

THE “DENVER MULTIMEDIA DATABASE”:
A FORENSIC DATABASE FOR DIGITAL
AUDIO, VIDEO, AND IMAGE MEDIA

by

CRAIG ANDREW JANSON

B.A., University of Richmond, 1997

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado Denver in partial fulfillment
of the requirements for the degree of
Master of Science
Recording Arts Program

2019

This thesis for the Master of Science degree by

Craig Andrew Janson

has been approved for the

Recording Arts Program

by

Catalin Grigoras, Chair

Jeff M. Smith

Cole Whitecotton

Date: May 18, 2019

Janson, Craig Andrew (M.S., Recording Arts Program)

The “Denver Multimedia Database”: A Forensic Database for Digital Audio, Video, and Image Media

Thesis directed by Associate Professor Catalin Grigoras

ABSTRACT

To date, there have been multiple databases developed for use in many forensic disciplines. There are very large and well-known databases like CODIS (DNA), IAFIS (fingerprints), and IBIS (ballistics). There are databases for paint, shoeprint, glass, and even ink; all of which catalog and maintain information on all the variations of their specific subject matter. Somewhat recently there was introduced the “Dresden Image Database” which is designed to provide a digital image database for forensic study and contains images that are generic in nature, royalty free, and created specifically for this database. However, a central repository is needed for the collection and study of digital audios, videos, and images. This kind of database would permit researchers, students, and investigators to explore the data from various media and various sources, compare an unknown with knowns with the possibility of discovering the likely source of the unknown. This paper introduces “The Denver Multimedia Database”. A database developed to collect and catalog multimedia content across the three major sources of video, audio, and images.

The form and content of this abstract are approved. I recommend its publication.

Approved: Catalin Grigoras

To D, A, and W. You are my lodestar.

ACKNOWLEDGEMENTS

This would not have been possible without Jeff Smith, Catalin Grigoras, and Leah Haloin. Your collective dedication, vast knowledge, undying support, and strong commitment have been amazing. Cole Whitecotton, you have inspired and helped me probably more than you know. Thank you all.

CONTENTS

INTRODUCTION	1
CHAPTER	
I. LIBRARIES AND DESIGN.....	3
II. EXISTING RESEARCH.....	14
III. METHODOLOGY AND LOGIC	17
IV. THE INTERFACE AND DATABASE	19
<i>Front End</i>	19
<i>Audios, Videos, Images</i>	20
<i>Admin Interface</i>	23
<i>WordPress Administration</i>	29
<i>Database</i>	31
V. GOING FORWARD	33
CITATIONS	36
BIBLIOGRAPHY	38
APPENDICIES	
A. CLASS LISTING	40
<i>Database</i>	40
<i>Sample</i>	41
<i>Sanitize</i>	44

Users 45

Utility 46

B. TECHNICAL NOTES..... 47

FIGURES

1.1 Output from Mutagen processing of MP3 file.....	5
1.2 A partial listing of the resulting JSON after encoding of the General data block extracted using php-MediaInfo.....	6
1.3 A partial listing of array dump from php-MediaInfo General data block after JSON decoding and conversion	8
1.4 A partial listing of the JSON result from pymediainfo processing of an audio file.	9
1.5 A partial listing of the array dump which is the result of EXIF data extraction using the PHP native function <code>exif_read_data()</code>	12
4.1 The DMDB Home Page.....	19
4.2 Audios, Videos, and Images Filtering Options.....	20
4.3 A Listing (Audio) After Sample Attributes Selected. Audio and Video Sample Listings are Similar.....	21
4.4 Image Listing After Sample Attributes are Selected	21
4.5 Basic Information Expanded for an Audio Sample	22
4.6 The Sample Details Page with All Attributes for an Image Sample (Not the Full Page).....	23
4.7 Contributor Icon Bar	24
4.8 The Sample Upload Form.....	25
4.9 Step 1 of Multiple Sample Upload Process	26
4.10 Step 2 of the Multiple Sample Upload Process	27
4.11 During Upload A Scrollable Box with Status of Each Sample's Progress is Available for Review.	27
4.12 Administrative Icon Bar.....	28
4.13 User Listing.....	28
4.14 User Edit/Add Management Interface	29
4.15 The WordPress Dashboard	30

4.16 DMDB Database Tables and Relationships.....	31
--	----

ABBREVIATIONS AND DEFINITIONS

PHP – A recursive name which refers to PHP: Hypertext Preprocessor, PHP is a scripting language that is often used for web development and can be embedded into web pages.

LAMP – Linux (operating system), Apache (web server), MySQL (database), PHP (scripting language).

WordPress – Originally a software platform for blogging, this has grown into a robust content management system with a large community of contributors that develop plugins for various functionality.

CMS – Content Management System.

Daemon – On the Linux platform, this is a process that runs in the background and provides services to the server. This is similar to a “process” or “service” on the Windows operating system.

Instantiation – The initial creation of a variable or object within a software code environment.

JSON – JavaScript Object Notation; JSON provides a syntax for organization, storage, and exchange of data in a human readable format. Properly implemented, it provides for a logical method by which the data can be shared, accessed, and converted to arrays for use within code on either side of a programmatic conversation.

SQL Injection Attack – A hacking method where specially formed strings are entered in to the fields on a web browser form. These typically take the form of SQL commands and are designed to elicit a response from the database that was not intended by the developer. The type of attack is used for the purpose of revealing data in a database or revealing information about a database for use in getting to the data.

INTRODUCTION

Despite decades of existence, multimedia forensics is still a nascent discipline and may remain so for some time to come. Digital technology in audio, imaging, and video is continually changing and with those changes come changes to the forensic artifacts that researchers, investigators, and academics want or need to understand and study.

There are a number of databases that exist for use by forensic examiners in various disciplines. Databases exist for digital imaging (UCID, ImageCLEF, Caltech-256, 80 million tiny images, and more[1]-[4]), fingerprints (IAFIS[5]), software hashes (National Software Reference Library[6]), DNA (CODIS[7]), ballistic information (NIBIN[8]), and the list goes on. Moreover, the database developed by DARPA for the Media Forensics Challenge covers a wide range of images and video utilizing a precise methodology such that manipulated images and high provenance images can be made available for assessment of state-of-the-art forensic systems[9]. The Denver Multimedia Database Database (DMDB) does not presume to match the level, complexity, nor deep scientific nature of these databases. It is designed to provide a central repository for audio, video, and image samples for the purpose of maintaining them in a centralized way for reference and for open sharing of that reference with others who may benefit.

This paper details the design and development of this database which is designed to be a straightforward system having limited technological requirements (required software, platform, etc.) and meets the goal of manageability both on the front end and within the underlying code. In turn, it is hoped that this database and its interface can be used by researchers, students, academics, and forensic professionals to upload test data and media files, and have those media files cataloged and characteristic data extracted. Thereafter, the database can then be used to compare the items within the corpus to outside media or media within the database for similarity,

common elements, and other useful traits. The final result will be a website which will allow for both population of the database and searches, comparisons, testing of unknowns, and, perhaps, a number of other uses as it evolves.

The DMDB will initially be small but with the variety of test samples that have already been conducted and archived on various researchers' local systems, it is expected that this centralized database will grow quickly to be of use within a short period of time. Moreover, it is hoped, in time, this database will have the capability to answer questions such as, what recording device created this audio? What recording device created this video? Does this image match a camera on file? What other devices have similar characteristics to the audio device that created this sample? All valuable answers in the multimedia forensics discipline.

The remainder of this paper is organized as follows: Chapter I presents the design and evaluation of libraries available for extraction of metadata from the media samples. Chapter II provides an overview of existing research. Chapter III describes the methodology and logic behind the collection of the samples, structure of the database, and approach to the user interface. Chapter IV presents the resulting web site where a user interface is available for exploring and adding to the database, and Chapter V concludes this paper with some possibilities for the future of the system.

CHAPTER I

LIBRARIES AND DESIGN

Ideally, each media file in a database such as the DMDB would be evaluated at the binary level and the individual characteristics catalogued to understand the source of the file.

However, that is an impractical approach to building a database for use across a wide range of expertise and outside the scope of this paper. The manual aspect of doing those analyses in order to build the rules for each would take a significant amount of time to achieve and there would be no guarantee that all variations could be accounted for. Therefore, existing tools for cataloguing the media have been used to build the sample processing engine for the DMDB. This allows for focus on the data derived from the media rather than the mechanics of how that data is derived.

A more automated methodology also allows students and others who may not be familiar with the forensic techniques of media evaluation, a method by which they can contribute to the database and use it for their own education, research, or investigations. Furthermore, the core component to extracting data from a media file for migration to a database is to ensure that as much useful information as possible is extracted from the media and that information is consistent across samples which furthers the quality of the data contained therein. Consistency of a tool in how it handles data extraction across media file types is important since jumping between libraries could pollute the results with different naming conventions, tags, and the possibility that some data could be interpreted differently between methods. For these reasons, it was decided to limit, as much as logical, the mechanisms used to extract and archive the data as opposed to using multiple libraries and combining the data. In the end, having a like data set between samples will make search and comparison consistent and more reliable.

A number of media extraction libraries were tested with currency of the software, active

maintenance and update availability, support, and, where applicable, the existence of a community surrounding them considered. Another consideration was the existence of hooks for internet-based development to make integration with the web interface more efficient as well as secure. The following libraries were reviewed for this purpose:

- Mutagen – Audio and Video – by Joe Wreschnig, Michael Urman, Lukáš Lalinský, Christoph Reiter, Ben Ockmore & others;
- pymediainfo – Audio and Video – by Patrick Altman, Louis Sautier;
- php-mediainfo – Audio and Video – by Maxime Horcholle & others;
- Mp3Info – Audio – by Chris Nandor & Dan Sully;
- Mp3Info – Audio – by Ricardo Cerqueira, maintained by Cedric Tefft;
- phpExifTool – Images – Romain Neutron
- Native PHP `exif_read_data()` function – Images

What follows is a discussion of the results of experimentation with these libraries and the final outcome for the libraries that were used for data extraction of audio, video, and image samples to the DMDB.

Mutagen[10]

Mutagen is a self-described “Python module to handle audio metadata.”[11] It supports a large list of media types, such as ASF, FLAC, MP4, Monkey’s Audio, MP3, Musepack, Ogg Opus, Ogg FLAC, Ogg Speex, Ogg Theora, Ogg Vorbis, True Audio, WavPack, OptimFROG, and AIFF as well as ID3v2 and ID3v2.4 standard frames[12] and has the capability to parse this information for output. While it does read the more technical information of bitrate, format (including version and profile), sampling frequency, length, and number of channels, all other information is limited to the general information about the audio data and more specific to the one might see on the display of an MP3 player (song, artist, record date, album, etc.) The

utilities available do not derive deeper information such as compression mode, stream size, bit rate mode, etc.

