“NEP-TUNE”

PREFACE

Contact

Please email tobythetenor@gmail.com for any enquiries.

Additional proposals

To the end of this paper has been appended a page summarizing further project proposals.

Result

This paper received a distinction (91%) in December 2016.

Below follows the dissertation produced for the OU in 2016.

TM470

SPEED-OPTIMISED INDEXING SOLUTION FOR FILE METADATA, WITH MOBILE-FRIENDLY FRONT-END

"NEP-TUNE"

THE FINAL PROJECT REPORT

Toby Scholz

Supervisor: Charly Lowndes

B6960379

12 September 2016

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

(only figures in the main body are listed here, figures in the appendix are referred to separately)

TABLE OF CODE EXCERPTS

By recommendation of my tutor, hyperlinks to references, the glossary, and appendices, are provided.

1. PROJECT REPORT

1.1. Motivation

This project seeks to develop a solution for indexing and searching large collections of files.

1.2. Nature and context of the problem

Most people will have had some experience with digitising a physical artefact (e.g. scanning a drawing), and trying to retrieve the digitised version at some point in the future. A small collection of such files can easily be managed, but in larger collections, such as may be found in corporate, educational, or governmental environments, searching and finding individual documents can present a challenge.

A traditional approach to this problem is to use a commercial document management system. Evaluation presented in appendix 4.11 reveals that such systems commonly exhibit one or more of the following drawbacks:

1) They tend to store the actual documents in a database, which in turn has the following downsides:
   - documents can often be accessed only using the software that archived it
   - that software (and therefore the documents collection) tends not to be portable, as it requires licensing, and specific hardware to run on
2) Traditional document management systems often are restricted to a pre-defined set of file types, leaving some documents effectively "unmanageable".
3) Traditional document management systems rely on file metadata (which may either be appended to the file itself, or held separately in a proprietary database) being correctly categorised and populated.

Whilst analysing the use of commercial document management systems within academic institutions, Lixandroiu et al. (2015, pp 207 - 210) state that there is a "strong demand for improvement" for managing the "increasing amount of [heterogeneous, digital] information", and point out that existing commercial solutions can have an "extremely high initial cost" (both in terms of money and time, since the software relies on "classification according to [proprietary] classification schemes or taxonomies").

Particularly the heterogeneity of documents within a collection presents a substantial hurdle for many systems ('documents' here can mean anything from "text materials […], to recordings of native speakers of indigenous languages to videos"), which has an impact on both "scalability" (the systems evaluated by the authors reside on a single server without a lightweight, public interface) and "data mobility" (data cannot be exchanged between systems), (Lixandroiu *et al*., 2015, p 208).
A common manifestation of this problem are collections of heterogeneous media files, reaching from

amateur recordings of concerts, to original compositions, and digitised historical recordings (for example those found in ECLAP – see section 1.4, analysis of likely impact), all of which may exist as digital files, but with only partially or incorrectly populated metadata.

Personally, I am experiencing this situation with a sizeable collection of digital music recordings, many of which are non-, or semi-professional recordings of performances I have participated in (an example is shown in figure 1).

Figure 1: Towards New Possibilities - a university project in 2007, which I co-organised, recorded, and performed in

1.3. Proposed solution or recommendations

The core idea is to make large collections of heterogeneous files more easily accessible to the end user.

Three key concepts are fundamental to the proposed solution:

1) indexing the collection can greatly improve the speed of search (as opposed to iterating over files to extract information)
2) extending search to include auxiliary information, such as the file name, whilst at the same time not enforcing a categorisation of the search term, will ensure maximum visibility of all available information
3) delivering the actual file via a standardised interface reduces access overhead, and improves cross-platform compatibility

Hence, the proposal is to build a software solution, which indexes files on a given network share, and provides a fast search facility that is not restricted to the type of metadata, which is also able to return a requested file via the same interface (the Nep-Tune server component).
In addition, a mobile phone application shall be provided to illustrate the functionality of the software (the Nep-Tune client component).
All computationally heavy work shall be handled by the server component, thus enabling a client solution to work on computing devices with limited processing power.
The proposal results in the following draft of features for the software:

1.3.1. Features of the proposed server component

- interacts with files in a read-only manner (meaning the collection of files remains unchanged)
- offers a standardised interface, which allows the creation of custom client software
- returns an actual binary file via the same standardised interface to the client for further processing
- is open source, and can thus be enhanced to index any type of document directly by its users
- offers some level of platform independence, so it is portable across systems
- scales for performance, meaning the addition of more processors results in quicker search
- transcodes multimedia file to HLS on the fly for Apple iOS compliance

1.3.2. Features of the proposed client component

- runs on Apple iOS devices, written in Swift
- offers real-time search, leveraging the performance of the server component
- implements playback of media files and HLS streams

The proposed software is underpinned by the following underlying concepts and fundamental principles:

1.3.3. Underlying concepts and fundamental principles

1.3.3.1. Server component

- multithreaded, distributed implementation, based on the principles taught in M362 (The Open University, 2008a)
- networked communications, based on further concepts presented in the above module
- highly efficient search algorithm, based on the fundamental mathematical principles introduced in M269 (The Open University, 2014a)
- engineered according to the standards and principles taught in TM354 (The Open University, 2015f)
- persistence using a NoSQL document-oriented database implementation, based on research, and prior experience

1.3.3.2. Client component

- written in the Swift language
- leveraging HLS to facilitate reliable multimedia streaming over a network connection

…both of which are newly acquired skills based on research and knowledge acquired throughout the project.

1.4. Analysis of likely impact

For large indexing tasks, Lixandroiu et al. (2015, p 208) recommend the use of FLOSS components to ensure maximum deployability, but continue to point out that existing FLOSS solutions suffer from the same drawbacks as commercial software, the biggest of which is that software packages are typically self-contained, meaning they do not offer the ability to interact with third-party clients.

The proposed server component overcomes this problem by being both open source (thereby offering the possibility of being enhanced by its users), and providing a standardised interface (meaning users can provide their own client component (such as existing corporate software) to process the returned information to suit their needs).

Bellini et al. (2014) introduce the ECLAP library system (a search interface for a library collecting information about performing arts projects and artefacts within Europe), a concrete candidate for benefitting from the proposed solution. The ECLAP system (website shown in figure 2) deals with the same problem this project tries to address – large collections of files with heterogeneous metadata – but does not seem to do so particularly well.

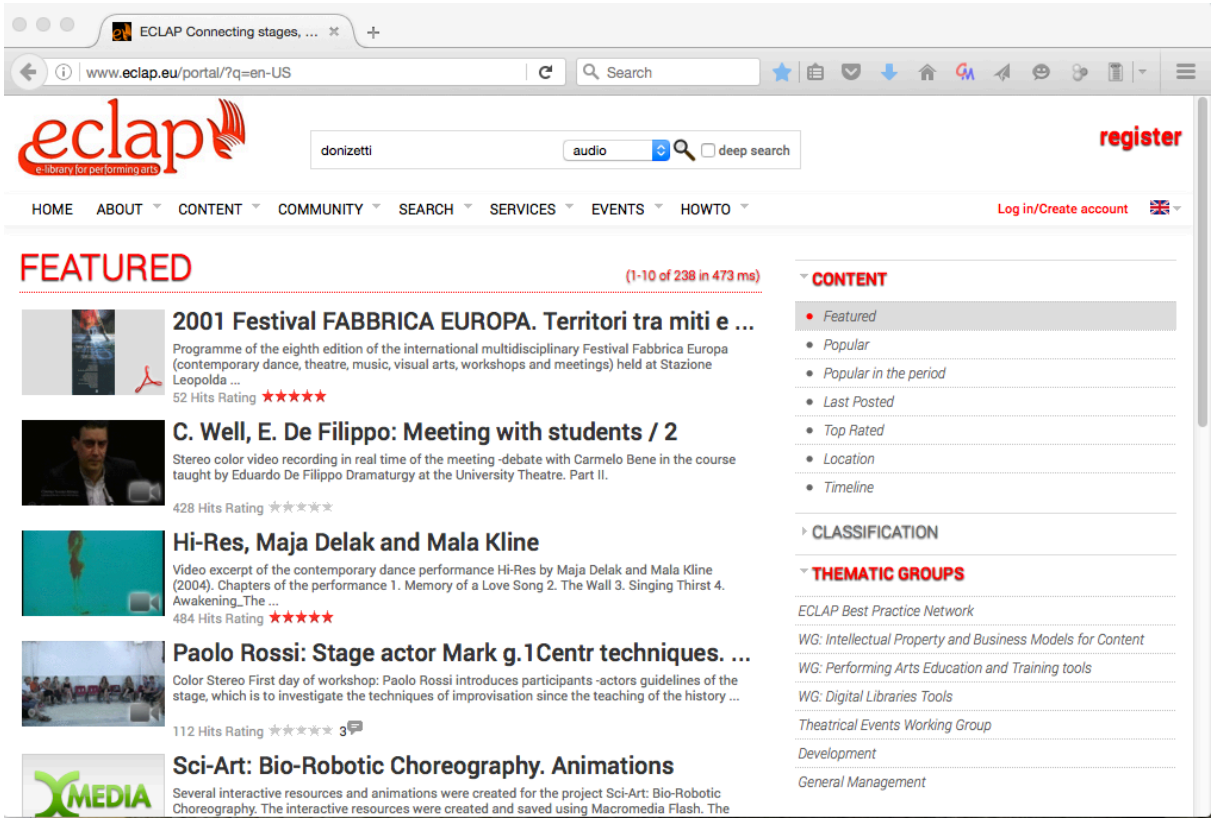


Figure 2: The ECLAP website

Searching for keyword “donizetti” took an average of 24.9 seconds (see appendix 4.4 for details) to return (see figure 3) - in my view an unacceptably long time.

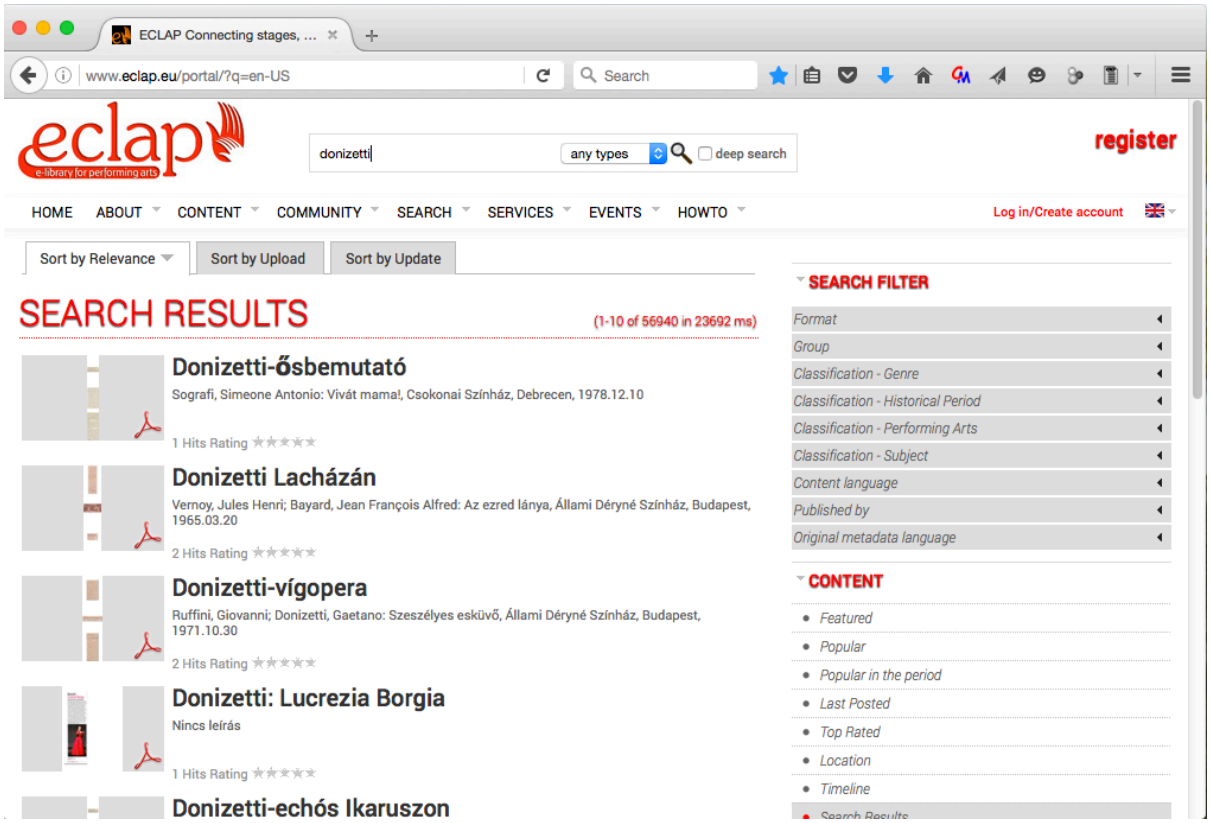


Figure 3: ECLAP search result

From the paper, it is unclear whether Bellini at al. (2014) propose an improvement to the current ECLAP library system, or whether the recommendations in the paper have actually been implemented. In the former case, this project may present a valuable alternative to the proposed solution built on Apache Solr (Apache, 2016d), whereas in the latter case, it seems that the recommendations have had a significant performance impact, to which this project may also offer a remedy.
Of course any assumption as to why ECLAP performs so poorly is pure speculation, since neither the size of the dataset, nor specifics of the underlying hard- or software are known.

An immediate use case was identified in a conversation with my friend Colin Dunn from Boosey & Hawkes (Dunn, 2016). Currently, the music publisher uses many different software packages to find digitised content in its various archives, which can make it laborious to find a certain document if its location is unknown. An initiative is now in progress to consolidate access into a single user interface. The Nep-Tune server component would have the ability to make the whole collection accessible at once, provided access is offered via a samba share, while the standardised interface should make integration with existing corporate software simple, retaining the desired layout without requiring users to learn using a new system.

Taking this approach should make deployment, compared to deploying a discrete system, significantly cheaper, as Nep-Tune can act as a "Component-off-the-shelf", as discussed in *Developing New Processes for COTS-Based Systems* (Brownsword at al., 2000, pp. 49-55).

A few more general areas that might benefit from the application of the project software are identified in table 1 below.

Type of Organisation	Justification
Libraries	Libraries epitomise collections of heterogeneous data, from digitised books to voting ledgers. The proposed system could offer real-time search through vast archives.
Universities	Once home to philosophy and humanities only, universities have hugely diversified, and it has become possible to study everything from sports to circus performance. Thus, the type of artefacts produced varies widely, presenting a challenge to archivists. A type-agnostic system would aid accessibility of data.
Museums	Projects, such as *Science of 3D* (Hess, 2013) produce a completely new type of data. So long as there is any parsable information associated with it, the proposed system will ensure digital files produced from scanning historic artefacts will remain accessible to a wide audience for a long time to come.
Government organisations	Government documents are often changing. For example, it is likely that the type of information stored in an electronic passport chip is going to change once the UK will have severed its ties with the European Union. This makes such data unsuitable for a traditional, relational indexing scheme, benefitting from the NoSQL approach outlined in this document.
Home theatres	This project was inspired by (and offers an immediate solution to) searching a collection of media files on a NAS. The client playback capabilities would make it a viable replacement for commercially available solutions, such as those presented in appendix 4.11.

Table 1: Possible areas of application for Nep-Tune

The overall goal of the project is to offer an alternative, easy to use, lightweight, and fast software application, which is able to appeal to a wide range of users.

By offering an indication of both general areas of possible deployment, and selected usage scenarios, the likely impact of the project outcome has been comprehensively described.

1.5. The project lifecycle

An agile approach to software creation champions an iterative style to development, such as the *Iterative Enhancement* model described by Basili at al. (1975), which proposes a "step-wise refinement" of the development process.

Adapting this model, a project plan was developed sectioning the project into three major task phases, each of which comprises a defined set of features facilitating testing and demonstrating certain elements of the software (aligned with the deadlines for each of the three required assignments), while also allowing time to address issues and complete academic work.

Interleaving between server and client component, the schedule of phases (presented in full in appendix 4.5) approximately yields the following major tasks (excluding academic work, details of which are accounted for in the detailed timetable in the appendix 4.6):

1) Providing infrastructure (setting up server, installing dependencies, and creating an Apple developer account)
2) Create basic executables for server and client component, integrate with third-party libraries and develop unit tests to ensure proper operation
3) Add indexing and search facility to server component, and basic audio player to client component
4) Provide connectivity between server and client via proposed standardised interface
5) Implement on-the-fly transcoding for media files
6) Performance analysis
7) Bug fixes and minor enhancements

Care has been taken to use meaningful variable names, follow conventions, and provide comments in order to ensure maintainability of the code for the longest possible time.

1.6. Account of related literature

The idea for the project was inspired by a personal notion of an ever-increasing amount of stored data, which was difficult to access on low-powered devices, such as mobile phones.

The Worldwide Storage in Big Data Forecast, 2015–2019 study (DuBois, 2015) affirms this notion by predicting "double-digit" growth of stored data in the ensuing three years, approximately doubling every two years. At the same time, Intel's authoritative annual report 2016 (Krzanich, 2016, p 14) highlights a slowdown in advances of processor development (predicted in *The end of Moore's Law? Why the theory that computer processors will double in power every two years may be becoming obsolete* (Green, 2015), which led to the annual report), as the "world's largest chipmaker" (Bylund, 2015) "lengthens the amount of time [they] will utilize [their] 14nm […] technologies" (Green, 2015), moving from a two- to a three-year development cycle. Together, these sources affirmed that it will not be possible to rely on increases in processing power alone to manage ever-growing collections of data, thereby justifying the project.

In *A system architecture based on open source enterprise content management systems for supporting educational institutions*, Lixandroiu et al (2015, pp 207 - 214) comprehensively researched the type of data found in educational institutions, and how this data is handled. It was credibly shown that:

- organisations now have to cope with "numerous new sources of content"
- there is a "strong demand for improvement" with regards to harmonising data structures and accessing data
- institutions are becoming "increasingly dependent" on electronic documents
- current document management solutions suffer from "scalability and sustainability", "data mobility" (platforms are not able to exchange data), and "user adoption" problems

Together with a detailed evaluation of existing commercial document management software (presented in appendix 4.11), aided by the *Best Document Management Software and Systems* review (Brooks, 2016), the research conducted by Lixandroiu et al. was instrumental in determining the requirements for the project software, which include standardised input-, and output interfaces, support for a large variety of content, and an open source release that will enable adding custom content parsers easily (see requirements analysis in appendix 4.7).
A survey on indexing techniques for big data: taxonomy and performance evaluation (Gani at el, 2014, p 241) recognised that "Available solutions for efficient data storage and management cannot fulfil the needs of […] heterogeneous data where the amount of data is continuously increasing", and proceeded to comprehensively evaluate database *types* that would be suitable to best tackle this problem. While Gani et al make no recommendations, it is clear from their deep and comprehensive categorisation of data management and indexing techniques that a document-oriented NoSQL

database solution seemed best suited for this project, since it excels at storing data for collections of heterogeneous files.

Data management in cloud environments: NoSQL and NewSQL data stores (Capretz et al., 2013, p 15) compared a variety of database *implementations* by functionality and speed, highlighting MongoDB as the ideal candidate, stating that "MongoDB can achieve strong consistency", while at the same time offering an access policy ("multiple readers, single writer"), which is perfectly suited to this project. This opinion was further underpinned by the cursory *Survey on NoSQL Database* (Du et al, 2011), which praises MongoDB's "support […] to store complex data types" and "High-speed access to mass data".

Having decided on the database implementation, a lifecycle model was needed.
Iterative Enhancement: A Practical Technique for Software Development (Basili et al., 1975, pp. 390 – 396) proposes a "step-wise refinement" structure that divides the development process into discrete tasks, beginning with important core functionality, and placing lesser features at the end of the cycle. Basili et al. pre-empt many elements that form part of contemporary agile practices (such as those used at my place of work), and their focus on "structure, modularity, modifiability, usability, reliability and efficiency" helped devise the schedule of phases (discussed above), which led to the successful and timely completion of this project.

The design and analysis process was guided by the TM354 module materials (The Open University, 2015c); having recently completed this module, I had previously analysed its weaknesses, and made recommendations as part of academic submissions to the Open University, which meant optimal approaches to software design and analysis were still fresh in my mind.

An Efficient Design and Implementation of an MdbULPS [MongoDB-based unstructured log processing system] *in a Cloud-Computing Environment* (Cui et al., 2015, pp. 3182 – 3202) provided invaluable analysis concerning the service model, real-time information work-flow, database configuration, and API integration. While the topic of log processing is rather different than the indexing of file collections this project seeks to address, it was possible to extrapolate a lot of information from this article, particularly with regards to working with unstructured data, parallelisation of the access process, memory management and scalability.

The multithreaded, distributed implementation of the server component in Java was closely modeled on the chat server example from the M362 module materials (The Open University, 2008b), which demonstrated an unusually high amount of integrity and accuracy.
For the client component, *Developing iOS 8 Apps with Swift* (Hegarty, 2015) provided a basis to start development.

This series of lectures provides an excellent bridge from well-known languages, such a C++, to Swift, by highlighting the peculiarities of Swift (such as 'Optionals', and initialisation), explores how to

interact with the iOS front-end interface, and introduces the auto-layout feature.
The first few lectures are based on designing a calculator app, which I followed along and built. Stanford University is a highly reputable institution, and its course materials have guided me many times in the past, when Open University module materials seemed contradictory.

Anže Rehar's paper *Implementacija protokola HTTP Live Streaming v programskem jeziku Java* (Rehar, 2012) explored ways of realising HLS in Java, and, despite the language barrier, provided a good starting point for evaluating possible approaches the transcoding media files.

The literature review would not be complete without a reference to technical documentation. Oracle's Java documentation (Oracle, 2016d) proved an accurate and mature source of information, providing a good amount of examples, which makes it very easy to work with.
Unfortunately, Apple's *iOS Developer Library* (Apple, 2016g) is rather harder to work with, for two reasons: firstly, iOS still uses the Objective-C programming language, and many references document its usage (as opposed to Swift, which my client component is written in). Secondly, Swift itself is a very young language, and still changing fast. As a result, I found in places that the documentation referred to a different version of Swift than the one I was working with (2.2), stating a syntax different to the one required. Thus, information from the *iOS Developer Library* often had to be complemented with trial and error, or additional resources, such as online forums.

Having completed most of the development items, test cases were developed again based on the TM354 module materials. However, *Optimization of information retrieval for cross media contents in a best practice network* (Bellini at al., 2014) allowed me to perform a usability analysis against a real-world application. Bellini et al. researche possibilities to optimise the ECLAP (ECLAP, 2016) system, which currently struggles with unacceptably high search times (see section 1.4 Analysis of likely impact).
This paper allowed me to take its analysis of the type of data in question, and the proposed solution, and compare it against this project, which seems to have implemented most of the recommendations Bellini et al. made.
In addition, *Computer Architecture and Amdahl's Law* (Amdahl, 2013, pp 38-46) helped quantify the potential performance gains of Nep-Tune, evaluated against additional computing resources.

Besides the above key literature, research on the problem, and the implementation of its solution, was aided by podcasts, such as *Science Weekly* (The Guardian, 2014), which were very valuable less for their content, but rather for their implicit permission to download and store them, aiding the proposition of creating reproducible test results. This becomes particularly important in a time when the use of downloaded media files, including those that were regularly purchased, is increasingly

restricted by inescapable regulations, such as *The Copyright and Rights in Performances (Personal Copies for Private Use) Regulations 2014* (Great Britain, 2014).

The occasional search on the internet seems unavoidable for software development, and yields articles and blogs such as *Compiler error: Method with Objective-C selector conflicts with previous declaration with the same Objective-C selector* (O. P., 2015). Such sources will almost always only pertain to a very specific problem, and offer little use or credibility beyond that problem.

1.7. Legal, Social, Professional, and Ethical Issues

1.7.1. Legal issues

At least some of the files in the collection aiding to test the outcome of my project may be governed by *The Copyright and Rights in Performances (Personal Copies for Private Use) Regulations 2014* (Great Britain, 2014). Under these regulations, I would not be permitted to continue to use those files to carry on the development of my project for commercial purposes, once the project has been delivered, as this would constitute a violation of section 28B(1)(c-d). I am however permitted to use those files under section 28B(5)(b), so long as I ensure that no third party could obtain a copy, intentionally, or unintentionally.

During the project, I restricted myself to working with files that are liberally licensed, such as podcasts, and the project software is packaged without any actual content files.

Since the implementation of Apache's Tika library (Apache, 2016c), the vast majority of development can be completed using files unaffected by the above act, with the exception of development specifically relating to copyrighted content (e.g. correct handling of DRM enforcement), which remains a consideration beyond the scope of this project.

The licenses of all third-party libraries that form part of the software have been carefully checked to ascertain their use is legitimate within the context of this project.
Modified libraries are no longer used (Apache Tika replaced the previously modified mp3agic (Patricios, 2013) library).
JCIFS (JCIFS, 2014) is released under the LGPL license (reproduced in appendix 4.18), Apple's Swift and Apache Tika under the Apache license (reproduced in appendix 4.19), and Oracle Java under the Oracle developer license (reproduced in appendix 4.20), all of which generally allow non-commercial use with a few conditions, such as the requirement to include the appropriate license text in distributions (a commercial release of the project software is not under consideration at this point in time).

While Apple imposes some restrictions on the distribution of an iOS app in its App store (Apple, 2016e and Apple, 2016f), this concern is beyond of the scope of the project.

Even though it could be argued that the software produced as part of this project could aid the distribution of pirated content (by enabling better indexing facilities for files of potentially unknown content), the target audience of the software being libraries, academic institutions, and professional bodies, makes any such concern negligible.
Similarly, a hacker gaining illegitimate access to the software's standardised interface could with ease obtain copies of files illegally. The server component currently offers no facility for authentication or non-repudiation, placing the onus of restricting access appropriately on the client, or the network the software is deployed within.

No part of the software is deliberately designed, or makes a significant contribution, to engaging in illegal activities.

Although encryption of communications remains beyond the scope of the project, it is planned to be added immediately after submission. Meanwhile, confidentiality must be ensured by restricting use of the software to a Local Area Network, or securing access via a VPN or similar means.

As for any networked application, cross-border legal issue should be considered at the deployment stage. The software has been vetted against UK and US law, but no assurances can be given that the software complies with the law of any other countries.

1.7.2. Social issues

Since the software is designed to ease access to stored (media) files, it could be argued that it could contribute to social isolation by making it more attractive to listen to music, as opposed to interact socially. However, in the light of commercially available streaming services, such as Spotify (Spotify, 2016), I do not think that the project software will significantly aid social isolation, even if it were widely available.

To the contrary, I feel it could make an overall positive contribution by shortening the time it takes to find content in digital archives, such as libraries or research facilities, allowing the user more time to dedicate himself to the task at hand (or even socialise). After all, the point of addressing this problem is to overall integrate existing (and future) data in digital archives, which has not been diligently tagged with metadata, as well as bridging different approaches to data management in general.

1.7.3. Ethical issues

There appear few ethical concerns associated with this project.

The target audience is unspecified in terms of class, race, gender, or any other social dividers – the project may serve to aid managing wind farms in rural Africa equally well as indexing ancient artefacts in the Smithsonian museum.

It may be considered an issue that this project does not actively seek to address inequalities (for example in relation to specific gender- or age groups), but since this project does not look at the nature of such inequalities, it is not feasible to also try to address them.

Not directly involving other participants, there is no measurable ethical impact as a direct result from producing this report.

One aspect that remains beyond the scope of this project is to evaluate how well the client app will work, when operated by users with visual, haptic, or auditory impediments. The iOS framework will automatically be able to read out any on-screen text, and understand various accessible types of button actions. However, full integration with the iOS accessibility protocol (Apple, 2016k) was not possible during the project, owing to time constraints, but is planned after the submission of this paper.

