NTNU – Trondheim
Norwegian University of
Science and Technology

Mini-MAM

Trine Rein

Electronics System Design and Innovation
Submission date: June 2014
Supervisor: Andrew Perkis, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

The amount of digital content outside the broadcasting industry is rapidly increasing, and the existing professional tools are often outside the scope of this new wave of content owners. To manage an increasing amount of content, as well as metadata, content owners require access to Media Asset Management (MAM) systems [I]. Most MAM systems are large and costly to implement and maintain, and are specifically designed for the broadcasting industry.

The task of this project is to design, implement and test a Mini-MAM system which take the most important functionalities of a complete MAM system in order to satisfy the requirements of small organisations, entities and individuals. The Mini-MAM system must be fully functional, however, the complexity and price must be significantly less than existing professional systems. The starting point will be the requirements outlined in the report Media Asset Management, Prosjektrapport [II]. The prototype is to be evaluated with a subjective usability test, and compared to a commercial content management system.

[I] Karl Paulsen. *Moving Media Storage Technologies: Applications and Workflows for Video and Media Server Platforms.* Focal Press, 2011.

[II] Andrew Perkis and Håvard Ferstad. Media asset management, prosjektrapport. Technical report, NTNU, June 2013. Found 28.8.2313.

Preface

This thesis presents my work throughout my final semester at the Department of Electronic and Telecommunication at the Norwegian University of Science and Technology (NTNU). The purpose of this thesis was to develop a prototype of a mini-media asset management system.

I would like to thank my supervisor Professor Andrew Perkis for his good help, and patience throughout this semester. I would also like to thank the people at Expology for providing me with their XMS system, which has served as a benchmark for my own system. I would also like to thank Klaus-Peter Engelbrecht at TU Berlin for his advice regarding the design of the Graphical User Interface. And finally I would like to thank all those who participated in the usability study, in addition to Kristian Rygh Jerndahl for good help and support.

Trondheim 09.06.2014
Trine Rein

Abstract

As the amount of digital content outside the broadcasting industry increase, the need for Media Asset Management systems among this new wave of content owners has increased. An efficient, user friendly and cheap MAM system is required to satisfy their demands, a system we will denote as "Mini-MAM". A MAM system is the core component for a content owner in organizing their content and associated metadata. Traditionally, MAM systems are complex and expensive systems, where most features are unnecessary for the intended users of a Mini-MAM. The goal of this thesis, is to design, implement, and test a proposed Mini-MAM system, that fits the demands of these new content owners.

The thesis will explain the basics of a MAM system, and explore the importance of metadata in terms of content management. The thesis will also indentify the primary requirements of the Mini-MAM system in order to meet the demands of small and non traditional content owners. This thesis highlights the important decisions made during the development of the system, as well as the obstacles that had to be overcome. The Mini-MAM is written using Python, relying primarily on the Tkinter library to create the interface. The Mini-MAM also relies on the SQLite database management system to store persistent data, and the ffmpeg media framework for transcoding.

The system has also been compared to a commercial exhibition management system in subjective assessment of the two systems. The other system to be assessed was the Expoloy XMS system, designed to manage the content of exhibition centres and museums. The assessment had 14 subjects perform basic tasks on the two systems, before answering a questionnaire where the subjects rate the user friendliness and usability of each system. The results from the subjective assessment showed that the Mini-MAM was the more user friendly system.The participants rated the systems on a scale from 1-5, where 1 is bad, and 5 is excellent. The participants rated their general impression of the Mini-MAM's user friendliness a score within the 95% confidence interval of 3.60-4.25. In comparison, the participans rating of the XMS System that fell inside the 95% confidence interval of 1.64-2.51. 100% of the users also reported that, as non-experts, they preferred the Mini-MAM system.

From the results of the subjective assessment, is clear that the Mini-MAM fulfils its role as a user friendly and intuitive system. Despite the XMS system being a far more extensive and powerful system, the complexity of the system confused the untrained participants. The Mini-MAM fulfils the basic requirements outlined in the thesis, and is easy to for new users. Despite being a prototype, the Mini-MAM shows promise, and can with further development, become a commercially viable system.

Sammendrag

Siden mengden av digitalt innhold utenfor kringkastingsindustrien øker, har det dukket opp et behov for nye struktureringsverkøy. For disse nye innholdsaktørene, er essensielt at dette verkøyet er brukervennlig, effektivt, og billig. I et MAM system har man oversikt over både innhold, og den tilknyttete metadataen, tradisjonelle MAM systemer er ofte store og krevende systemer, som er uegnet for mindre brukere. Målet i denne masteroppgaven er å designe, implementere og teste et mini-MAM system som utfyller behovene til disse nye innholdsaktørene.

Rapporten gir en innføring i hva et MAM system er, og forklarer viktigheten av metadata for god håndtering av mediainnhold. I rapporten identifiseres hovedoppgavene til et mini-MAM system: legge inn innhold, organisere metadata, i tillegg til å organisere og spille av medie innholdet. I designprosessen gjennomgåes de viktige avgjørelsene som er blitt tatt under utviklingen av systemet, i tillegg til utfordinger man har måttet løse. Mini-MAM systemet er skrevet i Python, og brukergrensesnittet baserer seg i hovedsak på Tkinter biblioteket. Systemet bruker også en SQLite database til lagring av data, og ffmpeg media rammeverket til transkoding.

I en subjektive test, er Mini-MAM systemet blitt sammenlignet med XMS systemet til Expology. XMS er kommersielt media-håndteringssystem, som fokuserer på å organisere innhold på et opplevelsessenter, I testen deltok 14 deltakere, som gjennomførte enkle oppgaver på de 2 systemene. Etterpå besvarte deltagerne et evalueringsskjema hvor de ratet brukervennligheten, tidsbruken og menyoppsettet til de 2 systemene. Resultatene fra testen viste at mini-MAMen var et mer brukervennlig og effektivt system. På en skala fra 1-5, der 1 er dårlig, og 5 svært bra, ga brukerne deres generelle inntrykk av mini-MAMens brukkervennligheten en score på 3.60-4.25 innenfor et 95% konfidensintervall. Til sammenligning, falt resultatet til XMS innenfor 1,64-2,51. Ut ifra perspektivet til en ny bruker, svarte 100% av deltagerne at de ville foretrukket å bruke Mini-MAM systemet.

Ut ifra den subjektive vurdering, kommer det klart frem at Mini-MAM prototypen er et svært brukervennlig og intuitivt system. Til tross for at XMS er et mer omfattende og kraftigere system, virker det som om at de mer avanserte egenskapene førte til forvirring blant testpersonene fra en ny bruker sitt synspunkt. Mini-MAMen tilfrettstiller de krav som er fremlagt i denne oppgaven, samtidig som den er enkel å sette seg inn i for nye brukere. Til tross for at den er en prototyp, viser Mini-MAMen at den har potensiale, og med mer utvikling kan bli et kommersiell produkt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A few decades ago, multimedia content was primarily distributed through the broadcasting industry. With today's technology, an increasing amount of multimedia content is created and consumed by individuals, as well as smaller entities [1]. With the increasing amount of multimedia content, comes the need to organize and manage this content. Broadcasting companies have been using Media Asset Management (MAM) systems to manage their multimedia content for several years, and as such, most MAM systems are catered towards entities with large amounts of content and resources. A typical MAM system will include the entire work-flow for a broadcasting entity, beginning with the ingest of the raw material, and ending with a fully edited media asset that may be published to different channels [2][3]. For private actors, as well as smaller enterprises and entities, some of the tools and functionality of a broadcast oriented MAM system may be unnecessary. These MAM systems are generally large systems, that require manpower as well as resources to maintain. For a smaller entity, the cost of both acquiring and maintaining these systems may be too cost intensive.

As a non-broadcaster, the need for advanced post production, and editing tools is greatly reduced. Entities, such as museums, schools, or private individuals primarily require a MAM system to catalogue and store their media content [4], as well as providing a streamlined method for providing and editing metadata to their content [5][6]. For a non-broadcaster, a simple system that allows for easy ingest, and content retrieval, is becoming a vital requirement. In a world of conflicting standards [7][8][9], large repositories, a MAM-system must be able to properly store, embed [10] and maintain metadata.

Our goal for this thesis, is to investigate the most important functionality and requirements of such a Mini-MAM, and implement them in a prototype. Starting with the requirements outlined in [11], this report aims to further refine the core functionality and requirements of a Mini-MAM. To help limit the scope of the prototype, the prototype will be limited to a single computer, instead focusing on the metadata related aspects of a MAM system. Once the requirements are established, a prototype of a Mini-MAM will be designed, developed and tested. As the proposed users of the system are private individuals, small entities and enterprises, the user interface of the Mini-MAM should be as user friendly as possible [12].

To elaborate on the implementation of the prototype we will examine, and explain the libraries [13][14][15][16], and tools [17][18][19][20] required to implement the system as well the challenges we face during development [21]. The system will be a traditional graphical user interface, focused on the interaction between keyboard,mouse and screen, despite new interesting opportunities arising in the field of user interface development [22][23]. To test the system, we will perform a subjective assessment [24], where the Mini-MAM is compared to a commercial content management system.

Chapter 2

Theory

This chapter will introduce the basic theoretical elements required to understand the needs, and methodology behind the Mini-MAM Prototype. It will also explain a few basic concepts related to the handling of metadata.

2.1 Media Asset Management

2.1.1 MAM - Introduction

Media Asset Management has traditionally been associated with keeping track of, and maintaining large media storage systems. In the past, this included keeping track of actual physical tapes, cataloguing information about the tapes, and their physical location in a storage system. Today, a MAM system will attempt to fill a much larger role in the media handling chain. Instead of simply being a tool used to store media assets, a modern MAM system will contain the entire tool-chain, from ingest to extraction [2].

Generally, a MAM system, refered to simply as a MAM can bee seen as a subset of a Digital Asset Management (DAM) system. Where the DAM deals with generic digital assets, the MAM deals with media assets, usually referring to images, video, or audio assets [2][3].

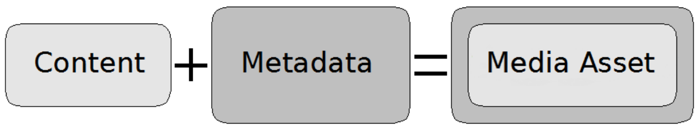


Figure 2.1: Definition of media asset.

As seen in figure 2.1, the media assets contained in a MAM contain more than the actual content, or 'essence'. In addition to the actual content data, we have the metadata, the information about the content. In figure 2.1, we see that the metadata acts as a wrapper, or second layer of information. In a MAM system, this will typically be information that aids in the process of tracking, editing or playing the media essence. The metadata should explain what the media essence contains, keywords to help link media items together, as well as technical information about the essence. The handling of metadata is rapidly becoming one of the most important aspects of a MAM, as proper usage of metadata empowers the other stages in a typical MAM toolchain.

A modern MAM system, should attempt to fill a set of basic requirements, and may also include other functionality depending on the context of the MAM [2].

Search: A MAM should be able to perform searches of the media content. It is vital to be able to search through both structural and descriptive metadata, to be able to find the right content, in the proper format.

Organize: The MAM should also be able to automatically organize, and link relationships between different media content. This is relevant for series of videos, or various forms of media used in a presentation or program.

Transfer: A MAM should also be facilitate the transfer of media content to other locations when necessary. This could be moving centralized content to local servers, or sharing media content with other platforms according to a pre-defined standard.

Edit: Advanced MAM systems may incorporate full-fledged editing tools when these are required. In the basic case, allowing users to segment or join media essences should be available. Combining two short video segments into one, or extracting a short clip from a longer video are examples of common tasks that should be supported.

What should be noted though, is that the main purpose of a MAM is to keep track of the media content [3]. This means that as files are moved, edited, and relationships edited, the MAM has to update and maintain both structural and descriptive metadata.

One of the challenges associated with metadata management in a MAM, is how to store the metadata. Two principle philosophies exist when dealing with metadata storage: embedding, or a separate database [2]. Embedding the metadata ensures that the link between descriptive metadata and the media essence persists through the tool chain, assuming all tools are able to work with the embedded format. Embedding the metadata does however lead to a few issues with archiving. If a proper version control system is implemented, storing a new backup every time the metadata changes, could lead to storage bloat, and hinder the system. The alternative is to use a separate database of metadata, where the system keeps track of the connections between metadata and media essence, updating the database whenever the essence is edited, or new metadata is to be added. As in many cases, the best option, seems to be the middle ground [4], and support both. This means the MAM needs the ability to strip internal metadata during content ingestion, and add it to its internal storage. Once a file is selected for exporting, the updated metadata can then be re-embedded, before checkout.

2.1.2 Mini-MAM

For the purposes of this thesis, we want to focus on the most important aspects of a Mini-MAM. From the specifications outlined in [11], we see that the ingest component of the Mini-MAM is responsible, and the basis for most operations. Therefore it is natural to choose the ability to ingest content as one of the focus points of the prototype. The second major focus point in our prototype, is the ability to add, and edit metadata for the media content in the Mini-MAM. Without metadata, the Mini-MAM is nothing more than a fancy folder with an interface. By choosing to focus on content ingestion, as well as searching metadata, we fulfil two of the basis requirements outlined by [2], namely *Organize* and *Search.*

In addition to metadata, we also want to focus on displaying the content. To help limit the scope of the prototype, we also chose to limit the Mini-MAM to a single computer. This is primarily to avoid having to deal with network code, as the primary functionality should remain the same.

