not update any fields within the database. | N/A |
| 69 | This test will look at leaving everything blank and trying to run the settingsScript.php | 2/04/17  08:24 | The script should ignore all functions and send back a message telling the user that changes were saved. This will be successful if the database does not change | Database did not change. However wrong error message was presented to the user upon completion of the script | Problem was with an if statement. Was looking for multiple failings all at once, instead of one or multiple. This has been rectified and the correct message shows. |
| 70 | A add to favourites function has been added to the viewPage.php file. This is to test that the function and its associated script, favouriteSctipt.php will add the record to the favourites table in the database | 2/04/17  13:21 | Upon a successful run, the viewPage.php will post the video ID to the external script. The script will then decide whether or not the table should be updated. If it is updated, a record will be added to the database and a message will be shown to the user. | favouriteScript.php failed. Missing a semicolon after the header. | Semicolon added |
| 71 | As above with new fixes in place | 2/04/17  13:23 | As above | Script now produces the required outcome. The favourites table is updated, however there is an issue with receiving the response from the script. | Issue was with a previous echo statement used within the code. This has now been rectified and removed |
| 72 | Now we will test the same script again with the same user. This is to test that the favouriteScript.php will not repeat data | 2/04/17  15:12 | The script should not update the database in anyway and respond with a message telling the user that they have already added the video to their favourites | The test was successful. The database was not updated and the user received a message as needed | N/A |
| 73 | PHP code has been added to allow admin’s to upgrade users access levels. This test will check to make sure that the enableUserScript.php is working correctly and will send a response back to the user | 2/04/17  16:07 | Upon a successful run of the script, the user that has been selected should be now access level 2 in the database and the upgrade field be set back to 0. A message response will then be sent back to the user | Test was unsuccessful. Incorrect value used for the ajax function, meaning that the value of the div was null | Value has now been added to the enable submit button. |
| 74 | As above with the new fixes in place | 2/04/17  16:15 | As above | Test was somewhat successful. The upgrade column in the users table was set to 2 while the access level was set to one | Values have been swapped by mistake. These have now been rectified. |
| 75 | As above with new fixes in place | 2/04/17  16:32 | As above | Test was successful this time. The script updated the database as requested | N/A |
| 76 | adminPage.php now has php code in place to show videos that have been banned or ignored. This test is to make sure that the code to display this data runs appropriately | 4/04/17  15:38 | The page should display another row of thumbnails that will show banned or ignored videos. The user should then be able to click on these thumbnails to show information that relates to them | Error was thrown. No such array for these videos existed. | Typo error on line 108 so the array for these videos was not named properly. This has now been fixed. |
| 77 | As above with new fixes in place | 4/04/17  15:45 | As above | No errors were thrown. However no data was displayed. | Another typo error was made on line 123. Was going into if statement by accident. This has now been fixed and the data displays properly |
| 78 | Make sure that the new history videos in place on the adminPage.php open up the previously flagged comments. | 4/04/17  15:51 | Each thumbnail should open comments associated with them when selected. | Comments did not open as intended. Wrong function was being called within this div. | Correct function has been applied to the correct divs. |
| 79 | Test that the ban button for the ignored videos will still work for a different thumbnail | 4/04/17  16:02 | The button should run the ban script once again and update the relevant details within the database | The test was successful | N/A |
| 80 | Test the undo ban button. This should run an ajax function which will run the undoBanScript.php | 4/04/17  16:04 | The script should change the banned field for the related video to 0 making it viewable again to users | Script performed the function as it should have done. Test was successful | N/A |
| 81 | The viewPage.php file has had a ban option and a un-ban option for an admin user. This first test will make sure than the video will be banned upon selecting the ban video button | 4/04/17  16:15 | The button should run the banScript.php, which in turn will ban the video to any non-administrative users | Test was successful | N/A |
| 82 | This will now test the un-ban button which uses the undoBanScript.php | 4/04/17  16:18 | The button should run the undoBasnScript.php which will then make the video available for general viewing once again. | Test was successful | N/A |