Moreover, the data output from the function that extracts all information is cryptic (fig. 1.1) and partially encoded which would require complex parsing in order to extract key/value

```
>>> import mutagen
>>> mutagen.File("AC-DC - Back in Black.mp3")
{u'COMM:ID3v1 Comment:eng': COMM(encoding=<Encoding.LATIN1: 0>, lang='eng', desc=u'ID3v1
Comment', text=[u'00001609 000017BB 0000455F']),
u'PRIV:ZuneAlbumMediaID:\x00\x00\x00S\x00\x04\xdb\x11\x89\xca\x00\x19\xb9*93':
PRIV(owner=u'ZuneAlbumMediaID', data='\x00\x00\x00S\x00\x04\xdb\x11\x89\xca\x00\x19\xb9*93'),
u'PRIV:WM/WMCollectionGroupID:\xd8\xc1\xc8\x91(\x08@\xa0\x0c\x02dc_\x01\xa6':
PRIV(owner=u'WM/WMCollectionGroupID', data='\xd8\xc1\xc8\x91(\x08@\xa0\x0c\x02dc_\x01\xa6'),
u'PRIV:WM/WMCollectionID:\xd8\xc1\xc8\x91(\x08@\xa0\x0c\x02dc_\x01\xa6':
PRIV(owner=u'WM/WMCollectionID', data='\xd8\xc1\xc8\x91(\x08@\xa0\x0c\x02dc_\x01\xa6'),
u'PRIV:WM/Provider:A\x00M\x00G\x00\x00\x00': PRIV(owner=u'WM/Provider',
data='A\x00M\x00G\x00\x00\x00'), u'PRIV:PeakValue:\xa1\x7f\x00\x00': PRIV(owner=u'PeakValue',
data='\xa1\x7f\x00\x00'), 'TDRC': TDRC(encoding=<Encoding.LATIN1: 0>, text=[u'1980']),
u'PRIV:ZuneMediaID:\x00\x1d\x7f>\x00\x05\xdb\x11\x89\xca\x00\x19\xb9*93':
PRIV(owner=u'ZuneMediaID', data='\x00\x1d\x7f>\x00\x05\xdb\x11\x89\xca\x00\x19\xb9*93'),
u'PRIV:WM/MediaClassPrimaryID:\xbc}\xd1#\xe3\xe2K\x86\xa1H\xa4*(D\x1e':
PRIV(owner=u'WM/MediaClassPrimaryID', data='\xbc}\xd1#\xe3\xe2K\x86\xa1H\xa4*(D\x1e'), 'TPE2':
TPE2(encoding=<Encoding.LATIN1: 0>, text=[u'AC/DC']), 'TPE1': TPE1(encoding=<Encoding.LATIN1:
0>, text=[u'AC/DC']), 'TALB': TALB(encoding=<Encoding.LATIN1: 0>, text=[u'Back in Black']),
u'PRIV:ZuneAlbumArtistMediaID:\x00\x00\r\xa8\x00\x06\xdb\x11\x89\xca\x00\x19\xb9*93':
PRIV(owner=u'ZuneAlbumArtistMediaID',
data='\x00\x00\r\xa8\x00\x06\xdb\x11\x89\xca\x00\x19\xb9*93'), 'TCOM':
TCOM(encoding=<Encoding.LATIN1: 0>, text=[u'Angus Young/Brian Johnson/Malcolm Young']),
'TPUB': TPUB(encoding=<Encoding.LATIN1: 0>, text=[u'Albert Productions']), u'APIC':
APIC(encoding=<Encoding.LATIN1: 0>, mime=u'image/jpeg', type=<PictureType.OTHER: 0>, desc=u'',
data='\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x01,\x00\x00\xff\xdb\x00C\x00\x08\x06\x06\x0
7\x06\x05\x08\x07\x07\t\t\x08\n\x0c\x14\r\x0c\x0b\x0b\x0c\x19\x12\x13\x0f\x14\x1d\x1a\x1f\x1e\x1
d\x1a\x1c\x1c$.')
```

Figure 1.1 Output from Mutagen processing of MP3 file

pairs for insertion into a database. While this parsing and decoding is possible, this adds a layer of complexity to the processing of the files that does not appear to be necessary when compared to other libraries tested.

Documentation is sparse but Mutagen appears to be a library used within other contexts to support the manipulation of tag data in MP3 and MP4 audio files and appears directed toward the production of commercial audio and not more generic audio files from recording devices.

Processing a sample file from a handheld digital recorder through Mutagen yielded only the text “{}” and nothing more which indicates a null array and the likelihood it could not dig deeper into the file to get additional salient data.

php-mediainfo[13]

This library was the most promising for the efficient processing of media files. It uses the PHP scripting language which is the foundation of the DMDB and its underlying CMS system. It leverages the well-known, open source MediaInfo application maintained by

```
{
  "count": "331",
  "count_of_stream_of_this_kind": "1",
  "kind_of_stream": {
    "shortName": "General",
    "fullName": "General"
  },
  "stream_identifier": "0",
  "count_of_audio_streams": "1",
  "audio_format_list": "MPEG Audio",
  "audio_format_withhint_list": "MPEG Audio",
  "audio_codecs": "MPEG Audio",
  "complete_name":
  "*****\\media\\audio\\f2a3d748c5c38ac38be9cb214518ca68.mp3",
  "folder_name": "*****\\media\\audio",
  "file_name": ["f2a3d748c5c38ac38be9cb214518ca68.mp3",
  "f2a3d748c5c38ac38be9cb214518ca68"],
  "file_extension": "mp3",
  "format": {
    "shortName": "MPEG Audio",
    "fullName": "MPEG Audio"
  },
  "format_extensions_usually_used": "m1a mpa mpa1 mp1 m2a mpa2 mp2 mp3",
  "commercial_name": "MPEG Audio",
  "internet_media_type": "audio\\mpeg",
  "file_size": {
    "bit": 25980760
  },
  "duration": {
    "milliseconds": 1623771
  },
  "overall_bit_rate_mode": {
    "shortName": "CBR",
```

Figure 1.2 A partial listing of the resulting JSON after encoding of the General data block extracted using php-MediaInfo

MediaArea[14], putting a wrapper around it for use with PHP. While MediaInfo itself is available under Linux as a command line tool, having this type of wrapper and abstracting the functionality for use in PHP is advantageous due to the elimination of the need to run MediaInfo directly at the operating system level and then pulling the output in to the application. This prevents a possible malicious access vector which could compromise server security. Version v0.7.82 of MediaInfoLib under Linux, which is the version installed using the package installer *apt-get* (Debian, *yum* under CentOS and others), is significantly outdated and for php-mediainfo to successfully process a media file, an upgrade to MediaInfoLib v18.12 (as of this writing) is needed and has been used in this testing. See technical notes in the Appendix B for further details. Working with php-mediainfo is a fairly straightforward process of including and importing the proper libraries, instantiating the class, and passing the location of the digital media file to the class to obtain the output. Four primary classes exist to create objects that can then be processed in whatever manner necessary for the application. The first class, `getInfo()`, creates a container which is comprised of the objects containing the media data. `getGeneral()` retrieves the general, more high-level, data from the media file (this is the data that appears in the “General” area within the MediaInfo stand-alone software.) `getAudios()` parses the audio stream data from this container and creates arrays with the attributes of one or more audio streams if they exist in the media file. `getVideos()` performs the same basic tasks as `getAudios()` but operates on the video stream data and parses that into arrays of attributes specific to any present video stream or streams in the media file. Once the media information has been processed by these classes, it is a simple matter of converting to JSON (fig. 1.2) and arrays (fig. 1.3) for use

within the coding environment and for display.

Object : stdClass	
count	331
count_of_stream_of_this_kind	1
kind_of_stream	Object : stdClass
	shortName General
	fullName General
stream_identifier	Empty String
count_of_audio_streams	1
audio_format_list	MPEG Audio
audio_format_withhint_list	MPEG Audio
audio_codecs	MPEG Audio
complete_name	//audio/f2a3d748c5c38ac38be9cb214518ca68.mp3
folder_name	//audio
file_name	Array : Dimension 1
	idx:0 f2a3d748c5c38ac38be9cb214518ca68.mp3
	idx:1 f2a3d748c5c38ac38be9cb214518ca68
file_extension	mp3
format	Object : stdClass
	shortName MPEG Audio
	fullName MPEG Audio
format_extensions_usually_used	m1a mpa mpa1 mp1 m2a mpa2 mp2 mp3
commercial_name	MPEG Audio
internet_media_type	audio/mpeg
file_size	Object : stdClass
	bit 25980760
duration	Object : stdClass
	milliseconds 1623771
overall_bit_rate_mode	Object : stdClass
	shortName CBR
	fullName Constant

Figure 1.3 A partial listing of array dump from php-MediaInfo General data block after JSON decoding and conversion

pymediainfo[15]

The pymediainfo package, written in Python, operates in a similar manner to php-mediainfo in that it provides a wrapper around the native MediaInfo module on the Linux operating system. As such, it performs the same processing and derives the same information which is passed back to the programmatic front end for further manipulation. Pymediainfo can be included in a python script, instantiated as a MediaInfo object and then passed a media file location to perform the task of data extraction. Once the data has been parsed, a JSON encoded

array of that data can be generated (fig. 4) and in turn converted to name/value pairs and inserted into the database in any way the developer sees fit in the context of the coding environment.

```
{
  "tracks": [{
    "file_name": "WS700235",
    "file_extension": "MP3",
    "file_size": 19635910,
    "format_extensions_usually_used": "m1a mpa1 mp1 m2a mpa2 mp2
mp3",
    "stream_size": 1024,
    "duration": 1227154,
    "stream_identifier": "0",
    "other_overall_bit_rate_mode": ["Constant"],
    "overall_bit_rate_mode": "CBR",
    "codec": "MPEG Audio",
    "complete_name": " WS700235.MP3",
    "other_overall_bit_rate": ["128 Kbps"],
    "file_last_modification_date__local": "2017-04-30 06:18:42",
    "count_of_stream_of_this_kind": "1",
    "other_kind_of_stream": ["General"],
    "track_type": "General",
    "other_codec": ["MPEG Audio"],
    .
    .
    .
  }, {
    "format_version": "Version 1",
    "other_channel_s": ["2 channels"],
    "track_type": "Audio",
    "other_compression_mode": ["Lossy"],
    "other_bit_rate": ["128 Kbps"],
    "bit_rate": 128000,
    "format_profile": "Layer 3",
    "stream_size": 19634469,
```

Figure 1.4 A partial listing of the JSON result from pymediainfo processing of an audio file.