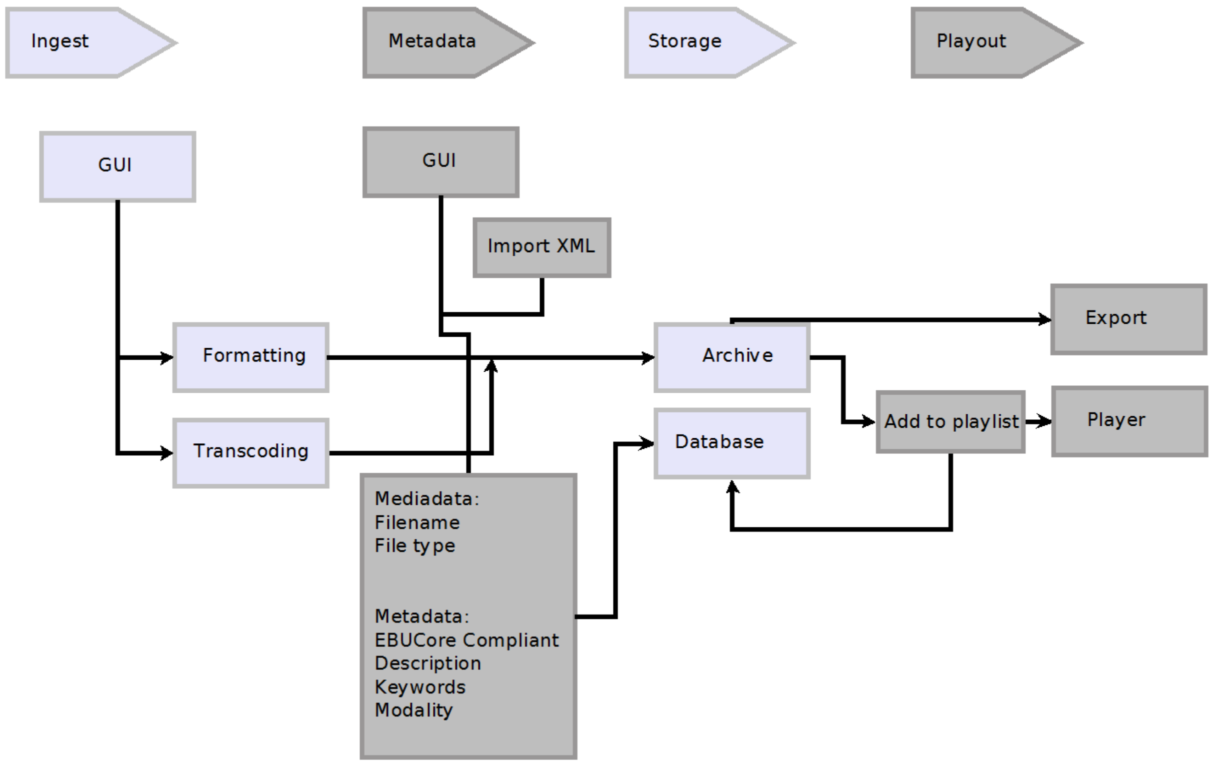


Figure 2.2: System Schematic of the Mini-MAM.

Figure 2.2 shows the proposed system schematic of the Mini-MAM. The schematic attempts to display the relationship between the various tasks and elements contained in the Mini-MAM prototype.

2.2 User Interface design

In the field of Human Computer Interaction (HCI), user interface design is an important subset. The user interface is the users primary method of interacting with any computer software [12]. In the interaction between a user and a computer, the user interface is what the users sees, hears, pushes, presses and interacts with. It handles both the input, as well the output of the process. Traditionally, the most common implementation of a user interface, involves a mouse and keyboard as the method of input, and a screen to handle the output. However, with today's advances in cellphone and tablet technology, touch screens are become and more popular. With several companies working on Virtual Reality prototypes [22][23], new types of user interfaces will have to be developed.

The goal when designing a user interface, is to provide the user with efficient tools to input information to the system. An ideal interface is one that is never truly noticed by the user, it allows the user to interact with the system without having to focus on how the interaction occurs. The user interaction should be as natural and intuitive as possible.

The general principles behind a good user interface can be summarized by a few select keywords [12].

- *Aesthetically Pleasing*
  By being Aesthetically Pleasing, a user interface takes advantage of graphical elements to help display or convey information to the user. By visually differentiating items, a user interface can create groups of items or elements that are similar, and visualize relationships between UI elements.

- *Clarity*
  Every aspect of the user interface should be clearly defined, and intuitive. It should be obvious to the user what an action's result is, before the action is performed.

- *Compatiblity*
  A user interface should be compatible in many areas. It should be compatible with different users, as users come in all varieties, the interface should cater to, and be understood by all. The interface should also be compatible with the task it controls, providing easy transition between supported tasks.

- *Comprehensibility*
  A system needs to be easy to learn and understand. A user should be able to discern the function of each element simply by seeing its location, description, or icon.

- *Configurability*
  By allowing configurability, the user gets an enhanced sense of control over the system. It also makes it easier for the user to understand the interface, as it encourages an active role.

- *Consistency*
  Design consistency is the first cardinal rule of user interface design [12]. It helps the user familiarize him/herself with the system, as understanding one element, will help the user understand other elements. As a primarily rule, if an element exists in more than one screen, it should have the same position, and functionality across screens. Elements that perform the same task, should look the same. As a rule of thumb, the same action should always produce the same result, and similar actions should produce similar results. It is also important to note that consistency can draw from other user interfaces, and standards/guidelines. Using similar elements other popular interfaces can drastically help reduce the time it takes for a new user to familiarize himself with the system. On the other hand *inconsistency* can be a major hindrance to good user interface design. An inconsistent interface will increase user error rates, and often lead to users spending more time with each screen. It also makes it harder to teach new users, and requires more detailed documentation.

- *Control*
  Users should always feel in charge of each action. Elements that change without user input, or limitations on the users action can be very frustrating. It should also be noted, that limiting the user, using modes should be avoided, if an element is visible, it should be usable.

- *Directness*

The user should always know what options and alternatives are available for an element. A user should also be able to directly see the effects of any actions on an object.

- *Efficiency*
  A user should be able to quickly transition between tasks. This applies on many levels, in terms of both direct input, as well as visual information. Wasting hand or eye movements should be avoided when possible. Familiar hotkeys, and other efficiency tools should also be available if possible.

- *Familiarity*
  Build on existing concepts and tools. This applies to language in the interface as well as elements.

- *Flexibility*
  A system must cater to different types of users. A flexible system responds well to the individual differences in the users. Flexibility does come with a cost, by having different methods to complete tasks, it can be difficult for inexperienced users to understand the system. By creating different methods to complete the same tasks, users can also end up only learning one method, and choosing to stick with it.

- *Forgiveness*
  A system has to incorporate that humans are prone to errors. Checks, prompts and queries to the user should be in place to prevent the user from making mistakes. The system should also be able to provide constructive error messages when an error occurs, allowing users to understand what went wrong. An important aspect of learning a new system, is trial and error. A good interface will facilitate the user to learn this way, instead of preventing it, or even worse, crashing.

- *Predictability*
  The user should be able to predict how different actions unfold. Closely related to consistency and directness, predictability is important for the learning process of a new user, and the efficiency of experienced users.

- *Recovery*
  The most important aspect of recovery, is that a user should never lose information permanently. It should always be possible to revert, or abolish an action, and return to previous point. A good interface will make sure that all interconnected data is stored and secure, even in the case of a critical failure.

- *Responsiveness*
  The system should aim to quickly and consistently respond to the user's actions. This can come in the form of a visual, a textual, or an auditory response. The user should never be left in the dark, if anything in going on the background, updating the user is extremely important. In essence, any action should have some sort of reactonary feedback.

- *Simplicity*
  A good user interface is a simple intuitive interface. Primarily, we have five ways to provide simplicty: Use Progressive disclosure - present information when it becomes required. Provide Defaults. Minimize screen alignment points. Make common actions simple. Provide uniformity and consistency.

- *Transparency*
  The user should be able to focus on the task, without having to worry about how the mechanics behind the interface function. The focus of the interface is to facilitate the tasks, not to focus on the underlying mechanics.

As seen on this list, there are some conflicts between certain items. A good interface design will balance the conflicting items to suit the needs of the desired interface. Another potential conflict is that of technical requirements versus the needs of the users. It is then important to remember the goal of a user interface, to help the user interact with the program. This is described as the second cardinal rule of graphical system developmenp: Human requirements should always take precedence over technical requirements [12].

2.3 Metadata

2.3.1 About Metadata

Metadata, or the 'data about the data' is an essential tool for handling media content. Without metadata, the only information about a file is what the operating system stores, name, date and other minor details. To provide the foundation for a proper MAM, additional information is necessary. Metadata is generally divided into two main categories, structural, and descriptive metadata [2].

Structural Metadata

Structural metadata primarily deals with what the media content is, it describes format, duration, aspect ratio, bit rate, and other technical information. This information holds the key to how the content is played out, and changes in the structural metadata will have an enormous effect on how the content is played out. Typically, this information is embedded in the media file, and handled by the recording and playback devices [2].

Descriptive Metadata

Descriptive metadata describes the content of the media. This will typically be genre descriptions, track names, episode number, names of characters or actors etc. Generally the descriptive metadata is limited to terms describing the basic properties of the content. Certain standards also allow for the adding time-stamped metadata, allowing the tagging of events in the content. Time stamping allows for tagging of for instance goal scoring in sports videos, or special events in movies, or even music [2][6].

Descriptive metadata is usually the aspect of metadata that is handled, or provided by a user, or MAM. A common method is to categorize the metadata using the Extensible Markup Language (XML) [5], where a standard, or *schema* is followed. Several such standards excist, usually catering to different types of media formats.

Writing Descriptive Metadata in XML

When actually writing the metadata, most standards operate in a similar manner. Under a base element, XML elements are added, and given appropriate values and descriptions. What separates two standards, is the different types of elements that may be used, and the different parameters that are linked to each element. This makes it difficult for a system to support more than one standard for metadata, as conversion from one standard to the other, require a complete mapping of elements from one schematic to elements of the other schematic. Such a mapping may be problematic, as an element in one standard can be either not be implemented, or be incorporated in other elements in a different schematic. Different methods on how to do this have been presented [9], and work continues on how to collaborate between excising metadata standards.

2.3.2 EBUCore

The EBUCore metadata set is designed, and maintained by the European Broadcasting Union [8]. It is based on the generic DublinCore [7] , and is targeted towards audio and video content. The metadata set is designed with simplicity in mind, allowing only generic metadata elements. This simplicity makes the standard very flexible, allowing it to describe a wide range of different material. Having the most generic and elementary metadata information available in a common format, allow entities to share media content and metadata, despite using proprietary standards for more advanced metadata entries. EBUCore offers basic functionality for the descriptive tagging of content, allowing simple keywords and descriptions, but reserves more advanced structures for information regarding rights owners, contributors, and technical information.

Apart from simple descriptions of media content, the MPEG-7 standards also define methods for entering a massive array of different metadata [7]. The standard defines descriptors for different shapes, colour schemes, facial recognition, and motion tracking for multimedia content. For audio content, the MPEG-7 allows the tracking of envelope, information about frequency components, and a broad spectre of other parameters. The MPEG-7 standard allows for the tracking, and storing of just about every aspect of a media file, making it an incredibly powerful standard.

2.3.3 Material Exchange Format

The Material Exchange Format (MXF) is a file format designed to help with the transfer, and maintenance of metadata in audio-visual work-flows [10]. The file format supports the embedding of metadata, as well as multiple ways of storing audio-visual essence data. MXF acts as a container format, storing media essences independent on the compression coding, it also allows for the storing multiple essences in the essence container, allowing users to store whole play lists in a single file. MXF also supports streaming, making it a valuable tool for server-client systems.

By allowing the embedding of metadata, the mxf format allows for easy transferring of metadata between entities, as well as between steps in a production chain.

2.4 Usability

The term usability is described as "the capability to be used by humans easily and effectively" [12]. Assessing the usability of a system is an important aspect of the design process, and usability should continuously be assessed throughout the development.

The most important point to remember when testing usability, is that usability is a quality that is very difficult to quantify. Users with different levels of experience will prefer different systems, and different elements in similar systems. Although usability is notoriously hard to quantize, the two primary attributes of a usable system interface, is efficiency and flexibility.

2.4.1 Subjective Assessments

The best way to test the usability of a system, is trough subjective assessments. By carefully selecting a set of observers, it is possible to get a fairly accurate assessment of how your system will perform in a true environment. By selecting non-experts that are unfamiliar with the development of the system [24], it is possible to assess the usability and user friendliness of a system.

As it can be difficult to get a large enough set of observers to get a statistically significant result, it can be necessary to perform statistical analysis on the results to verify them. The required analysis is presented in Chapter 3.

Chapter 3

Method

The Method chapter explains the process of designing,implementing and testing the Mini-Mam. In this chapter we will explain the decisions that were made during the design phase, and show how the design of the prototype evolved throughout the process. To elaborate on the development process, we look at obstacles, and challenges that were overcome during the implementation of the prototype. In the final parts of the chapter, we also examine the process behind the subjective assessment that is used to test the user friendliness of the Mini-MAM, and XMS systems.

3.1 Designing the Mini-MAM

3.1.1 Initial Design Process

The first step in the design process, was to identify the basic needs of the Mini-MAM system. Using the ideas from [11], as well as the core functionality of a Mini-MAM outlined in the theory chapter, three major tasks were chosen as core functions for the Mini-MAM prototype. The chosen tasks, 'Ingest', 'Search' and 'Play' were each envision to have their own main screen on the Mini-MAM. Using these three tasks, some very simple mock-ups were created, displaying the desired screens. Building upon the ideas from the theory chapter, it was natural to chose Ingest, and Search as main tasks, the ability to add content and metadata, and the ability to organize the metadata are two of the primary tasks of a Miin-MAM. By also choosing to add a Player, we hope to make the Mini-MAM a more user friendly system, that fit the basic needs of a larger target audience. By having a play screen, it is easier for smaller users to see the effect of a Mini-MAM, as having to open a new window to view a located media item can be cumbersome and frustrating.