Another concern could be that the additional server I set up to run the server component will draw additional electricity, and equally, electricity is used up by me writing the code and the assignments. This of course has a negative impact on our environment, since additional fuel will be required to meet this need.
In my view, if my software becomes successful, the benefits will however outweigh this concern, since time will be saved searching collections of files in future, thereby reducing the electricity consumption of the eventual users of the software.

The software will make files more easily accessible, which may hitherto have escaped discovery. For example, a keyword search may yield a result of a file that was accidentally included in the path, but was not meant to be. This does raise the issue of how easy it will be to maintain confidentiality, which is not governed by authentication, or other access-restricting mechanisms.
Since the software returns the actual file to the client on request, it will not be possible to undo the action of downloading a file.

1.7.4. Professional issues

Taking guidance from the *Legal, Social, Ethical and Professional* section of the TM470 module (The Open University, 2016), the following observations can be made:

- ✓ I have no vested interest in any of the technologies used in this project, nor are any third parties influenced by its outcome. I therefore consider myself free from any conflict of interest.
- ✓ In line with the schedule of lifecycle phases (appendix 4.5), the deliverables of this project are fit for purpose, as far as the limited scope of the project allowed.
- ✓ The tools used are adequate choices for the project. As renowned analyst firm RedMonk reports, Java is the world's second most popular programming language, with an estimated 9 million skilled developers worldwide (RedMonk, 2015). While performance considerations may have made it desirable to realize the implementation of the project in C++, the project being based on the Open University's M362 module (The Open University, 2008) tilted the choice strongly in Java's favour.

  As Du et al. elaborated in their paper *Survey on NoSQL Database* (Du et al, 2011), MongoDB is the preferred choice of document-oriented database, despite its young age.

  With Apple's iOS being the world's second most popular mobile platform, according to IDC (2015), Android may have been the obvious choice for the client implementation. However, a strong developer base, fast growing parent company (Forbes, 2015), and my own familiarity with the platform tipped the choice in favour of iOS, with plans to offer client software on other platforms (including desktop operating systems) being beyond the scope of the project.

The libraries employed fully serve their purpose, and may be replaced by different libraries, should it turn out to be necessary in the future

Based on the above justification, I believe the choice of the technologies made were the most appropriate in the given context.

- ✓ No pirated, or otherwise illegally acquired software was used to produce the project software. The software is written using the xCode and Netbeans IDEs, both of which are freely available. The server runs the Ubuntu Linux distribution (Ubuntu, 2016), which is free for non-commercial purposes. I rightfully and outright own the hardware used to complete the project.
- ✓ Legal issues were addressed above.
- ✓ I previously marked the absence of some skills as a risk factor in the risk analysis in appendix 4.10, but have been able to acquire the necessary skills to meet the schedule of lifecycle phases (appendix 4.5) as planned in the detailed timetable in appendix 4.6.
- ✓ Progress of the project in relation to the project plan is addressed in section 1.10 – Review of project management.
- ✓ Although it is unclear who will maintain the software in future, it was designed with readability in mind, and the server component was published under the LGPL license.

Having comprehensively identified the relevant issues above, particular attention was paid to taking them into account in order to maintain professional behaviour, and apply a suitable approach in all areas of the project.

1.8. Account of project work and its outcome

1.8.1. Analysis

1.8.1.1. The modelling process

The modelling process builds upon elements of the agile analysis approach as introduced in TM354 (The Open University, 2015c), here grouped into four distinct activities:

Domain modelling, which is concerned with modelling the existing business domain in an effort to understand the problem, and how the problem is currently addressed.

Software (or structural) analysis, which proposes a software solution to the problem(s) identified during domain modelling.

Requirements analysis, which defines testable properties the composed software must have to address the problem.

Design analysis, which defines how the software will fulfil the requirements architecturally and functionally.

In addition to the four activities described above, an evaluation against existing products was carried out, which served two purposes: firstly, as inspiration for approaching some of the particular challenges faced while developing the software, and secondly, to analyse the shortcomings of those products, to avoid making similar mistakes, and underline the validity of this project.

1.8.1.1.1. Domain modelling

Considering the domain to be a collection of media files, the following observations can be made:

- files come in a variety of file formats
- collections can contain a large amount of individual files, which may be grouped in directories
- large collections of files are not typically kept on a local drive, but network storage
- the metadata for individual files may be incomplete, or incorrect
- the metadata available will vary by file type, and individual files
- only read access is required; no changes to the underlying files are necessary
- consumption happens increasingly on mobile phones
- finding a particular file in the collection by search is typically very time consuming, and really only feasible with knowledge of the underlying directory structure once the collection exceeds a certain size
- traversing a file system may take an indeterminably long time

- transmission of large media files over a network can lead to complications (dropped / slow connections), for which reason Apple imposes a limit on the file size that can be transmitted without transcoding to HLS (Apple, 2016e)
- a common use case for storing files on a network share are media files (e.g. films), which tend to be large in size
- To ensure best performance, computationally heavy tasks are often delegated to servers

1.8.1.1.2 Software analysis

Evaluating the findings of the domain analysis above, the following statements can be made about a possible software implementation:

Server component

- a standardised HTTP interface will offer maximum compatibility with third-party clients. The interface should be file-type agnostic, and be able to return the file itself
- metadata is handled differently for each file type, probably requiring a software library per file type
- a persistence solution will have to cope with a heterogeneous dataset for metadata
- samba is a widely used protocol for connecting networked data storage to multiple clients. It should be supported

Client component

- a client for Apple iOS will allow me to test the software on my own devices
- the client needs to provide means for searching and selecting an item
- the client needs to provide basic playback functionality
- an Apple iOS client requires using the HTTP Live Streaming protocol

Desired distinctive features (or unique selling points)

- read-only approach leaves files untouched
- the software can be readily adapted to support additional file types
- standardised HTTP interface ensures cross-system compatibility
- the software can easily be integrated with third-party systems
- out-of-the-box compatibility with any samba share
- out-of-the-box compatibility with a large number of file types

Evaluation against existing solutions

To ensure this project does not duplicate existing functionality, an evaluation of the proposed software against existing commercial solutions was carried out (see appendix 4.11 for details).

In summary, existing solutions can roughly be grouped into two categories:

1) Media-focussed solutions, i.e. software, the primary purpose of which is to manage and play back media files

2) Document-management systems, i.e. software, the primary purpose of which is to manage collections of documents.

The proposed solution bridges both categories, by aiming to deliver some elements typically found in document-management systems (e.g. indexing of content), and other elements typically found in media-focussed solutions (e.g. playback of media files).

The evaluation shows that typical document management solutions will take ownership of the documents they manage, by saving them to a database, and allowing modifications, whilst not offering a solution for the retrieval and playback of multimedia files.
In contrast, typical multimedia applications do not offer an indexing solution. Both types of solutions usually offer search facilities, but in all cases, the search functionality of the evaluated media-focused solutions was restricted to searching the local device, not extending to an attached network share.

Thus, it can be concluded that the functionality offered by Nep-Tune is not offered by any of the existing solutions surveyed.

1.8.1.2. Requirements analysis

The process of eliciting requirements asked questions concerned with producing a software that conforms to the six criteria introduced in the TM354 module: usefulness, usability, reliability, flexibility, availability, and affordability (The Open University, 2016h).
The requirements were formulated by looking at each of the observations made during domain analysis, and defining a testable property, which was vetted against the six above criteria. This process is presented in appendix 4.7, with the resulting requirements summarised below:

Server component

To address the problems at hand, the software needs to be able to index the content of a samba share, and offer a standardised interface that allows it to return both search results and (multimedia) content, also offering a transcoding facility for those file types that require transcoding for HLS. Particular attention should be paid to search performance.

Client component

In order to showcase the server component, the client component should be able to interact with it, play back files retrieved from it, and offer an intuitive user interface.

1.8.1.3. Design analysis

Server component

To achieve the desired qualities of maximised cohesion and low coupling, it is recommended practice to separate functionality into several packages (The Open University, 2015j), comprised of components, each of which can be replaced with other components, which match or lower preconditions, and match or strengthen postconditions.

Using this premise, functionality can be grouped into packages as follows:

- client interaction (including network connectivity), accepting client connections and handling queries
- interaction with the database for persistence
- the indexing process, to build the index
- search
- a “live” data store, to keep the index in memory for fast access
- file and stream handling, to return data to the client
- media file transcoding
- file type specific libraries
- interaction with the samba share, enabling the retrieval of files for indexing and further processing
- maintenance, which keeps the index up-to-date
- configuration, logging, and error handling utility classes

This grouping is further illustrated in figure 4 below:

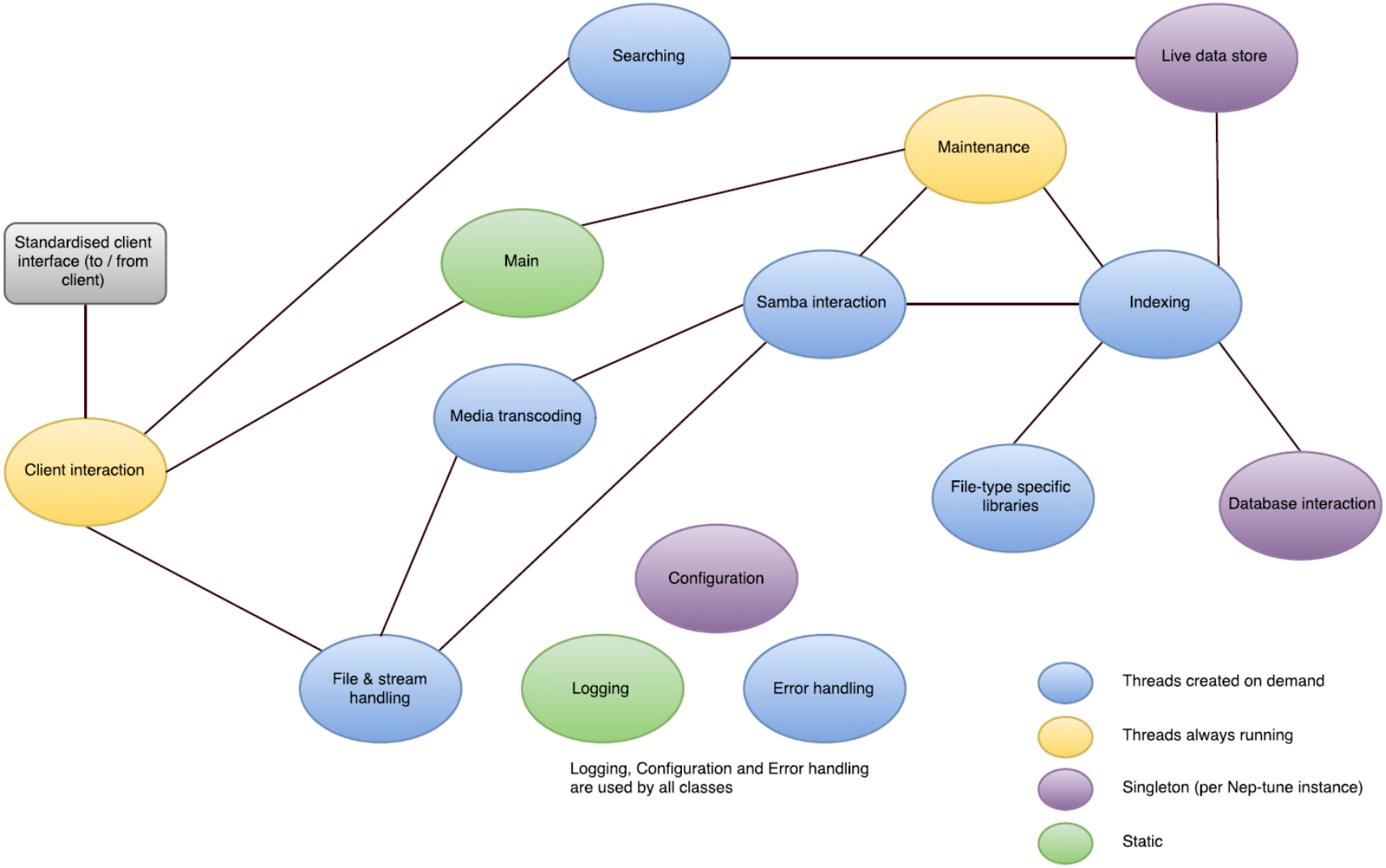


Figure 4: Diagram of the server component

Client component

Client functionality can be separated as follows:

- User interface
- Connection handling, which interacts with the server
- multimedia playback

A conceptual diagram of the client component is provided in figure 5.

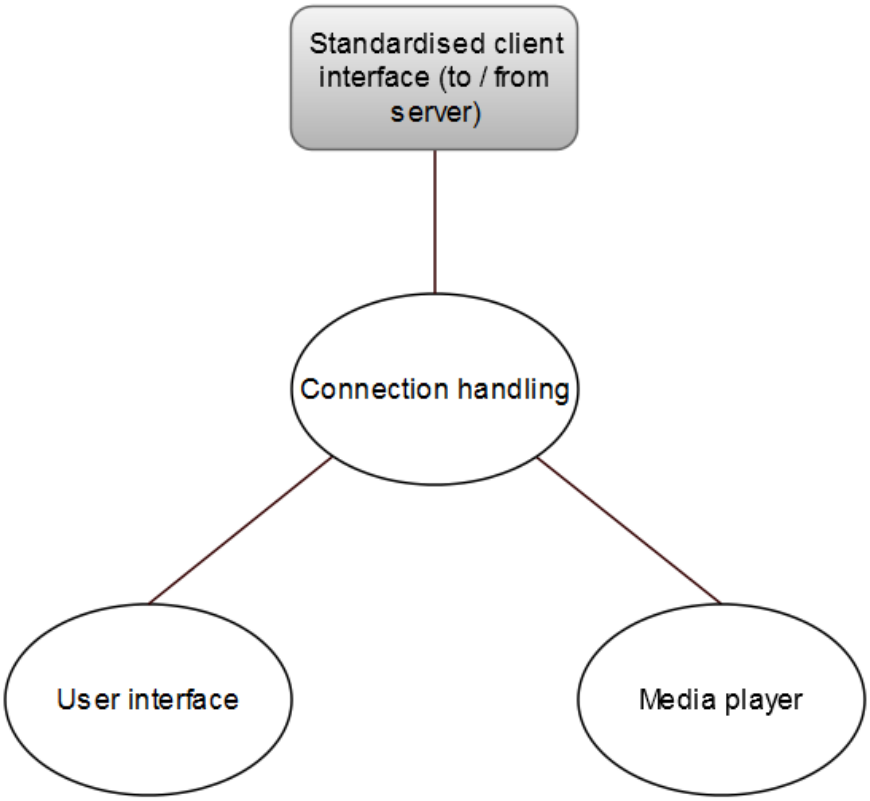


Figure 5: Diagram of the client component

1.8.2. Synthesis

1.8.2.1. Analysis structural model

From the domain analysis conducted above, the Open University's TM354 module recommends building an analysis structural model, which is concerned with "the structure of a software solution", e.g. defining the implementation of the previously suggested architecture (The Open University, 2015c).

1.8.2.1.1. Server component

Client interaction

Unit 8 of the Open University's M362 module analysed and reviewed a number of ways to communicate over a network.

HTTP GET is the recommendation for safe and idempotent queries (The Open University, 2008a, unit 8, pp. 11-20), such as the ones needed for querying the server component.

Java offers a native HTTP implementation via the sun.net.httpserver library (Oracle, 2016e), and HTTP being a familiar and widely used protocol, it does not seem justified to consider alternatives.

After having accepted the connection, the client query is processed by the ClientConnector class, which can be found in appendix 4.9.2.

Database interaction

The choice of MongoDB as a solution for persistence (as opposed to a traditional, relational, database) was discussed in the Account of related literature section. MongoDB offers a Java API (MongoDB, 2016), which was used for the interaction with the database, resulting in a simple class only supporting the necessary methods, presented in appendix 4.9.3.

MongoDB will persist individual documents as shown in code excerpt 1 below:

```
{
        "_id" : ObjectId("57c6d0019041ea035eec3c0b"),
        "xmpDM:genre" : "Podcast",
        "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
        "creator" : "BBC Radio 3",
        "xmpDM:album" : "Late Junction Sessions",
        "xmpDM:releaseDate" : "2011",
        "meta:author" : "BBC Radio 3",
        "xmpDM:artist" : "BBC Radio 3",
        "dc:creator" : "BBC Radio 3",
        "xmpDM:audioCompressor" : "MP3",
        "title" : "LJS 23 Jun 11: Hans-Joachim Roedelius and Christopher
Chaplin",
        "xmpDM:audioChannelType" : "Stereo",
        "version" : "MPEG 3 Layer III Version 1",
        "xmpDM:logComment" : "eng - \nA collaborative session by German
ambient music pioneer Hans-Joachim Roedelius and British composer
Christopher Chaplin",
        "xmpDM:audioSampleRate" : "44100",
        "channels" : "2",
        "dc:title" : "LJS 23 Jun 11: Hans-Joachim Roedelius and Christopher
Chaplin",
        "Author" : "BBC Radio 3",
        "xmpDM:duration" : "1299606.375",
        "Content-Type" : "audio/mpeg",
        "samplerate" : "44100",
        "fileName" : " LJS 23 Jun 11",
        "filePath" : "smb://tm470Server/Test/Test/Subdirectory/",
        "fileType" : "mp3",
        "hash" : "76262befcb34d956c39e47556efc052b"
}
```


Code Excerpt 1: Representation of a JSON object

Samba interaction

Samba is a ubiquitous protocol for making file shares available over a network. It is well maintained, offers both high performance, and high compatibility, and is the protocol I have traditionally used to connect to network shares (as opposed to alternatives, such as AFS, NFS, HDFS or AFP, implementation of which remains beyond the scope of this project).

The JCIFS library (The JCIFS Project, 2014) appears to be a mature and well-documented component that allows Java software to interact with samba shares. Since it fulfils all the requirements for interacting with the samba share in question, I did not find it necessary to evaluate further alternatives.

Indexing

Since no suitable third-party solution was found to take care of the indexing process, a simple algorithm, which needs not be exceptionally efficient, was composed from scratch, as demonstrated in code excerpt 2 (the complete class can be found in appendix 4.9.4).

Code Excerpt 2: The indexing algorithm

```
void storeData() {
        for (String ii : samba.populatePathSet() )
        {
             Document doc = new Document();
             Map file = splitPath(ii);
             […]
             MetaDataParser_tikaGeneric parser = new MetaDataParser_tikaGeneric();
             parser.decodeFile(filePath, doc);
             try
             {
                doc = addFileDetails(fileDetails, doc, filePath);
             }
             […]
```

Searching

With search performance being the particular focus of this project, it was necessary to evaluate the search algorithm from scratch.
An initial search algorithm, shown in code excerpt 3, proved too slow (see appendix 4.13.4 on performance testing):

```
         for (Document doc : data )
         {
             Set<String> setOfKeys = doc.keySet();
             for ( String key : setOfKeys )
             {
                 String val = doc.get(key).toString();

                 if (
 doc.get(key).toString().toLowerCase().contains(searchText.toLowerCase()))
                 {
                     result.add(doc);
                     break; // else we get multiple copies, unnecessary
                 }
             }
         }
```

Code Excerpt 3: Simple search

As I knew from the M269 module discussion on search algorithms (The Open University, 2014a, pp 72 ff), search was improvably slow because of the following aspects:

a) the nested `for` loop increased complexity by the power of 2

b) the unwrapping of the key/value pair in the BSON document demands resources, and would be faster if it was done linearly
c) the current algorithm would only use a single processor core

Hence, the revised implementation now:

a) Unwraps values into a single long string, which is held in a map with the UUID as key, thus reducing complexity
b) Splits the map of maps into equally sized chunks, based on the number of processor cores for concurrent processing.

The resulting search flow is presented in figure 6:

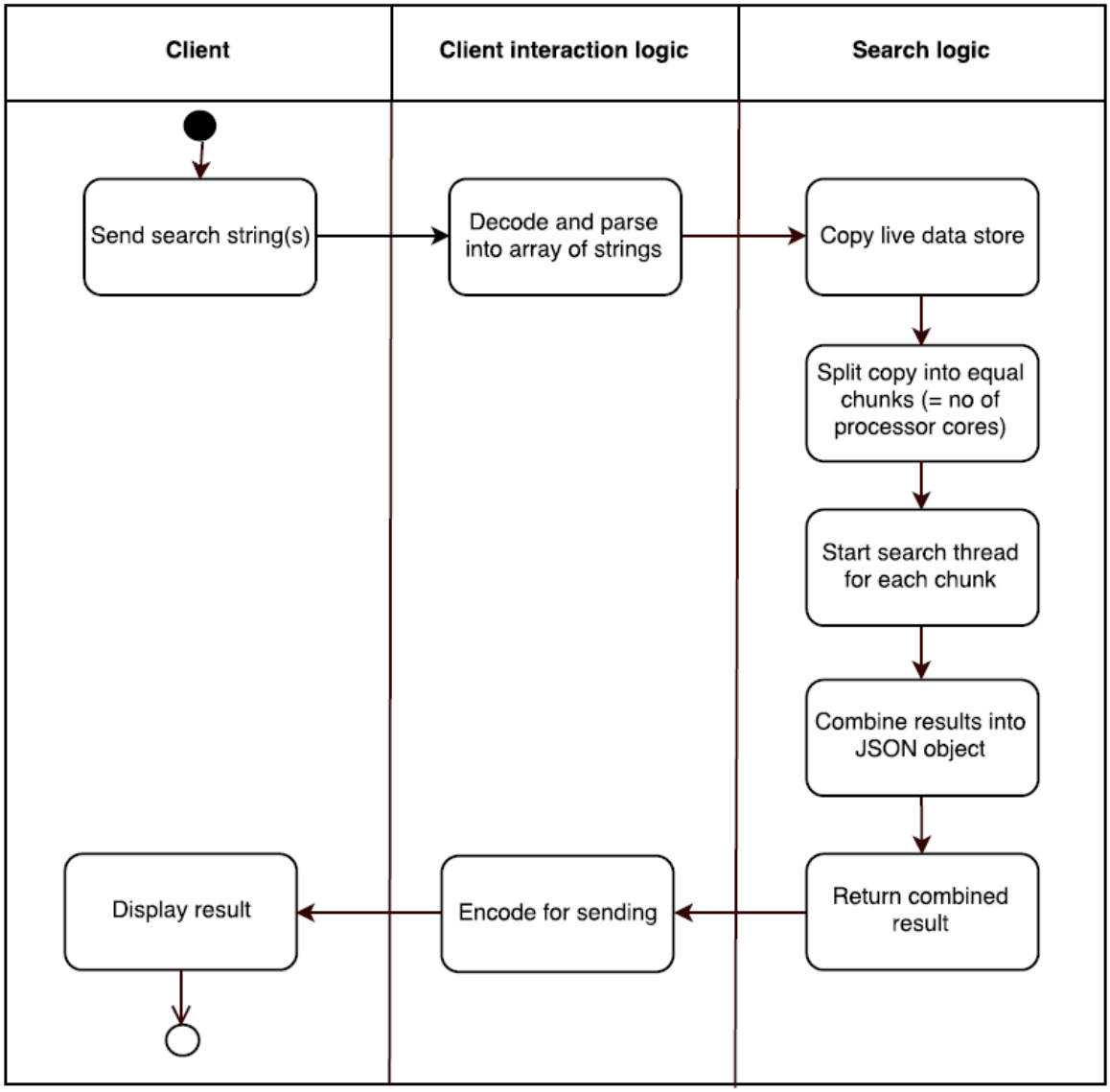


Figure 6: Search workflow

Implementing the flow illustrated, we end up with the code in code excerpts 4 & 5 (full class in appendix 4.9.1).

First, the data is split:

```
private void splitData() {
    int ii = 1;
    int multiplier = 1;
    Map<String, String> temp = new HashMap<>();
    for ( Map.Entry<String, Document> entry : slowDocMap.entrySet() )
    {
        temp.put( entry.getKey(), entry.getValue().toString() );

        if ( ii >= sectionDivisor * multiplier || ii == slowDocMap.size() )
        {
            fastDocList.add( (multiplier -1), new HashMap<>(temp) );
            multiplier++;
            temp.clear();
        }
        ii++;
    }
}
```

Code Excerpt 4: SpeedyGonzales multithreaded search – splitting data

Then, each chunk is submitted for processing, and finally re-united:

```
for ( int ii = 0; ii < fastDocList.size(); ii++ ) {
final int jj = ii;
threadList.add( executor.submit(new Callable<List<Document>>() {
    @Override
    public List<Document> call() throws Exception {
    List<Document> resultList = new ArrayList<>();
        try {
            Map<String,String> quickMap = fastDocList.get(jj);
            for (Map.Entry<String,String> entry : quickMap.entrySet() )
            {
                boolean found = false;
                for ( String substring : searchTerms ) {
                 if ( entry.getValue().contains(substring) )
                    {
                        resultList.add(slowDocMap.get(entry.getKey()));
                        break;
                    } }
            } }
        catch(Exception e){ […] }
        return resultList; } }) ); }
executor.shutdown();
for (Future<List<Document>> entry : threadList) {
    result.addAll(entry.get());  }

for (List<Document> entry : searchResult) {
    result.addAll(entry); }
return result; } }
```

Code Excerpt 5: SpeedyGonzales multithreaded search – processing data

As the performance analysis in appendix 4.13.4 shows, despite considerable overhead because of unwrapping the BSON objects and splitting them into searchable chunks, search performance improved by 14% to 38% using the “SpeedyGonzales” approach (see appendix 4.9.1) on the same hardware.

Live data store

The data store needs to be no more than an in-memory representation of the BSON objects stored in the MongoDB database (memory usage is not of concern for this project, but it can be said that the test server worked well with 2GB of memory).

All operations are always based on the UUID MongoDB assigns when storing an object.

To speed up further processing, the UUID of the actual BSON document was used to create a Map singleton (shown in code excerpt 6) storing the UUID, and its associated document.

```
private static Map<String,Document> MAP_OF_DOCS = new LinkedHashMap<>();
```

Code Excerpt 6: BSON map instantiation

The `Document` class (part of the MongoDB API) is a representation of the BSON objects stored in the MongoDB database, which can be accessed and modified much like an ordinary map. Its integration is provided by the MongoDB java driver.
Various methods are provided to access the map, which is populated by an 'init' mode the software can run in, or periodic maintenance (see appendix 4.9.6), which adds or removes documents on the fly.

File and stream handling

A stream is really no more than a large file, sectioned up into smaller individual files, which are sent one after another (see media transcoding below).
Thus, all that is needed is an implementation that can send a file via HTTP, which is achieved by reading the source file from the samba share into a Java byte array, and writing that byte array back to the HTTP socket, as illustrated in code excerpt 7.

```
byte [] bytearray  = new byte [(int)smbFile.length()];
[…]
ostream.write(bytearray,0,bytearray.length);
```

Code Excerpt 7: File handling - writing data to socket

The type of file described in the file header is recorded during indexing (for example "`audio/mpeg`" for files of type mp3), which will tell the client how to handle the file received.

The realisation of file delivery (which is represented in appendix 4.9.5), like the handling of client connections is handled in a multi-threaded fashion by implementing the Java `Runnable` interface, as described in the 'Chat room server example' (The Open University, 2008, unit 8, pp 39 ff.).

Media transcoding

Apple requires any apps handling streamed media files above a certain size (Apple, 2016i) to be delivered via HTTP Live Streaming, or, shorter, HLS (Apple, 2016h). Apple provides its own transcoding software, which unfortunately only runs on macOS (which itself is a derivative of the BSD Unix fork), not Ubuntu (my OS of choice, a fork of Linux).
An (excellent) alternative is to use FFmpeg (FFmpeg, 2016), an open source project dedicated to encoding and transcoding media files.
Although an FFmpeg Java cli wrapper is available (FFmpeg Java, 2016), its use requires implementing additional libraries, and does little to aid the implementation of my requirements.
Hence, I opted to call FFmpeg using the Java ProcessBuilder service (Oracle, 2016f).