The JSON output is very similar to that which is found using php-mediainfo and although there are differences in the arrangement and formatting of the data, generally, the same data is present between both sets. Both php-mediainfo and pymediainfo would be sufficient for use in

extracting the media data from audio and video files, however, pymediainfo introduces another language on which the system would be dependent and the requirement for maintenance and compatibility an added consideration.

MP3Info (Merelo-Guervós)[16]

The version of MP3Info by J.J. Merelo-Guevós is a Perl based script which takes an MP3 file as input and extracts ID3 and ID3v2 information from the file[17]. This information is limited to MP3 audio files and more robust utilities exist which can derive deeper file characteristics. Moreover, as a Perl based utility, this adds a layer of complexity and additional compatibility requirements to any systems on which the database resides, negating the key goal of The Denver Multimedia Database of close compatibility across platforms. No further testing of this library was pursued.

Mp3Info (Tefft)[18]

This utility, last released November 14, 2006, is comprised of a Windows binary, RedHat/Fedora Linux binary, and the C language source code[19]. Attempts to compile the source code for Debian 7 (the operating system on which the DMDB system was installed for this project) were unsuccessful. It is suspected, based on the errors thrown during compile, that this failure is partially due to outdated code, and partially due to libraries being required which were incompatible with the target system. It is beyond the scope of this project to debug and correct these issues. No further testing was pursued for this library in favor of more successful results with others. Moreover, similar to the results using the Mutagen library, this version of Mp3Info was limited to “[modification of] the ID3 tags of MP3 files”[20] and it is suspected that it would not have derived the deeper information about an audio file that is desired for this database.

phpExifTool (Romain Neutron)[21]

The phpExifTool library is a PHP-based library for reading EXIF data from image files. It is a very robust and well written module utilizing an object-oriented architecture. However, it has one drawback which is that it relies on a Perl based sub-system with which it derives the EXIF data and which is what actually processes the media sample. As with Merelo-Guervós' MP3Info it adds a layer of complexity and requirement for compatibility that is not as desirable as having a system that uses only a few technologies but provides a robust sample processing mechanism to extract both consistent and reliable data elements from each media type. For that reason, no further testing of this library was pursued beyond rudimentary setup and experimentation to determine underlying technology.

Native PHP function `exif_read_data()` [22]

As a function that exists within the main component of the architecture, `exif_read_data()` lends itself to be the desirable selection for processing image files. Rudimentary testing was done with sample images from an iPhone 7 and the amount of data derived was impressive for a simple function (fig. 5). Further review of the data showed that it was more raw than desired but a review and understanding of the EXIF standard[23] allowed for translation of the raw values (e.g. GPS data, f-stop, etc.) to more readable and familiar values.

Array : Dimension 1		
idx:FILE	Array : Dimension 2	
	idx:FileName	a11bc924c1c368b5897158cd971aa6ce.jpg
	idx:FileDateTime	1554476995 (2019-04-05 09:09:55)
	idx:FileSize	1159685
	idx:FileType	2
	idx:MimeType	image/jpeg
	idx:SectionsFound	ANY_TAG, IFD0, THUMBNAIL, EXIF, GPS
idx:COMPUTED	Array : Dimension 2	
	idx:html	width="3264" height="2448"
	idx:Height	2448
	idx:Width	3264
	idx:IsColor	1
	idx:ByteOrderMotorola	1
	idx:ApertureFNumber	f/2.2
	idx:Thumbnail.FileType	2
	idx:Thumbnail.MimeType	image/jpeg
idx:IFD0	Array : Dimension 2	
	idx:Make	Apple
	idx:Model	iPhone 6
	idx:Orientation	3
	idx:XResolution	72/1
	idx:YResolution	72/1
	idx:ResolutionUnit	2
	idx:Software	10.0.1
	idx:DateTime	2016:09:15 12:15:05
	idx:YCbCrPositioning	1
	idx:Exif_IFD_Pointer	206
	idx:GPS_IFD_Pointer	1692

Figure 1.5 A partial listing of the array dump which is the result of EXIF data extraction using the PHP native function `exif_read_data()`

The required processing of the raw data was not such that it would be difficult to implement. It was found that the required translations only entail some mathematical functions and derivation of units (meters, time zones, etc.)

Results

Based on review and experimentation of various libraries used for processing the three types of media files, it was determined that phpMediaInfo and the native PHP function `exif_read_data()` would be used to process audio and video samples; and image samples, respectively. These two tools meet the initially stated goals of working within the main architecture of the system, extraction of a sufficient volume of the desired data from the samples, and a consistency in the naming of that data such that the database will have a cohesive across media types. They also help to limit the number of technologies relied upon in building the system making it more straightforward on both the front and back ends.

It should be noted that this does not preclude the use of Python or other language-based modules for further enhancement of the system. Use of these modules would not have a detrimental effect on the system other than adding another layer of technology. Python is a powerful language and libraries exist for processing of media files that may enhance and add value to the database. However, in this first iteration, for reasons of consistency, portability, and creation of a solid foundation on which to build, and because both sufficiently extract necessary data, the PHP modules were selected for the extraction of audio, video, and image media information.

CHAPTER II

EXISTING RESEARCH

It is likely that there are a number of databases that exist where the creators have catalogued and recorded information about audio and video sample files. These could take the form of spreadsheets, directory structures with precise naming conventions, or actual databases. The natural tendency when addressing any collection of items is to try to make sense of them and understand each element in the context of the whole. But these databases often exist for the owners' use without any intent to provide access and, perhaps, no realization that access might be beneficial to others.

However, there do exist databases and their attendant collections of media that expressly exist for study and a bringing together of like media for the purpose of keeping a central repository. One such database, and that which was the inspiration for this paper, is The Dresden Image Database which is a database of over 14,000 images created under controlled conditions for the purpose of providing a corpus for use in digital image forensics[24]. The creators of the database set out to create a publicly available database to be used in the study of digital cameras and how they produce images, including sensor noise patterns, the use of compression algorithms, and other general characteristics of the resulting images and camera settings themselves[25]. The intent being to answer the questions as to whether an image purported to have come from a given camera model is authentic and can images be linked back to the camera by which it was made.

Another database, and one of the larger endeavors in the media forensics discipline, is that designed for DARPA's MediFor program. The dataset strives to meet the needs of various evaluation tasks specifically designed to test forensic systems. These tests not only set out to

measure performance but also effects on performance, diversity, complexity, and quality and quantity of media required for results from those system to be considered accurate[26].

Khurram, et. al. present a novel dataset for use in evaluating audio forensic algorithms[27]. The researchers propose a database which can be used to estimate background noise and reverberation levels, acoustic environment identification, automatic identification of telephone handsets and microphones, and identification of audio recorder[28]. They assert that a challenge within the discipline of multimedia forensics is the lack of publicly available data with which testing of algorithms can be performed. Their study consisted of evaluation of acoustic features of various recording devices and acoustic environments in order to fill the gap in this part of the discipline.

Another approach to image databases is to mine the data of the images in non-image-specific databases. For example, toolmark, footwear, handwriting, cartridge casings, drug tablets, and faces[29]. In their approach, Garadts and Bijohd strive to evaluate the usefulness of the surrounding data of these databases to provide methods for linking cases or other valuable derivations with the idea that it could further enable examiners to limit the number of returned results or permit the discovery of links between elements of seemingly unrelated databases making investigation more efficient and fruitful.

There are also commercial databases available which provide listings of characteristics of media files. One such database is an audio and video recording database named DB-Expert[30]. It provides a method for acquisition and storage of audio and video media with the added benefit of file structure parsing and feature/characteristic extraction for storage in the database similar to what is being accomplished with the DMDB. The drawback is that it is commercial and there is a cost associated with being able to use the software. This expense is

not always possible to take on in a variety of forensic labs so a lot of the time, low cost or free alternatives are sought out and the DMDB strives to exist under that model.

CHAPTER III

METHODOLOGY AND LOGIC

The primary goal of the DMDB was to create a database similar to The Dresden Image Database but extend it to audio and video so a corpus of all three media types could be compiled under a single system.

Just as important to this goal are ease-of-use; a simplified platform for users, contributors, and administrators; a code base that can easily be understood for anyone with a little development experience; and, finally, a platform that is well-known and has a large install base such that if the need arises for finding a developer to implement enhancements, it would neither be expensive nor time consuming to extend the system's usability and features into the future.

The WordPress platform was selected as the base platform of the DMDB. The is prolific use of WordPress as a content management system (CMS). With 50% of the market share and an install base of 313, 957 installed in the top 1 million websites[31], it is currently the most used CMS. Therefore, the likelihood of finding someone familiar with both the underlying code base and the management tools with the administrative environment are increased. The vision for the DMDB is that articles or other writings are posted to the system to provide context for the tools available there. It could not only be a central repository for the media files and their data, but an information sharing portal for multimedia forensics-related information. Moreover, it is open source, well-supported and continually kept current with regard to security patches. There are a variety of plugins available for many different tasks should the need to add functionality to the front end arise. Finally, WordPress is PHP based which marries well with the technologies used for extraction of the media data for the database. All of this negates the need to maintain security across multiple languages and, from a technical standpoint, does not require the need for

someone who understands multiple languages and platforms in order to make enhancements or changes.

For the same reasons of flexibility, support, portability, and maintenance, the server architecture of the DMDB was chosen to be Debian Linux, utilizing the MySQL database and Apache web server. The LAMP stack (Linux, Apache, MySQL, and PHP) is a very widely used architecture for service of web technologies with some variation in the web server (e.g Nginx instead of Apache) and database (e.g. MariaDB instead of MySQL) but the code base of the DMDB is compatible with any of those technologies so can be ported if necessary, with little change to the system overall.

Security was also a consideration as was mentioned above relative to the reticence to use methods that executed functions or software directly through hooks to the operating system. Linux is a well-supported and well-known community with the installation of patches and updates nothing more than issuing a few *apt-get* commands (similar to *yum* on CentOS and other flavors of Linux). Automattic, the developers of WordPress, are vigilant in ensuring that the WordPress code base is patched as soon as a vulnerability is found.

Overall, it is believed that with the feature set of WordPress, the technologies selected for acquisition of the data from the media files, and the underlying server architecture, the DMDB is set to perform well into the future and has many possibilities for enhancement and expansion.