| 83 | A video settings page, vSettings.php has now been added. This will test that the image file handler and the video file changer will perform properly in the editFileVideoScript.php script | 13/04/17  09:27 | The script should unlink the previous image file when a new image file is presented to it and replace it with the said new one. The video file location should also be changed by copying the video file and deleting the old file in the previous directory. The new directories should then be stored within the database | Image file did not upload correctly. No image files were being found by the script when an image was being uploaded to it. Instead the previous image was being used. Video file also relocated however the wrong file path was recorded to the database. | Wrong post was being used to receive the image file. This has now been corrected. The video file used the wrong variable to specify the name of the video file. This has also been rectified |
| 84 | As above with the new fixes in place. | 13/04/17  10:12 | As above | Test was still unsuccessful. Image path was stored incorrectly. | Image file path was using the old genre from an incorrect variable once again. This has now been fixed |
| 85 | As above with the new fixes in place | 13/04/17  10:30 | As above | Test is now successful, both image file and video file were uploaded or relocated respectively | N/A |
| 86 | This will now test the text boxes within the vSettings.php. All of these will have information typed into them so that they can be updated within the database. | 13/04/17  10:34 | The editVideoScript.php should update the database with the new data put towards it | Test was succesful | N/A |
| 87 | The vSettings.php form will now be left blank. When the save changes button is used the editVideoScript.php will be used with no posted values | 13/04/17  10:48 | The script will retrieve the old image and video paths and all other data associated with the video. The script should then perform all of the necessary checks as before but then revert all database data to the previous iteration. | Test was successful no other errors occurred | N/A |
| 88 | A delete function has been added to the vSettings.php page. This connects to a deleteVideoScript.php script, which will delete the video information in the database and its image and video files | 13/04/17  12:43 | The vSettings.php page should run an ajax function which will in turn run the delete video script. All tables within the database that have a video ID will be deleted from it. The script will then unlink the video and image file from the file host. Upon completion the user will be sent back to the homepage with a confirmation message | Test was successful, all data pertaining to the video was deleted correctly from the database and the file host. | N/A |
| 89 | A search feature has been added to the website. This involves using a search bar input box which connects to an ajax script. The script connects to a php search script which fetches data that is relevant. This test will check that the script will export data upon a key up event within the input box | 13/04/17  19:21 | The ajax script on every web page should call on a search script (searchScript.php). The script will then find videos within the database that is like the value posted to it. The script will then bind all data retrieved from the database to an array and then print it. The ajax will then print out the array results within the search field. These thumbnails should also be clickable to send the user to the video view page | Search script performed correctly as it should have. However the ajax scripting did not work as previously planned. Instead multiple videos were printed within the search field | Ajax was adjusted to run on success function when the server script had finished running completely. |
| 90 | As above with new fixes in place | 13/04/17  20:32 | As above | The correct number of videos now display as required. However the video thumbnails are still not clickable. | Script now forcibly uses double quotations to surround the link within the thumbnail. |
| 91 | As above with the new fixes in place | 13/04/17  21:47 | As above | The test is now successful. Video thumbnails and their links both show and work correctly | N/A |
| 92 | This test will check that the new history and favourites pages perform their delete functions. A separate deleteSpecific.php script will delete a specific video from a user’s history or favourites list. This test checks that the records are removed from the database when the user requests it. | 13/04/17  23:13 | Upon selecting the x in the top corner of one of the video thumbnails the deleteSpecificScript.php should delete the record from the database which associates with the user who is logged in. | Thumbnail kept overruling the x button. Although the video was deleted from both pages on both instances, the video thumbnail automatically directed them to the video that the thumbnail is linked to. | The link now no longer applies to the entire thumbnail div, instead the link only applies to the elements inside of it |
| 93 | As above with new fixes in place | 13/04/17  23:25 | As above | Test is now successful. Both the favourites page and history settings page both display and can delete videos associated with a specific user with no problems | N/A |
| 94 | A new reset password system has now been put into place. The user can now reset their password if they have forgotten it. The resetPassword.php page will be tested to check to see if a reset password link will be sent to the user. | 15/04/17  13:39 | Firstly we will check to see if the resetMailingScript will work correctly. This script should take the token associated with the user’s registration, reassemble it and store it as a hash within the database. The users email and this new temporary password will be sent as a link to the user through email to be used in a secondary script. | Test was successful. The link was created as needed and sent to the users email. The database was also updated with the correct password. | N/A |