With reuse in mind, the source file is supplied using the generic file interface of the server component, as shown in code excerpt 8:

```
sourceFile = "http://127.0.0.1:" +
properties.getProperty("httpServerPort") + "/file/" + objectId;
```

Code Excerpt 8: HLS encoding - fetching the source file

Code excerpt 9 shows an array of the necessary command line options, which is created depending on the desired transcoding type:

```
List<String> command = new ArrayList<>();
        command.add(ffmpegPath);
        command.add("-i"); command.add(sourceFile); //input file
        command.add("-map"); command.add("0"); // which stream to encode
        command.add("-f"); command.add("segment");
        command.add("-acodec"); command.add(audioCodec);
        command.add("-segment_list_type"); command.add(segment_list_type);
        command.add("-segment_time"); command.add(segment_time);
        command.add("-segment_format"); command.add(segment_format);
        command.add("-segment_list_entry_prefix"); command.add(baseUrl);
        command.add("-segment_list"); command.add(segment_list);
        command.add(outputFileTemplate);
```

Code Excerpt 9: Building the FFmpeg command structure

And code excerpt 10 demonstrates how a new FFmpeg process is created and started:

```
        ProcessBuilder procBuil = new ProcessBuilder(command);
        Map<String, String> environ = procBuil.environment();
        Process process = procBuil.start();
```

Code Excerpt 10: Starting the transcoding process

Once the playlist exists, the client can start streaming, while the FFmpeg process continues transcoding the file.

```
String expectedPlaylist = properties.getTempDirectory() + "/" + objectId +
".m3u8";
int count = 0;
            while(!expectedPlaylist.exists() && count <= 500)
            {
                Thread.sleep(50);
                count++;
            }
                   if (count > 499)
            {
                new ErrorSender("Could not find playlist", socket,
ostream).sendError(); // To client

                LOG.ERROR("Could not find playlist for id " + objectId); // To log
file
                return; // Prevent locking
            }
```

Code Excerpt 11: Delivering the HLS playlist

File types

Nep-Tune uses the Tika 1.13 library (Apache, 2016c) to support several 100s of file types out of the box - the main reason the Tika library appealed for this project.
Nep-Tune is designed to make it easy to add support for additional file types. To do so, one can simply add a class for it, and intercept parsing in code as highlighted in code excerpt 12:

```
// Check file extension for custom file types
        if (false /* add custom file type here */ )
        {

        }
```

Code Excerpt 12: Adding custom file types for parsing

Samba interaction

Samba interaction consists of a very simple class, which implements the JCIFS (2014) library. It can make a connection to the samba share specified in the configuration file, return individual files for parsing and sending to the client, and further produce a map of the directory structure, which is used when the index is created initially.

Maintenance

The Maintenance class (see appendix 4.9.6) has two purposes:

1) It periodically scans the samba share for file changes, using a relatively efficient hash algorithm, allowing it to add / remove files to / from the index that have been added / removed to / from the samba share, and re-parsing files the metadata or content of which have changed. Code excerpt 13 shows the code responsible for adding new files, and updating files the properties of which have changed:

```
void addEntryIfWasAddedToSambaOrIfExistingEntryMatchesFile() throws
NoSuchAlgorithmException, SmbException
    {
        Map<String,Document> existingEntries = ds.getMap();
        for (String ii : samba.populatePathSet() )
        {
            boolean found = false;
            String entryToRemove = null;
            String path = null;
            String file = null;
            String type = null;
            for ( Map.Entry<String, Document> entry :
existingEntries.entrySet() )
            {
                path = (String) entry.getValue().get("filePath");
                file = (String) entry.getValue().get("fileName");
                type = (String) entry.getValue().get("fileType");
                String filePath = path + file + "." + type;
                if (ii.equals(filePath))
                    {
                        found =
compareFiles(entry.getValue().get("hash").toString(), ii);
                    }
                    entryToRemove = entry.getKey();
                    break;
                }
            }
            if (!found)
              [ add to index ]
        }
    }
```

Code Excerpt 13: Maintenance – adding and updating the index

2) It cleans up temporary files (which are mainly produced by the HLS transcoding process) periodically, and on shutdown.

Installation script

Having settled on using FFmpeg (2016) for transcoding media files, I soon realised FFmpeg does not build on CentOS 7.2 (see Scholz, 2016b for details), prompting me to switch to Ubuntu. On switching operating system, it became clear that the set-up process had become sufficiently complicated to warrant automation.

The resulting bash script (complete script in appendix 4.12) prompts the user to enter configuration details, installs dependencies, writes the various files required by Nep-Tune, and configures a service for systemd.

For example, below code excerpt 14 prompts the user to enter a password (which remains obscured from view):

```
function getPassword {
read -s -p "Please enter the password for the samba share`echo $'\n> '`" passWord1
read -s -p "Please re-enter the password`echo $'\n> '`" passWord2
}
getPassword
while [ "$passWord1" != "$passWord2" ]
do
echo Passwords do not match, please try again
getPassword
done
sambaPassword=${passWord1}
```

Code Excerpt 14: Installation script - password query

The script is based on prior work experience, where bash is used extensively.

Utility classes

The simplicity of the utility classes in this project means their inclusion here is not warranted.

1.8.2.1.2. Client component

The client component makes use of the Model-View-Controller (MVC) principles introduced in the Open University's TM354 module (2015d), which is to "split the user interface interaction into three distinct roles: the model of the domain, the view representing that domain, and the controller of changes to the domain.".

Graphical User interface

The basis for the client component was provided by the UISearchController Tutorial by Nicolas Martin et al. (2015). Since the user interface is designed and maintained in the storyboard xml file in XCode (see code excerpt 15), separation of user interface and underlying control mechanism are given. The DetailViewController gained a play button and volume slider, while overall, the colour scheme and logo were changed:

```
<color key="barTintColor" red="0.24313725490196078" green="0.23921568627450981"
blue="0.53725490196078429" alpha="1" colorSpace="calibratedRGB"/>

<imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES"
image="Nep-Tune.PNG" translatesAutoresizingMaskIntoConstraints="NO" id="p6J-Tb-
Q0Q">

<rect key="frame" x="-54" y="0.0" width="349" height="40"/></imageView>
```

Code Excerpt 15: Nep-Tune logo and colour

This xml file represents the 'View' part of the component.

Server communication

The main class to interact with the application, the MasterViewController gained the ability to talk to the server component to fetch the resulting JSON object, and further process this JSON object as shown in code excerpt 16:

```
    func retrieveFilteredResultsFromServer(searchText: String) {
        let startTime: NSTimeInterval = NSDate().timeIntervalSince1970

        if (searchText.characters.count > 1) {

            var searchURL: String = baseURL + "search/%22" +
searchText.stringByAddingPercentEncodingWithAllowedCharacters(.URLHostAllowedCharact
erSet())! + "%22"

            // This assumes that "client" is rarely used
            if searchMode == "client" {
                searchURL = baseURL + "data/"
                print("In client-only search mode")
            }
            json = getJSON(searchURL)
            let returnedData = parseJSON(json)!
            populateIndexWithResults(returnedData)
            print("Server returned \(returnedData.count) documents")

        } else {
            populateIndexWithResults (["Enter 2 or more characters to start
searching", ""])
        }

        let endTime: NSTimeInterval = NSDate().timeIntervalSince1970
        let timeTaken = (endTime - startTime) * 1000
        print("Search string \(searchText) took \(timeTaken)ms to return")

    }

    func getJSON(urlToRequest: String) -> NSData? {
        return NSData(contentsOfURL: NSURL(string: urlToRequest)!)
    }


    func parseJSON(inputData: NSData?) -> [Dictionary<String,AnyObject>]? {
        do {
            if let jsonArray: NSArray = try
NSJSONSerialization.JSONObjectWithData(inputData!, options:
NSJSONReadingOptions.MutableContainers) as?
                NSArray {

                    if let swiftArrayOfDictionaries: [Dictionary<String, AnyObject>]
= jsonArray as? [Dictionary<String, AnyObject>]
                    {
                        return swiftArrayOfDictionaries
                    }
            }
        } catch let error as NSError {
            print(error.localizedDescription)
        }
        return nil
    }
```

Code Excerpt 16: Retrieving data from the server

The Master- and DetailViewControllers represent the ‘Control’ part of MVC.

Media player

The media player implementation, part of which is shown in code excerpt 17, is built on the Apple AVPlayer framework.

It fetches the media files from the server (e.g. the playlist, which describes the components of the stream).

```
func preparePlayer(baseURL: String, objectId: String) {

        url = NSURL(string: "\(baseURL)mediafile/\(objectId)")

        if let fileURL = url {
            player = AVPlayer(URL: fileURL)
            let playerController = AVPlayerViewController()
            playerController.player = player
        } else {
            […] // Error handling
        }
    }
```

Code Excerpt 17: Retrieving media file from the server

Furthermore, it can start and stop playback, as illustrated in code excerpt 18:

```
func play() {
        if (player == nil) {
            preparePlayer()
            player!.play()
        } else {

        if (isPlaying()) {
            player!.pause()
        } else {
            player!.play()
            }
        }
    }
```

Code Excerpt 18: Player functions

Lastly, the class contains some other functionality, such as changing the volume.

The Audio player forms part of the ‘Model’ of the MVC principles, although most of the model can be found in the server component.

1.8.3. Evaluation

To ensure the software fulfills the requirements elicited during the analysis stage, a set of functional tests was designed, following the recommendations for software testing in TM354, "Unit 11 – Strategy for creating test cases" (The Open University, 2015g). Details can be found in appendix 4.13.

Although problems were encountered on the way (for example, metadata for comments was not correctly parsed at some point), the server component has no known issues at the time of writing.

However, functional testing revealed that a number of issues present in the client component, which are summarised in the table of known issues in appendix 4.15.
Since the main purpose of the client component was to showcase the capabilities of the server component, addressing the observed issues was not considered vital for completion of the project, but will be attended to after submission of this paper.

1.8.3.1. Performance analysis & non-functional testing

Since the client component of this project is an iPhone app, comparative testing ought to be performed against other iPhone apps of similar functionality. Therefore, a performance analysis compared to the apps identified in appendix 4.11 would have been an ideal test scenario. However, this was not possible, for reasons summarised in table 2.

Name	Reason cannot be compared against
Airplayer	Cannot search beyond folder / playlist
Kodi	Cannot search beyond folder / playlist
OPlayer	No search functionality
Player Xtreme	Can only search iPhone local library
Twonky	No longer starts up on iOS 9

Table 2: Reasons why Nep-Tune performance cannot be compared to similar mobile applications

In conclusion, no available iPhone app can, to my knowledge, perform search similar to the project software.

To provide some benchmark value, I have included the search performance measurements of the built-in Windows Explorer, and Apple's OS 10.10 Finder.

Averaging the results of the performance tests on a set of approximately 15,000 files presented in tables 9 & 10 in appendix 4.13.4 yields the values shown in table 3, while the technique of taking the actual measurements is described in appendix 4.13.3.

	Nep-Tune Server mode (SpeedyGonzales)	Nep-Tune Server mode (simpleSearch)	Nep-Tune Client mode	Windows 8.1 Explorer	Apple 10.10 Finder
Average time taken	265ms	330ms	3139ms	5519ms	1640ms
Percentage change	100% (benchmark)	124.5%	1184%	2082%	618.9%

Table 3: Averaged performance results

1.8.3.2. Test results evaluation

The requirements (see appendix 4.7.3) found that "the search shall return within 500ms for a collection smaller than 10,000 files in 99% of test cases on the test system".
A result of under 300ms for a set of nearly 15,000 entries, as shown in table 3, comfortably exceeds this target.
Furthermore, the power of the multi-threaded search approach becomes apparent, when comparing the multi-threaded search to the simple search implementation – a performance gain of approximately 25% can be observed, despite the additional overhead of sectioning the data.

Performing the identical search on the client alone takes more than ten times longer, a spectacular result illustrating how much weaker mobile processors still are, compared to their server counterparts. The results for Windows Explorer highlight the drawbacks of linear parsing – the file early in the file structure (the Outlook podcast, BBC, 2012) was found in less than one second, whereas the file at the end of the file structure took 10 seconds to find. Both results are significantly slower than the Nep-Tune implementation.

The real surprise came when using Apple OS X Finder. While the Outlook podcast was found in a time comparable to Windows, the Science Weekly podcast was never found. I left the search continuing for eight hours in one attempt (there is no visual indication whether a search has completed), and was still faced with an empty results window.

To enable prediction of performance for different sizes of collections, I applied Microsoft Excel's FORECAST function (Microsoft, 2016) to extrapolate the graph in figure 7 from the measured values recorded in table 11:

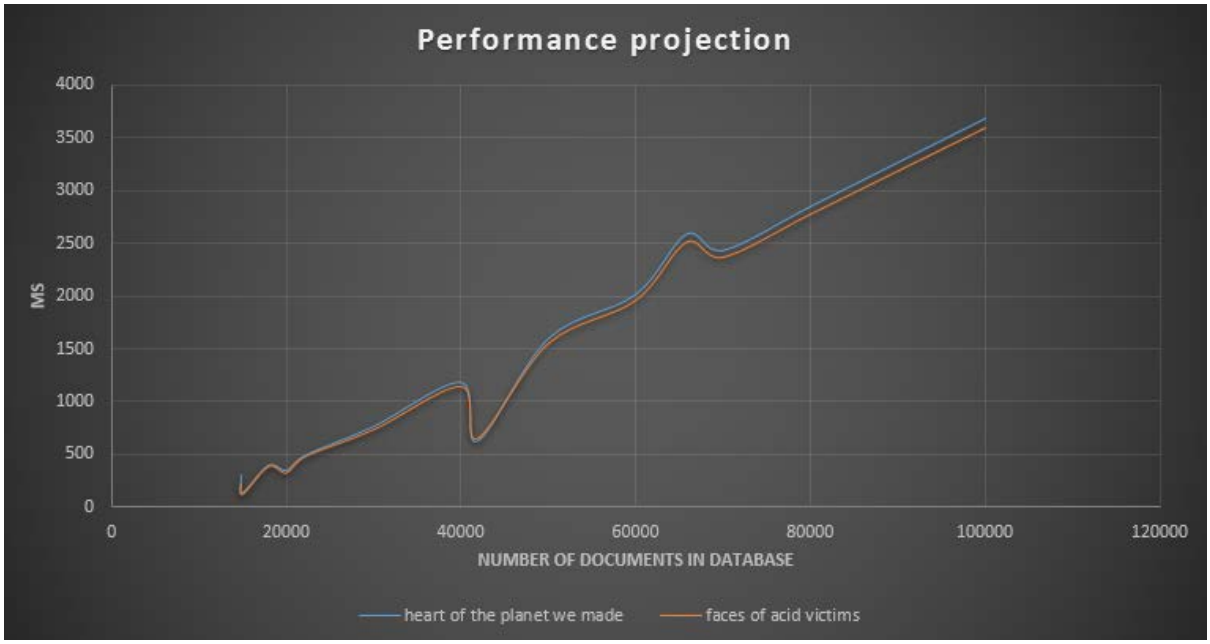


Figure 7: Nep-Tune performance projection for large collections (milliseconds vs number of documents in the collection)

The graph shows that, with the same hardware, search time correlates approximately linearly to the size of the collection.

From the 25% performance increase between SpeedyGonzales and simple search in tables 9 & 10, it can be deduced that the parallelisable part of the program is approximately 50% providing an input for Amdahl's famous law to describe performance characteristics of multi-processor systems ($S_{latency}(f,S) = \frac{1}{(1-f)+\frac{f}{s}}$) (Amdahl, 2013, p 45), where f is the parallelisable part, (1-f) represents the serial part, and s the number of processor cores.

Plugging those figures into Amdahl's equation, we end up with $S_{latency} = \frac{1}{(1-0.5)+\frac{1}{[1,24]}}$ resulting in the graph in figure 8:

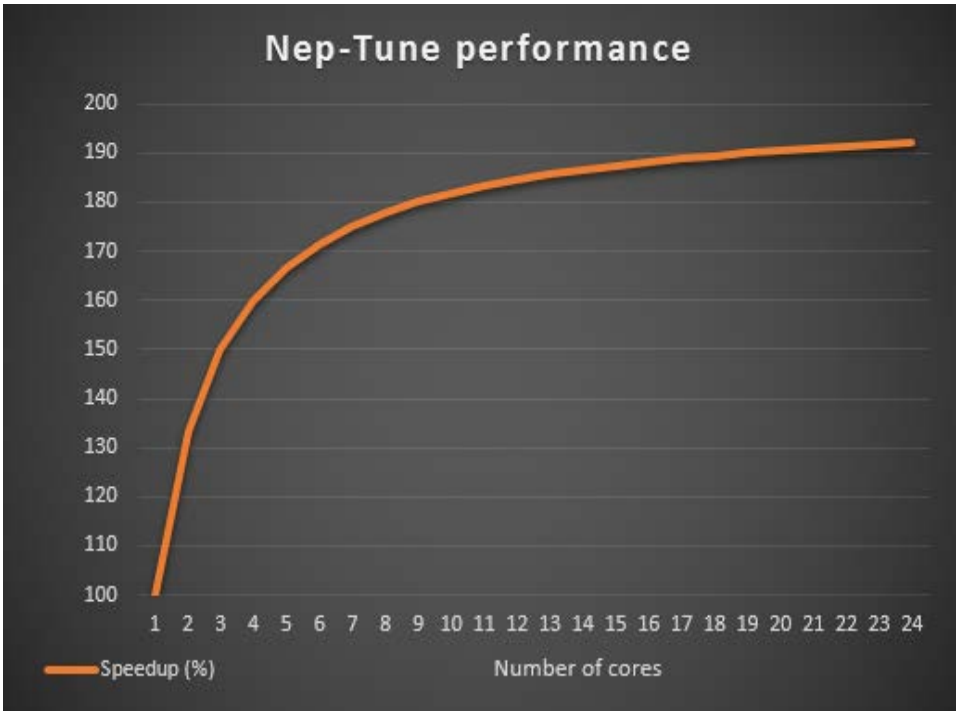


Figure 8: Nep-Tune projected performance (single-core performance is benchmark)

However, since the parallelisable part of the program is a function of the size of the collection, we can make the following observations:

1) The larger the collection, the smaller the serial fraction (1-f) becomes
2) The serial part of the program does not grow significantly with an increased number of available processors
3) The number of parallelisable chunks (equating to the number of documents in the collection) will, in practice, always be greater than the number of available cores (at the time of writing).

In summary, this means that, the larger the indexed data set is, the more Nep-Tune will benefit from additional hardware resources.

Overall, the results testify an immense success for the Nep-Tune project.
Not only does Nep-Tune open up large file structures to devices with low processing power (such as mobile phones), but it also beats the two incumbent desktop operating systems at their own game.

1.8.3.3. External evaluation

James Hatton, a fellow TM470 student, kindly offered to give me feedback (Hatton, 2016) on the application in its current state.

Various aspects of the project were discussed during the meeting, such as the potential of the server component in James' realm of work (vehicle management), as well as its potential impact on existing solutions.

The overall feedback James gave was very positive, and a list of feature suggestions was compiled during the process, with items such as recognition for binary data (e.g. tell whether an image is green or blue by looking at it), semantic understanding of phrases, and a bias that values certain information more highly than other. Table 11, "Feature suggestions", is presented in appendix 4.14.

1.9. Review of current stage of project work

1.9.1. Server component

The server component is feature-complete according to the initial requirements, and free of (known) errors. It has been publicly released on Bitbucket (Atlassian, 2016), and I have received a request to evaluate its deployment in a commercial setting (Dunn, 2016).

Encrypted communications will be necessary – this relatively mundane feature was deliberately left out of the schedule of lifecycle phases in appendix 4.5 (as no special skills are required, yet setting up the infrastructure will take some time), and should be complete by the end of the year.

More features, such as the processing of streamed data for "Big Data" applications, search bias, and recognition of binary data are not currently planned, and will be considered when a clear need arises.

1.9.2. Client component

I myself am the biggest fan of the client component, and have a vested interest in developing it further, as I already use it regularly.

In its current stage, it is still quite far away from being accepted into the Apple app store, but the goal is to achieve this within a year or so.

To make acceptance possible, the list of known issues in appendix 4.15 will need to be addressed, and presumably further issues will arise that are going to lead to one or more rejections on submission.

Besides personal use, its main purpose was to showcase a possible implementation of a client to complement the server component, and it is likely that another, simpler, client component will be developed, for example a web interface or desktop application.

1.9.3. Academic work

The structure of the module, which means effectively submitting three marked draft versions of this paper before submitting the paper itself, is very much a double-edged sword in my view.

On the upside, the previous submissions provided useful guidance for improving the workflow for producing the report, as well as valuable hints for the flavour of submission expected.

On the downside, the poor marks received for previous submissions seem to push the possibility of achieving an overall distinction out of reach (in contrast to previously completed modules), which had a negative impact on motivation.

Looking forward, I have already applied to begin advanced studies in computer science later this year, and the project has provided great inspiration for the ensuing research project.

Together with the skills acquired, it must be acknowledged that producing this paper was a very

valuable academic project, which will have a lasting impact both on my future academic, as well as professional career as a software developer.

In summary, I believe that, by producing fully working software, the project work has successfully addressed all of the core aspects of the problem.

1.10. Review of project management

The initial schedule had not allowed sufficient buffer time to cope with unforeseen requirements and events (for example the necessity to use HLS for media transfer, FFmpeg not building on CentOS 7 prompting a switch to Ubuntu, and illness), a mistake to be avoided in future projects.
While ultimately some client features were sacrificed compared to the original plan, the chosen *Iterative enhancement* lifecycle model (Basili et al., 1975, pp. 390-396) allowed me to produce an overall working solution on time, with a server component which is feature-complete and free of (known) errors, thus proving an excellent choice for the project.
Having identified all the resources, skills, and activities needed to complete the project in a timely and successful manner during the analysis stage described in section 1.8.1, continuously using a variety of sources, which were carefully judged for significance and credibility in section 1.6, I was able to succinctly identify the contributory role to the project of each, and thus, despite problems along the way, was able to follow the orginal schedule closely, mitigating all but one of the risks identified in table 6 in appendix 4.10 on time for project completion.
The boundaries and expected outcome of the project were clearly set, core aspects of the problems addressed have not changed since inception, and the problem description was revised to enhance clarity and increase definition.

1.11. Review of personal development

Having approached the project with some years of experience as a C++ developer in the financial sector, it greatly contributed to advance my skills in several areas, including those of project management, report writing, and aspects of software development.
Some useful patterns have evolved that reflect on how I learn and work most effectively, for example setting small development challenges for evenings, while leaving academic and learning tasks for longer stretches of time, such as weekends. In the process, some weaknesses were uncovered, too, most notably that reading academic papers sent me on a trail to find out more related information, most of which was not relevant to the project, costing valuable time (and producing a sizeable bibliography).

1.11.1. Scheduling

The project taught me techniques, such as setting personal milestones and better estimation of the duration of tasks, and its self-motivated nature meant I was working under my own supervision, which overall required improvement of self-discipline and time management (assisted by regular

communication with my tutor in the form of project logs and emails (see appendix 4.16 and 4.17 respectively)).
It particularly highlighted the necessity of buffer time to cope with events such as unforeseen requirements (as discussed above), while at the same time triggering some "out-of-the-box" thinking (i.e. overcoming the lack of file locking support by looking at modification time).

Future projects will benefit from having identified both examples of effective, and wasteful work, and from considering how the setbacks of this project were managed (in particular by planning for alternative approaches early on, and measuring progress).

1.11.2. Report writing

Many aspects of adequate report writing were unclear to me at the start of this project. Thanks go particularly to Charly Lowndes for readily answering my questions. Examples include the question of the appropriateness of time measurement tools for an academic paper ("a stopwatch should be fine" (Lowndes, 2016c, 29 April)), and the extent of describing skills acquisition in the report ("…acquiring skills along the way can be summarized more briefly", (Lowndes, 2016b, 29 April 2016)). Selected email exchanges can be found in appendix 4.17.

Overall, I feel I have gained a much clearer understanding of how to structure reports, which elements are important to include in detail, and which can be touched upon more lightly.
In particular, the resources identified in section 1.6 and 1.8 helped me to justify opinions and judgements well, which I learnt to communicate concisely using diagrams, illustrations, code excerpts, and arguments where appropriate.

The project logs (appendix 4.16) were an extremely useful tool for structuring my report in order to communicate the approach I took to certain issues and problems, as well as helping me to keep to the schedule.

1.11.3. Software development

At the beginning of the project, I had no experience with Swift, and was only vaguely familiar with Java through previous university modules.
Specific achievements include:

1.11.3.1. Java

Through my work with the server component, I now feel confident that I am able to write moderately complex applications in Java, including using features new to the latest version of Java (8), such as Lambda expressions and streams, an example of which is shown in code excerpt 19.

Example from HLSCreator.java

```
command.stream().forEach((String string) -> {LOG.DEBUG(string + " ");} );
```

Code Excerpt 19: Java lambda expression and stream

I consider it an enormous success that the server component not only implements all requirements elicited at the beginning of the project, but is also free of known issues (at the time of writing).

1.11.3.2. MongoDB

MongoDB's rapid growth in popularity (see appendix 4.21) makes it an important cornerstone in many contemporary IT projects.

Learning how to integrate MongoDB's Java library into an autonomous piece of software (for which MongoDB's Getting started with MongoDB (Java edition) guide provided a great introduction (MongoDB, 2016)), as well as enhancing my dexterity in interacting with a MongoDB database appear to be an invaluable skill in today's digital realm.

1.11.3.3. Swift

The steepest learning curve was undoubtedly Swift, a new programming language used for iOS app development, which superseded Objective-C.

While I was looking forward to the challenge, I also realised that this was the biggest risk factor in the project.

Thankfully, I discovered the Stanford University lectures on developing iOS apps in Swift (Hegarty, 2015), which served as an excellent introduction to starting app development.

The next step was to build the client app, for which the UISearchController tutorial (Martin et al., 2015) provided a suitable template.

From here, Apple's *iOS Developer Library* (2016g) answered the vast majority of my questions.

Undeniably, there is still a long way to go before I can claim mastery of Swift, particularly in the area of GUI interaction, but I feel the foundations are now there, and it is my ambition to ready the client app for release on the Apple app store after submission of this paper.

1.11.4. Personal goals

I set myself the following goals at the beginning of the project, which I believe I have successfully achieved:

- ✓ write a complete, distributed application from scratch
- ✓ master the Java programming language
- ✓ introduce myself to Swift, the language used for Apple iOS devices
- ✓ deepen my understanding of NoSQL databases
- ✓ apply the agile development techniques taught in the Open University's TM354 module (The Open University, 2015f)
- ✓ enhance my understanding of developing concurrent, distributed systems, based on the Open University's M362 module (2008a)

1.11.5. Summary

The breadth and depth of new skills acquired give me confidence in my own ability to learn independently, while reflection highlighted some areas that could be improved upon, which will not only be useful for tackling similar projects in the future, but are also likely to transcend into other areas of my work and life.

I feel I have achieved my personal goals, and am very pleased that I was able to demonstrate not only technical skills, but also an enhancement of understanding in tertiary fields of expertise, such as archiving, and aspects of transmitting digital media over a network.

To extend the project further, additional skills in the implementation of cryptography, as well as a deeper understanding of GUI interaction in Swift are required. An implementation of the client on Android would be desirable, which opens up a partially new area of development all together (partially since Android apps are written in Java).

As a music college graduate, the combination of music and technology was always a hobby of mine, and it was an eye-opening experience to lift the lid on some of the technologies that I would have used as a black box prior to this project.