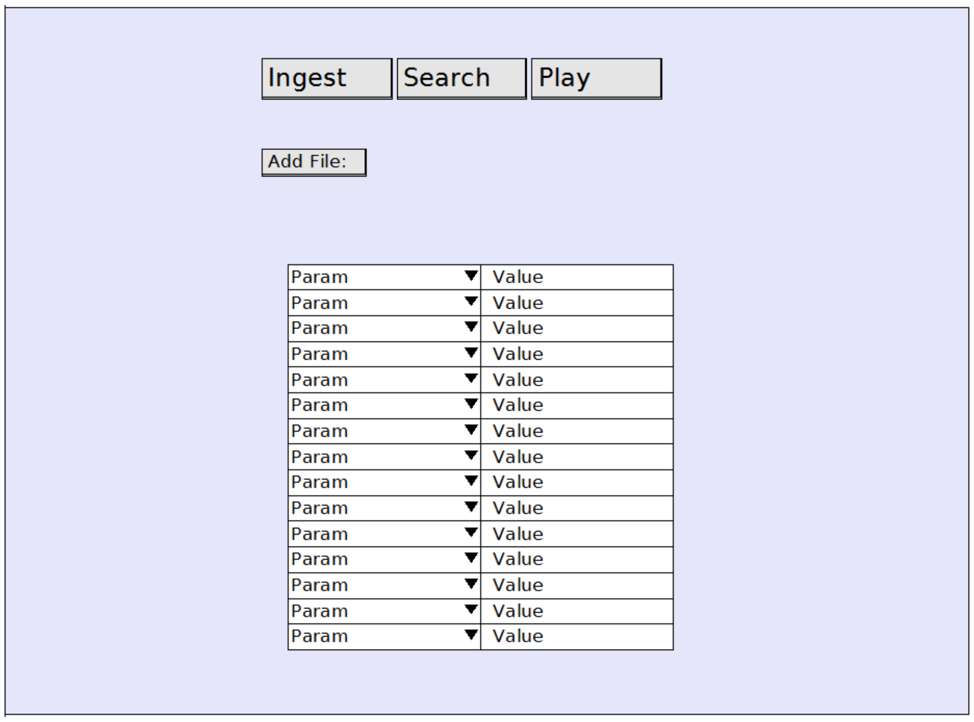


Figure 3.1: Ingest Screen of first iteration.

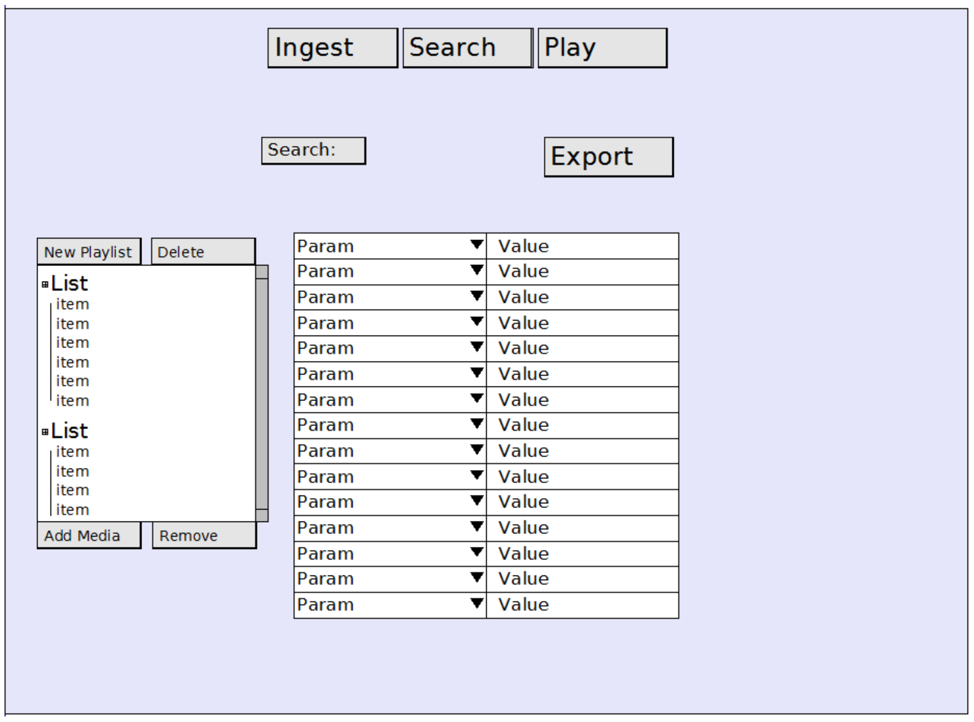


Figure 3.2: Search Screen of first iteration

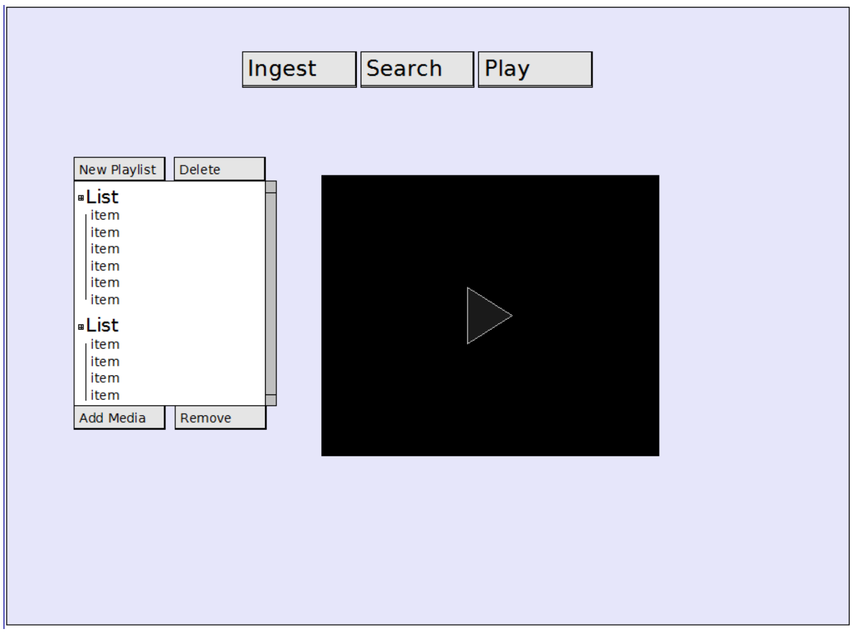


Figure 3.3: Player Screen of first iteration

As we see in figure 3.1, the initial idea for the ingest page, was a simple screen to add metadata parameters to the selected file. The thought behind the mockup is that each parameter is either a metadata field, or other information about the file, such as location, file type and name. In the Search screen mock-up in figure 3.3, we use the same structure. This is because it made creating the mock-up easier, but also because how important it is to maintain consistency in a Graphical User Interface (GUI) design . [12]. We also maintain consistency when using the same playlist view on the mock-up of the Play screen in figure 3.2.

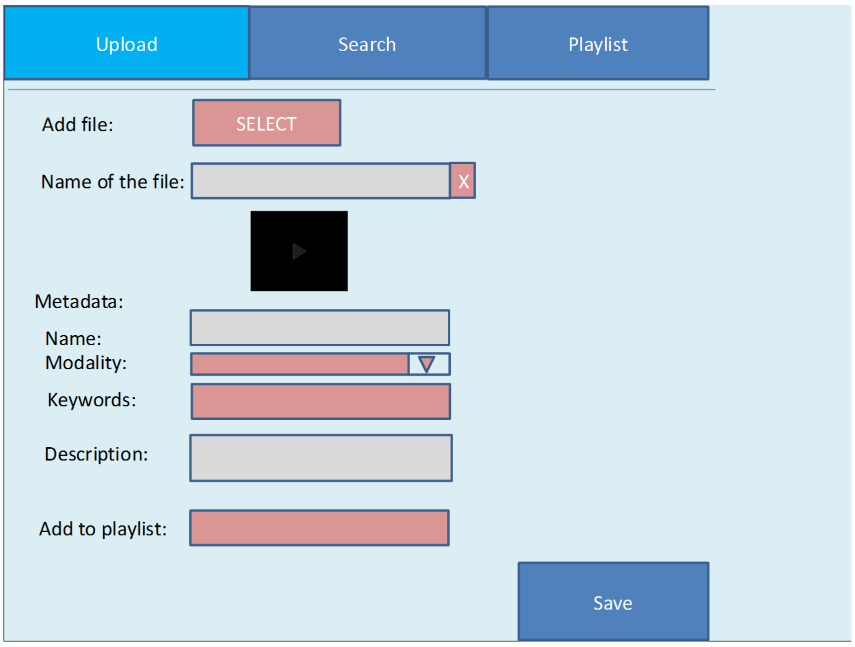


Figure 3.4: Ingest Screen from 2nd design iteration.

The next design iteration focused on refining the ideas from the original mock-ups. By expanding the generic 'Param - Value' pairs into the actual values that were deemed important, the new mock-ups look more like a finished interface. An important difference between this version, and the previous version, is that the 'Play' page is now dedicated to playlists, as well as the media player. It is also envisioned that both the search and ingest menu display either thumbnails, or short previews of the content that is found, to help the viewer determine if the correct media item has been added/found.

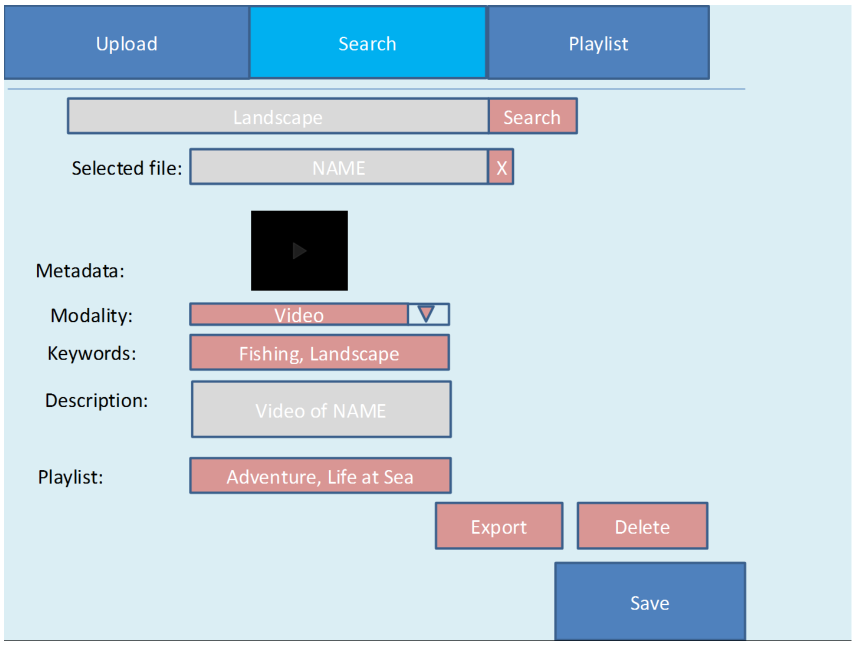


Figure 3.5: Search Screen from 2nd design iteration.

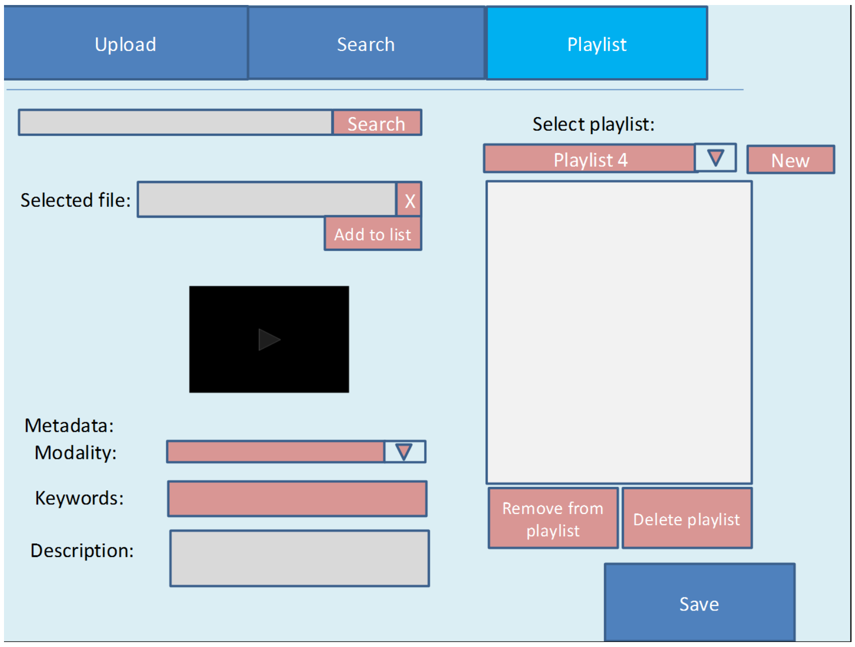


Figure 3.6: Playlist Screen from 2nd design iteration.

Figure 3.4, 3.5 and 3.6 display the main screens of the new mock-up, we see that the new mock-ups focus more on the interaction with the interface. These mock-ups are originally from a power point presentation (found in the included MASTERZIP.zip), where the interaction with buttons, and pop-ups are displayed. The main features highlighted in this presentation, is the ability to add an item to playlist through a dropdown menu, as well as adding keywords through a similar dropdown menu. The presentation also features the use of top level notification boxes to display information to the user. It should also be noted that when these mock-ups were made, the scaling of different elements were not taken into consideration. This is quite apparent in the 'Playlist' menu in figure 3.6, where the media player (icon in the center of the screen), is noticably smaller than it should be. When moving on into development of the Mini-MAM, this mock-up serves as a basis for how the elements should line up.

3.2 Development of the Mini-MAM

3.2.1 Initial Goals

When starting development of the Mini-MAM, the initial goal was to get the main selection menu working. This turned out to be the first of many small challenges that had to be overcome during the development. As the Tkinter library primary deals with placing single elements on the screen, having different elements taking up the same space was difficult. The solution was to use a python dictionary to store the three main pages, and use the buttons to set the selected frame as top-level. Technically, the act of using a dictionary to store three different objects, and call them as required is not exactly a new innovation, or even a difficult task. However, as this was the first task in creating the GUI, it took some time to understand the process of GUI creation in the context of using python code.