The DMDB is by no means a finished product. As with any piece of software or web-based tool, it is never really complete and can always be improved and enhanced as new ideas come forth or new requirements arise for its use. Those improvements are possible with the flexibility, openness, and straightforward nature of the DMDB.

CHAPTER IV

THE INTERFACE AND DATABASE

There are actually three distinct areas within the site with which users and administrators can interact. They are the main, front-facing, interface, the WordPress administration area, and the system administration area.

Front End

As with any online system there is the initial landing page at which users arrive when visiting a website. The homepage for the DMDB is simply an introduction to the system and its purpose. It also displays a “ticker” of the number of audio, video, and image samples plus counts of codecs for audio and video. The text is the abstract of this paper but can be whatever text is deemed appropriate in the long term.

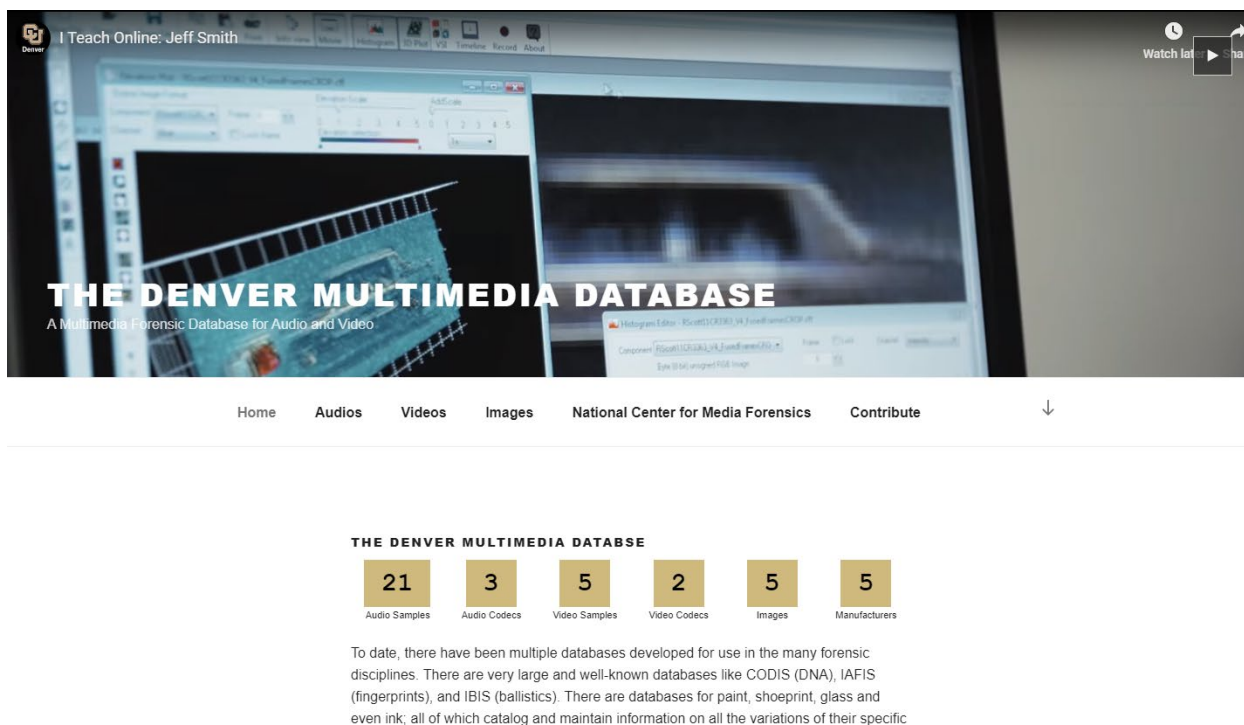


Figure 4.1 The DMDB Home Page

Audios, Videos, Images

On these three pages, there exists a sidebar with which to filter samples. These filters are derived from the specific media type attributes within the database and only those items applicable to that media type are available for selection. Initially, the page is blank on the right side with only the filtering selections visible to the left.

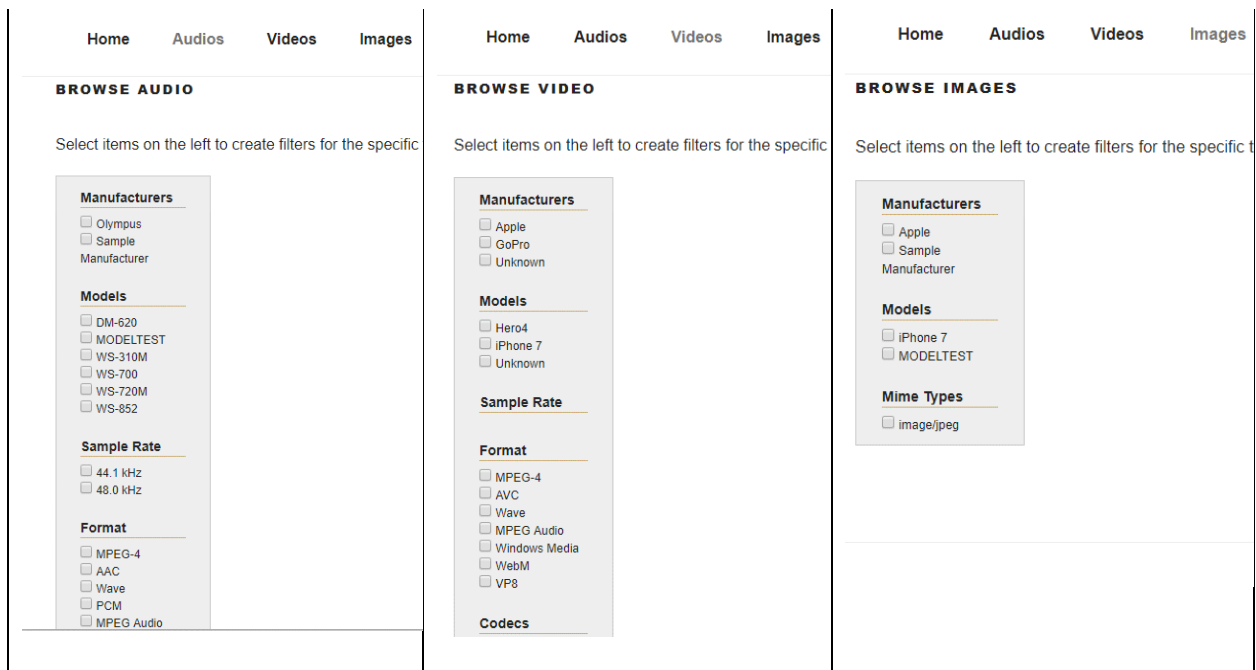


Figure 4.2 Audios, Videos, and Images Filtering Options

Selecting any one or more of the check boxes under Manufacturers, Models, and the other specific headings on the left side will make a call to the database and retrieve the basic information for samples with those attributes. As more attributes of each selection group are present in the database, these areas will automatically add those to the selection groups.

THE DENVER MULTIMEDIA DATABASE

A Multimedia Forensic Database for Audio and Video

[Home](#)
[Audios](#)
[Videos](#)
[Images](#)
[National Center for Media Forensics](#)
[Contribute](#)

BROWSE AUDIO

Select items on the left to create filters for the specific types of audio samples you want to review.

Manufacturers

☒ Olympus
 ☐ Sample

Manufacturer

Models

☒ DM-620
 ☐ MODELTEST
 ☐ WS-310M
 ☐ WS-700
 ☐ WS-720M
 ☐ WS-852

Sample Rate

☐ 44.1 kHz

Olympus - DM-620	46,164 bytes	8d36b9dcaab214be31c254922521ec72
Test audio sample1		
SHOW DETAILS		
Olympus - DM-620	45,583 bytes	d2020fb4fcebcc0e9da5f1ec1749f864
Test audio sample2		
SHOW DETAILS		
Olympus - DM-620	7,227,350 bytes	88898e0063dc167b49a8c7bfb40537bd
Test sample audio		
SHOW DETAILS		

Figure 4.3 A Listing (Audio) After Sample Attributes Selected. Audio and Video Sample Listings are Similar.

THE DENVER MULTIMEDIA DATABASE

A Multimedia Forensic Database for Audio and Video

[Home](#)
[Audios](#)
[Videos](#)
[Images](#)
[National Center for Media Forensics](#)
[Contribute](#)

BROWSE IMAGES

Select items on the left to create filters for the specific types of image samples you want to review.

Manufacturers

☒ Apple
 ☒ Sample

Manufacturer

Models

☐ iPhone 7
 ☐ MODELTEST

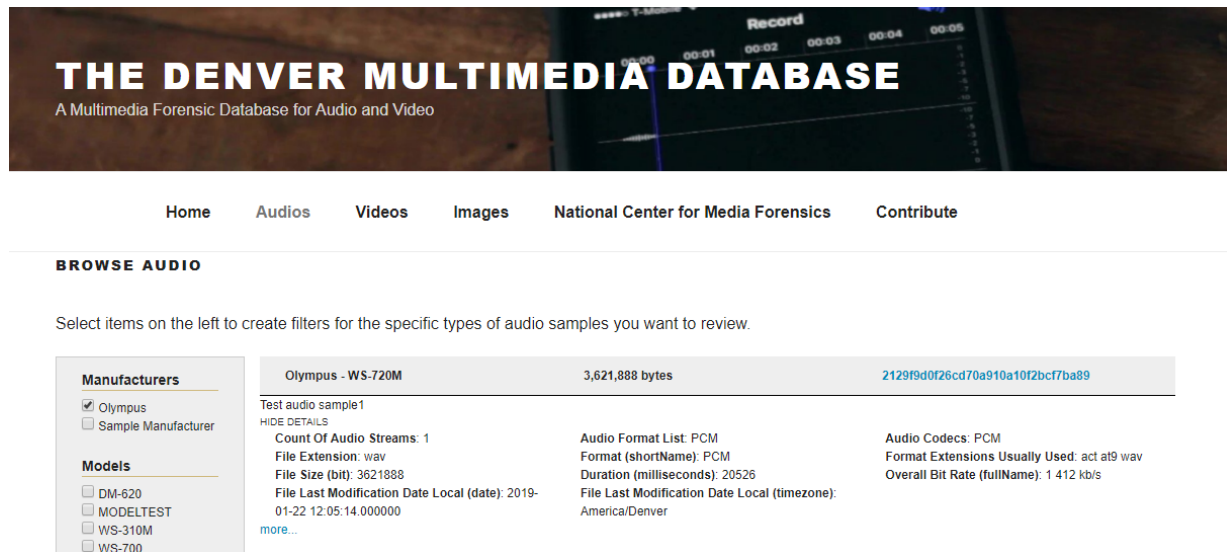
Mime Types

☐ image/jpeg

	Apple - iPhone 7	3,332,395 bytes	6609db94e89af5c4d12c3b826274788a
Test image for image import.			
SHOW DETAILS			
	Sample Manufacturer - MODELTEST	357,095 bytes	2708e24c0028bdb7d32d4c2b43d7e15f
testing single upload process			
SHOW DETAILS			
	Sample Manufacturer - MODELTEST	222,095 bytes	5e25a702873724becf51ac6500def578
testing single upload process			
SHOW DETAILS			

Figure 4.4 Image Listing After Sample Attributes are Selected

Once sample attributes are selected and a list appears, the “SHOW DETAILS” link under the user-entered sample description can be clicked to expand a basic feature list of the sample with a “more...” available for a full listing of all sample attributes.