1.12. Epilogue

1.12.1. Client component enhancements

Since the above was written, I have further worked on the client. It is now using Apple's built-in AVPlayer layer, which brings the following benefits:

- video playback (shown in figures 9 & 10, and code excerpt 20)
- enhanced playback features (e.g. progress bar and seek, background playback, full screen support, etc.)

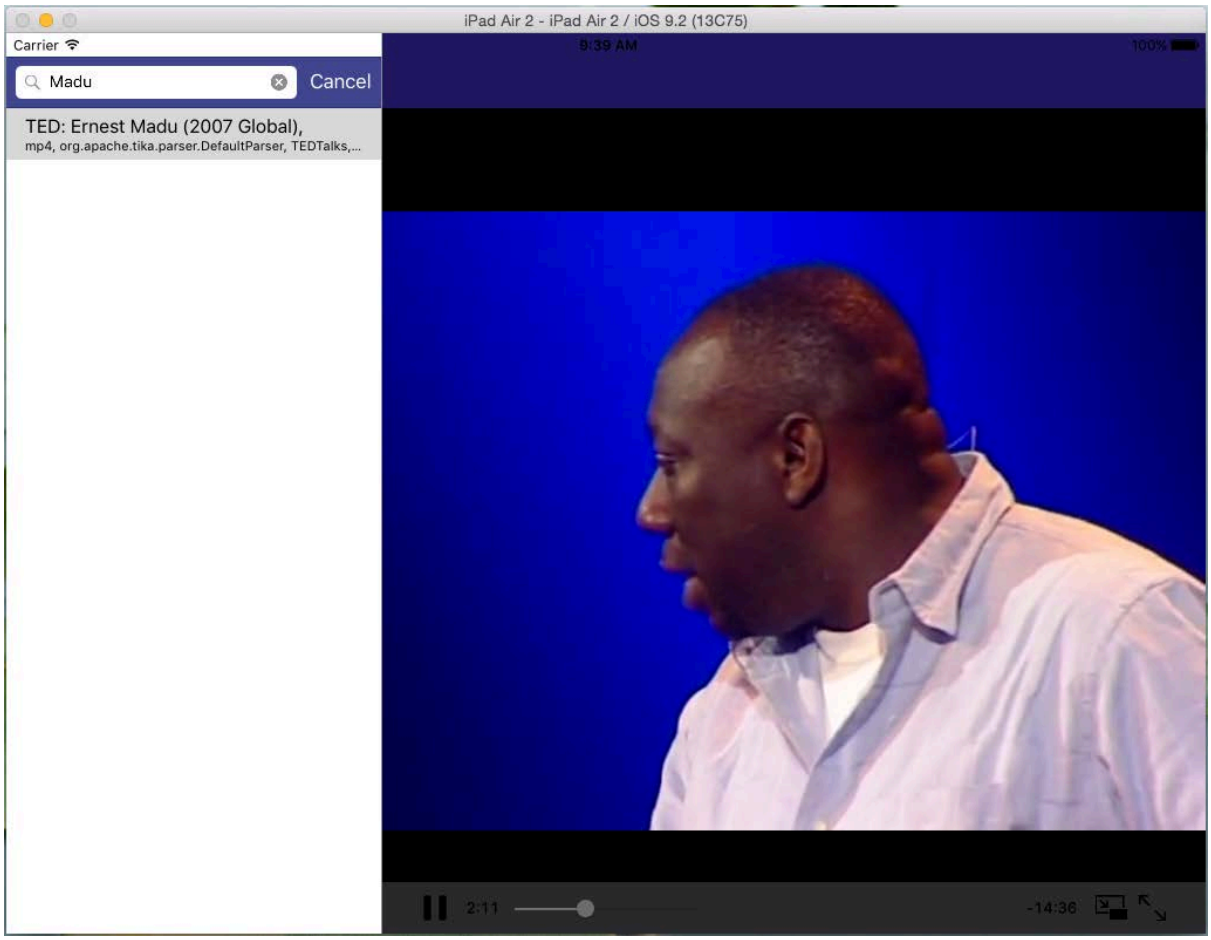


Figure 9: Video playing in split view (Madu, 2007)

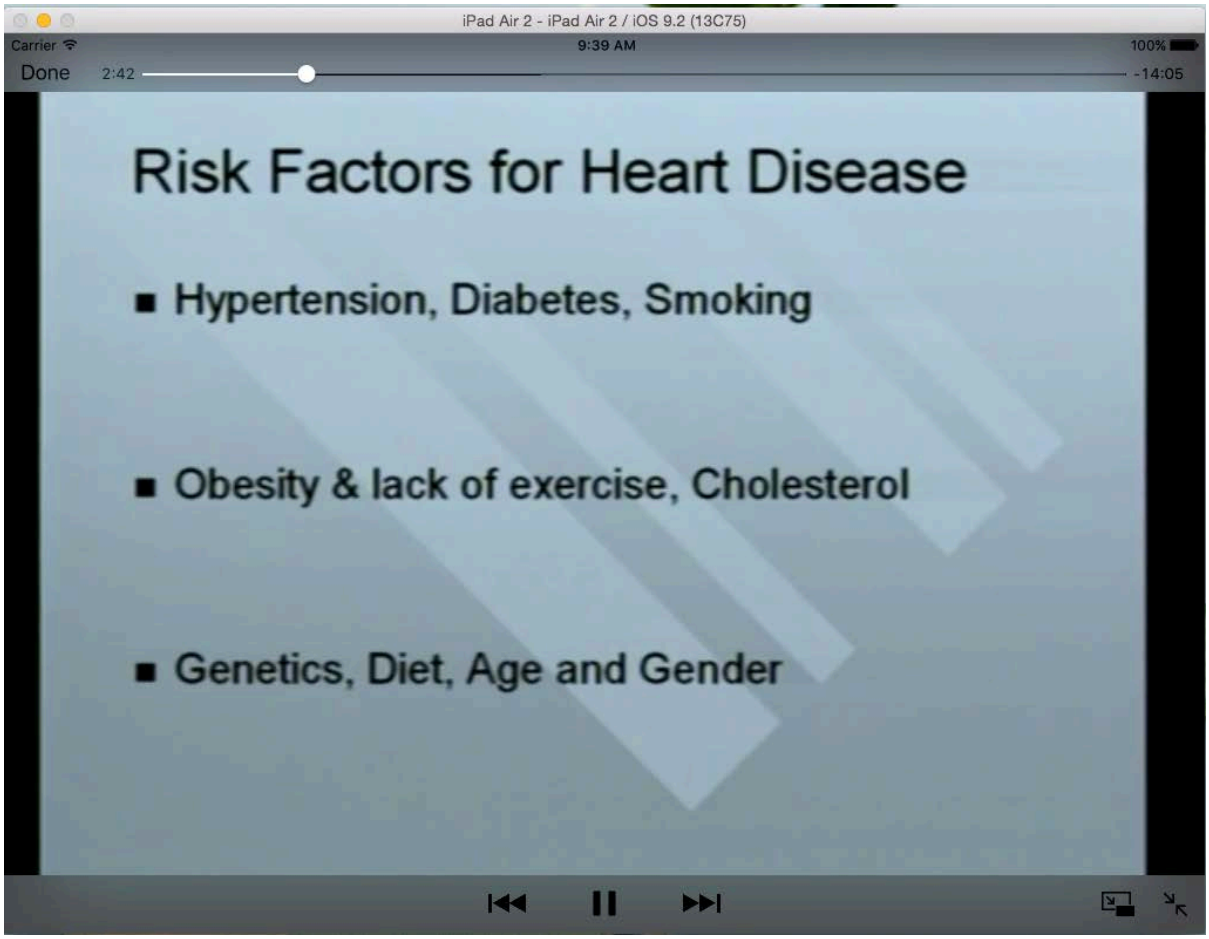


Figure 10: Video playing in full screen (Madu, 2007)

```
func setUpPlayer() {
        if let _ = detailIndex {
            if (detailIndex!.objectId != "nil" && detailIndex!.objectId != "" && baseURL != nil )
{
                objectId = detailIndex!.objectId
                print("DirectPlay is \(directPlay)")
                if ( directPlay! )
                {
                    url = NSURL(string: "\(baseURL!)mode=file&objectId=\(objectId)")
                }
                else
                {
                    url = NSURL(string:
"\(baseURL!)mode=playlist&objectId=\(objectId)&targetUrl=\(baseURL!)") // HTTP live stream
playlist
                }
                self.player = AVPlayer(URL: url!)
                self.avController = AVPlayerViewController()
                self.avController.player = self.player
                mediaLayer.frame = self.view.bounds
                avController.view.frame = mediaLayer.frame
                self.addChildViewController(avController)
                self.view.addSubview(avController.view)
                self.player.play()
            }
```

Code Excerpt 20: New AV player

By doing so, I have implemented phase 3.2.a (see appendix 4.5.3), which means only phase 3.2.b (locally persisted playlists), the very last (and thus least important) of the deliverable features proposed at the outset, was not achieved, which was owing to the unexpected requirement for implementing HLS (phase 2.2.e, appendix 4.5).

Also addressed were issues C-4 (Default entry can be selected for playback), and C-7 ("splitview doubles up detail view in portrait mode"), which were recorded in appendix 4.15.

1.12.2. Server component

1.12.2.1. Performance improvement

An improvement to search performance was found, which was achieved by replacing line 79 of the SpeedyGonzales class represented in appendix 4.9.1 with the code shown in code excerpt 21.

```
int kk = 0;
                    for (Map.Entry<String,String> entry : quickMap.entrySet() )
                    {
                        boolean found = false;
                        for ( String substring : searchTerms )
                        {
                        if ( entry.getValue().toLowerCase().contains(substring)
)
                            {
                                resultList.add(slowDocMap.get(entry.getKey()));
                                LOG.DEBUG("Run: " + jj + " Added " +
entry.getKey() + " to list" );
                                found = true;
                                break;
                            }
                        }
                        if (found && kk == quickMap.entrySet().size())
                        {
                            break;
                        }
                        kk++;
                    }
```

Code Excerpt 21: Performance enhancement to the SpeedyGonzales class

1.12.2.2. Encryption

Initial research has begun in enhancing the project software with encryption.

For the server component, there seems to be a choice between working with the 'com.sun.net.httpserver.HttpServer' library (Oracle, 2016g), creating a socket factory, or using the 'javax.net.ssl' library (Oracle, 2016h), which would integrate more smoothly with the code I have already written. In both cases, a self-signed certificate will need to be created.

For the client, I previously needed to explicitly allow unencrypted communications, and it seems everything required for encrypted communications is already provided by the application framework.

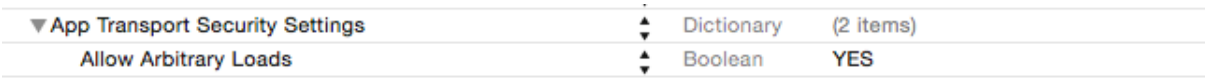


Figure 11: Setting in info.plist to allow unencrypted communications

2. REFERENCES

With agreement of my tutor, the numbering of references for identical authors (e.g. 2016a, 2016b …) is not sequential.

Apache (2016c) Apache Tika [Online]. Available at https://tika.apache.org/ (Accessed 06 August 2016). (See http://tika.apache.org/1.13/formats.html for supported file types.)

Amdahl, G. (2013) Computer Architecture and Amdahl's Law, *Computer*, Vol 46(12) pp. 38-46 [Online]. DOI: 10.1109/MC.2013.418 (Accessed 27 August 2016)

Apache (2004) *Apache License* [Online]. Available at http://www.apache.org/licenses/LICENSE-2.0 (Accessed 21 August 2016).

Apache (2016c) Apache Tika [Online]. Available at https://tika.apache.org/ (Accessed 29 August 2016)

Apache (2016d) 'SOLR', *Apache Lucene* [Online]. Available at http://lucene.apache.org/solr/ (Accessed 26 June 2016).

Apple (2016e) 'Distributing on the AppStore', *Apple* [Online]. Available at https://developer.apple.com/app-store/ (Accessed 6 April 2016).

Apple (2016f) 'Apple Developer Agreement', *Apple* [Online]. Available at https://developer.apple.com/programs/terms/apple_developer_agreement.pdf (Accessed 6 April 2016).

Apple (2016g) *iOS Developer Library* [Online]. Available at https://developer.apple.com/library/ios/navigation/ (Accessed 17 February 2016).

Apple (2016h) *HTTP Live Streaming* [Online]. Available at https://developer.apple.com/streaming (Accessed 12 June 2016).

Apple (2016i) *Technical Q&A QA1767* [Online]. Available at https://developer.apple.com/library/ios/qa/qa1767/_index.html (Accessed 12 June 2016).

Apple (2016k) *iOS Developer Library* [Online]. Available at https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAccessibility_Protocol/index.html#//apple_ref/doc/uid/TP40008786 (Accessed 6 April 2016).

Atlassian (2016) 'Bitbucket', *Atlassian* [Online]. Available at https://bitbucket.org/ (Accessed 30 August 2016).

Basili, V. R., Turner, A. J. (1975) Iterative enhancement: A practical technique for software development, *IEEE Transactions on Software Engineering*, Vol.SE-1(4), pp.390-396, [Online] DOI: 10.1109/TSE.1975.6312870 (Accessed 6 February 2016).

BBC (2012) ‘Plastic surgeon Mohammad Jawad on transforming faces of acid victims’ *Outlook* 14 February 2012 [Podcast]. Available at http://www.bbc.co.uk/programmes/p00nkzzv (Accessed 19 June 2016).

Bellini, P., Cenni, D., & Nesi, P. (2014). Optimization of information retrieval for cross media contents in a best practice network. *International Journal of Multimedia Information Retrieval*, 3(3), 147-159. [Online] DOI: 10.1007/s13735-014-0058-8 (Accessed 25 June 2016).

Boosey (2016) *Boosey & Hawkes* [Online]. Available at https://www.boosey.com (Accessed 28 August 2016).

Brooks, C. (2016) ‘Best Document Management Software and Systems’ *Business News Daily*, 24 March 2016 [Online]. Available at http://www.businessnewsdaily.com/8038-best-document-management-software.html (Accessed 5 June 2016).

Brownsword, L. Oberndorf, T. Sledge, C.A. (2000) 'Developing new processes for COTS-based systems', *IEEE Software*, vol. 17, no. 4, pp. 48–55 [Online]. Available at http://ieeexplore.ieee.org (Accessed 23 April 2016).

Bylund, A. (2015) ‘10 Largest Companies by Market Cap in Semiconductors’ *The Motley Fool*, 14 April [Online]. Available at http://www.fool.com/investing/general/2015/04/14/10-largest-companies-by-market-cap-in-semiconducto.aspx (Accessed 6 April 2016).

Capretz, M.A., Higashino, W.A., Tiwari, A. and Grolinger, K., (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: advances, systems and applications*, 2(1), [Online]. DOI: http://dx.doi.org/10.1186/2192-113x-2-22 (Accessed 5 April 2016).

Cui, Y., Kim, M., Lee, H. (2015). ‘An efficient design and implementation of an MdBULPS [Mongo Database-based unstructured log processing system] in a cloud-computing environment’. *KSII Transactions on Internet and Information Systems* (9.8), p 3182.

Debian (2016). ‘systemd – system and service manager’, *debian wiki* [Online]. Available at https://wiki.debian.org/systemd (Accessed 4 September 2016).

Du, J., Han, J., Haihong, E., Le, G. (2011) ‘Survey on NoSQL database’, *Pervasive Computing and Applications (ICPCA), 6th International Conference on Pervasive Computing and Applications*. Port Elizabeth, 26-28 Oct. pp. 363-366. DOI: 10.1109/ICPCA.2011.6106531 (Accessed 6 April 2016).

DuBois, L., Feng, I., Nadkarni, A. (2015) *Worldwide Storage in Big Data Forecast, 2015–2019*. Framingham, International Data Corporation

Dunn, C., (2016). Conversation between Colin Dunn (from Boosey & Hawkes) and Toby Scholz, 21 July 2016

ECLAP, 2016. *European Collected Library of Artistic Performance* [Online]. Available at http://www.eclap.eu (Accessed 14 August 2016).

Edavs (2016) 'Airplayer', *Olimsoft* [Online]. Available at http://www.edavs.com/ (Accessed 19 June 2016).

ErpNext (2016) 'ERP Made Simple' *ERPNext* [Online]. Available at https://erpnext.com/ (Accessed 19 June 2016).

FFmpeg (2016) 'A complete, cross-platform solution to record, convert and stream audio and video', *FFmpeg* [Online]. Available at https://ffmpeg.org (Accessed 12 June 2016).

FFmpeg Java (2016) 'bramp/ffmpeg-cli-wrapper', *Github* [Online]. Available at https://github.com/bramp/ffmpeg-cli-wrapper (Accessed 12 June 2016).

FileHold (2016) 'Document & Record Lifecycle Software' *FileHold* [Online]. Available at http://www.filehold.com (Accessed 19 June 2016).

Forbes (2015) 'The World's Most Valuable Brands – 2015 ranking', *Forbes* [no date] [Online]. Available at http://www.forbes.com/powerful-brands/list (Accessed 6 April 2016).

Gani, A., Hanum, F., Siddiqa, A., Shamshirband, S. (2015) 'A survey on indexing techniques for big data: taxonomy and performance evaluation', *Knowledge and Information Systems*, vol. 46(2), pp. 241-284 [Online]. DOI: 10.1007/s10115-015-0830-y (Accessed 6 April 2016).

GNU (1999), 'GNU Lesser general public license' [Online]. Available at: http://www.gnu.org/licenses/lgpl-2.1.txt (Accessed 21 August 2016).

Great Britain. (2014) ''The Copyright and Rights in Performances (Personal Copies for Private Use) Regulations 2014', *Copyright, Design and Patents Act 1988* [Online]. Available at http://www.legislation.gov.uk/uksi/2014/2361/regulation/3/made (Accessed 6 April 2016).

Green, C. (2015) 'The end of Moore's Law? Why the theory that computer processors will double in power every two years may be becoming obsolete' *The Independent*, [online]. Available at: http://www.independent.co.uk/life-style/gadgets-and-tech/news/the-end-of-moores-law-why-the-theory-that-computer-processors-will-double-in-power-every-two-years-10394659.html (accessed on 27 February 2015).

The Guardian (2014) 'A journey to the heart of the planet we made' *Science Weekly* [Podcast] 30 June 2014. Available at https://www.theguardian.com/science/audio/2014/jun/30/planet-adventures-anthropocene-gaia-vince-podcast (Accessed 19 June 2016).

Hatton, J. (2016) Online meeting with Toby Scholz, 13 June 2016

Hegarty, P. (2015) Developing iOS 8 Apps with Swift [Download]. Available at https://itunes.apple.com/en/course/developing-ios-8-apps-swift/id961180099 (Accessed 6 February 2016).

Hess, M., Nelson, T. (2013) *Science of 3D* [Online]. Available at http://www.ucl.ac.uk/museums-static/science-of-3d/ (Accessed 9 April 2016).

IDC (2015) 'Smartphone OS Market Share, 2015 Q2', *IDC* [Online]. Available at http://www.idc.com/prodserv/smartphone-os-market-share.jsp (Accessed 6 April 2016).

The JCIFS Project (2014) 'The Java CIFS Client Library', *JCIFS* [Online]. Available at https://jcifs.samba.org/src/docs/api/ (Accessed on 17 February 2016).

Jörg, S., Normoyle, A., Safonova, A. (2012) 'How Responsiveness Affects Players' Perception in Digital Games', *Proceedings of the ACM Symposium on Applied Perception (SAP 12)*, pp. 33 - 38 [Online]. DOI: 10.1145/2338676.2338683 (Accessed 29 August 2016).

Kodi (2016) 'Kodi', *Kodi* [Online]. Available at http://kodi.tv/ (Accessed 6 April 2016).

Krzanich, B. (2016) 'FORM 10-K (Annual Report)', *Intel Corp* [Online]. Available at http://www.intc.com/secfiling.cfm?filingID=50863-16-105 (Accessed 2 April 2016).

Kuali (2016) 'Software solutions for higher education' *Kuali* [Online]. Available at https://www.kuali.org/ (Accessed 19 June 2016).

Lixandroiu, R., Maican, C. (2015) 'A system architecture based on open source enterprise content management systems for supporting educational institutions', *International Journal of Information Management*, vol 36, no 2, pp 207 – 214 [Online]. DOI: 10.1016/j.ijinfomgt.2015.11.003 (Accessed 5 June 2016)

LogicalDOC (2016) 'Document Management System' *LogicalDoc* [Online]. Available at http://www.logicaldoc.com/product/overview.html (Accessed 19 June 2016).

Lowndes, C. (2016b) Comments on TMA02, 18 April 2016

Lowndes, C. (2016c) Selected email exchanges, various dates, represented in appendix 4.17

Madu, E., (2007) 'World-class health care', *TED Ideas worth spreading.* Video clip [Online]. Available at http://download.ted.com/talks/ErnestMadu_2007G-480p.mp4?apikey=TEDDOWNLOAD (Accessed 1 September 2019).

Martin, N., Pereira, A. (2015) 'UISearchController Tutorial: Getting Started', *RayWenderlich Tutorials for Developers & Gamers* [Online]. Available at https://www.raywenderlich.com/113772/uisearchcontroller-tutorial (Accessed 10 Feburary 2016).

Marxists Internet Archive (2007) ‘Soviet History Archive’, *Marxists Internet Archive* [Online]. Available at https://www.marxists.org/history/ussr/sounds/mp3/soviet-anthem.mp3 (Accessed 26 June 2016)

Microsoft (2016) ‘FORECAST function’, *Microsoft* [Online]. Available at https://support.office.com/en-us/article/FORECAST-function-50ca49c9-7b40-4892-94e4-7ad38bbeda99 (Accessed 18 June 2016).

MongoDB (2016) ‘Getting started with MongoDB (Java Edition)’, *mongoDB* [Online], New York City, NY. Available at https://docs.mongodb.org/getting-started/java/ (Accessed on 2 April 2016).

O., P. (2015) Re: ‘Compiler error: Method with Objective-C selector conflicts with previous declaration with the same Objective-C selector’, *Stackoverflow*, 5 April [Forum comment]. Available at http://stackoverflow.com/questions/29457720/compiler-error-method-with-objective-c-selector-conflicts-with-previous-declara (Accessed 2 April 2016).

Odoo (2016) ‘All-in-one management software’ *Odoo* [Online]. Available at https://www.odoo.com/ (Accessed 19 June 2016).

OfficeGemini (2016) ‘Dokmee’ *OfficeGemini* [Online]. Available at https://www.dokmee.com (Accessed 19 June 2016).

The Open University (2008a) ‘Unit 8’, *M362, Developing concurrent distributed systems*. The Open University Press, Milton Keynes

The Open University (2008b) *M362, Developing concurrent distributed systems* [Online]. Available at https://learn2.open.ac.uk/course/view.php?name=M362-15B (Accessed 2 April 2016).

The Open University (2014a) ‘7.3 Computational Complexity’, *M269 Algorithms, data structures and computability* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=467577&section=3.1 (Accessed 2 April 2016).

The Open University (2015c) ‘From domain to analysis models’, *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739476&section=4 (Accessed 12 June 2016).

The Open University (2015d) ‘Model-view-controller pattern’, *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739465&section=5.4 (Accessed 12 June 2016).

The Open University (2015f) *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/subpage/view.php?id=739368 (Accessed 12 June 2016).

The Open University (2015g) ‘Unit 11 Product quality: verification, metrics and testing’, *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739467&section=4.4 (Accessed 26 June 2016).

The Open University (2015h) ‘Agile development’, *TM354 Software engineering* [Online]. Available at ‘https://learn2.open.ac.uk/mod/oucontent/view.php?id=739472&section=3.2.3 (Accessed 2 July 2016)

The Open University (2015j) ‘Architecture, quality attributes and tactics’, *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739466&section=4.4 (Accessed 2 July 2016)

The Open University (2016) ‘Legal, Social, Ethical and Professional issues’, *TM470-16B* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=767416 (Accessed 6 April 2016).

Oracle (2014), *Oracle Technology Network License Agreement*, [Online]. Available at http://www.oracle.com/technetwork/licenses/standard-license-152015.html (Accessed 21 August 2016).

Oracle (2016d) ‘Java™ Platform, Standard Edition 8 API Specification’ *Java SE Documentation* [Online], Redwood Shores, CA. Available at https://docs.oracle.com/javase/8/docs/api/ (Accessed 2 April 2016).

Oracle (2016e) ‘Class Http Server’ *Java HTTP Server – Oracle JRE* [Online]. Available at https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpServer.html (Accessed 19 June 2016).

Oracle (2016f) ‘Class ProcessBuilder ‘, *Java Platform* [Online]. Available at https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html (Accessed 12 June 2016).

Oracle (2016g) ‘Https Server’, *Java SE Documentation* [Online]. Available at ‘https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpsServer.html (Accessed 11 September 2016).

Oracle (2016h) ‘Package java.net.ssl’, *Java SE Documentation* [Online]. Available at https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/package-summary.html (Accessed 11 September 2016).

Patricios, M. (2013) A java library for reading mp3 files and reading / manipulating the ID3 tags (ID3v1 and ID3v2.2 through ID3v2.4), *mpatric / mp3agic* [Online]. Available at https://github.com/mpatric/mp3agic (Accessed 8 March 2016).

Pentaloop (2016) ‘Player Extreme’, *Pentaloop* [Online]. Available at http://pentaloop.devcontact.com/support (Accessed 2 July 2016).

Pinpoint (2016) ‘Document Management and Storage Systems’ *Pinpoint* [Online]. Available at http://www.pinpointdigital.co.uk (Accessed 19 June 2016).

Rehar, A. (2012) ‘Implementacija protokola HTTP Live Streaming v programskem jeziku Java’, *Univerza v Ljubljani fakulteta za ra cunalni stvo in informatiko* [Online]. Available at http://eprints.fri.uni-lj.si/1929/1/Rehar-1.pdf (Accessed 6 April 2016).

RedMonk (2015) ‘The RedMonk Programming Language Rankings: June 2015’, *RedMonk* [Online]. Available at http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/ (Accessed 6 April 2016).

The Royal Society (2010) ‘Beyond limits’, *R.Science podcast* [Online]. Available at http://downloads.royalsociety.org/audio/rscience/100927.mp3 (Accessed 16 June 2016).

Scholz, T. (2016a) *TM470 TMA02*, submitted to The Open University as part of TM470 assessment

Scholz, T. (2016b) ‘Building x264 on CentOS 7.2 | undefined reference errors | please help’, *The VideoLAN Forums* [Online]. Available at https://forum.videolan.org/viewtopic.php?f=32&t=132956&p=441572&hilit=centos+7#p441572 (Accessed 18 June 2016).

Spotify (2016) ‘Spotify’, *Spotify* [Online]. Available at https://www.spotify.com (Accessed 6 April 2016).

Twonky (2016) *Twonky* [Online]. Available at http://twonky.com (Accessed 2 July 2016).

Ubuntu (2016) ‘Ubuntu’, *Ubuntu* [Online]. Available at http://www.ubuntu.com (Accessed 18 June 2016).

Wikpedia (2016c) ‘List of UPnP AV media servers and clients’, *Wikipedia* [Online]. Available at ‘https://en.wikipedia.org/wiki/List_of_UPnP_AV_media_servers_and_clients (Accessed 19 June 2016).

3. BIBLIOGRAPHY

Alphabet (2016) 'Your search - speed-optimised indexing solution for file metadata with mobile-friendly front-end - did not match any articles.' *Google Scholar* [Online]. Available at https://scholar.google.co.uk/scholar?q=speed-optimised+indexing+solution+for+file+metadata+with+mobile-friendly+front-end&hl=en&as_sdt=0,5 (Accessed 19 June 2016).

Apache (2016a) Apache HTTP SERVER PROJECT [Online]. Available at https://httpd.apache.org (Accessed 22 March 2016).