| 95 | The newPassword.php page will now be tested. This test will use the correct user link first. | 15/04/17  15:07 | Upon entering this with the correct link sent to the users email. The page will display their email and ask to input a new password which will be updated in the database. The user will then be logged into the site | Test was successful, user was able to update their password upon entering the page. They were then subsequently logged in. | N/A |
| 96 | Now an incorrect link will be used to try and enter the new password page. The same email will be used but a random password will be given to the url link | 15/04/17  15:16 | The page should automatically redirect the user to the homepage and tell the user to start the process once again | Test was successful. | N/A |
| 97 | A subscription type service has now been put into effect. A user can add another user to their subscription list using a button on a video view page. This test will check that the viewPage.php will add the selected user to the database. | 17/04/17  16:03 | The viewPage.php should call on an ajax function which will in turn call the subUserScript.php script. This should add the user associated with the video to the database along with the user who is currently signed in. upon the ajax request being successful the users subscription list should be updated with the new user shown in the list | Test was not successful. The user was added to the database as required. However the subscription was not updated within the users subscription area | The ajax was not updating the correct div. This has now been corrected. |
| 98 | As above with new fixes in place | 17/04/17  16:21 | As above | Test was successful. | N/A |
| 99 | This now tests the un-subscribe function within the viewPage.php page. The user can remove the subscription from their list using the same button. | 17/04/17  19:38 | The page again will call an ajax function which will run a removeSubScript.php script. This will remove the user from the subscription list table associated with the user that Is currently signed in. | Test was successful. The users subscription was removed from the database as requested. | N/A |
| 100 | A CRON job has now been created in order to remove users who have made an account but not registered to the database. A CRON job script has been created to do this | 23/04/17  19:57 | The CRON job should delete an intentionally un registered user within a 24 hour period. | Test was successful, the CRON job deleted the user without the need for administrator intervention. | N/A |

## User testing and feedback

Here is a list of a number of users that have used and tested the site after deployment. The results were gathered by using google forms in order to get different opinions which could be compared and contrasted for the same questions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Timestamp** | **Did you find navigation easy to use?** | **Is the websites text clear to read?** | **How would you rate the website as a whole?** | **Is there any further comments you would like to make about the site?** | **Do the video features meet your expectations? Please explain what you liked and disliked about this** | **How does the website compare to other popular online platforms? (e.g. YouTube, Twitch)** |
| 4/21/2017 15:01:48 | Yes | No | 3 | I think that the home page may overwhelm viewers once it is full of content due to the number of thumbnails on screen. Also the green font on the comments section is jarring and could be tricky to read for some people. | Maybe adding a like/dislike system to make finding high quality content easier. | Good |
| 4/21/2017 21:09:55 | Yes | Yes | 2 | Less luminous green. Go to basic colours. | Yes, just like on YouTube. The videos loaded very quickly. | Good |
| 4/22/2017 17:15:36 | No | Yes | 2 | Some screen size issues, had to zoom to 80% when viewing a video. Also when uploading a video could do with a bit of text explaining file size restrictions or a completion percentage. However amount of features and usability of the website are all positive and good. | Videos play well and work as expected. | Good |
| 4/24/2017 14:29:10 | Yes | Yes | 2 | None right now. The search images section did not appear to work correctly. | Yes, but unable to upload .wav files for some reason. | Good |
| 4/25/2017 11:13:44 | Yes | No | 3 | While I like the idea I think the execution could be improved. Not sure about the color scheme but this is a personal preference. At first I found the hamburger menu confusing but soon got used to it, an improvement could be made here. Overall the function of the site is clean and no errors were detected when using it. I would be interested in discussing your search function and how you determine the search terms | Some of the videos open up full screen and could be set too large. Also from a personal point of view I would prefer videos to auto start when I click them. | Did not meet standards of other platforms |
| 4/25/2017 11:24:58 | Yes | Yes | 5 | Simple and easy to follow. Well designed. | Yes | Good |