The screenshot displays the 'THE DENVER MULTIMEDIA DATABASE' interface. The header includes navigation links: Home, Audios, Videos, Images, National Center for Media Forensics, and Contribute. Below the header is a 'BROWSE AUDIO' section with a filter instruction: 'Select items on the left to create filters for the specific types of audio samples you want to review.' On the left, there are two filter categories: 'Manufacturers' (with 'Olympus' checked and 'Sample Manufacturer' unchecked) and 'Models' (with 'DM-620', 'MODELTEST', 'WS-310M', and 'WS-700' all unchecked). The main content area shows a table with one row for an audio sample. The table has three columns: 'Olympus - WS-720M', '3,621,888 bytes', and a blue link '2129f9d0f26cd70a910a10f2bcf7ba89'. Below the table, the sample name 'Test audio sample1' is shown with a 'HIDE DETAILS' link. The details are expanded, showing a 'Count Of Audio Streams: 1' and a list of attributes: File Extension (wav), File Size (3621888 bits), File Last Modification Date Local (2019-01-22 12:05:14.000000), Audio Format List (PCM), Format (shortName) (PCM), Duration (20526 milliseconds), File Last Modification Date Local (timezone) (America/Denver), Audio Codecs (PCM), Format Extensions Usually Used (act at9 wav), and Overall Bit Rate (1.412 kb/s). A 'more...' link is also present.

Olympus - WS-720M	3,621,888 bytes	2129f9d0f26cd70a910a10f2bcf7ba89
Test audio sample1 HIDE DETAILS Count Of Audio Streams: 1 File Extension: wav File Size (bit): 3621888 File Last Modification Date Local (date): 2019-01-22 12:05:14.000000 more... Audio Format List: PCM Format (shortName): PCM Duration (milliseconds): 20526 File Last Modification Date Local (timezone): America/Denver Audio Codecs: PCM Format Extensions Usually Used: act at9 wav Overall Bit Rate (fullName): 1.412 kb/s		

Figure 4.5 Basic Information Expanded for an Audio Sample

[Home](#)
[Audios](#)
[Videos](#)
[Images](#)
[National Center for Media Forensics](#)
[Contribute](#)



aa9d4911bd0c7f4efc6014b09f529
2019-04-03 15:33:04
412,357 bytes
image

Keywords:
test keyword1, test keyword2

Administrator
P8240021.JPG
Sample Manufacturer
MODELTEST
SERIALTEST

testing single upload process

Computed

Height: 1280
ByteOrderMotorola: 0
UserCommentEncoding: UNDEFINED

Width: 960
ApertureFNumber: f3.8
Thumbnail.FileType: 2

IsColor: 1
UserComment:
Thumbnail.MimeType: image/jpeg

IFD0

ImageDescription: OLYMPUS DIGITAL CAMERA
XResolution: 72
Software: v874u-74
Exif IFD Pointer: 248

Make: OLYMPUS OPTICAL CO.,LTD
YResolution: 72
DateTime: 2002:08:24 16:36:53

Model: C960Z D460Z
ResolutionUnit: 2
YCbCrPositioning: 2

Thumbnail

Compression: 6
ResolutionUnit: 2

XResolution: 72
JPEGInterchangeFormat: 1244

YResolution: 72
JPEGInterchangeFormatLength: 3657

EXIF

ExposureTime: 0.0027700831024931
ISO Speed Ratings: 125
DateTimeDigitized: 2002:08:24 16:36:53
ExposureBiasValue: 0
LightSource: 0
MakerNote: OLYMP
ColorSpace: 1
FileSource:

FNumber: 3.8
ExifVersion: 0210
ComponentsConfiguration: ☐☐☐☐
MaxApertureValue: 3
Flash: 0
UserComment:
ExifImageWidth: 960
SceneType:

ExposureProgram: 2
DateTimeOriginal: 2002:08:24 16:36:53
CompressedBitsPerPixel: 2
MeteringMode: 5
FocalLength: 9.9
FlashPixVersion: 0100
ExifImageLength: 1280

Figure 4.6 The Sample Details Page with All Attributes for an Image Sample (Not the Full Page) Admin Interface

The administrative interface is where contributors can add sample content but also allows for adding or removing of users. There are three levels of access; user, contributor, and administrator.

Users can browse the sample database, download samples, and review sample details. At present the administrative User level is not used and access to samples is granted to anyone who can visit the site. If login capability is desired such that only a select group is allowed to review sample listings and sample details, this can easily be implemented. Contributors can do everything users can but may also contribute to the database and upload their own samples. Administrators have all access and can do what Users and Contributors can but have the added ability to add or remove users and will also have other functions available if they are added to the

system (e.g. review/approval of samples, removal of samples, sample updates, etc.).

The sample upload area consists of two general options. The user defines what the sample is, where it came from (if known), associated keywords (content information, specific details about conditions under which the sample was captured, etc.), and detailed notes about the sample. This is important information which the contributor would know and may not be available in the sample information extraction details. It provides a small kernel of “meta” data that defines the sample and provides a quick overview for users on the front end.

There are two way of uploading samples. A single sample upload through one of the Audio, Video, or Image icons or a multiple sample upload process where a user can drag and drop multiple samples to initiate upload and processing of those samples.

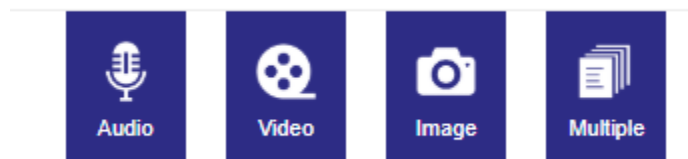


Figure 4.7 Contributor Icon Bar

In the single upload process the user can enter the “meta” information and click to choose the sample they want to upload in a single step.

Upload Audio Sample

Fill in the information below about the source of the file and click to add file. Then click "Save". The system will make every attempt to extract the metadata and other characteristics of the file and load that information to the database.

Source Information

Manufacturer	Choose File No file chosen
Model#	
Serial#	
Keywords (comma seperated)	
Notes about this sample	
Save	

Figure 4.8 The Sample Upload Form

Uploading multiple samples consists of two steps; entering the “meta” data and then proceeding to the drag and drop interface to begin the upload process of all samples (note the missing “Choose File” button in the Upload Multiple Samples interface.)

Upload Multiple Samples

Fill in the information below about the source of the files and click to add files. Each image will be associated with the information below so BE SURE you are uploading media samples from the same source. The system will make every attempt to extract the metadata and other characteristics of the file and load that information to the database.

Source Information

Audio	▼
Manufacturer	
Model#	
Serial#	
Keywords (comma seperated) 	
Notes about these sample	

Continue

Figure 4.9 Step 1 of Multiple Sample Upload Process

Upload Multiple Samples

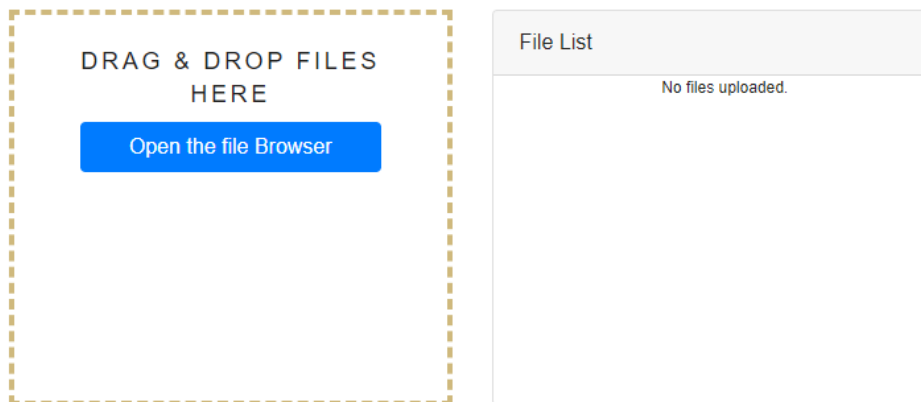


Figure 4.10 Step 2 of the Multiple Sample Upload Process

During the multiple sample upload process, once the files have been dropped into the hot zone, as designated by the dashed box, the File List area will list the samples and provide a status for each as it is uploaded.

Upload Multiple Samples

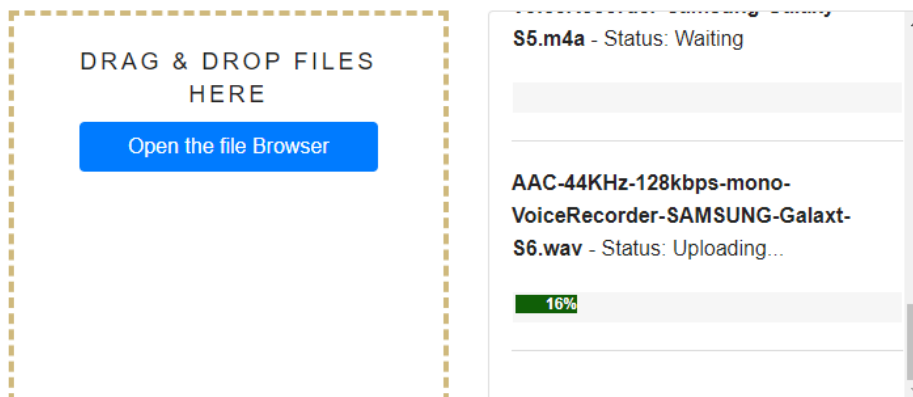


Figure 4.11 During Upload A Scrollable Box with Status of Each Sample's Progress is Available for Review

Whenever a sample is uploaded, each file is associated with the user who uploaded it so that, if needed, the user can be contacted regarding additional information about the sample.