After properly setting up the ability to swap between the three main screens, the job of adding the ability to enter text began. Text is entered into Entry widgets, and by linking a Tkinter StringVariable to each entry widget, keeping track of the entered strings becomes easy. To make it easier to control, as well as reset the entries, and text labels between use, it was deemed necessary to collect the StringVariables in a dictionary. This makes it easy to clear the StringVariables by iterating over the dictionary containing them.

3.2.2 Development Process

When working with Tkinter, there are many 'best practices' to learn. One of those, is to use custom classes for larger sections of the GUI. Not only does this make it easier to structure your code, it also makes it more readable and easy to understand. Learning these tips and tricks took some time, and required some re-structuring of the code.

After having a decent framework for the GUI, work began on adding functionality. The first task, was to create the functions that allow a user to add a new media file. This meant creating a pop-up window, for the user to browse files on his/her computer, and returning the path of the selected file. From the file path, the files name and extension can be extracted, and by matching the extension with a list of common formats, the file's modality is determined. From there, the Mini-MAM has to be able to remember information stored between sessions. This could be solved in a number of ways, data could be recorded in a simple text file, a storage system using XML could be implemented, or the Mini-MAM could connect to a local database. Due to the amount of functionality provided by a SQLite database, this option was chosen, allowing the development to focus on the GUI instead of writing custom read, write and search functionality.

To connect with the database, the Mini-MAM uses the sqlite3 python library[19]. This library makes it easy to enter and retrieve information from the database, by using python variables. By using the library correctly, we also get input encapsulation, preventing any issues with SQL-injections. Once the ability to reliably store the information about each media item is available, the need to properly store the actual media item has to be addressed. Initially, the functionality to add files simply consisted of copying the media file to a subfolder of the Mini-MAM. As one of the goals of the Mini-MAM is to transcode video and other media items, work began on investigating possibilities on how to do this using python. Initially, a few python media library were investigated, but due to their lack of support for certain formats, the choice was made to attempt to integrate ffmpeg.

Integrating ffmpeg turned out to be a quite challenging task. As the ffmpeg software runs from the command window, running a command is not that difficult, but the Mini-MAM should also be able to read and interpret the output from the ffmpeg instance to give the Mini-MAM feedback. This turned out to be a bit tricky, as it required finding, and including some rather obscure python libraries that were initially designed for Linux. Once the winpexpect library was imported, it was also discovered that the winpexpect library does not work with Unicode characters, and only use ASCII, causing the Mini-MAM to crash whenever a file uses non-ASCII characters (such as Æ, Ø, and Å). It took some time to identify the issue, but once it was discovered that the string formatting was the issue, it was solved by temporarily renaming the imported file.

Once the ability to properly ingest a file was supported, work began on the search page. The search page follows the same layout as the ingest page, using entry widgets and StringVariables. To complete the search page functionality, the Mini-MAM has to communicate with the database. This is accomplished by using the sqlite3 library, and the search functionality present in SQLite. As the searches in the database are returned as an touple of strings contained in an array, it was desirable to arrange the returned data in a more structured manner. This lead to the decision of adding custom classes for data sets. The idea was already implemented in the project report, so the same class was used for the Mini-MAM.

To complete the search page, the results have to be displayed to the user. For this task, the Tkinter dropdown menu was chosen. This widget creates a dropdown menu displaying the names of the media items found during the search. When selecting a menu item, the search page updates all the internal StringVariables, using the data contained in the metadata object with the name displayed on the menu. Internally, each item in the dropdown menu is linked to a metadata object, and the text displayed is the name stored in the metadata object. With the ability to find, and edit information about the media items, the next step is to add the ability to store the changes made. Using SQLite commands to update information in the database, the save function is relatively straightforward. All the data in the current metadata object is updated in the database, using the fileid to update the correct entry in the database.

Once the ingest and search menus were functional, work began on the ability to add and edit playlists, as well as play media items. This turned out to be a quite time consuming task, as the Tkinter Treeview widget can be a bit cumbersome. The Treeview widget relies on a tree-like structure of nodes, where nodes are identified by a unique "iid" string. As the the 'iid' is the only way to differentiate between different nodes, some naming convention was required to link a tree node to either a playlist, or a media item. After some attempts at a naming convention, the eventual solution became using the unique ID of each media item, as well as each playlist, with a leading 'i' or 'p' to indicate if the item is a playlist or media item. However, this would cause errors when an item was added to more than one playlist at a time. To also prevent issues from a single media item being in the same playlist more than once, nodes indicating media items were given iids that include the item id, playlist id, as well as the media items position in the playlist. In terms of the actual playlists, the Mini-MAM implements a Playlist class. This class contains all the basic information about the playlist, as well as two dictionaries that contain the subplaylists and media items in the playlist.

To manage the playlists between sessions, new functions to create, update, and read playlists from the database also have to be created. Most of these functions are quite basic, but require some use of recursion to fetch all subplaylists from a playlist. During use, the Mini-MAM store the current top level playlists in a dictionary in the root Tkinter element. This leads to an interesting dilemma, where a playlist exists in the internal dictionary, the database, and the Treeview widget, and all three instances have to be kept up to date whenever a playlist changes. To keep this from becoming an issue, the updating of the Treeview widget is done by simply rebuilding the node structure from the internal dictionary whenever a change is made. This may not be the ideal solution, but was an easy way to make the playlist meny functional.

The next issue, was to create a media player inside the Mini-MAM. From the get-go, the goal was to integrate the VLC media player [17], although it was not a given that this was feasible. The main issue with integrating the media player, was to control where the video was to be displayed. The VLC python bindings [13], provide all the tools required to control a VLC player instance using python, so having the Mini-MAM launch an instance of VLC would not be an issue. The goal of actually integrating the player was a bit more challenging, as the examples provided by VLC do not use the Tkinter libraries. The solution to the issue, was to find the "console window handle" (HWND)[21] of the widget where you want the video displayed. This took quite a lot of trial and error, as well as looking through countless examples and online forums. Once the video was integrated, focus began on creating the buttons and menus required to control the video, fortunately this process was made easy by the examples provided by the VLC python bindings. Using the examples, the media player quickly took shape, and the functionality to play, pause, stop was quickyl added. Adding the progress bar, that displays your current location in the media item was a bit trickier, and required the use of the Tkinter timing functions.

3.3 Subjective Assessment

To determine the usability of a program, the program has to be tested. The goal of a subjective assessment, is to get unbiased feedback from a varied population of test subjects. The subjects are asked to rate the quality of the system-under-test, after having experienced different aspects of the system. The group of subjects should ideally be large amount of individuals, with different backgrounds and computer knowledge, and who are considered non-experts in the study of media asset management. Generally, a subject assessment is a good indicator of the usability of a system, the test itself can be difficult to properly set up. It is therefore recommended to follow a set of guidelines and rules on how to perform a proper subjective assessment [24]. Despite these guidelines primarily involving testing the quality of video content, the general ideas can be used for the usability test of the two management systems.

3.3.1 Equipment

The tests were performed on laptop, with both systems pre-installed. The proper tools and libraries required for the Mini-MAM prototype were installed, and a the software was set up with a sample video already added to the system. This was done to minimize the time spent during the test, as the transcoding of video material can be time-consuming. The laptop was also running a virtual Ubuntu server, using VirtualBox. This Ubuntu server hosted a copy of the XMS system, allowing the user to access it through the browser on the main computer. The test material used during the test was provided by the KOMOPP project [11].

3.3.2 Session

The evaluations were performed at the Sense-IT lab (A362). The participants were seated in front of the laptop, with a supervisor sitting next to them. During the tests, the room was empty, apart from the participant and the supervisor. To prevent bias, users were randomly selected to start with either the Mini-MAM, or the XMS system.

Each session consisted of the participant completing two series of tasks for each of the two systems. The first series of tasks is designated as a "Free" task, where they are asked to perform a set of tasks without any guidance on how to complete the tasks. The Free tasks are time-limited, and the user is asked to move on, once 5 minutes have passed. The second series of tasks, are a repeat of the initial tasks, this time with step by step guide on how to complete the tasks.

3.3.3 Participants

The participants of the subjective assessment consisted of 14 individuals. Having 6 females, and 8 males, we get a ratio of 57.14% males, and 42.86% females. The participants were all between 23 and 27 year old, and had no prior experience with either of the two systems. Most subjects were students at NTNU, and may have above average familiarity with computers. However, none of the participants were familiar with the subject of Media Asset Management, and the participants should therefore be considered non-experts. As the goal of the assessment was to test the usability of the two systems, the participants were given a step-by-step guide on how to complete the tasks. This was primarily done to limit the impact of each participants general familiarity with computers, and place the focus of the test on the systems, instead of the participants.

During the session, the time each participant spent on the 'step by step' part of the assessment was timed using a smartphone.

Before the actual assessments were performed, a trial assessment was performed. During this trial, it was noted that the trial subject simply followed the step-by-step guide blindly, without reflecting on the usability of the two systems. It was therefore decided to include a time-limited 'free' task, where the users were asked to perform basic operations without any guidance. Once the time ran out, the users moved on to the step-by-step guide. The assessments were completed within 25-40 minutes, and upon completion the participants were rewarded by selecting a chocolate bar from a small selection.

3.3.4 Evaluation

The evaluation of the subjective assessments were handled using a google form, this allowed for quickly and effectively collecting the data in a spreadsheet for analysis. The form consists of 4 pages, the first page collects basic information about the participant, as well as some information about their general familiarity with computers and media. The 2nd and 3rd page contain the answer forms, where the participant rate the user friendliness, menu layout, and time consumption of each task. The 4th page asks the participants what system they preferred to use, and what system they believe they spent the most time with.

Score Chart				
1	2	3	4	5
Bad	Poor	Fair	Good	Excellent

Table 3.1: Evaluation scale for the Subjective Assesment.

Participants rate each task on a scale from 1-5, as shown in table 3.1

3.3.5 Systems

The evaluation attempts to assess the usability and user friendliness of the Expology XMS exhibit management system, and the Mini-MAM Prototype. The XMS system is developed by Expology, a Norwegian company based in Oslo, and is primarily designed to manage the content and display screens of an exhibition center. Despite the differences between the main focus of the two systems, it is interesting to see the differences between a commercial system, and the Mini-MAM Prototype.

The tasks the participants completed were relatively similar on the two systems. The primary task was to add a simple media file, and provide some contextual metadata. The second task involve linking a set of media items together, in a playlist or presentation.

3.4 Statistical Analysis of Subjective Assessment

Presenting the results from the user survey require a certain amount of statistical analysis. As the amount of samples we have are far below the requirements for a 'true' mean score, we can estimate the true mean value using a 95% confidence interval [24]. We can then say with a probablity of 95%, that the absolute value of experimental mean error is smaller than the width of the 95% confidence interval.

First, we need to calculate the mean score $\bar{\mu}_j$, for each category.

$$\bar{\mu}_j = \frac{1}{N}\sum_{i=1}^{N} u_{ij} \tag{3.4.1}$$

Where u_{ijk} is the mean score of action i, to category j.

$$[\bar{\mu}_j - \sigma_j, \bar{\mu}_j + \sigma_j] \tag{3.4.2}$$

u_{ijk} is then used to calculate the 95% confidence interval, where:

$$\sigma_j = 1.96\frac{S_j}{\sqrt{N}} \tag{3.4.3}$$

And the standard deviation is given by:

$$S_j = \sqrt{\sum_{i=1}^{N} \frac{(\bar{\mu}_j - \mu_{ij})^2}{(N-1)}} \tag{3.4.4}$$

Chapter 4

Implementation

This chapter aims to explain the tools and libraries required for the implementation of the Mini-MAM prototype, as well as showcase the final version of the system. The chapter will explain each of the tools and main libraries used by the Mini-MAM prototype, explaining how they function, and why they are required. The final sections of the chapter display the finalized main screens of the Mini-MAM, and explain the important functionality of the system.

4.1 Motivation and Design philosophy

When designing the Mini-MAM, the main focus has been to create a simple and user friendly system, that performs the basic operations required to keep track of a media collection. To make the system as user friendly as possible, a Graphical User Interface has been developed, using the python [14] programming language. However, it should be noted that the current version of the system is considered to be a prototype, demonstrating that it is possible to implement the operations required for a Mini-MAM in a simple manner.

When designing the interface, it was important to focus on the primary tasks of the Mini-MAM, adding content, finding content, and displaying content. Using this focus, the three main screens of the Mini-MAM are the Ingest Page, the Search Page, as well as the Play Page. To help the users visualise their content, it was deemed necessary to add a Playlist Page in addition to the three other primary pages

4.2 Tools

4.2.1 The Tkinter library

To build a GUI, Python have many different libraries and packages. Tkinter[15] is the de-facto standad library, and is therefore a well documented library with many examples and tutorials available on the internet. The availability of tutorials and examples was an important factor in determining what library to use, and due to their abundance Tkinter was chosen for this project.

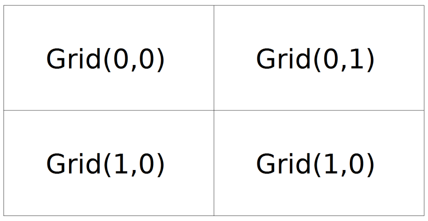


Figure 4.1: Tkinter grid example.

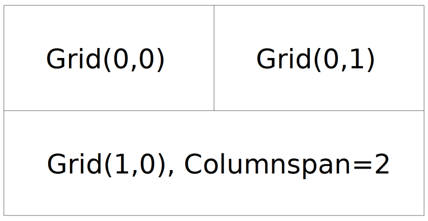


Figure 4.2: Tkinter grid example using columnspan.