Apache (2016b) 'Apache CouchDB', *Apache CouchDB* [Online]. Available at http://couchdb.apache.org (Accessed 6 April 2016).

Apple (2016a) 'Using HTTP Live Streaming', *iOS Developer Library* [Online], Cupertino, CA. Available at https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html#//apple_ref/doc/uid/TP40008332-CH102-SW5 (Accessed 24 March 2016).

Apple (2016b) 'AVAudioPlayer', *iOS Developer Library* [Online]. Available at https://developer.apple.com/library/ios/documentation/AVFoundation/Reference/AVAudioPlayerClassReference/ (Accessed 17 February 2016).

Apple (2016d) 'iTunes', *iTunes* [Online]. Available at http://www.apple.com/uk/itunes/ (Accessed 6 April 2016).

Apple (2016j) 'NSUserDefaults' *iOS Developer Library* [Online]. Available at https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/index.html#//apple_ref/occ/instm/NSUserDefaults/registerDefaults: (Accessed 18 June 2016).

Barbierato, E., Gribaudo, M., Maurolacono, I. (2014). Performance evaluation of NoSQL big-data applications using multi-formalism models. *Future Generation Computer System*. (37), pp 345-353.

BBC (2015) 'Ripping music and films illegal again after High Court overturns new law', *BBC News*, 17 July 2015 [Online]. Available at http://www.bbc.co.uk/newsbeat/article/33566933/ripping-music-and-films-illegal-again-after-high-court-overturns-new-law (Accessed 30 March 2016).

Bylund, A. (2015) '10 Largest Companies by Market Cap in Semiconductors' *The Motley Fool*, 14 April [Online]. Available at http://www.fool.com/investing/general/2015/04/14/10-largest-companies-by-market-cap-in-semiconducto.aspx (Accessed 6 April 2016).

Chodorow, K. (2010) ‘History of MongoDB’, *snail in a turtleneck* [Blog]. Available at http://www.kchodorow.com/blog/2010/08/23/history-of-mongodb/. (Accessed on 31 March 2016).

DB-Engines Ranking (2016) ‘DB-Engines Ranking’, *DB-Engines* [Online]. Available at http://db-engines.com/en/ranking (Accessed 9 April 2016).

Creative Commons (2016) ‘About CC0 - “No Rights Reserved”’, *Creative commons* [Online]. Available at https://creativecommons.org/about/cc0/ (Accessed 6 April 2016).

Dataone (2016) ‘Best Practices‘, *DataONE* [Online]. Available at https://www.dataone.org/best-practices (Accessed 6 April 2016).

DuBois, L., Feng, I., Nadkarni, A. (2015) *Worldwide Storage in Big Data Forecast, 2015–2019.* Framingham, International Data Corporation

Denegri-Knott, J. (2010) ‘The emergence of MP3 technology’, *Journal of Historical Research in Marketing*, 2010, vol. 2(4), pp. 397-425 [Online], DOI: 10.1108/17557501011092466 (Accessed 2 April 2016).

Gani, A., Hanum, F., Siddiqa, A., Shamshirband, S. (2015) ‘A survey on indexing techniques for big data: taxonomy and performance evaluation’, *Knowledge and Information Systems*, vol. 46(2), pp. 241-284 [Online]. DOI: 10.1007/s10115-015-0830-y (Accessed 6 April 2016).

Github (2016) ‘Apache License’, *apple/swift* [Online]. Available at https://github.com/apple/swift/blob/master/LICENSE.txt (Accessed 6 April 2016).

GNU (2016) ‘GNU GENERAL PUBLIC LICENSE 3.0’, *GNU* [Online]. Available at http://www.gnu.org/licenses/gpl-3.0.en.html (Accessed 6 April 2016).

Gracenote (2016) ‘gracenote’, *Gracenote* [Online]. Available at http://www.gracenote.com/ (Accessed 6 April 2016).

Grafov, A. (2016) ‘Parser and generator of M3U8-playlists for Apple HLS’, *GitHub* [Online]. Available at https://github.com/grafov/m3u8 (Accessed 6 April 2016).

Gustafson, J. (1988) ‘Reevaluating Amdahl’s Law’, *Communications of the ACM*, vol. 31, no. 5, pp. 532-3.

Hazra, T. (1995) ‘Parallel computing Defining what it is and the doors it opens via transputers’ *IEEE Potentials*, vol. 14, no. 3, pp 17 – 20 [Online]. DOI: 10.1109/45.464689 (Accessed 2 April 2016).

Hill, M., Marty, R. (2008) ‘Amdahl’s Law in the Multicore Era’, *Computer*, vol 41(7), pp 33-8 [Online]. DOI: 10.1109/MC.2008.209 (Accessed 27 August 2016).

ID3v2 (2012) ‘Low Tech History’, *ID3v2 The Audience is informed* [Online]. Available at http://id3.org/History (Accessed 9 March 2016).

Intel (2016) Intel® Xeon® Processor E7 v3 Family, *intel* [Online]. Available at http://ark.intel.com/products/family/78585/Intel-Xeon-Processor-E7-v3-Family#@Server (Accessed 6 April 2016).

Is NoSQL fit for the Enterprise? (2016). [online video clip] London: The Register, Collins J. Available at: http://whitepapers.theregister.co.uk/paper/view/4265/ (accessed on 27 February 2016).

Kim, M., Cui, Y., Lee, H. (2015). 'An efficient design and implementation of an MdBULPS [Mongo Database-based unstructured log processing system] in a cloud-computing environment'. *KSII Transactions on Internet and Information Systems* (9.8), p 3182.

Lowndes, C. (2016a) OU Collaborate conversation with Toby Scholz, 29 March 2016

Microsoft (2016) 'Performance Optimizations for the XML Data Type in SQL Server 2005', *TechNet* [Online]. Available at https://technet.microsoft.com/en-us/library/ms345118(v=sql.90).aspx (Accessed on 6 April 2016).

Moore, G. (1965) 'Cramming more components onto integrated circuits', *Electronics.* (38.8).

Neumann, J., Plank, M. (2014) 'TIB's Portal for audiovisual media: New ways of indexing and retrieval', *IFLA Journal*, 3 March (40.1), pp 17-23 [Online]. DOI: http://dx.doi.org/10.1177/0340035214526531 (Accessed 28 June 2016).

Oard, D. W., Baron, J. R., Hedin, B., Lewis, D. D., & Tomlinson, S. (2010). 'Evaluation of information retrieval for E-discovery' *Artificial Intelligence and Law*, vol 18(4) pp347-386 [Online]. DOI: 10.1007/s10506-010-9093-9 (Accessed 18 June 2016).

O'Brian, N. (2013) 'HTTP Live Streaming – Videos On Demand (HLS – VOD), *NSProgrammer* [Blog, online]. Available at http://www.nsprogrammer.com/2013/08/http-live-streaming-videos-on-demand.html#more (Accessed 18 June 2016).

Ofcom (2016) 'Broadband Wireless Access / Spectrum Access', *Ofcom* [Online], London. Available at http://licensing.ofcom.org.uk/radiocommunication-licences/mobile-wireless-broadband/cellular-wireless-broadband/policy-and-background/broadband-fixed-wireless/ (Accessed 2 April 2016).

Oracle (2016a) 'Deploying JavaFX Applications', JavaFX Documentation [Online]. Available at http://docs.oracle.com/javafx/2/deployment/deploy_quick_start.htm#A127674 (Accessed 6 April 2016).

Oracle (2016b) 'Oracle Binary Code License Agreement for the Java SE Platform Products and JavaFX', *Oracle Technology Network* [Online]. Available at http://www.oracle.com/technetwork/java/javase/terms/license/index.html (Accessed 6 April 2016).

Oracle (2016c) ‘Setting the class path’, *Java SE Documentation* [Online], Redwood Shores, CA. Available at http://docs.oracle.com/javase/7/docs/technotes/tools/windows/classpath.html (Accessed 2 April 2016).

Peek, H. (2010) ‘The Emergence of the compact disc’, *IEEE Communications Magazine*, vol 48(1) pp 10-17 [Online]. DOI: 10.1109/MCOM.2010.5394021 (Accessed on 2 April 2016).

Qualcomm (2016) ‘Snapdragon 820 Processor’, *Qualcomm* [Online]. Available at https://www.qualcomm.com/products/snapdragon/processors/820 (Accessed 2 April 2016).

Redmond, E., Wilson, J. (2012). *Seven databases in seven weeks: a guide to modern databases and the NoSQL movement* [Online], Pragmatic Programmers. Available at http://eu01.alma.exlibrisgroup.com/view/action/uresolver.do?operation=resolveService&package_service_id=2643447480002316&institutionId=2316&customerId=2315 (Accessed 6 April 2016).

Strasser, C., Cook, R., Michener, W. & Budden, A., (2012) DataONE, *Primer on data management: what you always wanted to know* [Online]. Available at https://www.dataone.org/sites/all/documents/DataONE_BP_Primer_020212.pdf (Accessed 6 April 2016).

The Open University (2013a) *M364 Fundamentals of interaction design* [Online]. https://learn2.open.ac.uk/mod/subpage/view.php?id=416136 (Accessed 12 June 2016).

The Open University (2014b) ‘Block 3 Units 9-12 From architecture to product’, *TM354 Software engineering* [Online], Available at https://learn2.open.ac.uk/mod/repeatactivity/view.php?id=781328 (Accessed 6 April 2016).

The Open University (2015e) ‘Manage system capacity’, *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739466&section=4.3 (Accessed 12 June 2016).

Unknown. (2011). *Code of Conduct for BCS Members*. London. BCS, The Chartered Institute for IT

VideoLAN (2013) ‘x264’, *VideoLAN organization* [Online]. Available at http://www.videolan.org/developers/x264.html (Accessed 18 June 2016).

Wheeler, D. (2004) ‘SLOCCount’, *David Wheeler’s Personal Home Page* [Online]. Available at http://www.dwheeler.com/sloccount/ (Accessed 18 June 2016).

Wikipedia (2016a) *Gracenote*, Wikipedia [Online], 28 March 2016. Available at https://en.wikipedia.org/wiki/Gracenote (Accessed 30 March 2016).

Wikpedia (2016b) *Napster (pay service)* Wikipedia [Online], 14 February 2016. Available at https://en.wikipedia.org/wiki/Napster_%28pay_service%29 (Accessed on 30 March 2016).

Williams-Grut, O. (2015) ‘Apple's iPhone: The most profitable product in history’, *The Independent* 29 January [Online]. Available at http://www.independent.co.uk/news/business/analysis-and-features/apples-iphone-the-most-profitable-product-in-history-10009741.html (Accessed 6 April 2016).

4. APPENDICES

4.1. Glossary

AFP: Apple Filing Protocol. A technology proprietary to Apple to represent a file system over a TCP/IP network connection. Support for AFP by vendors other than Apple exists, but is not mature.

AFS: Andrew File System. A distributed file system with support for a large number of clients. AFS heavily influenced NFS, but is now rarely used.

API: Application Programming Interface. The set of public interfaces belonging to a software library, which allow its use within another system. The API equates to the functional specification of a software library.

Boosey and Hawkes: Large British music publisher, founded in 1930, now owned by the Dutch Stichting Pensionenfonds. See (Boosey, 2016).

BSON: Binary JSON. Binary representation of a JSON object.

CentOS: A popular Linux distribution, which is effectively a clone of Red Hat Enterprise Linux, but without Redhat's support. It may be used freely for non-commercial purposes

CIFS: Common Internet File System. See SMB.

CPU: Central processing unit.

Debian: A Linux-based operating system that stands out for only offering software that can be used free of charge.

DRM: Digital Rights Management. Copyrighted files may include information that can be used to enforce legitimate use.

ECLAP: European Collected Library of Artistic Performance. A European library project seeking to archive performing arts related materials throughout Europe. See ECLAP, 2016.

FLOSS: Free / Libre / Open Source Software. An acronym describing software, the source code of which is known, and that generally imposes few restrictions with regards to its deployment.

Fork: An implementation of software based on a certain software core. The various implementations will share some characteristics defined by the core, but differ in other characteristics, defined by the implementation. It is often important to know which core a software is forked off, since it has an impact on compatibility for third-party software and hardware support.

HDFS: A young, distributed file system based on the Hadoop framework, which supports very large deployments.

HLS: HTTP Live Streaming. A method of transferring files via HTTP in small chunks, such that the receiver can process individual chunks immediately, as opposed to waiting for the entire file to load. A popular method to transfer large media files for instant consumption.

HTTP: Hypertext Transfer Protocol. A ubiquitous protocol used to transfer data over a network.

IDE: Integrated Development Environment. A tool that aids the development of software, by offering assistance with syntax checking, and auto-completion, and streamlines many processes, such as formatting, and compilation tasks.

iOS: iPhone Operating System. The iOS (not to be confused with Cisco IOS) is used as operating system for Apple's line of portable devices, such as iPhone, Watch, iPad, and iPod.

JSON: JavaScript Object Notation. A human-readable representation of an object. Although it derives from the JavaScript language, it is used by many languages.

macOS: MACintosh Operating System. The operating system Apple provides with its line of desktop-class computers, such as the Mac Pro, Mac Book, and Mac Mini.

NAS: Network Attached Storage. A NAS is a device with the explicit purpose of storing data, and making that data available on the (local) network. This way, data can be shared across all devices connected to the network, such as televisions, smart phones, and refrigerators. Newer NAS devices often offer enhanced functionality, including native video playback, and CCTV. The enterprise equivalent would be a SAN.

Nep-Tune: The name Nep-Tune was inspired by a combination of an allusion to the Roman god of the sea, Neptune, where the sea represents the vast collection of documents that the god, Neptune, presides over, and a word play on the English word "tune", which represents the multimedia capabilities of the software.

NFS: Network File System. A technology to represent a file system over a TCP/IP network connection. Originating in the Linux community, it is functionally similar to SMB, but less frequently used, and arguably inferior in terms of stability for large file sizes.

NoSQL: An umbrella term for database technology, which does *not* use a relational database schema. Typically, NoSQL implementations will use a key / value strategy, where any table can hold entries with any number of tags (representing the key), which replace the columns found in relational databases.

Samba: Although "Samba" is technically a re-implementation of the SMB/CIFS protocol, "samba" is now widely used synonymously to SMB and CIFS. In reality, many implementations exist that use the SMB protocol (e.g. JCIFS), but these are rarely referred to by their proper names.

SAN: Storage Area Network. A network dedicated to making storage accessible to servers and clients in a corporate network. A SAN offers full transparency, so what may appear to be a single system drive to a server may actually comprised of several stripes on many devices within the SAN, optimised for safety, performance, or auditability.

SMB: Server Message Block. A technology to represent a file system over a TCP/IP network connection. Since SMB was initially proprietary to Microsoft, the Common Internet File System (CIFS) was developed as an open source clone to SMB. SMB has since become open source, and the terms SMB and CIFS are now interchangeable.

SQL: Structured Query Language. Although SQL originally referred to the language used to query relational databases, SQL has become the umbrella term for any relational database implementation.

Ubuntu: A popular Debian distribution, best known for including a graphical user interface, and its focus on productivity integration.

UUID: Universally Unique IDentifier: An identifier, which is unique within any given system

XFS: eXtended File System – a common file system for Linux distributions; the standard file system for CentOS 7.

4.2. Abstract

This project seeks to produce a software solution to index and search large collections of individual files with heterogeneous metadata, making the index, and any particular file within the collection, available via a standard interface to clients, focusing on search performance and scalability.
An iOS client is also provided to demonstrate the ability of the Java server component, and play back retrieved media files.

4.3. Acknowledgements

I would like to thank my tutor Charly Lowndes for his continued support throughout the project, and his guidance on the academic aspect of the project.

I am grateful to Peter Thomson and Dr. R. Westaway for their help with the choice of topic, and for preventing me from getting involved with some of the far more complex ideas I proposed.

Chun-Yin Chu deserves my gratitude for designing the logo for my client app, for never getting upset with my working through the night and weekends, and for enabling me to use the resources of University College, London.

Thanks to James Hatton, for providing a third-party view on the usability aspects of the software, and for his input on possible applications in the vehicle industry.

4.4. ECLAP search

Ten attempts were made on 26 June 2016, taking from 21.9 to 26.4 seconds to return. The search was repeated on 2 July 2016 with similar results.

4.5. Schedule of lifecycle phases

4.5.1. Phase 1

1. Install CentOS test server
    a. download & install OS
2. Install MongoDB
    a. install and configure MongoDB on server
3. Develop client
    a. set up an Apple developer account
    b. study API for music playback & file retrieval information
    c. implement basic UI to play back local file
4. Develop server component
    a. create Java executable
    b. integrate executable with MongoDB
    c. research parsing of file metadata for mp3
    d. read file system structure and metadata, and write both to the database

4.5.2. Phase 2

2. Develop client / server integration
    a. enhance executable to accept connections from client
    b. enhance client to request connection to server component
    c. research how to implement HLS to allow media streaming from server to client
    d. change operating system to Ubuntu, since libx264 (required for HLS) does not compile on CentOS 7.
    e. enhance server to stream HLS
    f. enhance client to play back the stream received
3. Comparative performance analysis, and software review

4.5.3. Phase 3

1. Enhance the server component
    a. monitor for file changes (using scanning or notifications)
    b. parse metadata for more file formats (mp4, aac, flac)
2. Enhance the client
    a. sophisticated playback functions (pause, skip, fast forward / reverse)
    b. create locally persisted playlists

4.6. Detailed timetable

Start date	Task	Related Phase	Completed
29/01/2016	Project Start		
15/02/2016	download & install OS	1.1.a	25/02/2016
26/02/2016	Install and configure MongoDB on server	1.2.a	27/06/2016
28/02/2016	set up an Apple developer account	1.3.a	01/03/2016
02/03/2016	study API for music playback & file retrieval information	1.3.b	06/03/2016
07/03/2016	implement basic UI to play back local file	1.3.c	13/03/2016
14/03/2016	Create Java executable	1.4.a	20/03/2016
21/03/2016	Integrate executable with MongoDB	1.4.b	27/03/2016
28/03/2016	Research parsing of file metadata for mp3	1.4.c	03/04/2016
04/04/2016	Research ways to query server component from client over HTTP	1.4.d	08/04/2016
09/04/2016	Revise TMA02 with tutor feedback		09/04/2016
12/04/2016	TMA02 due		-
13/04/2016	Start implementation of HTTP server in server component and enhance executable to accept connections from client	2.1.a	17/04/2016
18/04/2016	Enhance client to request connection to server component	2.1.b	24/04/2016
25/04/2016	Test client server integration		29/04/2016
30/04/2016	Review progress for TMA03		01/05/2016
02/05/2016	Research approaches for transcoding for HTTP Live Streaming	2.1.c	08/05/2016
09/05/2016	Implement HTTP Live Streaming on server component		13/05/2016
29/05/2016	Break for TM354 exam revision		-
06/06/2016	change operating system to Ubuntu to accommodate HLS transcoding using ffmpeg (this item was the unexpected result of FFmpeg not compiling on CentOS)	2.1.d	06/06/2016
08/06/2016	enhance client to HLS stream and play back the stream received	2.1.e/f	17/06/2016

18/06/2016	Review streaming enhancements for TMA03		19/06/2016
20/06/2016	Comparative performance analysis, and software review	2.2	21/06/2016
22/06/2016	Work on draft for TMA03		26/06/2016
26/06/2016	Complete draft for TMA03, discuss queries with Charly		01/07/2016
02/07/2016	Review TMA03 according to feedback		05/07/2016
05/07/2016	TMA03 due		-
06/07/2016	Research monitoring for file changes (could be via scan or notification)	3.1.a	09/07/2016
10/07/2016	Research parsing metadata for more file formats (mp4, aac, flac)	3.1.b	15/07/2016
16/07/2016	Implement findings on file change monitoring and additional metadata parsing		22/07/2016
23/07/2016	Analyse performance and make predictions for commercial application		05/08/2016
06/08/2016	Begin EMA		-
13/08/2016	Continue with EMA		-
15/08/2016	Include analysis findings in EMA		-
22/08/2016	If on time, research and implement sophisticated playback functions (fast forward / reverse, scrolling), and local playlists	3.2.a/b	10/09/2016
27/08/2016	Have draft EMA ready		29/08/2016
04/09/2016	Review code and EMA for final submission		12/09/2016
12/09/2016	Project End / EMA due		

4.7. Requirements analysis

Compiled using agile methodology, as introduced in TM354 (The Open University, 2015h), and informed by Lixandroiu at al., (2015) and Brooks (2016).

4.7.1. Server component

Observations from domain analysis	Comment	Resulting Requirement
To ensure best performance, computationally heavy tasks are often delegated to off-site servers	A client-server model seems desirable	The component must offer an interface, with which it can communicate over a network
Files come in a variety of file formats	Besides common file types, some files to be indexed may be proprietary	The component ought to support a large number of common file type out-of-the-box
Collections can contain a large amount of individual files, which may be grouped in directories	For a large number of files, traversing the directory structure for each query will not bring the desired query speed.	An index of the collection must be produced, which can be queried quickly
Large collections of file are not typically kept on a local drive	Access to corporate file shares is most commonly provided using the samba protocol, closely followed by NFS.	The component must be able to connect via a standard network file protocol, such as samba
Traversing a file system may take an indeterminably long time	If we traverse the file system once, and subsequently reload the index on restart, and maintain accuracy, this is not a problem	- The index must be persisted - The index must be updated to reflect changes in the file system
Finding a particular file in the collection by search is typically very time consuming, and really only feasible with knowledge of the underlying directory structure once the collection exceeds a certain size	Once a file has been found, it will be desirable for the server to return the file to the client.	- Search must be “fast” - an interface must be provided that allows searching for and returning the found file
Transmission of large media files over a network can lead to complications (dropped / slow connections), for which reason Apple imposes a limit on the file size that can be transmitted without transcoding to HLS	Multiple approaches to streaming media files exist, but Apple currently only supports HLS	multimedia files shall be automatically transcoded to HLS to comply with Apple’s (Apple, 2016a and Apple, 2016e) constraints
Metadata for individual files may be incomplete, or incorrect	It may be possible to glean additional information from the file path, and –name. In	- File path and –name must be included in the index. - The server component should

	addition, the server component should ignore tag type, to offer maximum success rate	not filter by tag type.
Metadata available will vary by file type, and files may be of many different types	If the type of metadata is unknown, it will be difficult to implement a relational database. A NoSQL solution will be more appropriate.	- Persistence should use a NoSQL approach to allow for heterogeneous metadata. - the server component should make it easy to add support for additional file types
Only read access is required; no changes to the underlying files are necessary	Write interaction need not be implemented	The server component must not make changes to the indexed files

Table 4: Functional requirements server component

4.7.2. Client component

Observations from domain analysis	Comment	Resulting Requirement
To ensure best performance, computationally heavy tasks are often delegated to off-site servers	A client-server model seems desirable	The component must offer an interface, with which it can communicate over a network
consumption happens increasingly on mobile phones, for which reason a mobile phone client is desirable	I primarily use Apple iOS devices, which would be the desired client platform	The client component must run on Apple iOS devices.
a common use case for storing files on a network share are media files (e.g. films), which tend to be large in size	It would be good to be able to playback a file retrieved from the server	The client component must offer media playback

Table 5: Functional requirements client component

4.7.3. Non-functional requirements of the system

Observed functional requirement	Comment	Resulting requirement
search must be "fast"	Research in the field of computer games shows that "delays of 500ms were rated as acceptable" for the player to receive a response to input (Jörg et al, 2012). Without overcomplicating the situation, 500ms seems a good target value for a response that should be perceived a "fast" (or "real-time").	The search shall return within 500ms for a collection smaller than 10,000 files in 99% of test cases on the test system.
the server component should make it easy to add support for additional file types	support for file types not supported by Apache Tika necessarily demands an additional libray, so the best result would be to make it possible to add support in only one place in the code, in additional to linking the additional library.	It shall take no more than two (conceptual) steps to add support for additional file types.

Table 6: Non-functional requirements

4.8 –

This appendix was deliberately removed prior to submission.

4.8.1 Code complexity

This easter-egg appendix is not referred to in the main body.

The Open University's TM354 module introduced counting the lines of discrete code as a measure of complexity. With the project being code complete, I was intrigued how many lines of code the project had yielded. SLOCCount (Wheeler, 2004) is a tool included with Ubuntu, which provides some surprising statistics alongside its output:

```
toby@ubuntu:~/src/NepTune/src/NepTune$ sloccount .
Creating filelist for NepTune
Categorizing files.
Finding a working MD5 command....
Found a working MD5 command.
Computing results.


SLOC    Directory       SLOC-by-Language (Sorted)
1648    NepTune         java=1644,sh=4


Totals grouped by language (dominant language first):
java:          1644 (99.76%)
sh:               4 (0.24%)




Total Physical Source Lines of Code (SLOC)                = 1,648
Development Effort Estimate, Person-Years (Person-Months) = 0.34 (4.06)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 0.35 (4.26)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 0.95
Total Estimated Cost to Develop                           = $ 45,651
 (average salary = $56,286/year, overhead = 2.40).
SLOCCount, Copyright (C) 2001-2004 David A. Wheeler
SLOCCount is Open Source Software/Free Software, licensed under the GNU GPL.
SLOCCount comes with ABSOLUTELY NO WARRANTY, and you are welcome to
redistribute it under certain conditions as specified by the GNU GPL license;
see the documentation for details.
Please credit this data as "generated using David A. Wheeler's 'SLOCCount'."
toby@ubuntu:~/src/NepTune/src/NepTune$
```

SLOCCount statistics

According to SLOCCount, the 1644 lines of code (server component only) should have taken approximately four months to develop (this seems correct, if the academic part of the project is discounted), and would cost around $45,500, or £35,000 at today's conversion rate.
Since I developed the software by myself, I should be able to claim the whole amount, making me worth £35,000 * 4 = £105,000 / year – minus expenses (and that is presumably based on estimates from 2004, the year SLOCCount was last updated).

4.9. Server component code excerpts

Comments and imports have been generally omitted in order to reduce page count. The full code can be downloaded from https://bitbucket.org/SpeedyGoneZales/neptuneserver.