Additionally, a hash value is generated for the binary data of the sample file, stored in the database, and the sample file renamed in the format of *[hash value].[original extension]*. This provides a way to ensure, if downloaded, a sample is the same as originally uploaded but also provides a method by which the system can check to verify that the incoming sample has not already been added to the system. The original filename is stored with the sample record in the database for reference.

If a user has administrative capability, they will also see a “Users” icon available in the secondary menu at the top of the page.

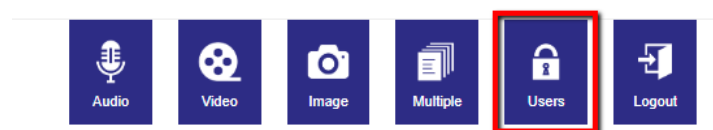


Figure 4.12 Administrative Icon Bar

This area is used for adding, changing, or deleting users and establishing their access level. Clicking this icon will take the administrator to the user listing.

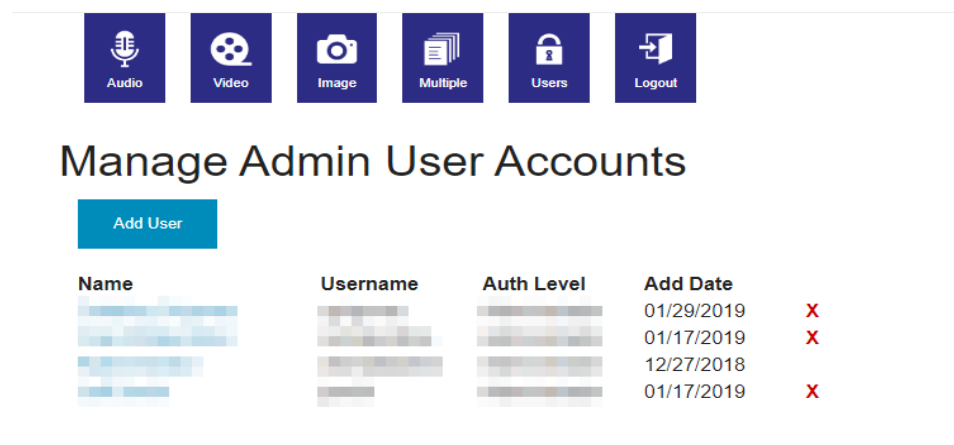


Figure 4.13 User Listing

From there the “Add User” button can be selected to add a user. If it is the intent to edit an existing user, the name of the user in the listing is hot and will link to the user edit page. This

page serves the dual purpose of allowing for addition or modification of a user account.

Audio

Video

Image

Multiple

Users

Logout

Manage Admin User Accounts

Name:



UserID:



Password:



(To keep password,
leave blank)

Auth Level:

Administrator

▼

Update User

Figure 4.14 User Edit/Add Management Interface

WordPress Administration

The WordPress administrative interface is quite robust and an in-depth explanation is beyond the scope of this writing. It is suggested that the documentation on WordPress be studied at <https://www.wordpress.org> for those interested and who have access. The following is a basic overview of the more salient features of WordPress that will likely be used most often. For those who are familiar with WordPress, this will not be new information.

Once logged in, the user is presented with a left sidebar with various options and a right area where the WordPress Dashboard is displayed.

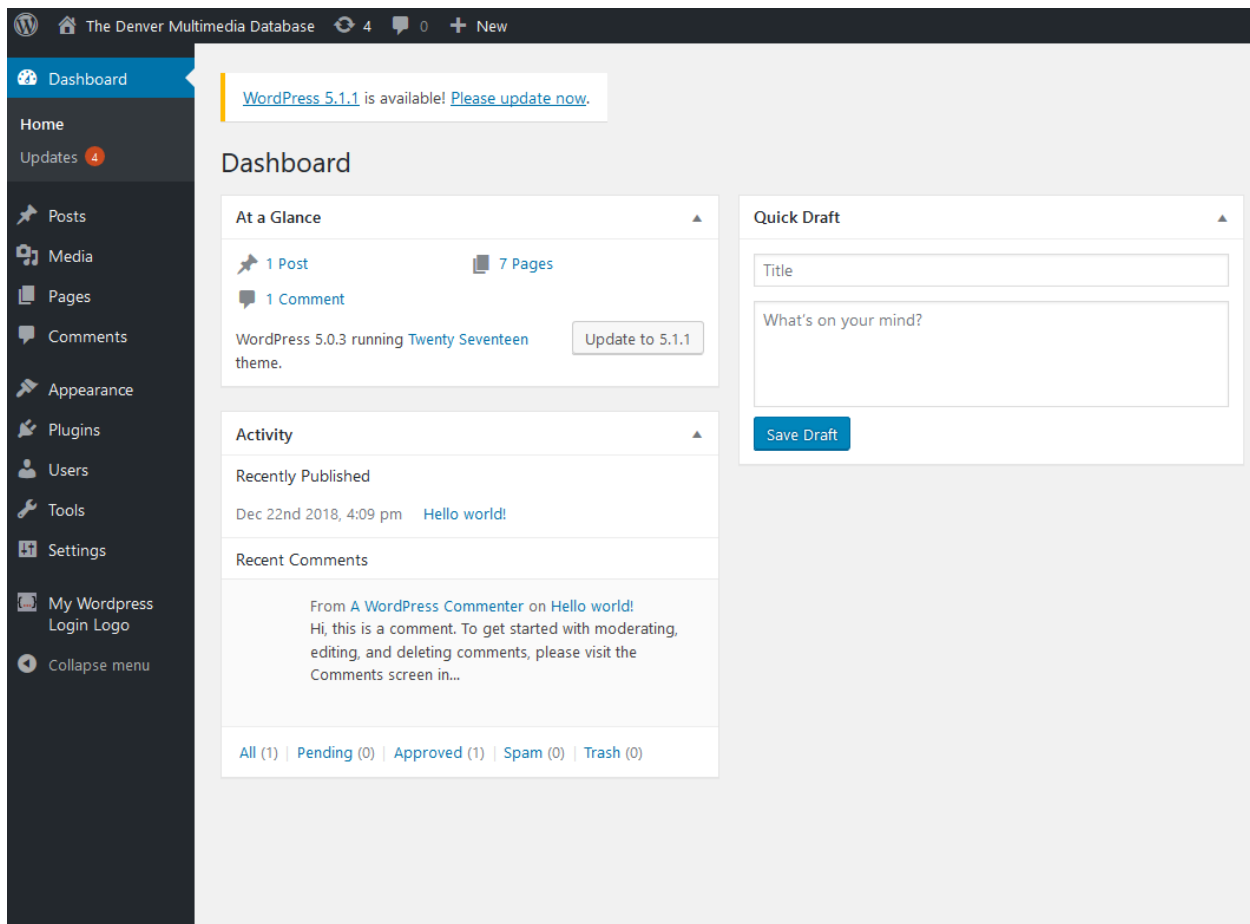


Figure 4.15 The WordPress Dashboard

What is available on the left side will vary depending on plugins installed and what role a user has been given (roles are not used currently in the DMDB.)

The primary areas are Posts, Pages, and Appearance. Posts are used for articles, information pages, and other like content which would be listed together for access by visitors to the site. Pages are the more static elements and can consist of “About” type pages, or other types of content that do not rotate out and typically have links in the main menu. Appearance is used to adjust the look and feel of the public facing interface and there are various options available for changing the header video currently in place, tagline, colors, fonts, font sizes, etc.

Database

As mentioned previously, the database resides under the MySQL database application. There are two sets of tables; tables which handle the CMS aspects of the system and are installed by WordPress, and tables that are specific to sample information and users of the admin area. Further information about the table structure of the WordPress database can be found at <https://www.wordpress.org>.

The database tables related to samples and users consist of only five tables.

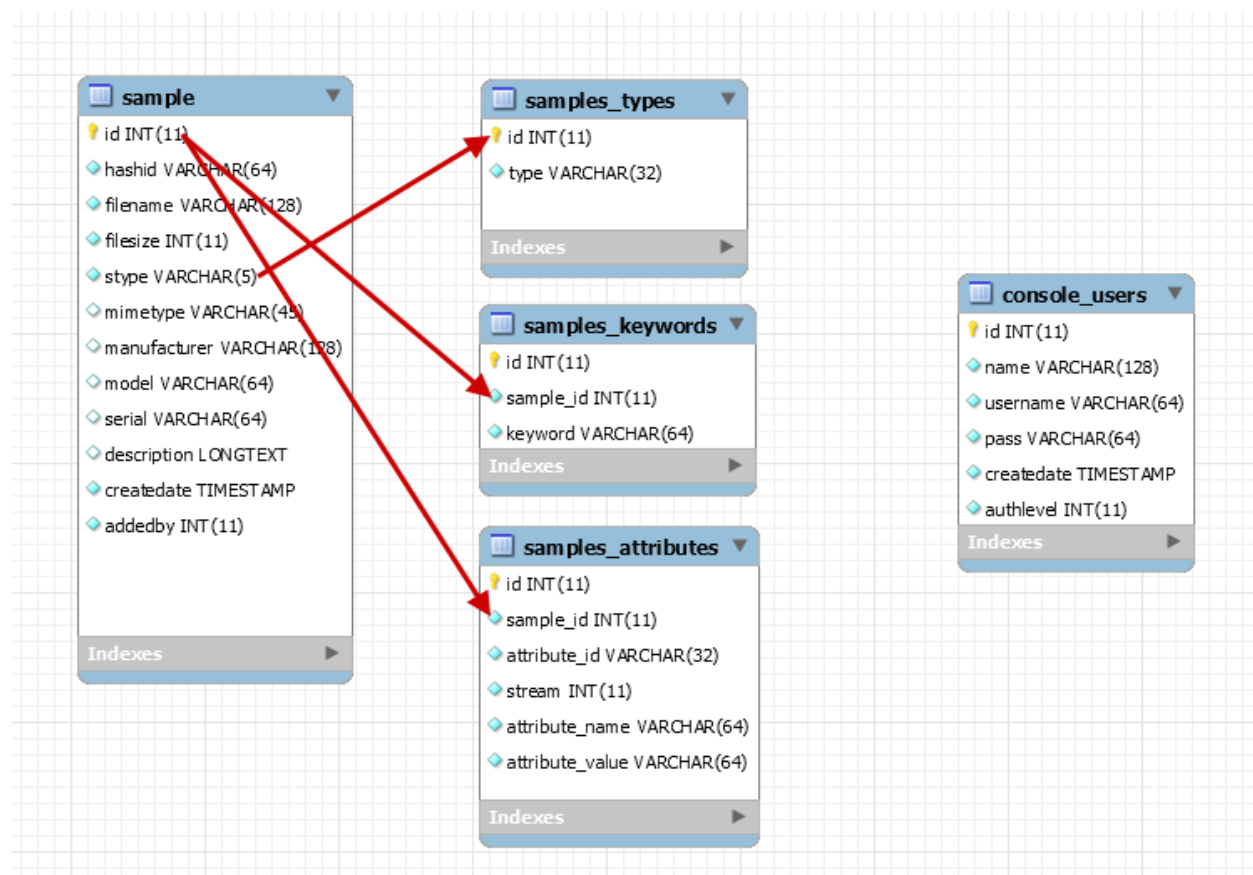


Figure 4.16 DMDB Database Tables and Relationships

A sample's general information such as original file name, hash value, the user-entered data, who entered it and date uploaded are all stored in the main *sample* table. There is a stype value which is linked to the *sample_types* table and translates the id into the name of that type.