Tkinter relies on so called 'widgets' to create a new graphical interface. Widgets refer to all types of graphical objects: buttons, empty frames, labels with text or images and a wide array of other specialized objects. To display the widgets on the screen, and to determine their locations in relation to each other, Tkinter uses a geometry manager. This manager can be called in several ways, but for this project, we use the grid manager. The grid manager gives each widget a position on grid, defined by its row, and column value as shown in figure 4.1. The manager also has options to allow widgets to take up more than one row or column, as well defining how the size of each column or row should scale, as seen in figure 4.2

To create a Tkinter window, you first create a Tkinter root widget. This root widget serves as a top level window, that you place other widgets in. In this way, the root widget shares a lot with the Frame widget. A Frame widget acts as a container for other widgets, and opens a wide array of options, especially in terms of placement. Another basic widget is the Label widget, this widget is the primary method for displaying information to the user. The primary use the Label is to display text, or images, but when left empty, the Label can be used as a simple colour display tool.

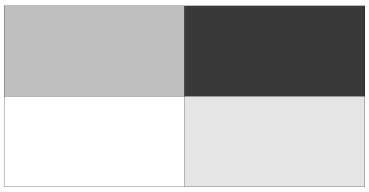


Figure 4.3: Tkinter grid example.

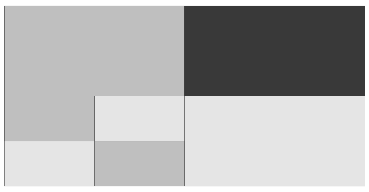


Figure 4.4: Tkinter grid example using columnspan.

The examples in figure 4.3 show how the relationship between Frames and other widgets can be used to create more advanced graphics. We see that the main element is initially divided into 4 Labels, each represented by a different color. The figure 4.4 shows what happens, when one of the Labels is replaced with a Frame that contain 4 other Labels.

4.2.2 SQLite

SQLite is a database management system, managed by a set of c based libraries [20]. As the name indicates, SQLite implements most of the SQL standard. However, in contrary to most other database management systems, SQL is integrated into the client application. This means that there is no SQLite process running in the background, as all calls are handled by the client application through different programming libraries. In our case, this is the sqlite3 python library[19].

For the Mini-MAM protoype, SQLite provides persistent storage between sessions, and provides a great framework to make sure the underlying database file is kept up to date and error free. The library operates by having a custom class, called a cursor connect to the database. The cursor can then execute SQL commands on the database, and store any potential results in internal variables. To access these variables, the call fetchone() or fetchall() returns either one of the entries returned by the SQL query, or all. These results can then be iterated over to select each column in the entry.

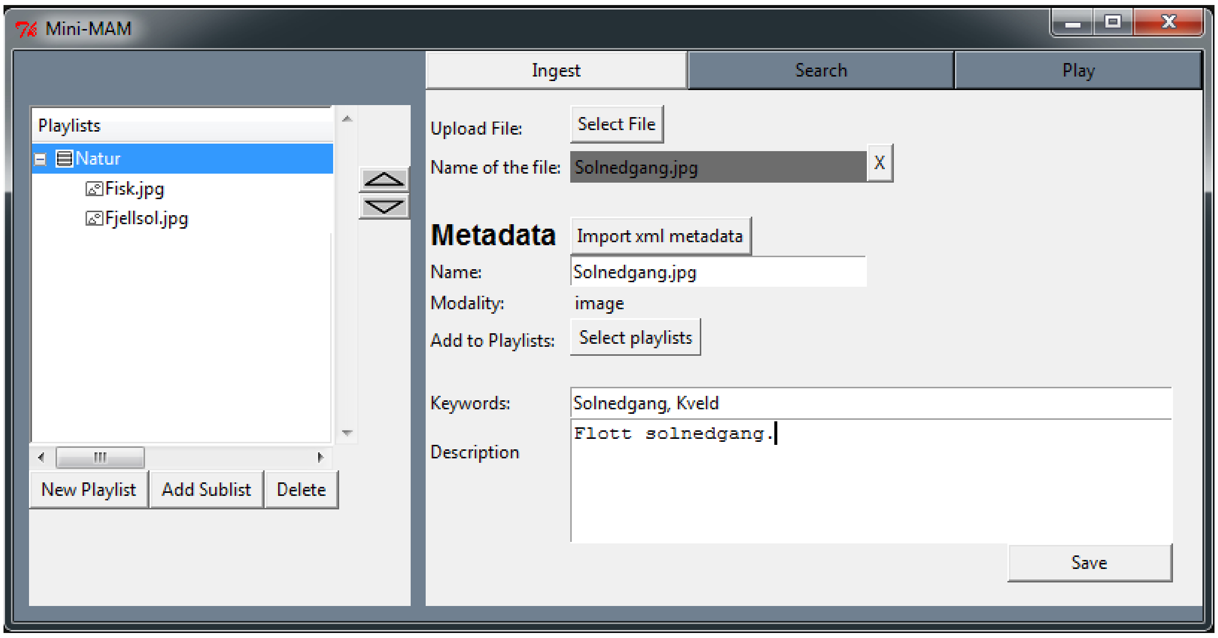
4.2.3 FFmpeg

The FFmpeg media framework [18] is a set of free software tools that allow a user to "decode, encode, transcode, mux, demux, stream, filter, and play pretty much anything that humans and machines have created.". Using a simple command line interface, the ffmpeg software makes it easy to transcode any new materials added to the Mini-MAM.

4.3 The Prototype

4.3.1 The Ingest Menu

The ingest menu is designed to allow the user to easily select, upload and add metadata to a new file. Upon pressing the "Select File" Button, the user can browse their computer for a media file. Once a file is selected, the Mini-MAM will store the file's path, and extract the file type, and file name from the full path. This information is stored, until the user presses the save button. Before saving the file, the user also has the option to add metadata.The user can either import EBUCore compliant XML files matching the method described in my previous project report, or manually enter a short description, as well as add keywords. If there are any playlists currently in the system, the user can mark them, so that the media file is added to the marked playlists once it has been succesfully added to the Mini-MAM.



Upload File:
Name of the file: Solnedgang.jpg
Name: Solnedgang.jpg
Modality: image
Add to Playlists:
Keywords: Solnedgang, Kveld
Description: Flott solnedgang.

Figure 4.5: Screencap of current Ingest Page.

Once the user clicks the save button, a media entry is created in the database, and the file's name, modality and metadata is added to the database entry. This call returns with the entry ID of the new entry, giving each file a unique identifier. This ID is used as the file's name once it is copied to the internal storage of the Mini-MAM. Currently, this internal storage is simply a named folder, and a more efficient method should be considered. If the selected file is video files, the Mini-MAM will use ffmpeg to transcode the video to the current default format. If the file is in any other format, the system will copy the original file to the internal storage folder.

Internally, all the functionality is supported by a structure of Tkinter StringVariables. These variables automatically update the text displayed to the user, using a set of automated callback functions.

4.3.2 Transcoding Standard

The ingest function will attempt to transcode any video material that is added to the Mini-MAM. Using ffmpeg, the imported video file is transcoded according to the parameters outlined below:

Resolution: 1280x720

Framerate: 50 p

Codec: h.264

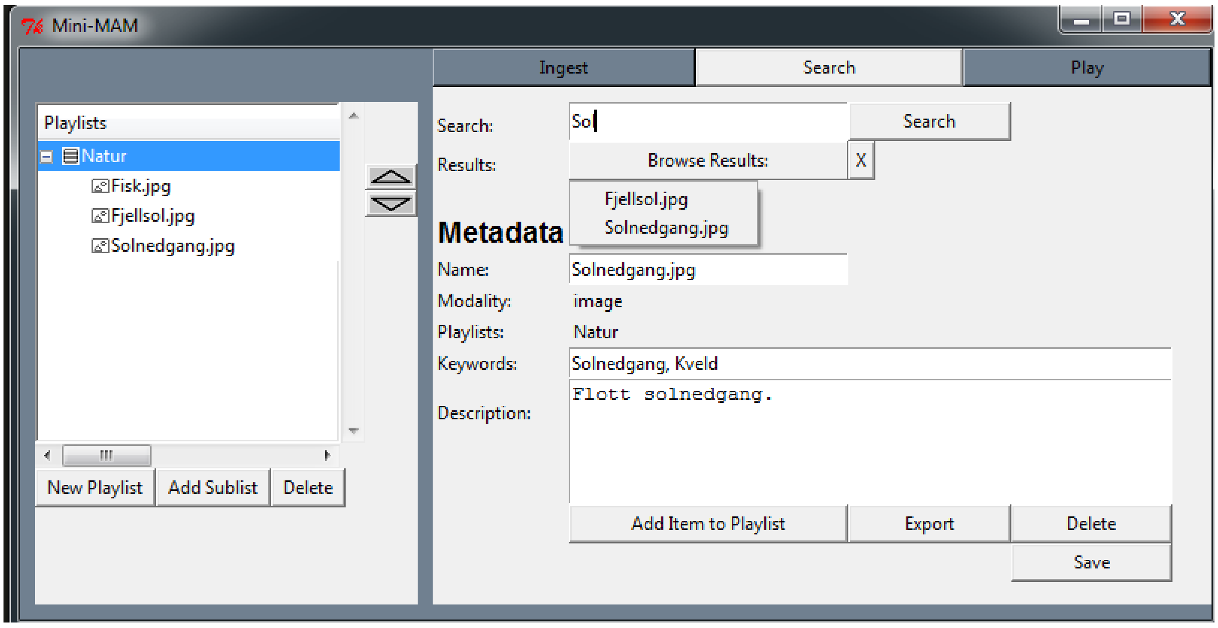
Container: mp4

Audio: 320kbps AAC Stereo

In terms of ingest content, the currently supports .mp4, .avi and .mpeg video. For images, the system supports .png, .jpg, .jpeg and .bmp. For audio, the system supports .wav and .mp3. These formats are the ones recognized by the system as either Video, Audio or Images, if other formats are desired, they can be added to the Importfile.py getFormat() function.

4.3.3 The Search Menu

The Search Menu builds on the same technical principles as the Ingest Menu, allowing users to search for media items that have already been added to the database. The search string is sent to the database, and all matching entries are built into a metadata-object, that encapsulates all the data into a easily handled manner. These objects are put into a list, that can be browsed by clicking the 'Browse Results' button. Upon selecting an option from the list, the information in the metadata object is loaded to the search page, allowing the user to edit the metadata of the selected media item, as well as add the item to any playlists. Once the user presses the save button, all changes are saved to the database.



Search: So

Results:

Name: Solnedgang.jpg

Modality: image

Playlists: Natur

Keywords: Solnedgang, Kveld

Description: Flott solnedgang.

Figure 4.6: Screencap of current Search Page.

In addition to the search and browse functions, the Search Menu allows a user to export the material from the Mini-MAM to a location on their computer. Currently, the export functionality is rather simple, but additional options may be added later, such as the ability to transcode material, and export metadata. The search menu also allows the user to add the currently selected media item to be added to any playlist highlighted in the Playlist Menu.

The basics behind the search menu are very similar to the Ingest Menu, a set of StringVariables control the text entries that display information to the user.

4.3.4 The Play Menu

The Play Menu allows the user to display the selected media item, using the VLC media player. To fully integrate the VLC media player, the Mini-MAM uses the VLC Python Bindings [13]. The VLC Python Bindings is a complete cover of the liblv API, generated from the original C sources. Using the VLC Bindings, the player can play both single items, as well as sets of items in a playlist. Technically, the VLC python bindings are used to spawn an instance of the VLC media player that runs in the background. By linking buttons in the interface to these commands, we are able to control the player, and extract information about the current media item in the player. For instance, the player progress bar, relies on continuously polling the VLC instance, and updating the Tkinter slider widget. The player currently supports playing both single media items, as well as full playlists. In the case of playlists, subplaylists will be ignored, and only content from the primary list is played.

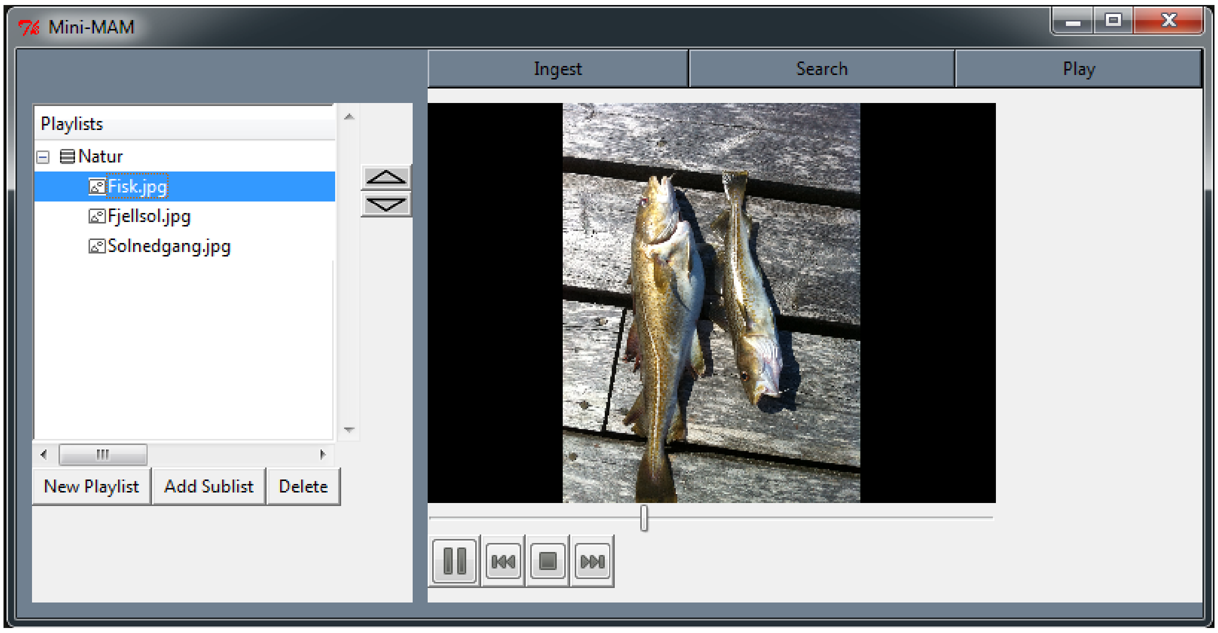


Figure 4.7: Screencap of current Play Page. Image Copyright Trine Rein

4.3.5 The Playlist Menu

During the development of the Mini-MAM, it became apparent that having constant access to the playlist menu was required. Therefore, the Playlist Menu was moved out of the swappable frames list, and given its own location on screen. The Playlist Menu consists of a Tkinter Treeview widget, that allows users to browse a tree-like structure of nodes. The nodes in Treeview widgets are built from a python dictionary containing the current top level playlists. These playlists in turn contain other playlists, as well as metadata objects for each media file in the playlist. This leads to an interesting issue, as all playlists essentially exist in three different locations, in the database, as treeview nodes, and in the internal list of playlists. This leads to the issue of having to update all three instances every time a change is made in any playlist. To help combat synch issues, the Treeview nodes are updated from the main list of playlists when a change is made, this is not an ideal solution, but saved a lot of time when writing the functions. This does however have the downside of having all currently expanded playlists being closed when a change is made. To help structure the media items in a playlist, icons have been made to distinguish between different media types. Currently, icons exist for audio, video, image type media items. To help users edit playlists, the ability to move an item up, or down in the currently selected playlist is available, by using the side buttons.