4.9.1. SpeedyGonzales.java

The SpeedyGonzales class represents the implementation of the multi-threaded search logic.

```
List<Map<String,String>> fastDocList = new ArrayList<>();
DataStoreSingleton ds = DataStoreSingleton.getInstance();
Map<String,Document> slowDocMap = ds.getMap();
int numberOfProcessorCores = Runtime.getRuntime().availableProcessors();
int sectionDivisor = ( Math.round(slowDocMap.size() / numberOfProcessorCores ));
```

```
SpeedyGonzales()
        super();
        splitData();
    }

    public List<Document> findString(List<String> searchTerms) throws InterruptedException, ExecutionException
    {
        List<Document> result = new ArrayList<>();
        List<List<Document>> searchResult = new ArrayList<>();
        ExecutorService executor = Executors.newFixedThreadPool(fastDocList.size());
        List<Future<List<Document>>> threadList = new ArrayList<>();

        for ( int ii = 0; ii < fastDocList.size(); ii++ )
        {
        final int jj = ii;
        threadList.add( executor.submit(new Callable<List<Document>>() {
            @Override
            public List<Document> call() throws Exception {
            List<Document> resultList = new ArrayList<>();
                try {
                    Map<String,String> quickMap = fastDocList.get(jj);
                    for (Map.Entry<String,String> entry : quickMap.entrySet() )
                    {
                        boolean found = false;
                        for ( String substring : searchTerms )
                        {
                        if ( entry.getValue().toLowerCase().contains(substring.toLowerCase()) )
                            {
                                resultList.add(slowDocMap.get(entry.getKey()));
                                break;
                            }
```

```
                    }
                }
            }
            catch(Exception e){
                LOG.ERROR("Error in the Multithreaded search logic: " + e.toString());
                throw new UnsupportedOperationException("Not supported yet.");
            }
            return resultList;
        }
    })
    );
    }
    executor.shutdown();
    for (Future<List<Document>> entry : threadList)
    {
        result.addAll(entry.get());
    }

    for (List<Document> entry : searchResult)
    {
        result.addAll(entry);
    }
    return result;
}

private void splitData() {
    int ii = 1;
    int multiplier = 1;
    Map<String, String> temp = new HashMap<>();
    for ( Map.Entry<String, Document> entry : slowDocMap.entrySet() )
    {
        temp.put( entry.getKey(), entry.getValue().toString() );

        if ( ii >= sectionDivisor * multiplier || ii == slowDocMap.size() )
        {
            fastDocList.add( (multiplier -1), new HashMap<>(temp) );
            multiplier++;
            temp.clear();
        }
        ii++;
    }
```

```
    }
}
```

4.9.2. ClientConnector.java

The client connector parses the requests from the client.

```
public class ClientConnector implements HttpHandler, Runnable {
    Socket socket;
    private static final String CHARSET = java.nio.charset.StandardCharsets.UTF_8.name();
    private static final int STATUS_OK = 200;
    private final static String CRLF = "\r\n";
    DataStoreSingleton dataStore = DataStoreSingleton.getInstance();
    ConfigReaderSingleton properties = ConfigReaderSingleton.getInstance();
    private final int waitBeforeSendingPlaylist;
    ClientConnector(Socket sock)
    {
        socket = sock;
        waitBeforeSendingPlaylist = Integer.parseInt(properties.getProperty("waitBeforeSendingPlaylist"));
        LOG.INFO("ClientConnector: Connection accepted from host: " + socket.getInetAddress().getHostName() + "," +
socket.getInetAddress().getHostAddress());
    }
    @Override
    public void handle(HttpExchange httpExchange) throws IOException
    {
        String response = "This is the response";
        httpExchange.sendResponseHeaders(STATUS_OK, response.length());
        OutputStream os = httpExchange.getResponseBody();
        os.write(response.getBytes(Charset.forName("UTF-8")));
        os.close();
    }
    @Override
    public void run()
    {
        try
        {
            processReq();
        }
        catch(Exception e)
        {
           LOG.ERROR(e.toString());
        }
    }
    private void processReq() throws Exception
    {
        InputStream istream = socket.getInputStream();
```

```
DataOutputStream ostream = new DataOutputStream(socket.getOutputStream());
BufferedReader breader = new BufferedReader(new InputStreamReader(istream, CHARSET));
String line = breader.readLine();
String connectionDetails = breader.readLine();
LOG.DEBUG("Connection to " + connectionDetails.substring(connectionDetails.indexOf(":") + 2));
Map<String, String> urlParameters = parseUrl(line);
LOG.DEBUG("Request from client was: " + urlParameters);
List<Document> docs = null;
String status = null;
String contentType = null;
String encoding = "charset: " + "UTF-8" + CRLF;
String mode = urlParameters.get("mode");
LOG.DEBUG("Query mode is " + mode);
if ( mode.equals("data") )
{
    docs = dataStore.getList();
    status = "HTTP/1.1 200 OK" + CRLF;
    contentType = "Content-type: " + "application/x-mpegURL" + CRLF;
}
else if (mode.equals("search"))
{
    String searchString = URLDecoder.decode(urlParameters.get("searchString"), CHARSET);
    LOG.DEBUG("Search string is " + searchString);
    List<String> searchList = new ArrayList<String>();
    Pattern regex = Pattern.compile("[^\\s\"']+|\"([^\"]*)\"|'([^']*)'");
    Matcher regexMatcher = regex.matcher(searchString);
    while (regexMatcher.find())
    {
        if (regexMatcher.group(1) != null)
        {
            searchList.add(regexMatcher.group(1));
        }
        else if (regexMatcher.group(2) != null)
        {
            searchList.add(regexMatcher.group(2));
        } else
        {
            searchList.add(regexMatcher.group());
        }
    }
    searchList.stream().forEach((String string) -> {LOG.DEBUG("Search term: " + string);}) ;
    docs = search(searchList);
    status = "HTTP/1.1 200 OK" + CRLF;
    //contentType = "Content-type: " + "text/html; charset=UTF-8" + CRLF + CRLF;
```

```
    contentType = "Content-type: " + "application/x-mpegURL" + CRLF;
}
else if (mode.equals("playlist"))
{
    String objectId = URLDecoder.decode(urlParameters.get("objectId"), CHARSET);
    Document doc = dataStore.getMap().get(objectId);
    String path = (String) doc.get("filePath");
    String file = (String) doc.get("fileName");
    String type = (String) doc.get("fileType");
    String filePath = path + file + "." + type;
    String expectedPlaylistPath = properties.getTempDirectory() + objectId + ".m3u8";
    LOG.DEBUG("Requested objectId is " + objectId);
    File expectedPlaylist = new File(expectedPlaylistPath);
    if (!expectedPlaylist.exists())
    {
        try
        {
            String targetUrl = "http://" + connectionDetails.substring(connectionDetails.indexOf(":") + 2) + "";
            if(urlParameters.containsKey("targetUrl"))
            {
                String url = URLDecoder.decode(urlParameters.get("targetUrl"), CHARSET);
                targetUrl = url.substring(0, url.indexOf("?"));
            }
            HLSCreator hc = new HLSCreator(objectId, type, targetUrl);
            Thread hcThread = new Thread(hc);
            hcThread.start();
        }
        catch(Exception e)
        {
            new ErrorSender(e.toString(), socket, ostream).sendError();
            LOG.ERROR("Failed to create HLS stream " + e.toString());
        }
    }
    LOG.DEBUG("M3U8 requested");
    int count = 0;
    while(!expectedPlaylist.exists() && count <= 500)
    {
        Thread.sleep(50);
        count++;
    }
    if (count > 499)
    {
        new ErrorSender("Could not find playlist", socket, ostream).sendError();
        LOG.ERROR("Could not find playlist for id " + objectId);
```

```
            return;
        }
        BufferedReader br = new BufferedReader(new FileReader(expectedPlaylist));
        String newLine;
        while(true)
        {
            newLine = br.readLine();
            if (newLine.startsWith("#EXTINF:"))
            {
                break;
            }
        }
        br.close();
        Thread.sleep(waitBeforeSendingPlaylist);
        File actualPlaylist = new File(expectedPlaylistPath);
        byte [] bytearray  = new byte [(int)actualPlaylist.length()];
        FileInputStream fis = new FileInputStream(actualPlaylist);
        LOG.DEBUG("Requested File is " +  actualPlaylist.getCanonicalPath());
        BufferedInputStream bis = new BufferedInputStream(fis);
        bis.read(bytearray, 0, bytearray.length);
        status = "HTTP/1.1 200 OK" + CRLF;
        contentType = "Content-type: " + "application/x-mpegURL" + CRLF + CRLF;
        ostream.write(status.getBytes(Charset.forName("UTF-8")));
        ostream.write(contentType.getBytes(Charset.forName("UTF-8")));
        ostream.write(bytearray,0,bytearray.length);
        LOG.DEBUG("M3U8 file" + System.lineSeparator() + new String(bytearray) + System.lineSeparator());
        ostream.close();
        bis.close();
        breader.close();
        LOG.DEBUG("M3U8 delivered");
        return;
    }
    else if (mode.equals("file"))
    {
        String filePath;
        String objectId = URLDecoder.decode(urlParameters.get("objectId"), CHARSET);
        if (objectId.endsWith(".ts"))
        {
            String expectedFilePath = properties.getTempDirectory() + objectId;
            LOG.DEBUG("Expected file is " + expectedFilePath);
            File streamFile = fileReadyForTransmission(new File(expectedFilePath));
            byte [] bytearray  = new byte [(int)streamFile.length()];
            FileInputStream fis = new FileInputStream(streamFile);
            LOG.DEBUG("Requested File is " +  streamFile.getCanonicalPath());
```

```
            BufferedInputStream bis = new BufferedInputStream(fis);
            bis.read(bytearray, 0, bytearray.length);
            status = "HTTP/1.1 200 OK" + CRLF;
            contentType = "Content-type: " + "video/MP2T" + CRLF + CRLF;
            ostream.write(status.getBytes(Charset.forName("UTF-8")));
            ostream.write(contentType.getBytes(Charset.forName("UTF-8")));
            ostream.write(bytearray,0,bytearray.length);
            ostream.close();
            bis.close();
            breader.close();
            return;
        }
        else
        {
            Document doc = dataStore.getMap().get(objectId);
            String path = (String) doc.get("filePath");
            String file = (String) doc.get("fileName");
            String type = (String) doc.get("fileType");
            contentType = "Content-type: " + doc.get("Content-Type") + ";" + CRLF + CRLF;
            filePath = path + file + "." + type;
        }
        try
        {
            FileStreamer fs = new FileStreamer(filePath, socket, istream, contentType);
            Thread fileThread = new Thread(fs);
            fileThread.start();
            return;
        }
        catch(Exception e)
        {
            new ErrorSender(e.toString(), socket, ostream).sendError();
            LOG.ERROR("Cannot find file " + e.getLocalizedMessage());
            return;
        }
    }
    else if (mode.equals("helloWorld"))
    {
        status = "HTTP/1.1 200 OK" + CRLF;
        contentType = "Content-type: " + "text/html; charset=UTF-8" + CRLF + CRLF;
        ostream.write(status.getBytes(Charset.forName("UTF-8")));
        ostream.write(contentType.getBytes(Charset.forName("UTF-8")));
        ostream.write("Welcome!".getBytes(Charset.forName("UTF-8")));
        ostream.close();
        breader.close();
```

```
            return;
        }
        else
        {
            new ErrorSender("Mode not recognized", socket, ostream).sendError();
            LOG.ERROR("Mode " + mode + " not recognized");
            return;
        }
        ostream.write(status.getBytes(Charset.forName(CHARSET)));
        ostream.write(contentType.getBytes(Charset.forName(CHARSET)));
        ostream.write(encoding.getBytes(Charset.forName(CHARSET)));
        ostream.write(CRLF.getBytes(Charset.forName(CHARSET)));
        if (mode.equals("search"))
            {
            PrintWriter writer = new PrintWriter(ostream);
            JsonWriterSettings settings = new JsonWriterSettings(JsonMode.STRICT);
            writer.write("[");
            try
            {
                for (Document doc : docs )
                {
                    writer.print(doc.toJson(settings));
                    writer.write(",");
                    writer.write(CRLF);
                }
            }
            catch(Exception e)
            {
                new ErrorSender(e.toString(), socket, ostream).sendError();
            }
            finally
            {
                writer.write("]");
                writer.flush();
                writer.close();
            }
        }
        ostream.close();
        breader.close();
        socket.close();
    }

    private synchronized List<Document> simpleSearch(List<String> searchText)
    {
```

```
    List<Document> data = dataStore.getList();
    List<Document> result = new ArrayList<>();
    for (Document doc : data )
    {
        Set<String> setOfKeys = doc.keySet();
        boolean found = false;
        for ( String key : setOfKeys )
        {
            String val = doc.get(key).toString();
            for(String searchString : searchText)
            {
                if ( doc.get(key).toString().toLowerCase().contains(searchString.toLowerCase()))

                {
                    result.add(doc);
                    found = true;
                    break;
                }
            }
            if (found)
            {
                break;
            }
        }
    }
    LOG.DEBUG("Result is: " + result.toString());
    return result;
}

private List<Document> search(List<String> searchStrings) throws InterruptedException, ExecutionException
{
    //return simpleSearch(searchStrings);
    SpeedyGonzales sg = new SpeedyGonzales();
    sg.setPriority(Thread.MAX_PRIORITY);
    return sg.findString(searchStrings);
}

private File fileReadyForTransmission(File file) throws InterruptedException {
FileLock lock = null;
FileChannel channel = null;
if (file.lastModified() != 0L) // Has attribute last modified
{
    while( LOG.NOW().getTime() < (file.lastModified() + 50) )
    {
```

```
        Thread.sleep(10);
    }
}
try
{
   channel = new RandomAccessFile(file, "rw").getChannel();
   lock = channel.lock();
   LOG.DEBUG("Channel locked");
   int count = 0;
   while(!file.canWrite() && count < 500)
   {
       Thread.sleep(50);
   }
   lock.release();
   LOG.DEBUG("Channel released");
}
catch(IOException e)
{
    LOG.ERROR("Error accessing file " + e.toString());
}
finally
{
   if (channel != null)
   {
       try
       {
           channel.close();
       }
       catch(Exception e) {};
   }
}
   return file;
}
private Map<String, String> parseUrl(String url) throws UnsupportedEncodingException
{
    url = url.substring(url.indexOf("?") + 1, url.indexOf("HTTP/") -1);
    final Map<String, String> parameters = new LinkedHashMap<>();
    final String[] content = url.split("&");
    for (String kv : content)
    {
        int index = kv.indexOf("=");
        String key = URLDecoder.decode(kv.substring(0, index), "UTF-8");
        parameters.put(key, kv.substring(index + 1));
    }
```

```
        LOG.DEBUG("Parameters are " + parameters.toString());
        return parameters;
    }
}
```

4.9.3. MongoConnectorSingleton.java

This class provides connectivity to the MongoDB database

```
public class MongoConnectorSingleton {
    MongoClient mongoClient;
    MongoDatabase db;
    private static MongoConnectorSingleton instance = null;
    public static MongoConnectorSingleton getInstance()
    {
      if(instance==null)
      {
         instance = new MongoConnectorSingleton();
      }
      return instance;
    }

    void createMongoConnection()
    {
        try
        {
            mongoClient = new MongoClient( "127.0.0.1" );
            db = mongoClient.getDatabase("neptune");
            LOG.INFO("Successfully connected to databse: " + db.getName());
        }
        catch (Exception e)
        {
            LOG.CRITICAL("Connecttion failed " + e.toString());
        }
    }

    Set<String> listCollections()
    {
        Set<String> collections = new HashSet<>();
        LOG.DEBUG("Collections are: ");
        for (String collection : db.listCollectionNames() ) {
            LOG.INFO(collection + System.lineSeparator());
            collections.add(collection);
        }
        return collections;
    }
    public FindIterable<Document> readCollection(String collectionName)
    {
```

```
        LOG.DEBUG("There are " + db.getCollection(collectionName).count() + " documents in collection " + collectionName);
        return db.getCollection(collectionName).find();
    }
    public Document getDocument(String collectionName, String fileName, String filePath, String fileType)
    {
        FindIterable docs =
          db.getCollection(collectionName).find(and (eq("fileName", fileName), eq("filePath", filePath), eq("fileType",
fileType)) );
        return (Document) docs.first();
    }
    public void addRecord(Document doc)
    {
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'", Locale.ENGLISH);
        try
        {
            db.getCollection("files").insertOne(doc);
        }
        catch(Exception e)
        {
            LOG.CRITICAL("Mongo exception while inserting documents " + e.toString());

        }
    }
    public void addRecords(List<Document> listOfDocuments)
    {
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'", Locale.ENGLISH);
        try
        {
            db.getCollection("files").insertMany(listOfDocuments);
        }
        catch(Exception e)
        {
            LOG.CRITICAL("Mongo exception while inserting documents " + e.toString());

        }
    }


    public void clearCollection()
    {
        db.getCollection("files").drop();
        LOG.INFO("Collection \"files\" dropped");
        if ( !listCollections().contains("files"))
        {
```

```
            db.createCollection("files");
            LOG.INFO("New, empty collection \"files\" created");
        }
        else
        {
            LOG.ERROR("Error clearing collection \"files\"");
        }
    }
    void removeEntry(String collectionName, String fileName, String filePath, String fileType)
    {
        db.getCollection("files").deleteOne(this.getDocument(collectionName, fileName, filePath, fileType));
    }
    public void close()
    {
        mongoClient.close();
    }
}
```

4.9.4. DataParser.java

This class is responsible for extracting metadata from files, and storing it in persistence.

```
public class DataParser {
    static ConfigReaderSingleton properties = ConfigReaderSingleton.getInstance();
    Set<String> pathSet = new HashSet<>();
    DataStoreSingleton dataStore = DataStoreSingleton.getInstance();
    private final MongoConnectorSingleton db = MongoConnectorSingleton.getInstance();
    private List<Document> docList = new ArrayList<>();
    SambaConnector samba = new SambaConnector(
                properties.getProperty("sambaUser"),
                properties.getProperty("sambaPassword"),
                properties.getProperty("sambaServer"),
                properties.getProperty("sambaPath"));


    public Document parseSingleDoc(String filePath)
    {
        Document doc = new Document();
        Map fileDetails = splitPath(filePath);
        if (false /* add custom file type here */ )
        {
        }
        else
        {
            MetaDataParser_tikaGeneric parser = new MetaDataParser_tikaGeneric();
            parser.decodeFile(filePath, doc);
            try
            {
                doc = addFileDetails(fileDetails, doc, filePath);
            }
            catch(Exception e)
            {
                LOG.ERROR("Error writing details for file " + filePath);
            }
        }
        LOG.INFO("Just added to list: " + doc.toString() + System.lineSeparator());
        return doc;
    }


    void storeData() {
        for (String ii : samba.populatePathSet() )
```

```
    {
        docList.add(parseSingleDoc(ii));
    }
    writeToDb();
    LOG.DEBUG("Just added to database were: " + docList.size() + " documents");
}



private Document addFileDetails(Map fileDetails, Document doc, String filePath) throws SmbException
{
    FileHasher fh = new FileHasher();
    SmbFile file = null;
    try
    {
        file = new SmbFile(filePath);
    }
    catch(Exception e)
    {
        LOG.ERROR("Error accessing file " + filePath);
    }
    if (file != null)
    {
        doc.append("fileName", fileDetails.get("name"));
        doc.append("filePath", fileDetails.get("path"));
        doc.append("fileType", fileDetails.get("type"));
        doc.append("hash", fh.generateHash(file));
    }
    else
    {
        LOG.ERROR("Error writing details for file " + filePath);
    }
    return doc;
}


public Map<String, String> splitPath(String path)
{
    Map<String, String> fileDetails = new HashMap<>();
    fileDetails.put("path", path.substring(0, path.lastIndexOf('/') +1));
    fileDetails.put("name", path.substring(path.lastIndexOf('/') + 1, (path.lastIndexOf('.') )) );
    fileDetails.put("type", path.substring(path.lastIndexOf('.') +1 ));
    return fileDetails;
}
```

```
    public void readDb()
    {
        FindIterable<Document> iterable = db.readCollection("files");
        iterable.forEach(new Block<Document>()
        {
            @Override
            public void apply(final Document document)
            {
                dataStore.addItem(document.get("_id").toString(), document);
            }
        }
        );

        LOG.INFO(dataStore.getSize() + " documents added to dataStore");
    }
    public void writeToDb()
    {
        db.addRecords(docList);
    }
}
```

4.9.5. FileStreamer.java

This class sends actual files to the client.

```
public class FileStreamer implements Runnable {

    String filePath;
    ConfigReaderSingleton properties = ConfigReaderSingleton.getInstance();
    SambaConnector samba = new SambaConnector(
    properties.getProperty("sambaUser"),
    properties.getProperty("sambaPassword"),
    properties.getProperty("sambaServer"),
    properties.getProperty("sambaPath"));
    private final static String CRLF = "\r\n";
    Socket socket;
    DataOutputStream ostream;
    InputStream istream;
    BufferedReader breader;
    String contentType = "";
```

```
FileStreamer(String fp, Socket sock, InputStream is, String contentType) throws IOException
{
    filePath = fp;
    socket = sock;
    istream = is;
    ostream = new DataOutputStream(socket.getOutputStream());
    breader = new BufferedReader(new InputStreamReader(istream, "UTF-8"));
    this.contentType = contentType;
}

@Override
public void run()
{
    try
    {
        SmbFile smbFile = samba.getFile(filePath);
        byte [] bytearray  = new byte [(int)smbFile.length()];

        if (smbFile.getLastModified() != 0L) // Has attribute last modified
        {
            while( LOG.NOW().getTime() < (smbFile.getLastModified() + 100) ) // Has been modified in last 100ms; let's wait
            {
                Thread.sleep(50);
            }
        }
        BufferedInputStream bis = new BufferedInputStream(smbFile.getInputStream());
        bis.read(bytearray, 0, bytearray.length);

        String status = "HTTP/1.1 200 OK" + CRLF;
        ostream.write(status.getBytes(Charset.forName("UTF-8")));
        ostream.write(contentType.getBytes(Charset.forName("UTF-8")));
        ostream.write(bytearray,0,bytearray.length);
        bis.close();
        ostream.close();
        breader.close();
    }
    catch(Exception e)
    {
        new ErrorSender(e.toString(), socket, ostream).sendError();
        LOG.ERROR(e.getLocalizedMessage());
```

```
            }
        }

}
```

4.9.6. Maintenance.java

This class updates the index when any changes to the file system occur.

```
public class Maintenance implements Runnable {


    SambaConnector samba = new SambaConnector(
            properties.getProperty("sambaUser"),
            properties.getProperty("sambaPassword"),
            properties.getProperty("sambaServer"),
            properties.getProperty("sambaPath"));

    DataStoreSingleton ds = DataStoreSingleton.getInstance();
    MongoConnectorSingleton mongo = MongoConnectorSingleton.getInstance();

    void checkIfDBEntryNoLongerExistsOnSamba()
    {
        Map<String,Document> existingEntries = ds.getMap();
        List<String> docsToRemove = new ArrayList<>();
        for ( Map.Entry<String, Document> entry : existingEntries.entrySet() )
        {
            String file = (String) entry.getValue().get("fileName");
            String path = (String) entry.getValue().get("filePath");
            String type = (String) entry.getValue().get("fileType");
            String filePath = path + file + "." + type;
            try {
                if (!samba.getFile(filePath).exists())
                {
                    mongo.removeEntry("files", file, path, type);
                    docsToRemove.add(entry.getKey());
                }
            }
            catch (SmbException ex)
            {
                LOG.ERROR("Exception while trying to access " + path + " (" + entry.getKey() + ")");
            }
        }
        removeItemsFromDataStore(docsToRemove);
    }
```

```
void removeItemsFromDataStore(List<String> docs)
{
    for( String entry : docs)
    {
        ds.removeItem(entry);
        LOG.INFO("Maintenance removed id " + entry + " from document store");
    }
}

void addEntryIfWasAddedToSambaOrIfExistingEntryMatchesFile() throws NoSuchAlgorithmException, SmbException
{
    Map<String,Document> existingEntries = ds.getMap();
    for (String ii : samba.populatePathSet() )
    {
        boolean found = false;
        String entryToRemove = null;
        String path = null;
        String file = null;
        String type = null;
        for ( Map.Entry<String, Document> entry : existingEntries.entrySet() )
        {
            path = (String) entry.getValue().get("filePath");
            file = (String) entry.getValue().get("fileName");
            type = (String) entry.getValue().get("fileType");
            String filePath = path + file + "." + type;
            if (ii.equals(filePath))
            {
                try
                {
                    found = compareFiles(entry.getValue().get("hash").toString(), ii);
                }
                catch (Exception e)
                {
                    LOG.ERROR("Exception comparing file " + e.toString());
                    found = false;
                }
                entryToRemove = entry.getKey();
                break;
            }
        }
        if (!found)
```

```
            /* Because mongodb creates the unique id of an entry, it is
            necessary to commit the new entry to db first, and then query the
            db to get the object id in order to add it to the live data store.
            */
        {
            if (entryToRemove != null)
            {
                ds.removeItem(entryToRemove);
                mongo.removeEntry("files", file, path, type);
            }
            DataParser dp = new DataParser();
            Map<String, String> fileDetails = dp.splitPath(ii);
            Document doc = dp.parseSingleDoc(ii);
            mongo.addRecord(doc);
            doc = mongo.getDocument("files", fileDetails.get("name"), fileDetails.get("path"), fileDetails.get("type"));
            String id = doc.get("_id").toString();
            ds.addItem(id, doc);
            LOG.INFO("Maintenance added document " + doc.toString());
        }
    }
}

private boolean compareFiles(String docHash, String filePath) throws SmbException
{
    FileHasher fh = new FileHasher();
    String fileHash = null;
    SmbFile file = null;
    try
    {
        file = new SmbFile(filePath);
    }
    catch(Exception e)
    {
        LOG.ERROR("Could not access file " + filePath);
        return false;
    }
    fileHash = fh.generateHash(file);
    return docHash.equals(fileHash);
}
```

```
void clearTempDirectory(int olderThan)
{
    long date = new Date().getTime();
    File tempFolder = new File(properties.getTempDirectory());
    File[] files = tempFolder.listFiles();

    for (File file : files)
    {
        if ((date - file.lastModified() > olderThan ))
                {
                    file.delete();
                }
    }
}

@Override
public void run() {
    while(true)
    {
        LOG.INFO("Starting Maintenance");

        try
        {
            addEntryIfWasAddedToSambaOrIfExistingEntryMatchesFile();
        }
        catch(Exception e)
        {
            LOG.ERROR("Error during file maintenance");
        }
        checkIfDBEntryNoLongerExistsOnSamba();
        clearTempDirectory((14400)*1000);
        LOG.INFO("Maintenance completed");
        try
        {
            Thread.sleep(Integer.parseUnsignedInt(properties.getProperty("maintenanceInterval")) * 60 * 1000 );
        } catch (InterruptedException ex)
        {
            LOG.ERROR("Error doing nothing in particular");
        }
    }
}
```

```
}
```

4.10. Risk analysis

4.10.1. Table of identified risks

Risks that were initially identified, but have successfully been mitigate are portrayed with a grey background. In this case, the “mitigation strategy” column reflects what was done in order to mitigate the identified risks.

Entries with a white background represent risks that remain associated with the project software beyond the point of its completion. In this case, the “mitigation strategy” column reflects recommendations for mitigating the identified risk in the future.

Risk	Probability	Initial impact assessment	Mitigation strategy
iOS App development. Difficulties in creating a suitable front-end app, that interfaces with the media facilities of the device, and connects to the server component.	Medium – it is uncertain whether the required skill will be acquired in good time.	Low - I have no experience with iOS App development. However, there is ample documentation of excellent quality, and observation of friends leads me to believe that a fairly low level of expertise is required to create a basic app, such as the one I propose. As long as basic functionality is provided, the impact of a clunky user interface is limited for my project, as it tries to solve a different problem	Viewing the Developing iOS 8 Apps with Swift lectures Paul Hegarty gave at Stanford University in 2015 (Hegraty, 2015) proved to be an excellent introduction into the app development workflow, and a great way to familiarise myself with the fundamental concepts and principles of Swift. Further, Apple’s own iOS Developer Library (Apple, 2016g) is very comprehensive, and only occasionally, I ventured to other sources to solve problems I experienced along the way. Aside from a few usability issues, app development is now complete.

Difficulties with the server core component development, i.e. Creating an executable that runs on the server.	Low – it is certain that the required skill will be acquired in good time.	Low - Creating a basic Java application formed part of M362 and M250, and I have similar work-related experience, albeit with C++ applications.	M362 and M250 proved a good basis for developing the Java server component, with additional queries answered by the official Oracle Java documentation (Oracle, 2016d). The documentation of each of the related libraries was sufficient to complete integration.
Server / Client communication Getting the client and server to communicate, and exchange the relevant information.	Medium – this skill is partially based on M362, so it is chiefly the client I need to be concerned with.	High – if the client fails to connect to the server, the desired outcome of the project cannot be achieved.	Server-side, the Oracle documentation for the Java Http Server class proved sufficient (Oracle, 2016e). Client-side, the Apple iOS Developer Library (Apple, 2016g) gives ample examples how to connect to a server for the required resource. Apple stipulates HLS is the only permissible protocol for transferring media of unknown size to an iOS device (Apple, 2016a), thereby removing the need to consider alternatives. Details for the HLS implementation were obtained from the FFmpeg project documentation (FFmpeg 2016).