| 4/25/2017 11:42:38 | Yes | Yes | 3 | Check your user validation functionality when a user registers. I purposely attempted to register a second account using the same email address but on the second instance, I used a mixture of upper and lower cased characters during the registration process. During your validation, could I suggest you do a to strtolower() function on the email address which is entered and validate against that. Also remove any whitespace from the email string. | The video features look quite basic, but assuming this is an ongoing project, then you are going the right way. | Could be improved |
| 4/25/2017 11:45:26 | Yes | Yes | 3 | Colour scheme isn’t pleasing on the eye and is quite shocking. I would definitely change the colour scheme. It’s consistent throughout but not the easiest to get on with. The colours are off putting. Not much text and a lot of empty space on the home page. When clicking on a video it opens up on a new page but is wider than the screen.  The registration process is simple to use although it did catch me out with just a username and password. Expected more details.  The site is easy to navigate.  Search function not working | The videos were ok | Could be improved |
| 4/25/2017 12:05:11 | Yes | Yes | 3 | I believe there is much potential for this site and its functionality. I like the browsing categories and after a brief visit to 'Twitch' which I do not use I feel Flixel is easier to navigate. I can read the text but some people may have difficulty reading the black font on grey background.  The videos loaded quickly and I enjoyed the fact I did not have to watch adverts or skip them so could get straight to the content I wanted which would in turn make me more likely to use it than. I have ranked it at 3 as I needed to scroll down to find the browsing categories and because I have a Gmail account I am unable to register. But overall I would use it if it meant getting to the content I wanted without adverts! | As I came to the site without specific expectations in mind this is hard to answer. I liked the fact that there are divisions for content to browse so if I was looking for a video on food I could go there. The videos which are currently available were funny. I generally would be looking for fitness films or educational films though rather than what is currently available and would prefer them to be slightly longer in length. | Could be improved |
| 4/25/2017 13:00:55 | Yes | Yes | 3 | Perhaps give the navigation its own bar as if feel the heading "newly uploaded videos". Is too close to it and makes the site feel a little "clunky". | It does meet my expectation it is easy to use and has a nice large screen to start with. It is easy to use, the only negatives are that you can’t see the title near the video(unless you scroll down)- this could be down to personal preference though and the comment blocks seems to be a bit "full" due to their being no padding in the box where the comment sits. The profile of the commented also needs to be vertically aligned. (Again personal preference) but overall the video features is nice to use. | Could be improved |
| 4/25/2017 13:16:06 | No | No | 3 | The black font on a grey background is not easy to read. The font itself would not be suitable for a lot of students with dyslexia.  The search box should be on the main page, rather than clicking on the three lines.  Likewise for the 'about' information - this could be in tabs at the top of the main page. You have to scroll down the page to read all the information in the box. Resize the font or box to make it fit on one screen. You are severely restricting your audience by not having gmail accounts allowed on the website. The sub categories are good but maybe these could appear on the left or right of the screen in a fixed menu. It would help having videos in each category too. The colours of grey and lime green work really well together.  I like how 'related videos' are suggested when you are watching one of the videos. | The videos played well with no buffering. | Could be improved |
| 4/26/2017 11:00:18 | Yes | Yes | 2 | Navigation is better than amazon firestick.  Improvements could include extended search for more than top 5. | Yes.  Videos are easy to upload with a simple to the point form.  Profile name is not an initially obvious link. | Good |
| 4/26/2017 14:09:54 | Yes | Yes | 3 | Whilst in register page, if window is minimized, text and boxes are not responsive. Making it difficult to read. | Video quality good, however video thumbnails appear too wide on list | Could be improved |
| 4/27/2017 22:24:11 | Yes | Yes | 3 | The site is quite easy to use. The use of the menu item that is easily recognisable is good as it follows common design standards. As you have separated your videos into different themes / topics / genres, the site would benefit from having an additional navigation menu at the top of this page which allow you to select the topic of choice and go to it. Bookmarking / anchoring I think this is called.  The About Page is useful as it contains some key points, including terms of use. The site would benefit from having a footer containing things such as links to social media sites, terms of conditions, copyright info etc.  The colour scheme contrasts well. The login and register areas are good features, as is the search box. I couldn’t get these to work though not sure if they’re supposed to at this stage. | The title of the video being listed is key to the engagement with the video, and the number of views each video has had is a nice touch. Many of the videos don't have a still image to further help promote it, which is a shame. The site would benefit aesthetically overall with this addition. | Could be improved |