The Playlist Menu allows the user to easily create and manage playlists. Buttons associated with the playlist page allows the user to create both root, and sub playlists, while the adding of content happens from either the Search, or Ingest menu. The delete button is also context dependant, and will delete the currently selected items, or playlist, depending on the users selection. If the user wants to delete a playlist, the system will properly delete the list, as well as any items or sub-playlists in the playlist. If the user deletes an item, the item is simply removed from the playlist, and retained in the database.

4.3.6 SQLHandler

The SQLHandler.py is a set of functions that allows the GUI to communicate with the database, providing functions to insert new media items,playlists or subplaylists into the database. The SQLHandler.py file also contain the definition for the metadata and playlist classes.

Metadata and Playlist classes

The Metadata and Playlist classes are designed as a method to incapsulate information, and make it easier to transfer data between functions. Whenever data is requested from the database, the database functions usually return information in the form of one of these classes. The Metadata class, contains the media items unique ID, the type, stored as the file extention, as well as the name, path, description and keywords associated with the media item. The Playlist class contains a few identifiers for the playlist, the unique ID of the list, its name, and in the case of sub-playlists, the ID of the parent list. In addition to these identifiers, the Playlist class contains two dictionaries, one containing Playlists, and the other containing Metadata entries. The playlists dictionary contains the playlists current sub-playlists, and uses the sub-playlists list id as the key for each entry. The items dictionary contain Metadata items, and uses the files track number as the key.

SQLite Functions

These functions primarily deal with the creation and updating of entries in the SQLite database. Using the sqlite3 [16] package, values in python variables can be written to database entries. The most interesting functions are the init function of the SQLDatabase class, the Search function, and the getPlaylists function.

The SQLDatabase init function checks to see if the a database exists in the '/data' folder, and if it cannot find a database, it creates a new one. Essentially, this means that the database will be re-created if it is deleted, but will remain untouched if it exists. The database consists of two main tables, the metadata, and tPlaylist tables. The metadata table contain all the information regarding each media item in the database, using the unique ID as the primary key. The playlist table however, simply hold the playlist ID, the playlists name, and a flag that says whether the playlist is a subplaylist or not. To manage items in playlists, we have two other tables, the tPlaylistItemMap, and the tPlaylistPlaylistMap. These Map tables provide the link between a playlist, and the items and sublists it contains. A map entry contain a unique Mapping ID, as well as the ID of the playlist and the ID item/subplaylist. For the tPlaylistItemMap also holds the track number of each Item, allowing an item to be mapped to the same playlist several times. The same principle is used in the tPlaylistPlaylistMap, each entry has a unique ID, as well as the ID of the root playlist, and the subplaylist.

The Search function is currently a rather large and contrived function, that will check the metadata table, and look through each column for the search term. If the search term is found, each entry is returned in its entirety, and a new Metadata object is created to hold the information and added to the list of found items. If the search term is found in both keywords, description, or file name, duplicate Metadata objects will be created. To prevent these duplicates from being sent back to the Mini-MAM, we do a culling of the list of Metadata objects before the list is returned.

As the Mini-MAM is reliant on having an internal list of all current playlists, we need a way to fetch the playlists from the database when the Mini-MAM is launched. The getPlaylists function performs this role, and does so by initially fetching all playlists that are not marked as sublists. It then uses the recursive function getSubPlaylists to find all subplaylists under the top playlists. The get SubPlaylist function calls itself, to find any playlists contained in the subplaylists, recursively finding all playlists. Both getPlaylists and getSubPlaylists call the getPlaylistItems function to find items within the playlists they find.

Chapter 5

Results

In this chapter, we display the most important and interesting results from the subjective assessment of the Mini-MAM and XMS systems. Answers from the questionnaire rating the two systems are reported in tables, sorted for each system. Finally we present the results from the general questions regarding computer usage among our participants. The full data from the subjective assessment, can be found in Appendix A.1

5.1 Subjective Assesment

During the subjective assessment of the two systems, the participants are asked to rate the tasks that they perform on a scale from 1-5, where 1 is bad, and 5 is excellent, as seen in table 3.1. As there were only fourteen participants, some statistical analysis was required before presenting the results. This is why the results are generally reported in the form of a 95% confidence interval.

Mini-Mam

	User Friendliness	Time Consumption	Menu Layout
$\bar{\mu}_{jkr}$	3.9592	4.0510	3.7653
S_{jkr}	0.4998	0.3317	0.3422
σ_{jrk}	0.3702	0.2458	0.2535
95% Conf	3.5890-4.3294	3.8052-4.2968	3.5118-4.0188

Table 5.1: The average score of the Mini-Prototype, across all tasks.

Initially, table 5.1 show that the average scores of the Mini-MAM system are quite good. The average score for all categories are all well above the average value of 2.5, and is on close to 4. From table 3.1 we see that a score of 4 represents 'Good'.

XMS

	User Friendliness	Time Consumption	Menu Layout
$\bar{\mu}_{jkr}$	2.4184	2.3571	2.9592
S_{jkr}	0.2622	0.3956	0.2137
σ_{jrk}	0.1943	0.2930	0.1583
95% Conf	2.2241-2.6126	2.0641-2.6502	2.8009-3.1175

Table 5.2: The average score of the XMS, across all tasks.

In table 5.2, we see that the average score of the XMS system is somewhere between 2 and 3, representing 'Poor' and 'Fair' scores. The XMS system has a noticably higher score in the Menu Layout category, compared to User Friendliness and Time Consumption.

Notable Results - Mini-MAM

Category	User Friendliness	Time Consumption	Menu Layout
General Impression	3.6060-4.2511	2.7905-3.9238	3.1632-4.4082
Search	2.4189-3.5811	2.8220-4.035	2.6892-3.8822
Playlists	4.1596-4.6976	3.8480-4.5806	3.6373-4.5056
Player	4.1593-4.8407	4.1593-4.8407	3.6638-4.7648
Upload	3.1632-4.4082	3.5490-4.5938	3.1543-4.2743

Table 5.3: Notable results the rating of tasks performed with the Mini-MAM, represented using 95% confidence interval.

In table 5.3, we see the 95% confidence intervals of the participants' rating of different tasks performed with the Mini-MAM. We see that the Search task is rated quite poorly compared to both the general impression, as well as the average score in table 5.1. It's interesting to note that the average score of the Mini-MAM is quite similar to the General Impression, for both user friendliness as well as layout, while the time consumption differs slightly. We also see that the playlist and play related tasks score quite well compared to the average, as well as general impression.

Notable Results - XMS

Category	User Friendliness	Time Consumption	Menu Layout
General Impression	1.6373-2.5056	1.3338-2.0947	2.0060-3.2798
Search	2.0834-3.3452	2.2352-3.3362	2.5101-3.6328
Template	1.6456-2.6401	1.5406-2.4594	2.1178-3.3108
Presentations	2.2819-3.1467	2.3182-3.2533	2.6267-3.8019

Table 5.4: Notable results the rating of tasks performed with the XMS System, represented using 95% confidence interval.

Table 5.4 shows that the trend where the XMS scores higher in the Menu Layout category continues.

5.2 Time Spent

Average Time Spent

XMS	Mini-MAM
05m23s	09m45s

Table 5.5: Average time spent per system for the step-by-step task.

5.3 Participant Survey

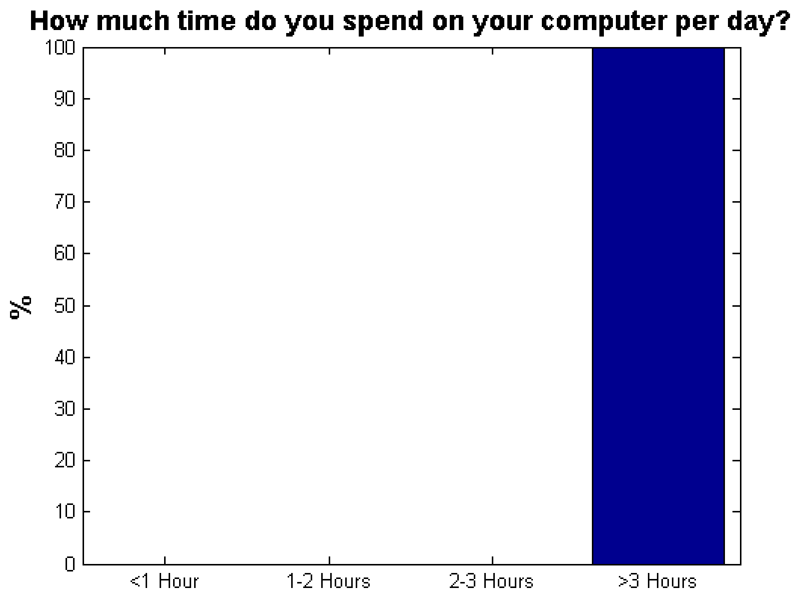


Figure 5.1: Result from user questionnaire.

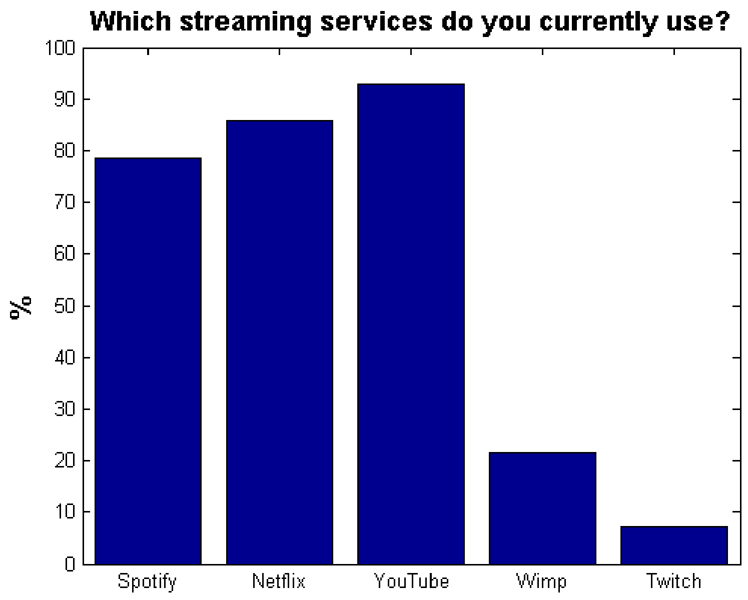


Figure 5.2: Result from user questionnaire.

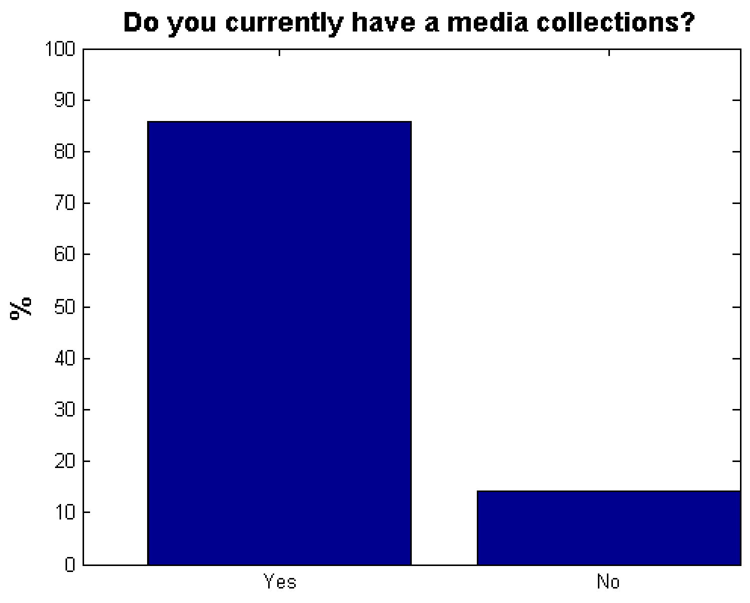


Figure 5.3: Result from user questionnaire.