Each of the keywords entered by a user (in comma delimited format) is broken out and saved as a separate record associated with the sample id. Finally, all of the attributes found by the extraction process are also separated into individual records associated with their sample id and include what “section” of the data it came from. For example, if a given datum was found in the audios section of a video file, it would have an attribute id of “audio” so it could be grouped and listed as such on the sample details page.

When requesting sample data from the database, the system simply joins the *sample*, *samples_attributes*, and *samples_keywords* table information using the common *sample_id* and returns the entire complement of data back to the front end for display and layout.

The *console_users* table is where users and authorization levels for each are stored.

CHAPTER V

GOING FORWARD

With any database application and especially with one involving a web-based front end, the project is never truly complete. New technologies, ideas, and enhancements to existing functionality will always play a factor its evolution. Furthermore, education of users and administrators is an important component to the proper use of any system but most especially one that is technical in nature and has the added requirement of needing clean data to be ingested into it.

Below are some considerations on what might be possible and what should be kept in mind as use of this system begins and its continued use into the future.

User Training

The quality of any system is predicated on the quality of data it contains and the attention to that quality given by the users. Therefore, it is paramount that users be educated in the importance of submitting clean samples to the database and ensuring that any data they enter be specifically related to the sample or samples they submit. Procedures and protocols may be necessary which will lay out the proper method for generating sample media which can include media length, file size, procedures for capturing audio samples. Additionally, close attention should be paid to matching sample attributes (that information submitted in the user-entered form data) accurately reflect the sample or samples being submitted.

An original function for the submission process was to verify that the type of sample being submitted was what the user was saying it was. It was decided that this should be impressed in the user training so that prior to any approach to the submission process, the organization of the media samples is performed. This serves two purposes; the media a user is

creating or working with is organized in their local library, and they begin to develop and practice the mindset of clean samples and data.

Enhancements

By its very nature, this database and its attendant features are living in the sense that they can be adjusted or improved upon into the future. With that in mind, possible enhancements can include mapping of EXIF GPS data. Rudimentary (or even more complex) analysis of media can be performed and attached to the sample record for a more robust presentation when reviewing a given sample and discovery of newly uploaded samples. One such library, called `pyAudioAnalysis`[32], could be implemented to do further analysis on sample audio files, providing graphs, graphic analysis results, and other features that could be of use for the database. This is a Python-based module which, as mentioned above, is counter to the desire to simplify technologies used on the platform, but it can be evaluated to discover the trade offs associated with layering additional technology and what benefits this module might yield. For that matter, additional Python libraries may be leveraged to do analysis on the image or video in this database if the need or desire arises. Limitations to the initial assertion of keeping it programmatically clean are impractical over the long term due to the robustness of, in particular, Python-based libraries for this kind of analysis.

Search could also be added to the system to allow for searches on specific hash values, searches on keywords, and any number of other data that would be of interest in grouping media samples together. The filtering mechanism (see The User Interface for further details) provides a search of sorts but is specific in its operating and groups media samples based on fixed characteristics. A search would provide a more fluid mechanism for any value within the media data itself.

Administrative tools can be added to edit or delete samples to ensure administrative control over the cleanliness of the database. Other sample-centric tools might be desired in an administrative context to ensure full control over the material that resides in the database and on the file system. It may be desirable to implement an approval mechanism such that no sample is visible until an administrator approves it.

Since each sample is associated with the user who uploaded it, a future enhancement could be to select only those samples from that user via a link on the sample detail page.

A regimen of articles, how-tos, and informational documents can be added to the content of the site (pages or posts), to allow for not only a central repository for the three media types but also one for information content that can assist in education and study. This expansion can take many forms and it is up to the imagination of the owners and caretakers as to where and how this evolves.

CITATIONS

- [1] G. Schaefer and M. Stich, UCID: an uncompressed color image database, Proceedings of the SPIE, 5307:472-480, 2003.
- [2] J. Kalpathy-Cramer, A. Herrera, D. Demner-Fushman, S. Antani, S. Bedrick and H. Müller, Evaluating performance of biomedical image retrieval systems - an overview of the medical image retrieval task at ImageCLEF 2004-2013, Computerized Medical Imaging and Graphics, 39:55-61, 2015.
- [3] G. Griffin, A. Holub, and P. Perona, Caltech-256 Object Category Dataset, California Institute of Technology. <http://resolver.caltech.edu/CaltechAUTHORS:CNS-TR2007-001>, 2007.
- [4] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition, PAMI, 30(11):1958–1970, 2008.
- [5] Federal Bureau of Investigation, <https://www.fbi.gov/services/information-management/foipa/privacy-impact-assessments/iafis>.
- [6] National Institute of Standards and Technology, <https://www.nist.gov/news-events/news/2018/09/database-software-fingerprints-expands-include-computer-games>
- [7] Federal Bureau of Investigation, <https://www.fbi.gov/services/laboratory/biometric-analysis/codis/codis-and-ndis-fact-sheet>
- [8] Bureau of Alcohol, Tobacco, Firearms and Explosives, <https://www.atf.gov/firearms/national-integrated-ballistic-information-network-nibin>
- [9] Guan, Haiying et al. "Mfc Datasets: Large-Scale Benchmark Datasets for Media Forensic Challenge Evaluation." 2019.
- [10] Wreschnig, Joe et al. "Mutagen - Python Multimedia Tagging Library." <https://mutagen.readthedocs.io/en/latest/index.html>
- [11] Ibid.
- [12] Ibid.
- [13] GitHub, <https://github.com/mhor>
- [14] MediaArea, <https://mediaarea.net/>
- [15] pymediainfo, <https://pymediainfo.readthedocs.io/en/stable/>
- [16] Meta::Cpan, <https://metacpan.org/pod/MP3::Info>.

- [17] Ibid.
- [18] iBiblio, <http://www.ibiblio.org/mp3info/>.
- [19] Ibid.
- [20] Ibid.
- [21] Github, <https://github.com/romainneutron/PHPExiftool>
- [22] PHP, `exif_read_data`, <https://www.php.net/manual/en/function.exif-read-data.php>
- [23] "Exchangeable Image File Format for Digital Still Cameras:Exif Version 2.31." *IPA DC-008-Translation*, Camera & Imaging Products Association, July, 2016 2016.
<http://www.cipa.jp/std/documents/e/DC-008-Translation-2016-E.pdf>
<http://www.cipa.jp/std/documents/e/DC-008-Translation-2016-E.pdf>.
- [24] Gloe, Thomas et al. "The 'Dresden Image Database' for Benchmarking Digital Image Forensics." *Proceedings of the 2010 ACM Symposium on Applied Computing*, ACM, 2010, pp. 1584-1590. doi:10.1145/1774088.1774427.
- [25] Ibid.
- [26] DARPA MediFor,
<https://www.darpa.mil/program/mediaforensics>,[https://www.fbo.gov/index?s=opportunity&mode=form&id=bfa29e5f04566fbb961cd773a8a8649f&tab=core &_cview=1](https://www.fbo.gov/index?s=opportunity&mode=form&id=bfa29e5f04566fbb961cd773a8a8649f&tab=core&_cview=1).
- [27] Muhammad Khurram, Khan et al. "A Novel Audio Forensic Data-Set for Digital Multimedia Forensics." *Australian Journal of Forensic Sciences*, vol. 50, no. 5, 2018, pp. 525-542, doi:10.1080/00450618.2017.1296186.
- [28] Ibid.
- [29] Geradts, Zeno J. and Jurrien Bijhold. *Data Mining in Forensic Image Databases*. vol. 4709, SPIE, 2002. *Aerosense 2002*.
- [30] OT Contact, <http://www.timbre.ru/product.php?WHAT=33&lang=en>
- [31] BuiltWith, <https://trends.builtwith.com/cms>
- [32] Github, <https://github.com/tyiannak/pyAudioAnalysis>

BIBLIOGRAPHY

- "Exchangeable Image File Format for Digital Still Cameras:Exif Version 2.31." *IPA DC-008-Translation*, Camera & Imaging Products Association, July, 2016 2016.
<http://www.cipa.jp/std/documents/e/DC-008-Translation-2016-E.pdf><http://www.cipa.jp/std/documents/e/DC-008-Translation-2016-E.pdf>.
- A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition, *PAMI*, 30(11):1958–1970, 2008.
- Boato, Giulia. "Multimedia Forensics: An Overview of Recent Advances and Open Issues." *Proceedings of the First International Workshop on Multimedia Verification*, ACM, 2017, pp. 1-1. doi:10.1145/3132384.3132388.
- BuiltWith, <https://trends.builtwith.com/cms>
- Bureau of Alcohol, Tobacco, Firearms and Explosives,
<https://www.atf.gov/firearms/national-integrated-ballistic-information-network-nibin>
- Cinthya Grajeda, Frank Breiting, Ibrahim Baggili. "Availability of Datasets for Digital Forensics – and What Is Missing." *Digital Investigation*, vol. 22, 2017, pp. S94 - S105, doi:<https://doi.org/10.1016/j.diin.2017.06.004>.
- DARPA MediFor, <https://www.darpa.mil/program/mediaforensics>
- Federal Bureau of Investigation, <https://www.fbi.gov/services/information-management/foipa/privacy-impact-assessments/iafis>.
- Federal Bureau of Investigation,
<https://www.fbi.gov/services/laboratory/biometric-analysis/codis/codis-and-ndis-fact-sheet>
- G. Griffin, A. Holub, and P. Perona, Caltech-256 Object Category Dataset, California Institute of Technology. <http://resolver.caltech.edu/CaltechAUTHORS:CNS-TR2007-001>, 2007.
- G. Schaefer and M. Stich, UCID: an uncompressed color image database, *Proceedings of the SPIE*, 5307:472-480, 2003.
- Geradts, Zeno J. and Jurrien Bijhold. *Data Mining in Forensic Image Databases*. vol. 4709, SPIE, 2002. *Aerosense 2002*.
- Giannakopoulos, Theodoros. "Pyaudioanalysis:An Open-Source Python Library for Audio Signal Analysis." *PLoS ONE*, vol. 10, no. 12, 2015, doi:10.1371.
- Gloe, Thomas et al. "The 'Dresden Image Database' for Benchmarking Digital Image Forensics." *Proceedings of the 2010 ACM Symposium on Applied Computing*, ACM, 2010, pp. 1584-

1590. doi:10.1145/1774088.1774427.