MongoDB database operations. Using the MongoDB interface to interact with the database.	Low – it is certain that the required skill will be acquired in good time	Low - MongoDB has a well-defined Java API, I have worked with MongoDB before, and – if needed – have help at hand.	Having had prior hands-on experience with MongoDB, I was able to integrate MongoDB quickly, fetching any missing information from the MongoDB Java Driver documentation (MongoDB, 2016).
Representing data retrieved from the database in Java objects.	Low – it is certain that the required skill will be acquired in good time	Low – the MongoDB Java library provides the necessary classes.	The current implementation of the index in Java works well.
Understanding how to make the file system work for my purpose.	High – it is unknown whether the required skill will be acquired in good time.	High - I have little trouble with working with the underlying Linux file system (xfs in this case), but am less familiar interfacing with samba. There are many unknowns on the client side. Should the client separately connect to the same SMB share? Should the server stream the file to the client? Will the client hold a copy of the file? Are there elements of SMB I can work with directly? A failure to understand the implications of the options available could have a direct impact on the usability of my software.	Having found the JCIFS library (The JCFIS project, 2014) for implementing samba in Java mitigated this risk, as this mature and well-documented library made implementation very easy.

Optimising HLS	Medium – it is uncertain whether all the problems can be fixed before project completion.	Low – the project software works, and demonstrates its purpose, without HLS. However, HLS is a concern for submission of the client app to the Apple App store.	A logic error was discovered that resulted in a malformed playlist, which was the root cause of the observed playback issues. This has now been fixed.
Implementing automatic updates of the index on file system changes	Medium – it is uncertain whether this goal will be achieved, as no research has been done yet.	High – for most live systems, it is very important to be able to update an index on the fly, both adding and removing information. Rerunning the indexing process from scratch is not a suitable workaround, as it takes too long.	A solution was implemented, which is a scan using hashing algorithms to detect changes, such as were discussed in M362.
Adding metadata parsing for additional file types	Low – it can be assumed that this is easily achievable.	Medium – one of the reasons I wanted to do this project was to provide a solution for myself to access my music library. Restricted only to files of type mp3, the solution would be rather unsatisfactory. From a general perspective however, the concept is proven, and the lack of a specific implementation feature is less of an issue.	Implementation of the Apache Tika library (Apache, 2016c) now allows Nep-Tune to understand many 100s of different file types.
Adding sophisticated playback features	Low – this is easily achievable.	Low – only usability will be increased with additional playback features.	The addition of sophisticated playback features was described in the Epilogue; they were ultimately achieved by leveraging the built-in playback interface.

Optimising server communication efficiency	Medium – it is uncertain whether the required skill will be acquired in good time.	Low – inefficient communication will have a performance impact, but not otherwise affect the solution.	Apple's restriction to the use of the HLS protocol prescribes the use of communication relatively strictly, meaning this risk is mitigated by the limited number of protocols available for use (Apple, 2016e). Further ways to improve efficiency, e.g. by using compression, can be evaluated during the dedicated time slot, later in the project (this was not achieved on time).

Table 7: Risk assessment

4.11. Evaluation against existing solutions

Existing solutions can roughly be grouped into two categories:

1) Media-focused solutions, i.e. software, the primary purpose of which is to manage and play back media files

2) Document-management systems, i.e. software, the primary purpose of which is to manage collections of documents.

The proposed solutions bridges both categories, by aiming to deliver some elements typically found in document-management systems (e.g. indexing of content), and other elements typically found in media-focused solutions (e.g. playback of media files).

For this reason, an approach to an evaluative comparison should take into account both document-management systems, and media-focused solutions.

It appears that many document-management solutions are aimed at a particular industry sector, but not a sector I necessarily have personal experience with.
Brooks (2016) reviewed 56 relatively universal commercial document management systems, of which he highlights five as being particularly adaptable, while Lixandroiu et al. (2015) offer a further three open source packages suitable for evaluation, providing a basis to start the evaluation process.

Finding media-focused solutions proved more difficult; Wikipedia (2016c) sports a list of related media software. Here, I particularly looked at the iOS section, since my own client is written for iOS. Ultimately, I chose the several I had prior experience with, namely Airplay, Kodi, Oplayer, PlayerExtreme and Twonky, having discounted those applications tied to a particular brand of hardware.

Below table 2 shows the feature comparison analysis, which was conducted by using the best information available from the documentation or presentation of the product. With the exception of Airplayer, Kodi, and Twonky, no third-party product was installed and test to aid this evaluation.

	Nep-Tune (this project)	Kuali (2016)	ERPNext (2016)	PinPoint Docuware (2016)	Dokmee Cloud (OfficeGemini, 2016)	LogicalDOC (2016)	FileHold (2016)	Odoo (2016)	Airplayer (Edavs, 2016)	Twonky (2016)	Kodi (2016)	Oplayer (Edavs, 2016)	Player Extreme (Pentaloop, 2016)
Open source	✓	✓	✓	×	×	✓*	×	✓	×	×	✓	×	×
Cross-platform	✓	×	×	⊘	×	✓	×	×	×	✓	✓	×	×
HTTP interface	✓	×	✓	×	✓	✓	×	⊘	×	✓	✓	×	×
Accessible data store (e.g. via samba)	✓	×	×	×	×	✓	⊘	×	✓	✓	✓	✓	✓
Support for media files	✓	×	⊘	×	×	×	×	⊘	✓	✓	✓	✓	✓
Metadata parsing	✓	✓	✓	⊘	✓	✓	⊘	✓	✓	✓	✓	✓	✓
API	✓	✓	×	✓	✓	✓	×	✓	×	×	✓	×	×
real-time search	✓	⊘	⊘	⊘	⊘	⊘	×	⊘	×	×	×	×	×

Table 8: Comparative feature evaluation of software solutions similar to Nep-Tune

✓ yes, × no, ⊘ cannot be determined from the information available.
** The open source version of LogicalDOC excludes some features of the paid-for version.*

In general, it can be said that typical document management solutions will take ownership of the documents they manage, by saving them to a database, and allow write modifications, whilst not offering a solution for the retrieval and playback of multimedia files.
In contrast, typical multimedia applications do not offer an indexing solution. One exception is Twonky, which unfortunately incurs a license fee, is restricted to only a small subset of available indexing information, such as title and artist, and the iOS client no longer starts up on iOS 9.
Both types of solutions offer search, but in all cases, the search of the media-focused solutions was restricted to searching the local device, not an attached network share.

4.12. The installation script

The installation script fulfils the following tasks:

- installs package dependencies
- configures MongoDB and auto-starts it
- fetches the ffmpeg source code and compiles it
- updates firewall rules to allow the Nep-Tune service
- creates a systemd service script for Nep-Tune
- configures the service to auto-start
- creates a configuration file
- prompts the user to enter credentials for the samba share
- creates a log file
- sets a few necessary variables

```
#!/bin/bash
# This script configures the server for Nep-Tune

####################Setting
variables#####################################################################
installingUser=$(who am i | awk '{print $1}')  #The user who is executing this
script should also be the user running Nep-Tune
packageManager=apt-get #Package manager, e.g. apt-get for Ubuntu, yum for CentOS /
RedHat etc
tempDirectory=/tmp #Temp directory for NepTune is primarily used for live
transcoded HLS files
thisDirectory=$(pwd)
yellow='\E[1;33m'
wipe="\033[1m\033[0m"

#Check user executed using "sudo"
executingUser=$(whoami)
if [ "$executingUser" != "root" ];
then
echo "Executing user is $executingUser"
echo "Script must be run with \"sudo\""
exit
fi

function getPassword {
read -s -p "Please enter the password for the samba share`echo $'\n> '`" passWord1
echo
read -s -p "Please re-enter the password`echo $'\n> '`" passWord2
echo
}

####################Reading in configuration via user
prompt#########################################
printf "$yellow";
read -p "Please enter the path to your samba share (e.g.
smb://my.server.net/share/), including the / at the end`echo $'\n> '`" sambaServer
read -p "Please enter the folder containing your files on above share (e.g.
ThisFolder/ThatFolder/), including the / at the end`echo $'\n> '`" sambaPath
read -p "Please enter the user name to access this share`echo $'\n> '`" sambaUser
getPassword

while [ "$passWord1" != "$passWord2" ]
do
echo Passwords do not match, please try again
getPassword
done
sambaPassword=${passWord1}
read -p "Would you like to initialize Nep-Tune after installation [Y/n]?`echo $'\n>
'`" initializeAfterInstallation
```

```
printf "$yellow"; echo Fetching public key for MongoDB; printf "$wipe"
 sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
 echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2 multiverse" |
sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
printf "$yellow";echo Updating;printf "$wipe";
 sudo $packageManager update -y
printf "$yellow";echo Installing dependencies; printf "$wipe";
 sudo $packageManager -y install autoconf automake build-essential libass-dev
libfreetype6-dev libsdl1.2-dev libtheora-dev libtool libva-dev libvdpau-dev
libvorbis-dev libxcb1-dev libxcb-shm0-dev libxcb-xfixes0-dev pkg-config texinfo
zlib1g-dev mongodb-org default-jre default-jdk yasm libx264-dev gcc make
printf "$yellow";echo Opening firewall port 7701; printf "$wipe";
 sudo ufw allow 7701/tcp
 mkdir ~/ffmpeg_sources
printf "$yellow";echo Compiling YASM; printf "$wipe";
 cd ~/ffmpeg_sources
 wget http://www.tortall.net/projects/yasm/releases/yasm-1.3.0.tar.gz
 tar xzvf yasm-1.3.0.tar.gz
 cd yasm-1.3.0
 sudo ./configure --prefix="$HOME/ffmpeg_build" --bindir="$HOME/bin"
 sudo make
 sudo make install
 sudo make distclean
printf "$yellow";echo Complling AAC encoder; printf "$wipe";
 cd ~/ffmpeg_sources
 wget -O fdk-aac.tar.gz https://github.com/mstorsjo/fdk-aac/tarball/master
 tar xzvf fdk-aac.tar.gz
 cd mstorsjo-fdk-aac*
 sudo autoreconf -fiv
 sudo ./configure --prefix="$HOME/ffmpeg_build" --disable-shared
 sudo make
 sudo make install
 sudo make distclean
printf "$yellow";echo Compiling FFmpeg; printf "$wipe";
 cd ~/ffmpeg_sources
 wget http://ffmpeg.org/releases/ffmpeg-snapshot.tar.bz2
 tar xjvf ffmpeg-snapshot.tar.bz2
 cd ffmpeg
 PATH="$HOME/bin:$PATH"
 PKG_CONFIG_PATH="$HOME/ffmpeg_build/lib/pkgconfig"
 sudo ./configure --prefix="$HOME/ffmpeg_build" --pkg-config-flags="--static" --
extra-cflags="-I$HOME/ffmpeg_build/include" --extra-ldflags="-
L$HOME/ffmpeg_build/lib" --bindir="$HOME/bin" --enable-gpl --enable-libass --
enable-libfdk-aac --enable-libfreetype --enable-libx264 --enable-nonfree
 sudo PATH="$HOME/bin:$PATH"
 sudo make
 sudo make install
 sudo make distclean
 sudo hash -r
printf "$yellow";echo Writing MongoDB config for systemd; printf "$wipe";

sudo cat > /lib/systemd/system/mongod.service << EOF
[Unit]
Description=High-performance, schema-free document-oriented database
After=network.target
Documentation=https://docs.mongodb.org/manual

[Service]
User=mongodb
Group=mongodb
ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf

[Install]
WantedBy=multi-user.target
EOF

 sudo systemctl daemon-reload
```

```
printf "$yellow";echo Starting and enabling MongoDB; printf "$wipe";
 sudo service mongod start
 sudo systemctl enable mongod.service
ffmpeg=$(sudo runuser -l $installingUser -c 'which ffmpeg')

##########################################Config
File################################################
printf "$yellow";echo Writing configuration file to /etc/nep-tune.properties;
printf "$wipe";
sudo cat > /etc/nep-tune.properties << EOF
# For supported file types, see: https://tika.apache.org/1.13/formats.html

# Set credentials for samba share, taking care to get the / in the right place
e.g.:
# sambaServer=smb://192.168.1.11/
# sambaUser=joe
# sambaPassword=passwordOfJoe
# sambaPath=Music/HardStuff/EvenHarder/
# Please note that entries are Case Sensitive

sambaServer=$sambaServer
sambaUser=$sambaUser
sambaPassword=$sambaPassword
sambaPath=$sambaPath

# Set logLevel one of debug, info, warning, error, critical. Default is warning.
Debug logging is expensive!
logLevel=warning

# HTTP Server properties
httpServerPort=7701

# Specify temp directory used for creating HLS stream file (/tmp/) by default
# tempDirectory=$tempDirectory

# Path to ffmpeg executable, specified during installation using 'which ffmpeg'
ffmpegPath=$ffmpeg

# Maintenance interval in minutes.
maintenanceInterval=15

# Playlist delay in ms. For HLS, the first .ts file needs to be available for the
client to retrieve
# before the playlist is sent (the playlist is written as stream files are
created).
# The code recognises when the first entry in the playlist is present, but it does
not know when
# the first stream file is available. 750ms after transcoding was started seems a
good compromise.
# Decrease this value to reduce the time between selecting a stream and starting
playback.
# Increase this value if nothing plays back after selecting a stream for playback.
waitBeforeSendingPlaylist=750
EOF

###################################Service#########################################
printf "$yellow";echo Writing service to /lib/systemd/system/nep-tune.service;
printf "$wipe";
sudo cat > /lib/systemd/system/nep-tune.service << EOF
[Unit]
Description=Nep-tune indexing service
DefaultDependencies=no
Before=networking.service

[Service]
Type=forking
RemainAfterExit=no
ExecStart=${thisDirectory}/src/NepTune/runNeptune --daemon
```

```
ExecStop=pid="\${ps aux | grep NepTune | awk '{print \$2}'}"
ExecStop=kill \${pid}

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload

#################################Run script#########################################
printf "$yellow";echo Writing run script to
${thisDirectory}/src/NepTune/runNeptune.sh; printf "$wipe";
sudo cat > ${thisDirectory}/src/NepTune/runNeptune.sh << EOF
#!/bin/bash
#
echo Compiling
javac -g -Xlint:unchecked  -Xlint:deprecation -classpath
.:${thisDirectory}/dist/lib/* ${thisDirectory}/src/NepTune/*.java
echo Finished compiling

#java -Xdebug -Xrunjdwp:transport=dt_socket,address=8800,server=y,suspend=y -cp
${thisDirectory}/src:${thisDirectory}/dist/lib/* NepTune.NepTune \$1 \$2 \$3
java -cp ${thisDirectory}/src:${thisDirectory}/dist/lib/* NepTune.NepTune \$1 \$2
\$3 >> /var/log/nep-tune.log 2>&1 &
EOF

sudo chmod +x ${thisDirectory}/src/NepTune/runNeptune.sh
sudo systemctl enable nep-tune.service # auto-start nep-tune

if [[ $initializeAfterInstallation =~ ^[Yy]$ ]]
then
${thisDirectory}/src/NepTune/runNeptune.sh --init
fi

printf "$yellow";
read -p "Installation has completed. Do you want to run the Nep-Tune daemon now
(y/n)?`echo $'\n> '`" runDaemon
if [[ $runDaemon =~ ^[Yy]$ ]]
then
service nep-tune start;
fi
echo "Type \"less +F /var/log/nep-tune.log\" to check the logfile"
echo Script complete
printf "$wipe";
```

4.13. Testing

4.13.1. Testing Maintenance.java

The purpose of this test is to ensure that:

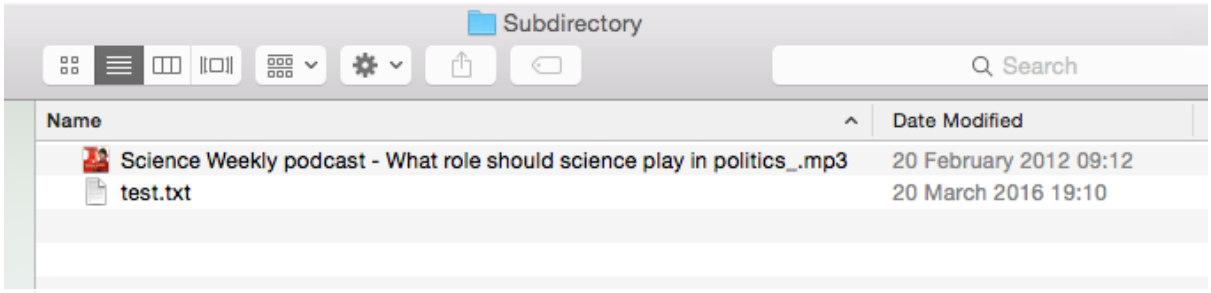
a) Files added to the samba share are added to the index

b) Files removed from the samba share are removed from the index

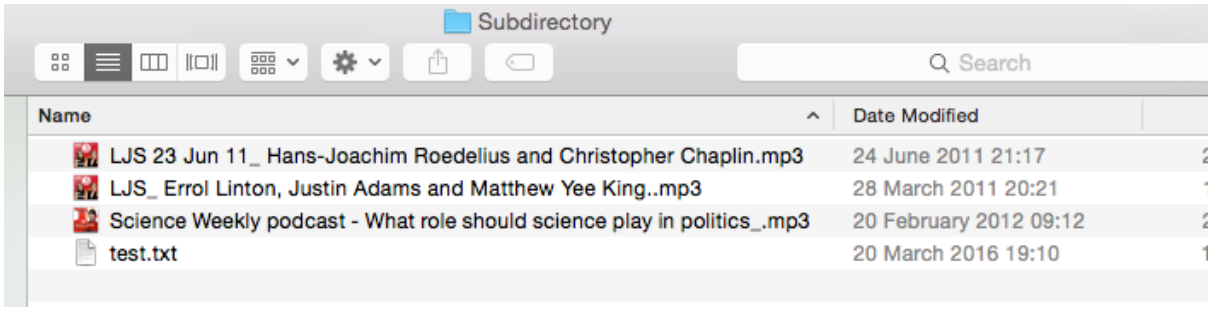
c) Files changed on the samba share are updated in the index

a)

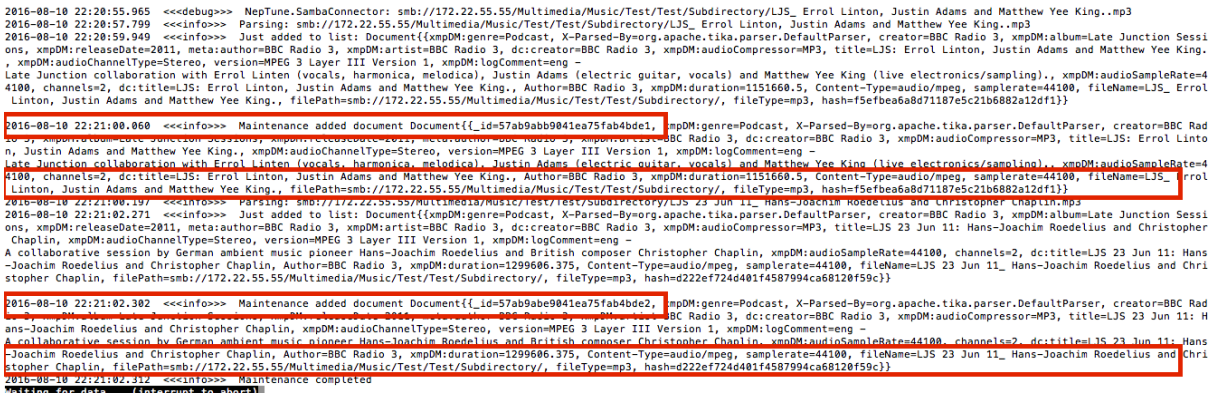
1) Starting out, we have two files present, “Science Weekly podcast - What role should science play in politics_.mp3” and “test.txt”



2) Two files were dragged into the directory, “LJS 23 Jun 11_ Hans-Joachim Roedelius and Christopher Chaplin.mp3” and “LJS_ Errol Linton, Justin Adams and Matthew Yee King..mp3”.



3) Nep-tune logging confirm these were added to the index shortly after:



4) Correct addition can be confirmed by querying the database, for example for a segment of the title of added files:

```
mongo neptune --eval "db.files.find({"title":/Linton/})"
```

```
toby@ubuntu:~$ mongo neptune --eval "db.files.find({"title":/Linton/})"
MongoDB shell version: 3.2.6
connecting to: neptune
{ "_id" : ObjectId("57ab9abb9041ea75fab4bde1"), "xmpDM:genre" : "Podcast", "X-Parsed-By" : "org.apache.t
ika.parser.DefaultParser", "creator" : "BBC Radio 3", "xmpDM:album" : "Late Junction Sessions", "xmpDM:r
eleaseDate" : "2011", "meta:author" : "BBC Radio 3", "xmpDM:artist" : "BBC Radio 3", "dc:creator" : "BBC
 Radio 3", "xmpDM:audioCompressor" : "MP3", "title" : "LJS: Errol Linton, Justin Adams and Matthew Yee K
ing.", "xmpDM:audioChannelType" : "Stereo", "version" : "MPEG 3 Layer III Version 1", "xmpDM:logComment"
 : "eng - \nLate Junction collaboration with Errol Linten (vocals, harmonica, melodica), Justin Adams (e
lectric guitar, vocals) and Matthew Yee King (live electronics/sampling).", "xmpDM:audioSampleRate" : "4
4100", "channels" : "2", "dc:title" : "LJS: Errol Linton, Justin Adams and Matthew Yee King.", "Author"
: "BBC Radio 3", "xmpDM:duration" : "1151660.5", "Content-Type" : "audio/mpeg", "samplerate" : "44100",
"fileName" : "LJS_ Errol Linton, Justin Adams and Matthew Yee King.", "filePath" : "smb://172.22.55.55/M
ultimedia/Music/Test/Test/Subdirectory/", "fileType" : "mp3", "hash" : "f5efbea6a8d71187e5c21b6882a12df1
" }
```

```
mongo neptune --eval "db.files.find({"title":/Roedelius/})"
```

```
toby@ubuntu:~$ mongo neptune --eval "db.files.find({"title":/Roedelius/})"
MongoDB shell version: 3.2.6
connecting to: neptune
{ "_id" : ObjectId("57ab9abe9041ea75fab4bde2"), "xmpDM:genre" : "Podcast", "X-Parsed-By" : "org.apache.t
ika.parser.DefaultParser", "creator" : "BBC Radio 3", "xmpDM:album" : "Late Junction Sessions", "xmpDM:r
eleaseDate" : "2011", "meta:author" : "BBC Radio 3", "xmpDM:artist" : "BBC Radio 3", "dc:creator" : "BBC
 Radio 3", "xmpDM:audioCompressor" : "MP3", "title" : "LJS 23 Jun 11: Hans-Joachim Roedelius and Christo
pher Chaplin", "xmpDM:audioChannelType" : "Stereo", "version" : "MPEG 3 Layer III Version 1", "xmpDM:log
Comment" : "eng - \nA collaborative session by German ambient music pioneer Hans-Joachim Roedelius and B
ritish composer Christopher Chaplin", "xmpDM:audioSampleRate" : "44100", "channels" : "2", "dc:title" :
"LJS 23 Jun 11: Hans-Joachim Roedelius and Christopher Chaplin", "Author" : "BBC Radio 3", "xmpDM:durati
on" : "1299606.375", "Content-Type" : "audio/mpeg", "samplerate" : "44100", "fileName" : "LJS 23 Jun 11_
 Hans-Joachim Roedelius and Christopher Chaplin", "filePath" : "smb://172.22.55.55/Multimedia/Music/Test
/Test/Subdirectory/", "fileType" : "mp3", "hash" : "d222ef724d401f4587994ca68120f59c" }
```

b)

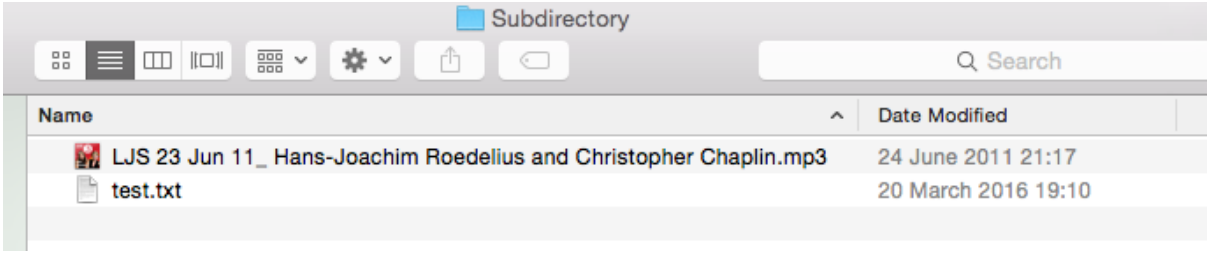
1) Confirm that the index contains the item “Science Weekly podcast - What role should science play in politics_.mp3”

```
mongo neptune --eval "db.files.find({"fileName":/role/})"
```

```
toby@ubuntu:~$ mongo neptune --eval "db.files.find({"fileName":/role/})"
MongoDB shell version: 3.2.6
connecting to: neptune
{ "_id" : ObjectId("5793510d9041ea1bff316639"), "xmpDM:genre" : "Podcast", "X-Parsed-By" : "org.apache.t
ika.parser.DefaultParser", "creator" : "guardian.co.uk", "xmpDM:album" : "Science Weekly", "xmpDM:releas
eDate" : "2012", "meta:author" : "guardian.co.uk", "xmpDM:artist" : "guardian.co.uk", "dc:creator" : "gu
ardian.co.uk", "xmpDM:audioCompressor" : "MP3", "title" : "Science Weekly podcast: What role should scie
nce play in politics?", "xmpDM:audioChannelType" : "Stereo", "version" : "MPEG 3 Layer III Version 1", "
xmpDM:logComment" : "eng - iTunNORM\n 00000174 00000173 00008EAE 00008EAE 000A8FAC 000A8FAC 00008590 000
0858F 0006020F 0006020F\u0000", "xmpDM:audioSampleRate" : "44100", "channels" : "2", "dc:title" : "Scien
ce Weekly podcast: What role should science play in politics?", "Author" : "guardian.co.uk", "xmpDM:dura
tion" : "1807293.5", "Content-Type" : "audio/mpeg", "samplerate" : "44100", "fileName" : "Science Weekly
 podcast - What role should science play in politics_", "filePath" : "smb://172.22.55.55/Multimedia/Musi
c/Test/Test/Subdirectory/", "fileType" : "mp3", "hash" : "9de794d9544f6884dea03cb97809e80a" }
```

2) Two files are now removed from the index, “Science Weekly podcast - What role should science play in politics_.mp3”, and “LJS_ Errol Linton, Justin Adams and Matthew Yee King..mp3” (the

presence of which we confirmed above).