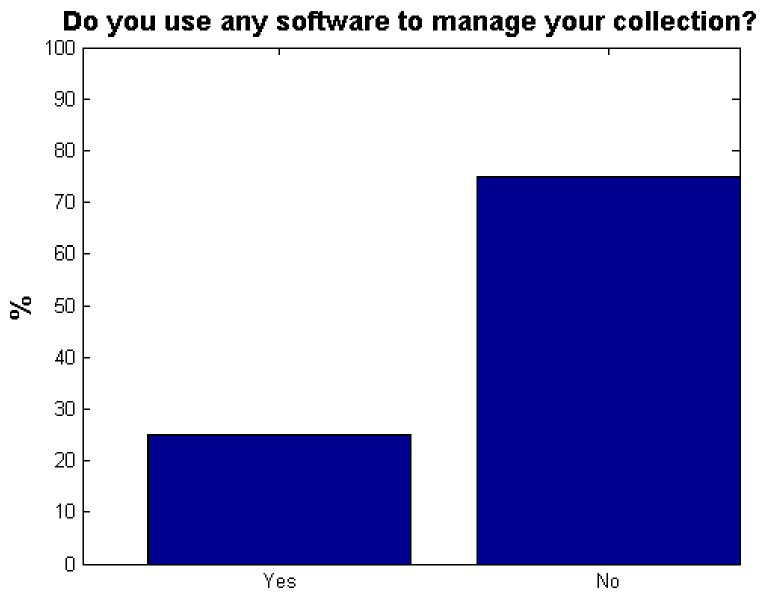


Figure 5.4: Result from user questionnaire.

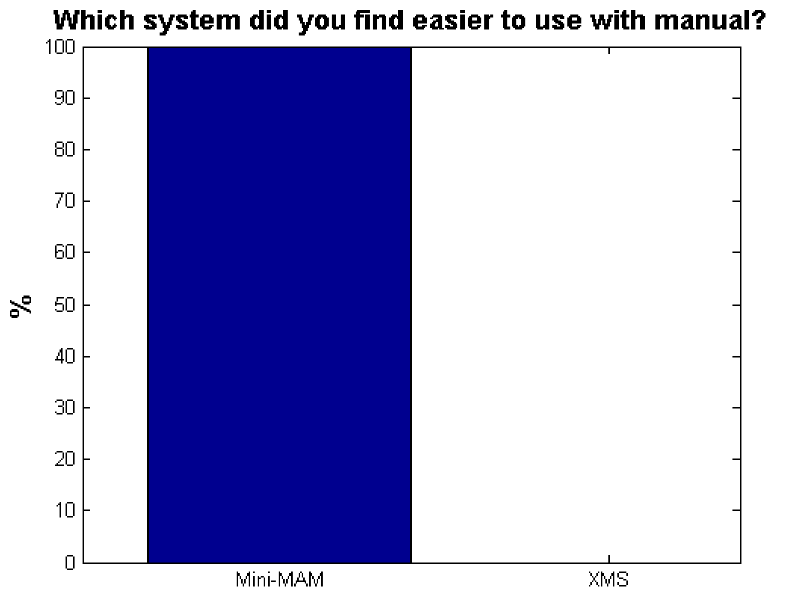


Figure 5.5: Result from user questionnaire.

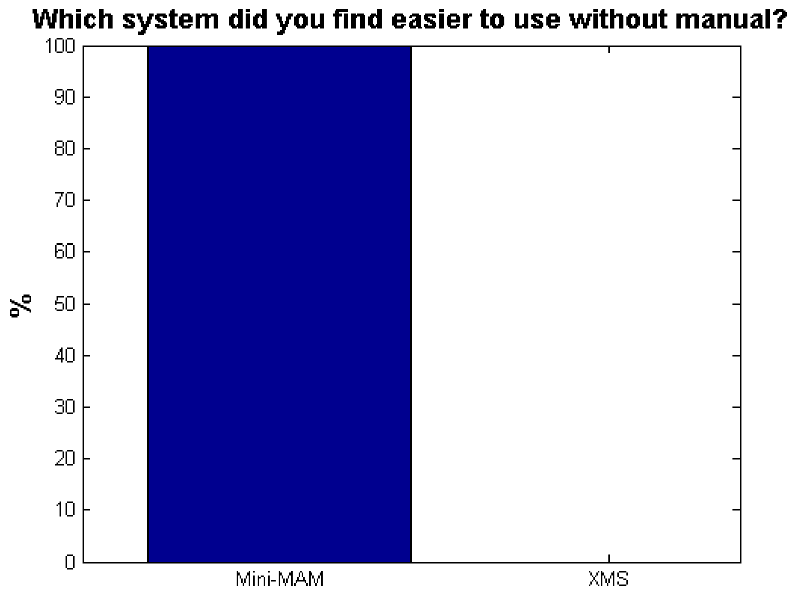


Figure 5.6: Result from user questionnaire.

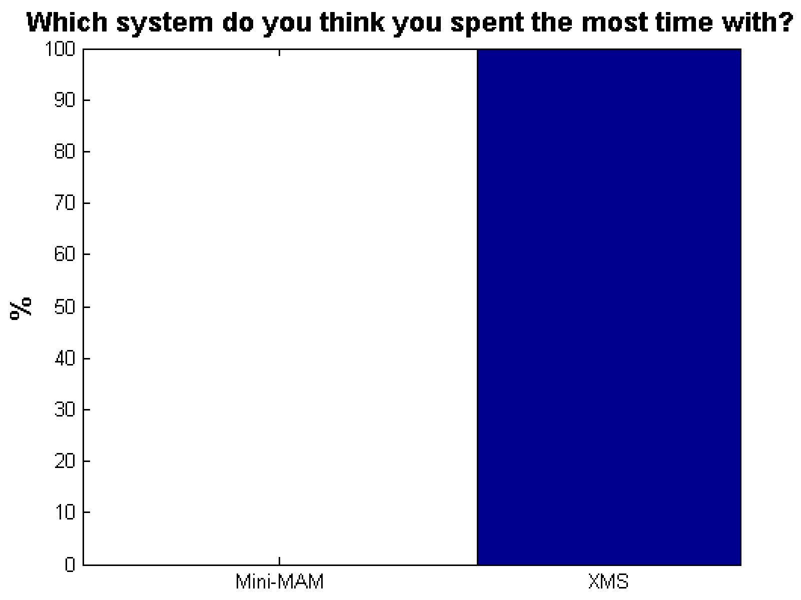


Figure 5.7: Result from user questionnaire.

Chapter 6

Discussion

In this chapter, the results of the subjective assessment, and the development of the current Mini-MAM prototype will be discussed. During the discussion, we want to highlight key results and features, and consider their potential implications.

6.1 Developing Mini-MAM Prototype

In this report, the focus has primarily been on the design and development of the Mini-MAM, as well as the usability test of the XMS system and Mini-MAM. The development process has been an interesting experience, with many different types challenges and rewards.

From design, through development, the Mini-MAM has been envisioned as a simple and user friendly system, that performs the basic tasks required to manage a set of media content. During the development, care has been taken to make sure that the primary processes of ingesting content, and editing existing material remain as easy as possible. The current system appears to allow users to perform the tasks outlined in previous chapters.

Despite appearing to be able to fulfil its role, the Mini-MAM does have some flaws. The current ingest menu can be considered a bit inflexible, as the system currently only allows for transcoding of video content. The ability to transcode, and store images and audio in a high quality format may not seem very important, as the quality of the media resource will not improve through transcoding. The importance of having a high quality base sample does however become necessary if more options are added to the export functionality of the Mini-MAM. By guaranteeing a default format, we can prevent issues when exporting the material, as well as allow for easier organization, and potentially embedding of metadata.

In general there are a lot of features that have been thought of, but not implemented due to time constraints. These are thoroughly explored in the Future Works chapter. It should however be noted, that we are developing a *Mini* Media Asset System, with only the most basic functionality. Too many features can increase the complexity of the system, and move focus away from the goal of having a very user friendly system. Adding more features to the system, would also increase the potential cost of the system, once again diverging from our primary focus of a cheap and simple MAM.

Currently, the Mini-MAM prototype suffer from a few issues, the primary one is the unintuitive method for browsing search results. Having to click on a slightly hidden "Browse Results:" button is far away from what is excepted in a modern GUI. However, a quick fix is not really possible, as there doesn't seem to be a way to open up a dropdown menu without having to press the actual drop down menu button. Ideally, the entire method of presenting search results should be re-done, and use a the Treeview widget, to display more information that just the name of the media item.

Despite the issues and shortcomings of the Mini-MAM, the current prototype has required a lot of work. Creating a GUI from the bottom up has been a slow process, with a lot of bumps in the road. The learning process has been quite intense, as the concepts behind the GUI libraries are somewhat different compared to other basic programming concepts. It has also been quite challenging to integrate other software tools, such as ffmpeg and vlc into the prototype. Although the actual integration may not be technically challenging, it is often hard to find proper documentation when you run into problems, as these are open source tools without a global technical support.

6.2 Usability Test

To screen the participants of the usability test, they were initially asked a few questions to examine their habits with computers and multimedia. From figure 5.1, we see that our group of subjects are very familiar with computers, all subjects reportedly spend more than 3 hours on their computer daily. From the initial survey, we also get some interesting results in terms of the media habits of our participants. From figure 5.2, we see that streaming services are very popular with our users. Music streaming services in particular are very popular, 100% of users report that they either use a music streaming service (spotify or wimp). During development of the Mini-MAM, it was natural to look to commonly used services for inspiration. The developed Playlist Menu is inspired by the layout and functionality of the Spotify playlists. As the subjects are familiar with Spotify, and similar systems, it possible that the elements inspired by such systems are better received than they might have been otherwise.

Looking at the results from the usability test in table 5.1, we see that the Mini-MAM performed fairly well. When compared to the results of the XMS system in table 5.2, the results are even more impressive. It should also be noted, that the results of the usability test should not be directly compared, due to the different properties of the two systems.

Despite the overall good scores from table 5.3, we also see that the search functionality of the Mini-MAM is noticably lower than the average score. This presumably because of how the browsing of the search results happen. In addition to the numerical data from the usability test, test participants did provide some feedback, and this was one of the issues mentioned. We also see that the playlist menu scores higher than the average score, indicating that the current layout of the playlist menu is a success.

In table 5.2, we see that the XMS system scores significantly lower than the Mini-MAM. Figures 5.5, 5.6 and 5.7 also show that the participants of the usability test preferred the Mini-MAM system. This can be contributed to several reasons, the most obvious being that the Mini-MAM is a more user friendly system. This should not come as a surprise, as the Mini-MAM is designed with user friendliness and usability in mind. Other factors can be attributed to the manner of the test, as well as some bias from the participants.

In terms of efficiency, we see from table 5.5, that the average time to complete the step-by-step tasks is far lower for the Mini-MAM. Despite the tasks being slightly different, the main goals are quite simple, indicating that the users are able to work more efficiently using the Mini-MAM. As the tasks performed during the step-by-step task work through almost every feature of the Mini-MAM, it speaks the system's efficiency that the users are able to complete them in such a short time.

The XMS system has more options, and tools than the Mini-MAM, making the first 'free task' the participants perform a lot harder. In addition, 'the Step-by-Step' tasks used close to every feature in the Mini-MAM, while it barely scratched the surface of what the XMS system can do. The XMS system is designed to be administrated by a trained individual with experience in IT systems, and not to be picked up by non-experts with no previous experience. The potential source of bias from the participants, stem from the fact that most of the participants were familiar with the author of the report, and were able to discern which system was developed by the author.

Despite the low scores from the usability assessment, the XMS system is an incredibly powerful tool. During the step-by-step tasks, the subjects were asked to create a presentation, this was considered by many subjects as a cumbersome process, where they had to move through many different steps and screens before being able to create an actual presentation. What the subjective assessment did not properly display, was that once this set-up process had been completed once, it became significantly easier to create more presentations, or add new fancy slides to already existing systems. Through a system of screen templates, and screen types, the XMS system allows users to create, and re-use their content in many different ways, effectively operating as content manager, as well as editing tool. The XMS system, apart from managing content, also manages computer screens, presentation cycles on these screens, and is in many ways a complete control system for an exhibition center. Compared to our prototype, the XMS system is far more advanced than any potential Mini-MAM.

6.3 System overview

In its current state, the Mini-MAM seems to be able to perform the primary tasks it was designed for. The current Ingest Menu should provide the user with the option to easily add new content, as well as tag it with basic metadata. From table 5.3, the task of uploading content is given a decet score, scoring above 4 (Good) in the User Friendliness category. The Search Menu also appears to be decent, scoring close to 3(Fair), despite the unintuitive method for displaying the search results. We see that the search task is the lowest scoring task for the Mini-MAM, and it is believed that the method for displaying the search results is the root cause for the low scores. The Play, and Playlist related tasks all score rather highly, indicating that the decision of adding an integrated media player was worth the effort. The Treeview element displaying the playlists also seems to be a well implemented, despite lacking the ability to drag and drop elements. The Playlist view also suffers from resetting the widget every time a change is made, something that could become an annoyance during extended use.

Despite some minor shortcomings, and issues, the Mini-MAM prototype should be able to fulfil the requirements envisioned. With the ability to manage both content and metadata in a user friendly manner, as well as display the content with an intuitive, embedded media player, the requirements appear to be filled.

Chapter 7

Conclusion

The goal of this theses has been to design, develop and test a simple and user friendly Mini-MAM, that is effective, cheap, and easy to maintain. The developed Mini-MAM fulfils the requirements outlined throughout this report. It is able to manage media content, organize and store metadata, and display the media content to the user. These tasks are made available to the user through a graphical user interface that is both user friendly, and efficient. Through the development of the Mini-MAM, and the subjective assessment, this thesis has shown that the demand for a Mini-MAM exists, and that the developed prototype fit the outlined requirements of such a system.

From the subjective assessment, it is clear that the Mini-MAM is the more user friendly system. Scoring higher in all measured categories, as well as being the preferred system. With further development outlined in the Future Works section, the Mini-MAM Prototype can become a commercially viable system that fit the requirements of many non-broadcaster content owners.

7.1 Future Work

The Mini-MAM is still a prototype, and there are issues that have not yet been fixed, or updated. The primary tasks that should be further developed are:

- *Improve Search Functionality*
  At the moment, the search function will search through every column of every entity. It should be possible to limit your search to only search names, keywords or descriptions. It should also be possible to only search for a selected modality.

- *Improve Search Results*
  The current method for browsing search results is the least user friendly menu in the prototype. It has been suggested to create a treeview widget, similar to the playlist menu, that is populated by the returned results.