Guan, Haiying et al. "Mfc Datasets: Large-Scale Benchmark Datasets for Media Forensic Challenge Evaluation." 2019.

J. Kalpathy-Cramer, A. Herrera, D. Demner-Fushman, S. Antani, S. Bedrick and H. Müller, Evaluating performance of biomedical image retrieval systems - an overview of the medical image retrieval task at ImageCLEF 2004-2013, *Computerized Medical Imaging and Graphics*, 39:55-61, 2015.

MediaArea, <https://mediaarea.net/>

Meta::Cpan, <https://metacpan.org/pod/MP3::Info>.

MP3Info, <http://www.ibiblio.org/mp3info/>.

Muhammad Khurram, Khan et al. "A Novel Audio Forensic Data-Set for Digital Multimedia Forensics." *Australian Journal of Forensic Sciences*, vol. 50, no. 5, 2018, pp. 525-542, doi:10.1080/00450618.2017.1296186.

National Institute of Standards and Technology, <https://www.nist.gov/news-events/news/2018/09/database-software-fingerprints-expands-include-computer-games>

National, Research Council. "Strengthening Forensic Science in the United States - a Path Forward." edited by The National Academies Press, 2009. doi:<https://doi.org/10.17226/12589>.

OT Contact, <http://www.timbre.ru/product.php?WHAT=33&lang=en>

PHP, exif_read_data, <https://www.php.net/manual/en/function.exif-read-data.php>

Phpexiftool, Github, <https://github.com/romainneutron/PHPExiftool>

Php-mediainfo, Github, <https://github.com/mhor>

Pyaudioanalysis, Gitub, <https://github.com/tyiannak/pyAudioAnalysis>

Pymediainfo, <https://pymediainfo.readthedocs.io/en/stable/>

Simson Garfinkel, Paul Farrella, Vassil Roussev, George Dinolta. "Bringing Science to Digital Forensics with Standardized Forensic Corpora." *Digital Investigation*, vol. 6, 2009, pp. S2 - S11, doi:<https://doi.org/10.1016/j.diin.2009.06.016>.

Wreschnig, Joe et al. "Mutagen - Python Multimedia Tagging Library." <https://mutagen.readthedocs.io/en/latest/index.html>

APPENDIX A

CLASS LISTING

Database

The database class provides an underlying connection layer to the database between the front end and administrative areas for the passage of records and instructions. It is a simple class which establishes the database connection and provides functions for access and use of that connection.

It primarily uses the PHP `mysqli_*` set of functions.

- Member Variables

For security reasons some member variables cannot be disclosed in this document.

\$hostname – The IP address of the local (or remote) host on which the software is installed and the database daemon is running

\$link – the link to the database hook that allows for creation, updates, and extraction of database records

\$dbname – The name of the database underlying the entire system

\$dbuser – The user specified which has access to the database tables and records.

- Member Functions

__construct()

This function establishes the default time zone for the system, establish the connection to the database via a set of parameters which include the member variables of this class.

This class is automatically triggered any time the class is instantiated.

getLink()

Returns the `$link` member variable value to the calling process.

Sample

This is the class that handles all manipulations, creation, and insertions into the database of audio, video, and image sample data. It manages parsing of inquiry results from the MediaInfo library and organizes the data into an efficient array. It also handles parsing and processing of EXIF raw data.

- Member Variables

\$db – This variable is assigned a passed in value that is derived from the Database class and is used in all member functions when database services are needed.

\$mediaData – This variable is an array which contains the results from an inquiry via MediaInfo on a specific media file whether it be an audio or video sample. The media data is the specific characteristics of a sample and corresponds with the “Video” and/or “Audio” data blocks of the MediaInfo library results.

\$mainData – This member variable provides basic high-level information about a sample such as name, format, files size, etc. This corresponds with the characteristics from the “General” data block of the MediaInfo library results.

- Member Functions

__construct()

This function assigns the passed in database link value to the member variable \$db in order to make it available for use within the class. This function is triggered automatically on instantiation of the Sample class.

save()

Once the \$mediaData and \$mainData variables have been hydrated in the object with the information from a given sample, this function saves that information to a database. The

data is laid out in standard array format such that any media data passed in will be consistently saved to the database tables alongside any other media data.

getSample(\$sid)

A call to this function will return an array with all of the data associated with the given sample ID. The sample ID is a database assigned id and has no other function than to connect all parts of the sample record. See data model for further details.

getBasicAudioAttributes(\$sid)

A specialized function, this is used for front end display and selects only specific and more readily recognized data elements of an audio sample for initial output when reviewing filter results. See “The User Interface” and member function `getBasicItemList()` for further details.

getBasicVideoAttributes(\$sid)

Similar to `getBasicAudioAttributes()` but for use with video samples.

getBasicImageAttributes(\$sid)

Similar to the audio and video functions above but for use with image samples.

writeAudioVideoData(\$mediadata,\$attid,\$stream)

This function takes the raw audio and video sample data derived from `phpMediaInfo` and processes it for insertion into the database. The raw data passed in has multilevel arrays and that data is flattened and saved to the Sample object. Once this data has been inserted into the Sample object, the `save()` function is called to pass the data to the database. The `$attid` variable designates to the type of media (audio, video) and the `$stream` variable handles the instance where there would be multiple video or audio streams in the media data.

writeExifData(\$exif)

Due to the differences in how the data is laid out and in methods by which that data was derived, this separate function processes the image EXIF data similar to how the writeAudioVideoData() function processes the stream data.

Within some of the raw EXIF data there are fractions as they were stored in the media stream. For example, the GPS coordinates, if they are present in the file, are saved as the fractions that represent the hours, minutes, and seconds values. Therefore, a simple function was written (exp2Num()) to convert these values to more usable floats for saving to the database and final display.

writeMainData(\$postdata, \$media)

The user-entered data is saved to the Sample object with this function. File information is extracted from the \$media variable and the information filled out on the form when submitting a piece of media comes from the \$postdata variable array. See “The User Interface” for further details.

getMediaData()

Returns the media data information component of a media sample.

getMainData()

Returns the user supplied information component of a media sample.

getBasicItemList(\$stype)

Where \$stype is one of audio, video, or image, this function will return the list of items to be pulled from the database for display on the initial filtering page. There is a check for which type was passed in and based on that value, an array with the names of those elements for display is returned. See “The User Interface” for further details.

dudeItsAlreadyThere(\$hash)

Runs a check on the database during sample submission to be sure the media has not already been uploaded. Process aborts if this is the case.

getGPS(\$exifCoord, \$hemi)

Converts the three hours, minutes, and seconds values in the Longitude or Latitude arrays from the EXIF data into decimal form for storage in the database.

exp2Num(\$exp)

A simple math function to convert the fraction values within some of the EXIF raw data into a float. Used by getGPS as well as conversion of other simpler data values, e.g. fstop, speed, etc. to floats for storage to the database.

Sanitize

Because data is being passed from a web page to ultimately be saved to the database, this function exists to scrub the data and be sure it is “clean” for insertion. This for the purpose of mitigating possible vectors for SQL injection attacks. It operates on the \$_POST, \$_GET, \$_REQUEST, and \$_COOKIE browser session objects.

- Member Variables

\$db – The database link which needs to be established in order to use MySQL functions.

- Member Functions

__construct(\$link)

Sets the \$db private variable to the passed in link and then executes the clean() function on the four browser session objects.

clean(\$input, \$db)

Loops through the values in the browser session object arrays and runs the

mysql_prep() function to sanitize all values and eliminate possible special string combinations that could result in a SQL injection

mysql_prep(\$input,\$db)

A small function that simply uses the native MySQL PHP function mysql_real_escape_string to scrub the \$input value of any possibly malicious content, escaping special characters and rendering them inert.

Users

A class used in administrative functions to manipulate and add user accounts to the system. Depending on authorization level, these user account have access to be able to contribute to and manage The Denver Multimedia Database system.

- Member Variables

\$db – The database link which needs to be established in order to use the MySQL functions.

- Member Functions

getAdminUserList()

A simple function to retrieve all records from the *console_users* table in the database. It is used for a listing of the users which provides a link for editing user accounts.

addAdminUser(\$req)

Takes a browser session request object and inserts the data into the database to add a new user record.

saveAdminUser(\$req)

Used for updating an existing user record in the database

getAdminUser(\$id)

Retrieves a user record from the database. Primarily used in the edit process.

deleteAdminUser(\$id)

Removes a user record from the database rendering that user unable to access the system.

Utility

A class for miscellaneous functions which can be used to perform minor and ancillary tasks alongside the other classes and their functions. This currently only has one function but other can be added as the system evolves.

- Member Variables

None

- Member Functions

getFileExtension(\$str)

A function to parse a passed in file name and derive the file extension. Uses simple string manipulation functions to break up the file name and pass back everything after the . in a filename.

APPENDIX B

TECHNICAL NOTES

The platform on which the underlying project for this paper is based is the Debian 7 (Wheezy) operating system on the amd64 architecture.

Using mediainfo to get the most possible data from the media files requires specific libraries to be installed but the default mediainfo installation that can be installed on Debian with *apt-get install mediainfo* is many versions behind the newest mediainfo and that which provides the best results (using the default yields very little when working with video files).

The following libraries must be installed for proper mediainfo data extraction with .deb package files available at <https://mediaarea.net/en/MediaInfo/Download/Debian>. Be sure to select the proper OS version and architecture for the application if installing this system from scratch or moving the current DMDB system. If using a different Linux flavor, this may serve as a guide to making the proper libraries available.

libmms0 (can be installed using *apt-get install libmms0*)

libzen0 (.deb file)

libmediainfo0 (.deb file)

mediainfo (.deb file)

For .deb package installation the following commands should be used once the package has been downloaded.

dpkg -i /path/to/.deb/file

apt-get install -f

...for each .deb file in the order above.