- *Ingest Status Bar* When adding a new video file, the transcoding process happens in the background, without any feedback to the user. This lack of feedback goes against all guides for good interface design, and should be remedied. As the ffmpeg conversion process does return some feedback, the framework for a proper progress bar is available.

- *Drag-and-drop support for playlist window.* It should be possible to edit the order of items in a playlist, by simply dragging and dropping the item.

- *Better Error handling, and code cleanup* At the moment, there is no uniform method for error handling in the code. A system with custom and specific exceptions, should be developed.

- *Bundling the Mini-MAM* Some sort of bundling should be performed, to allow users to easily install and set-up the Mini-MAM, as well as any required software tools.

Bibliography

[1] B.S. Manjunath Thomas Sikora, Philippe Salembier. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, 2002.

[2] Karl Paulsen. *Moving Media Storage Technologies: Applications and Workflows for Video and Media Server Platforms*. Focal Press, 2011.

[3] Lin Shan, Li Wei, and Zeng Ling. Research on novel solutions to existing problems of mam based on logical function framework. In *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pages 1–4. IEEE, 2011.

[4] Elizabeth Keathley. *Digital Asset Management: Content Architectures, Project Management, and Creating Order out of Media Chaos*. Apress, 2014.

[5] Fernando Pereira, Anthony Vetro, and Thomas Sikora. Multimedia retrieval and delivery: essential metadata challenges and standards. *Proceedings of the IEEE*, 96(4):721–744, 2008.

[6] William Y Arms. *Digital libraries*. The MIT Press, 2001.

[7] John R Smith and Peter Schirling. Metadata standards roundup. *Multimedia, IEEE*, 13(2):84–88, 2006.

[8] EBU. Tech 3293, ebu core metadata set. Technical report, EBU, february 2013.

```
https://tech.ebu.ch/docs/tech/tech3293v1_4.pdf -
  Found 7.10.2013
```

.

[9] Florian Stegmaier, Werner Bailer, Tobias Bürger, Mario Döller, Martin Höffernig, Wonsuk Lee, Véronique Malaisé, Chris Poppe, Raphaël Troncy, Harald Kosch, et al. How to align media metadata schemas? design and implementation of the media ontology. In *Proceedings of the 10th International Workshop of the Multimedia Community on Semantic Multimedia Database Technologies (SeMuDaTe 2009)*, volume 539, pages 56–69, 2009.

[10] Bruce Devlin. Mfx - the material exchange format. Technical report, EBU, July 2002.

```
https://tech.ebu.ch/docs/techreview/trev_291-devlin.pdf -
  Found 17.12.2013
```

.

[11] Andrew Perkis and Håvard Ferstad. Media asset management, prosjektrapport. Technical report, NTNU, June 2013. Found 28.8.2013.

[12] Wilbert O Galitz. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons, 2007.

[13] VLC. Python bindings - videolan wiki, June 2014.

`https://wiki.videolan.org/Python_bindings - Found 06.05.2014`

.

[14] Python. Welcome to python.org, June 2014.

`http://www.python.org - Found 10.04.2014`

.

[15] Python. Tkinter - python wiki, June 2014.

`https://wiki.python.org/moin/TkInter - Found 10.04.2014`

.

[16] Python. sqlite3 documentation, June 2014.

`https://docs.python.org/2/library/sqlite3.html - Found 14.04.2014`

.

[17] VLC. Videolan - official page for vlc media player, the open source video framework!, June 2014.

`http://www.videolan.org/vlc/index.html - Found 06.05.2014`

.

[18] FFmpeg. Ffmpeg, June 2014.

`http://www.ffmpeg.org/ - Found 20.04.2014`

.

[19] SQLite. Sqlite home page, June 2014.

`http://www.sqlite.org/ - Found 14.04.2014`

.

[20] Michael Owens and Grant Allen. *The definitive guide to SQLite*, volume 1. Springer, 2006.

[21] vvvv.org. Hwnd (windows), June 2014.

`http://vvvv.org/documentation/hwnd-%28windows%29 - Found 10.06.2014`

.

[22] Sony. Sony announces: Project morhpeus, June 2014.

`http://www.sony.com/SCA/company-news/press-releases/sony-computer-entertainment-a - Found 10.06.2014`

.

[23] Occulus Rift. The all new occulus rift development kit 2 8dk2) virtual reality headset | oculus rift - virtual reality headset for 3d gaming, June 2014.

`http://www.oculusvr.com/dk2/ - Found 10.06.2014`

.

[24] ITU-R. Methodology for the subjective assessment of the quality of television, June 2014.

`Technical Report BT.500-11 , 2002.`

.

Appendix A

A.1 Subjective Assessment Results

Results - Mini-MAM

Category	User Friendliness	Time Consumption	Menu Layout
General Impression	3,9286	3,9286	3,3571
Upload	3,7857	4,0714	3,7143
Search	3,0000	3,4286	3,2857
Playlist, Create	4,4286	4,2143	4,0714
Playlist, Edit	3,9286	4,0000	3,8571
Metadata, Edit	4,1429	4,2143	3,7857
Play	4,5000	4,5000	4,2143

Table A.1: Mean score for each category - Mini-MAM.

Results - Mini-MAM

Category	User Friendliness	Time Consumption	Menu Layout
General Impression	3,6060-4,2511	3,6060-4,2511	2,7905-3,9238
Upload	3,1632-4,4082	3,5490-4,5938	3,1543-4,2743
Search	2,4189-3,5811	2,8220-4,0351	2,6892-3,8822
Playlist, Create	4,1596-4,6976	3,8480-4,5806	3,6373-4,5056
Playlist, Edit	3,4483-4,4089	3,5406-4,4594	3,4043-4,3100
Metadata, Edit	3,6456-4,6401	3,7467-4,6818	3,2750-4,2964
Play	4,1593-4,8407	4,1593-4,8407	3,6638-4,7648

Table A.2: 95% confidence interval for each category - Mini-MAM

Results - XMS

Category	User Friendliness	Time Consumption	Menu Layout
General Impression	2,0714	1,7143	2,6429
Upload	2,2857	2,2857	3,0000
Search	2,7143	2,7857	3,0714
Presentation, Create	2,4286	2,4286	2,9286
Template	2,1429	2,0000	2,7143
Slide	2,5714	2,5000	3,1429
Presentation, Full Process	2,7143	2,7857	3,2143

Table A.3: Mean score for each category - XMS

Results - XMS

Category	User Friendliness	Time Consumption	Menu Layout
General Impression	1,6373-2,5056	1,3338-2,0947	2,0060-3,2797
Upload	1,8533-2,7181	1,7648-2,8067	2,3836-3,6164
Search	2,0834-3,3452	2,2352-3,3362	2,5101-3,6328
Presentation, Create	1,9374-2,9197	1,8579-2,9992	2,2965-3,5607
Template	1,6456-2,6401	1,5406-2,4594	2,1178-3,3108
Slide	2,0390-3,1038	1,9280-3,0720	2,6048-3,6809
Presentation, Full Process	2,2819-3,1467	2,3182-3,2533	2,6267-3,8019

Table A.4: 95% confidence interval for each category - XMS

How much time do you spend on your computer daily?

Option	<1 Hour	1-2 Hours	2-3 Hours	>3 Hours
Users	0	0	0	14

Table A.5: Observer survery

What Streaming service do you use?

Service	Spotify	Netflix	Youtube	Wimp	Twitch
Users	11	12	13	3	1

Table A.6: Observer survery

Do you currently have a media collections?

Option	Yes	No
Users	12	2

Table A.7: Observer survery

Which system did you find easier to use with manual?

Option	XMS	Mini-MAM
Users	0	14

Table A.8: Observer survery

Which system did you find easier to use without manual?

Option	XMS	Mini-MAM
Users	0	14

Table A.9: Observer survery

Which system do you think you spent the most time with?

Option	XMS	Mini-MAM
Users	14	0

Table A.10: Observer survery

Time Spent

	XMS	Mini-MAM
	05m57s	11m15s
	07m30s	08m30s
	04m30s	10m05s
	06m15s	10m31s
	05m32s	09m03s
	03m28s	07m41s
	07m15s	09m08s
	04m23s	09m18s
	05m20s	09m56s
	03m12s	07m45s
	06m09s	09m40s
	04m05s	08m16s
	06m20s	13m02s
	05m30s	12m23s
Average	05m23s	09m45s

Table A.11: Observer survery

Appendix B

B.1 Handout

Purpose of the test
The purpose of the test is to compare the Mini-MAM system to a commercial product XMS. The main focus will be to determine the usability of the two systems, from the viewpoint of a new user.

Instructions
You will be given a text document with a set of tasks you will perform on both systems. First you will try to manage the system without the step by step guide. After 5 minutes you will continue to the step by step guide. Here it's important that you follow the instructions step by step. There is no time limit, but your time spent on each system will be recorded.
If you experience any issues, please contact the supervisor.

After the usability-test you will be given a questionnaire with questions regarding the two systems. You will answer with a score from 1-5 on each question.

Score Chart				
1	2	3	4	5
Bad	Poor	Fair	Good	Excellent

For the purpose of the report, all personal details will be kept anonymous.

Thank you for participating.

B.2 Test Protocol

Mini-MAM - Free Task

- *Task:*Upload two files into the Mini-MAM system from the folder “XXX”, and add some keywords and a description
- *Task:*Create a new playlist. Give it a name. And add the 2 files you have uploaded into the system. Play the playlist.

Mini-MAM - Step by Step

- *Task: Upload file*
  Step 1) Choose file “file1” from Documents/Master/”XXX”
  Step 2) Fill in Keywords Y
  Step 3) Description
  Step 4) Save
  Step 5) Upload a new file: Choose file “file2” from Documents/Master/”XXX”
  Step 6) Import XML metadata. Find XML file from Documents/Master/xml
  Step 7) Add file to “Playlist 1”
  Step 8) Save

- *Task: Search for a file*
  Step 1) Enter the word Y
  Step 2) Start search
  Step 3) Browse search results
  Step 4) Select file “file1”
  Step 5) Add keyword Z
  Step 6) Save changes
  Step 7) Export file to Desktop
  Step 8) Delete file

- *Task: New playlist*
  Step 1) New playlist
  Step 2) Give the playlist a name “Playlist 2”

- *Task: Add media to playlist*
  Step 1) Search with the word “Y”
  Step 2) Start search
  Step 3) Browse search results
  Step 4) Select file “file1”
  Step 5) Add to playlist “playlist 1” and “playlist 2”

- *Task: Delete media from playlist*
  Step 1) Delete playlist 2
  Step 2) “Are you sure?”
  Step 3) Delete

- *Task: Add subplaylist*
  Step 1) Select playlist: “playlist 1”
  Step 2 Add subplaylist
  Step 3) Give the playlist a name: “playlist 3”
  Step 4) Save

- *Task: Play*
  Step 1) Select playlist: “playlist 1”
  Step 2) Play
  Step 3) Start playing

XMS - Free Task

- *Task: Upload two files into the XMS system from the folder "XXX", and add a comment.*
- *Task: Create a new presentation. Give it a name. And add the 2 files you have uploaded into the system, using templates and slides.*

XMS - Step by Step

- *Task: Upload File* Step 1) From the Main Menu: Select Exhibit Content
  Step 2) Under "Files"-"Add new file (advanced)" upload the file "file1" from Documents/-Master/"XXX".
  Step 3) Update filename (scriptcode) to "file1"
  Step 3) Select language "Norsk"
  Step 3) Add a comment
  Step 4) Select "Save"
  Step 5) Clear

- *Task: Search* Step 1) Search for file1
  Step 2) Edit the file "file1", and add two words to the comment
  Step 3) Save

- *Task: Create Screen Type* Step 1) From the Main Menu, enter the Broadcaster menu.
  Step 2) Enter the Configuration tab, and select Screen Types
  Step 3) Add a new screen type with name 1920x1080, set the Width to 1920 and the Height to 1080
  Step 4) Save

- *Task: Create Template* Step 1) Enter the Template menu, and add a new template.
  Step 2) Make sure the Template has a screen size of 1920x1080, and name it "Vertical Splitscreen – Image". Then add template
  Step 3) Add an image to the template, with ID 'Left', label 'Left', x positition 0, y positition 0, with a width of 960 and a height of 1080. Do not upload an image.
  Step 4) Add another image to the template, with ID 'Right', label 'Right', x position 960, y position 0, with a width of 960 and height of 1080. Do not upload an image.
  Step 5) Save

- *Task: Create Slide* Step 1) Enter the Slide menu, and add a new slide.
  Step 2) Name the slide "Vertical Images #1", with a screen type of 1920x1080
  Step 3) Select the Vertical Splitscreen – Image template, and upload two images From Documents/Master/"XXX Step 4) Save the slide

  Create a new template, this time adding a video field to the template. Give the video field an ID 'main, label 'main, x position 0, y position 0, with a width of 1920 and height of 1080.

  Create a new slide using this video template, with the file "Havert.mp4" from Documents/Master/"XXX".

- *Task: Presentation* Step 1) Enter the Presentation menu, and add a new Presentation, choose a suitable name.
  Step 2) Make sure the screen type is set to 1920x1080, other options should be kept

unchanged.
Step 3) Add the 2 slides to the timeline, and save the presentation.
Step 4) Delete presentation

Appendix C

C.1 Content of ZIP file

'DESIGN/':
MAMPowerPoint.odp - Presentation made to showcase MAM design ideas.

'SURVEY/':
SubjectiveAssessment.ods - Contains raw results, and calculations from subjective assessment.
SubjectiveAssessmentQuestions.pdf - Contains screenshots from the online questionnaire

'SOFTWARE/': - Required python code
'data/resorces': - Required images for the Mini-MAM
importfile.py
ProsjektGUI.py
sqlhandler.py
vlc.py
wexpect.py
xmlimport.py

'README.TXT' - Contains information about the .Zip folder, as well as instructions on how to run the Mini-MAM.

C.2 Test Material Subjective Assessment

The hard drive containing the test material from the KOMOPP project is administered by Andrew Perkis, for access contact him.