3) Nep-Tune logging confirm the removal of those files

```
2016-08-10 22:39:08.061  <<<info>>>  Maintenance removed id 5793510d9041ea1bff316639 from document store
2016-08-10 22:39:08.061  <<<info>>>  Maintenance removed id 57ab9abb9041ea75fab4bde1 from document store
2016-08-10 22:39:08.062  <<<info>>>  Maintenance completed
```

4) The query from b) repeated should now not yield any result:

```
toby@ubuntu:~$ mongo neptune --eval "db.files.find({"fileName":/role/})"
MongoDB shell version: 3.2.6
connecting to: neptune
toby@ubuntu:~$
```

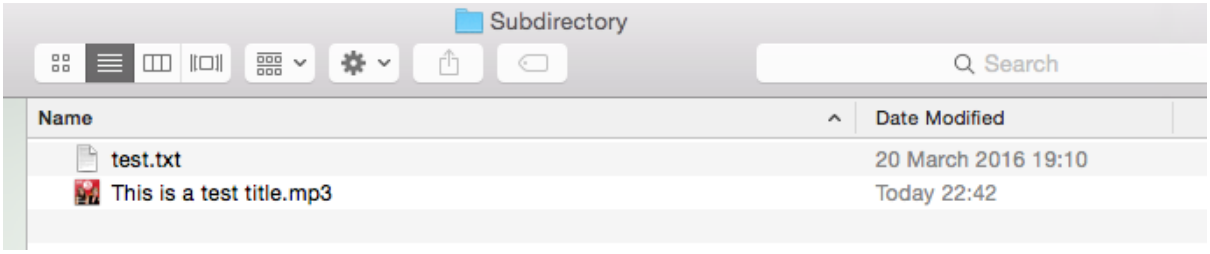
Likewise, the originally present file should have disappeared:

```
toby@ubuntu:~$ mongo neptune --eval "db.files.find({"title":/Linton/})"
MongoDB shell version: 3.2.6
connecting to: neptune
toby@ubuntu:~$
```

c)

1)

Remaining file “LJS 23 Jun 11_ Roedelius and Christopher Chaplin.mp3” was renamed to “This is a test title.mp3”



2)

The Nep-Tune logs confirm the “new” file was added, whilst the old entry was removed.

```
2016-08-12 21:38:46.930  <<<info>>>  Maintenance added document Document{{_id=57ae33d69041ea0bef07490a, xmpDM:genre=Podcast, X-Parsed-By=or
g.apache.tika.parser.DefaultParser, creator=BBC Radio 3, xmpDM:album=Late Junction Sessions, xmpDM:releaseDate=2011, meta:author=BBC Radio
3, xmpDM:artist=BBC Radio 3, dc:creator=BBC Radio 3, xmpDM:audioCompressor=MP3, title=LJS 23 Jun 11: Hans-Joachim Roedelius and Christopher
 Chaplin, xmpDM:audioChannelType=Stereo, version=MPEG 3 Layer III Version 1, xmpDM:logComment=eng -
A collaborative session by German ambient music pioneer Hans-Joachim Roedelius and British composer Christopher Chaplin, xmpDM:audioSampleR
ate=44100, channels=2, dc:title=LJS 23 Jun 11: Hans-Joachim Roedelius and Christopher Chaplin, Author=BBC Radio 3, xmpDM:duration=1299606.3
75, Content-Type=audio/mpeg, samplerate=44100, fileName=This is a test title.mp3, filePath=smb://172.22.55.55/Multimedia/Music/Test/Test/Su
bdirectory/, fileType=mp3, hash=76262befcb34d956c39e47556efc052b }
2016-08-12 21:38:47.128  <<<info>>>  Maintenance removed id 57ae33789041ea0bb4a60640 from document store
2016-08-12 21:38:47.129  <<<info>>>  Maintenance completed
```

3)
Querying for the new file name shows the renamed file can be found in the database:

mongo neptune --eval "db.files.find({"fileName":/title/})"

```
toby@ubuntu:~/src/NepTune/src/NepTune$ mongo neptune --eval "db.files.find({"fileName":/title/})"
MongoDB shell version: 3.2.6
connecting to: neptune
{ "_id" : ObjectId("57ae33d69041ea0bef07490a"), "xmpDM:genre" : "Podcast", "X-Parsed-By" : "org.apache.tika.parser.DefaultParser", "creator
" : "BBC Radio 3", "xmpDM:album" : "Late Junction Sessions", "xmpDM:releaseDate" : "2011", "meta:author" : "BBC Radio 3", "xmpDM:artist" :
"BBC Radio 3", "dc:creator" : "BBC Radio 3", "xmpDM:audioCompressor" : "MP3", "title" : "LJS 23 Jun 11: Hans-Joachim Roedelius and Christop
her Chaplin", "xmpDM:audioChannelType" : "Stereo", "version" : "MPEG 3 Layer III Version 1", "xmpDM:logComment" : "eng - \nA collaborative
session by German ambient music pioneer Hans-Joachim Roedelius and British composer Christopher Chaplin", "xmpDM:audioSampleRate" : "44100"
, "channels" : "2", "dc:title" : "LJS 23 Jun 11: Hans-Joachim Roedelius and Christopher Chaplin", "Author" : "BBC Radio 3", "xmpDM:duration
" : "1299606.375", "Content-Type" : "audio/mpeg", "samplerate" : "44100", "fileName" : "This is a test title.mp3", "filePath" : "smb://172.
22.55.55/Multimedia/Music/Test/Test/Subdirectory/", "fileType" : "mp3", "hash" : "76262befcb34d956c39e47556efc052b" }
```

Likewise, searching for the old file name asserts it is no longer in the database:

```
toby@ubuntu:~/src/NepTune/src/NepTune$ mongo neptune --eval "db.files.find({"fileName":/Hans-Joachim/})"
MongoDB shell version: 3.2.6
connecting to: neptune
```

Thus, it can be concluded that the implementation of the Maintenance class was successful.

4.13.2. Functional testing

Test cases were designed based on the recommendations for software testing in TM354, “Unit 11 – Strategy for creating test cases” (The Open University, 2015g).

4.13.2.1. Test case 1

Purpose

The purpose of this test is fourfold:

1) Verify that metadata is read in correctly

2) Verify that metadata is persisted correctly in the database

3) Verify that multiple files in a collection can be handled

4) Verify that files of unrecognised type are ignored

Fixture

The fixture prepares a well-known set of source file for indexing, the correctness of which can be cross-referenced by examining the source files with a third-party tool for metadata, such as iTunes (Apple, 2016b).
Required are:

- a shared folder containing files “A journey to the heart of the planet we made - podcast.mp3” (The Guardian, 2014), “Russian Red Army Choir - National Anthem Russia (1977).mp3” (Marxists Internet Archive (2007)) and a subdirectory called “Subdirectory” containing files “September 2010 - Beyond limits.mp3” (The Royal Society, 2010) and “test.txt”

- Server component is configured to access above shared folder

Actions

1) Run server component with “--init” flag.

2) Execute “mongo neptune --eval "db.files.find()"”

Expected Outcome

1) The metadata processed should match the metadata stored with the file, as shown in figure 8

2) The database query should match the metadata, in tag and content

3) The above shall be true for multiple files in the collection

4) Unrecognised files shall not appear in the database.

The following text should be printed on the shell, which can be used to verify the expected outcome, as described above:

MongoDB shell version: 3.2.3

```
connecting to: neptune
```

```
{ "_id" : ObjectId("56fd8b85c59640120deb2985"), "filePath" : "smb://<share>",
"fileName" : "Russian Red Army Choir - National Anthem Russia (1977)", "fileType" :
"mp3" }
```

{ "_id" : ObjectId("56fd8b85c59640120deb2986"), "filePath" : "smb://<share>Subdirectory/", "fileName" : "September 2010 - Beyond limits", "fileType" : "mp3", "album" : "The Royal Society - R.Science", "artist" : "The Royal Society", "comment" : "0", "data_length" : 11047, "frame_sets" : "{COMM=COMM: 2, PCST=PCST: 1, TALB=TALB: 1, TCAT=TCAT: 1, TCON=TCON: 1, TDES=TDES: 1, TGID=TGID: 1, TIT2=TIT2: 1, TIT3=TIT3: 1, TKWD=TKWD: 1, TPE1=TPE1: 1, TYER=TYER: 1, WFED=WFED: 1}", "genre_description" : "Podcast", "itunes_comment" : " 00000055 00000055 00000D44 00000D44 000AFBF9 000AFBF9 00004C74 00004C74 000BEA3F 000BEA3F", "length" : 11057, "title" : "September 2010 - Beyond limits", "version" : "3.0", "year" : "2010" }

{ "_id" : ObjectId("56fd8b85c59640120deb2987"), "filePath" : "smb://<share>Subdirectory/", "fileName" : "test", "fileType" : "txt" }

{ "_id" : ObjectId("56fd8b85c59640120deb2988"), "filePath" : "smb://<share>", "fileName" : "A journey to the heart of the planet we made - podcast", "fileType" : "mp3", "album" : "Science Weekly", "artist" : "guardian.co.uk", "comment" : " 000002AD 00000000 0000AF41 00000000 00015087 00000000 000084E8 00000000 0024F7C6 00000000", "composer" : "theguardian.com/science", "data_length" : 70125, "encoder" : "iTunes 10.6.3", "frame_sets" : "{COM=COM: 4, PCS=PCS: 1, PIC=PIC: 1, TAL=TAL: 1, TCM=TCM: 1, TCO=TCO: 1, TDR=TDR: 1, TDS=TDS: 1, TEN=TEN: 1, TID=TID: 1, TP1=TP1: 1, TT2=TT2: 1, TT3=TT3: 1, TYE=TYE: 1, WFD=WFD: 1}", "genre_description" : "Podcast", "itunes_comment" : " 000002AD 00000000 0000AF41 00000000 00015087 00000000 000084E8 00000000 0024F7C6 00000000", "length" : 70135, "title" : "A journey to the heart of the planet we made - podcast", "version" : "2.0", "year" : "2014" }

* Note: '_id' values will likely differ when this test is repeated.

Result

This test case passed only partially.

1)

In comparison with the iTunes screenshot (figure 8), the "comments" field has not been read in correctly. This issue has since been addressed.

Apart from this, the test passed successfully.

2)

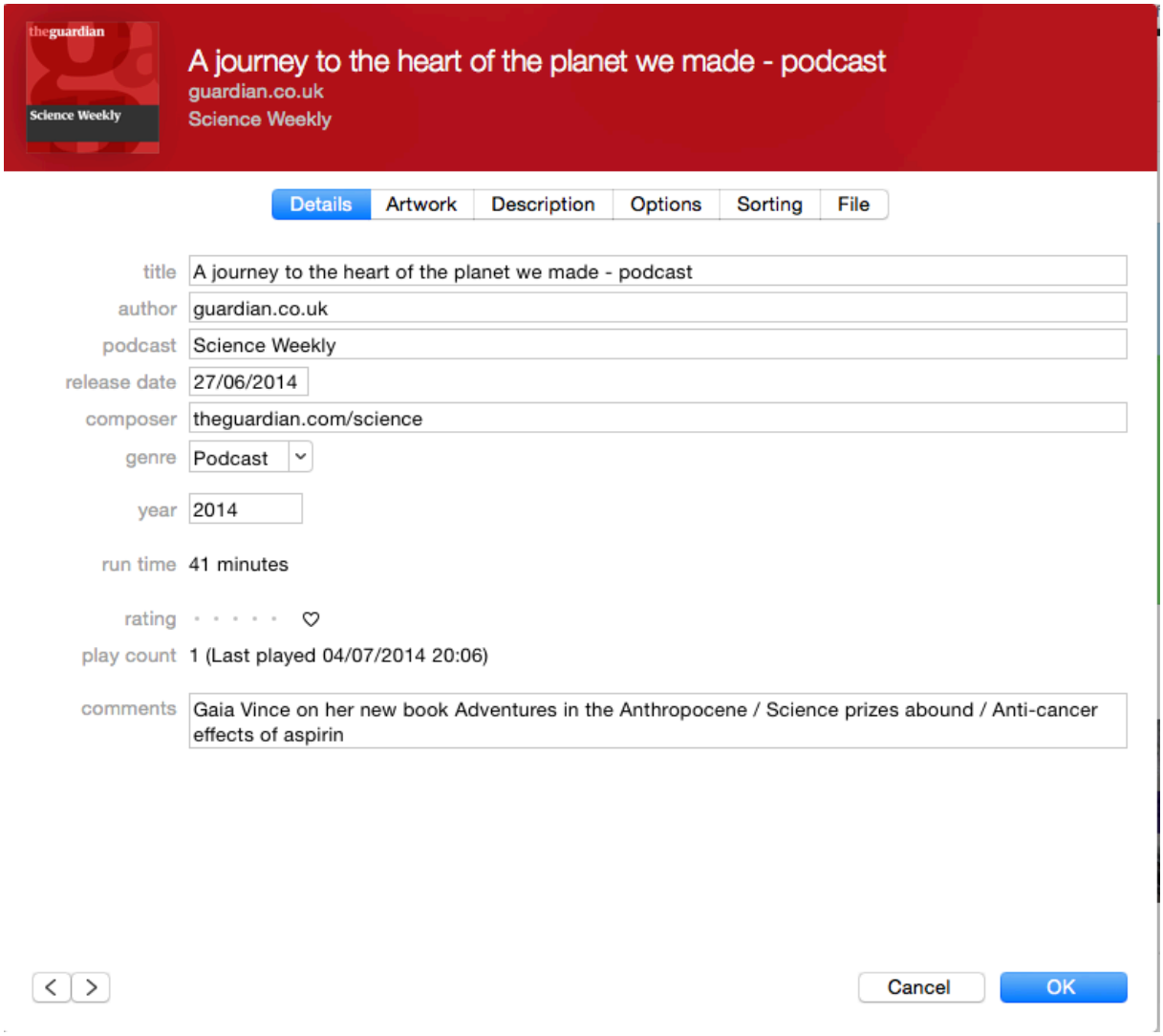
Querying the database shows that the metadata was persisted correctly (see figure 9).

3)

The second, and third file in the collection were processed and persisted equally well as the first..

4)

The file "test.txt" in "Subdirectory" is ignored..



Screenshot of iTunes showing metadata of the “A journey to the heart of the planet we made - podcast.mp3” file

Output of test case 1 (screenshot)

4.13.2.2. Test case 2

Purpose

The purpose of this test is to verify that a value not in the database is also not returned when querying for it.

Prerequisites

- the system has previously been initialized by Test case 1

- the word “grapefruit” is not contained in the title of any of the persisted items:

Actions

Execute “mongo neptune --eval db.files.find({title:\"grapefruit\"})” to verify this.

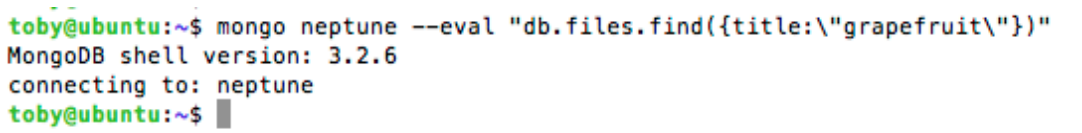
Expected Outcome

Nothing should be returned by this query.

```
toby@ubuntu:~$ mongo neptune --eval "db.files.find({title:\"grapefruit\"})"
MongoDB shell version: 3.2.6
connecting to: neptune
toby@ubuntu:~$
```

Result

The test performed as expected, nothing is returned. Pass.



Checking for negative result

4.13.2.3. Test case 3

Purpose

The purpose of this test is to verify that the software is able to correctly restore the persisted index on start-up.

Prerequisites

- the system has previously been initialized by Test case 1

- logLevel=debug is specified in the config file

Actions

1) Run server component with ‘--daemon’ flag

Expected Outcome

The following logs should be printed to the shell:

```
2016-04-01 12:48:14.289  <<<debug>>>  tm470v1.DataStoreSingleton: Document is
Document{{_id=56fd8b85c59640120deb2985, filePath=smb://<share>, fileName=Russian
Red Army Choir - National Anthem Russia (1977), fileType=mp3}}


2016-04-01 12:48:14.290  <<<debug>>>  tm470v1.DataStoreSingleton: Document is
Document{{_id=56fd8b85c59640120deb2986, filePath=smb://<share>Subdirectory/,
fileName=September 2010 - Beyond limits, fileType=mp3, album=The Royal Society -
R.Science, artist=The Royal Society, comment=0, data_length=11047,
frame_sets={COMM=COMM: 2, PCST=PCST: 1, TALB=TALB: 1, TCAT=TCAT: 1, TCON=TCON: 1,
TDES=TDES: 1, TGID=TGID: 1, TIT2=TIT2: 1, TIT3=TIT3: 1, TKWD=TKWD: 1, TPE1=TPE1: 1,
TYER=TYER: 1, WFED=WFED: 1}, genre_description=Podcast, itunes_comment= 00000055
00000055 00000D44 00000D44 000AFBF9 000AFBF9 00004C74 00004C74 000BEA3F 000BEA3F,
length=11057, title=September 2010 - Beyond limits, version=3.0, year=2010}}


2016-04-01 12:48:14.291  <<<debug>>>  tm470v1.DataStoreSingleton: Document is
Document{{_id=56fd8b85c59640120deb2987, filePath=smb://<share>Subdirectory/,
fileName=test, fileType=txt}}


2016-04-01 12:48:14.291  <<<debug>>>  tm470v1.DataStoreSingleton: Document is
Document{{_id=56fd8b85c59640120deb2988, filePath=smb://<share>, fileName=A journey
to the heart of the planet we made - podcast, fileType=mp3, album=Science Weekly,
artist=guardian.co.uk, comment= 000002AD 00000000 0000AF41 00000000 00015087
00000000 000084E8 00000000 0024F7C6 00000000, composer=theguardian.com/science,
data_length=70125, encoder=iTunes 10.6.3, frame_sets={COM=COM: 4, PCS=PCS: 1,
PIC=PIC: 1, TAL=TAL: 1, TCM=TCM: 1, TCO=TCO: 1, TDR=TDR: 1, TDS=TDS: 1, TEN=TEN: 1,
TID=TID: 1, TP1=TP1: 1, TT2=TT2: 1, TT3=TT3: 1, TYE=TYE: 1, WFD=WFD: 1},
genre_description=Podcast, itunes_comment= 000002AD 00000000 0000AF41 00000000
00015087 00000000 000084E8 00000000 0024F7C6 00000000, length=70135, title=A
journey to the heart of the planet we made - podcast, version=2.0, year=2014}}
```

* Note the '_id' values will correspond to the ones observed in test case 1, while time stamps will differ when the test is repeated.

Result

It can be observed in figure 11 that the data read in from the database on start-up matches the data that was persisted during initialisation in test case 1.

```
[toby@tm470 ~]$ run --daemon
Picked up _JAVA_OPTIONS: -Xmx512M
Picked up _JAVA_OPTIONS: -Xmx512M
2016-04-09 12:25:34.767  <<<info>>>  Starting application
2016-04-09 12:25:34.770  <<<info>>>  Connecting to database
Apr 09, 2016 12:25:35 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[127.0.0.1:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30
000 ms', maxWaitQueueSize=500}
2016-04-09 12:25:35.575  <<<info>>>  Successfully connected to databse: neptune
Apr 09, 2016 12:25:35 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: No server chosen by ReadPreferenceServerSelector{readPreference=primary} from cluster description ClusterDescription{type=UN
KNOWN, connectionMode=SINGLE, all=[ServerDescription{address=127.0.0.1:27017, type=UNKNOWN, state=CONNECTING}]}. Waiting for 30000
 ms before timing out
Apr 09, 2016 12:25:35 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:302}] to 127.0.0.1:27017
Apr 09, 2016 12:25:35 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=127.0.0.1:27017, type=STANDALONE,
 state=CONNECTED, ok=true, version=ServerVersion{versionList=[3, 2, 3]}, minWireVersion=0, maxWireVersion=4, maxDocumentSize=16777
216, roundTripTimeNanos=8271877}
Apr 09, 2016 12:25:35 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:303}] to 127.0.0.1:27017
2016-04-09 12:25:35.833  <<<debug>>>  tm470v1.MongoConnectorSingleton: There are 4 documents in collection files
2016-04-09 12:25:35.884  <<<debug>>>  tm470v1.DataParser: 4 documents added to dataStore
2016-04-09 12:25:35.885  <<<debug>>>  tm470v1.DataStoreSingleton: Document is Document{{_id=5708e5b0c596406989755668, filePath=smb
://                             /Test/, fileName=Russian Red Army Choir - National Anthem Russia (1977), fileType=mp3}}

2016-04-09 12:25:35.885  <<<debug>>>  tm470v1.DataStoreSingleton: Document is Document{{_id=5708e5b0c596406989755669, filePath=smb
://                             /Test/Subdirectory/, fileName=September 2010 - Beyond limits, fileType=mp3, album=The Royal Societ
y - R.Science, artist=The Royal Society, comment=0, data_length=11047, frame_sets={COMM=COMM: 2, PCST=PCST: 1, TALB=TALB: 1, TCAT=
TCAT: 1, TCON=TCON: 1, TDES=TDES: 1, TGID=TGID: 1, TIT2=TIT2: 1, TIT3=TIT3: 1, TKWD=TKWD: 1, TPE1=TPE1: 1, TYER=TYER: 1, WFED=WFED
: 1}, genre_description=Podcast, itunes_comment= 00000055 00000055 00000D44 00000D44 000AFBF9 000AFBF9 00004C74 00004C74 000BEA3F
000BEA3F, length=11057, title=September 2010 - Beyond limits, version=3.0, year=2010}}

2016-04-09 12:25:35.886  <<<debug>>>  tm470v1.DataStoreSingleton: Document is Document{{_id=5708e5b0c59640698975566a, filePath=smb
://                             /Test/Subdirectory/, fileName=test, fileType=txt}}

2016-04-09 12:25:35.886  <<<debug>>>  tm470v1.DataStoreSingleton: Document is Document{{_id=5708e5b0c59640698975566b, filePath=smb
://                             /Test/, fileName=A journey to the heart of the planet we made - podcast, fileType=mp3, album=Scien
ce Weekly, artist=guardian.co.uk, comment= 000002AD 00000000 0000AF41 00000000 00015087 00000000 000084E8 00000000 0024F7C6 000000
00, composer=theguardian.com/science, data_length=70125, encoder=iTunes 10.6.3, frame_sets={COM=COM: 4, PCS=PCS: 1, PIC=PIC: 1, TA
L=TAL: 1, TCM=TCM: 1, TCO=TCO: 1, TDR=TDR: 1, TDS=TDS: 1, TEN=TEN: 1, TID=TID: 1, TP1=TP1: 1, TT2=TT2: 1, TT3=TT3: 1, TYE=TYE: 1,
WFD=WFD: 1}, genre_description=Podcast, itunes_comment= 000002AD 00000000 0000AF41 00000000 00015087 00000000 000084E8 00000000 00
24F7C6 00000000, length=70135, title=A journey to the heart of the planet we made - podcast, version=2.0, year=2014}}

TODO: implement daemon
Apr 09, 2016 12:25:35 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Closed connection [connectionId{localValue:2, serverValue:303}] to 127.0.0.1:27017 because the pool has been closed.
[toby@tm470 ~]$
```

Output of test case 3 (screenshot)

4.13.2.4. Test case 4

Purpose

The purpose of this test is threefold.

1) verify client-server connectivity
2) verify the object requested object is returned by the server
3) verify the object is returned by the server such that it can be played back by the client.

Prerequisites

- the client app is configured correctly to connect to the server component
- the server component has previously been initialized with Test case 1, and is running as daemon as per Test case 3.

Action 1

- Enter “journey” into the search field

Expected Outcome 1

- “A journey to the heart of the planet we made – podcast […]” should be the single returned result. The result returns within 1 second

Action 2

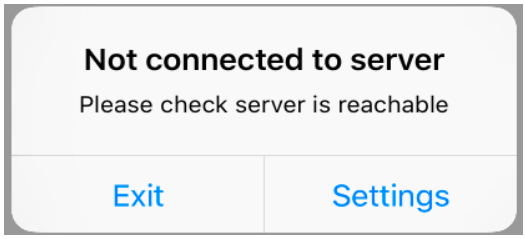
- Tap the single returned line

Expected Outcome 2

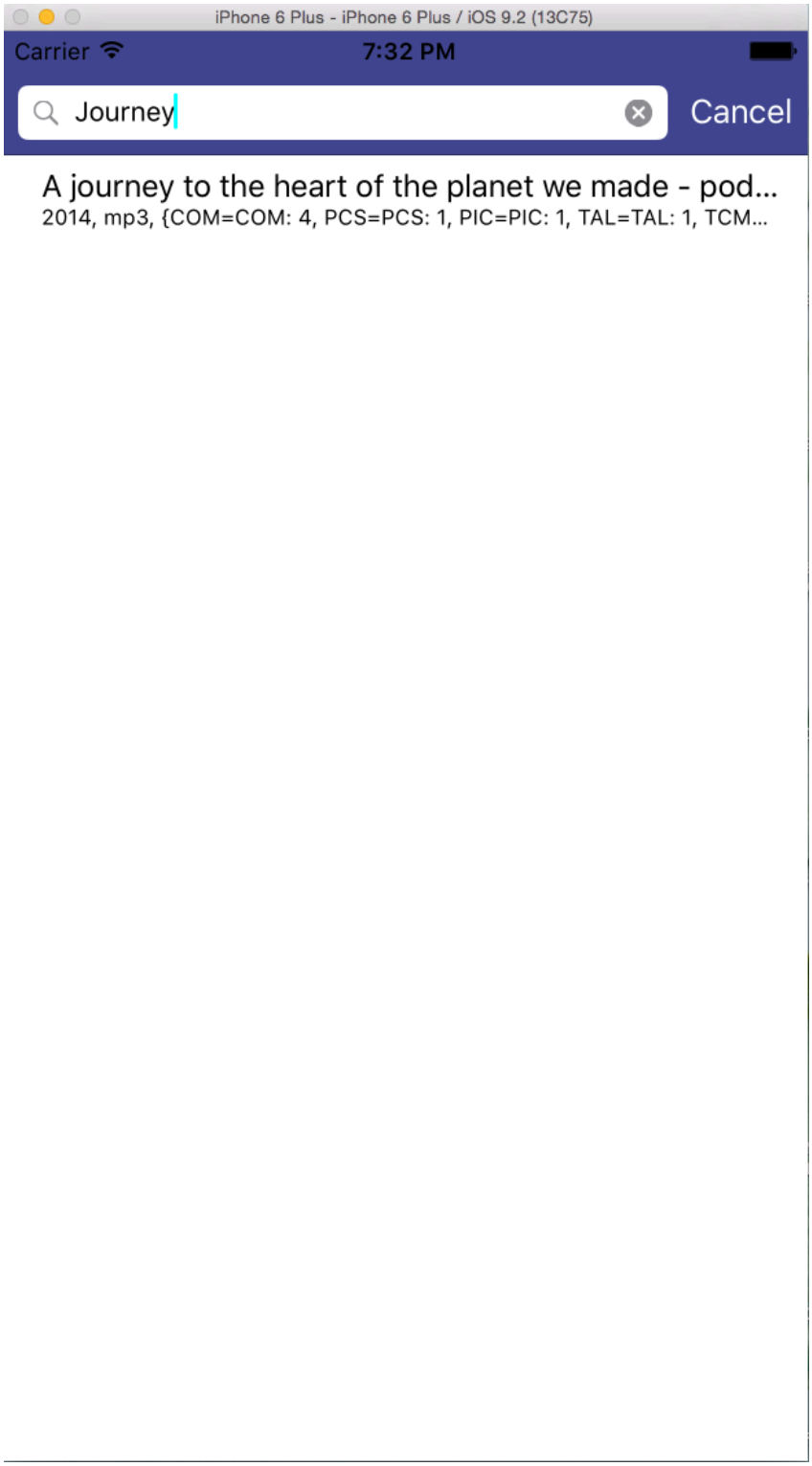
- the podcast should start playing within 3 seconds

Result

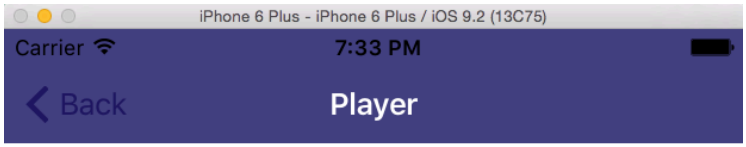
1) The client app connects correctly (else, an error, as in figure 12, would have been shown)
2) The requested result is correctly returned by the server, as shown in figure 13
3) The requested file is received, and the clients starts playing it in less than one second (compare figure 14).



This error would be shown, if the connection failed



The returned search result is shown (the meaningless information shown in the secondary row has been recorded as known issue C-6, and the fact that not all information is displayed as C-7 in table 7).



The selected file is played back

4.13.3. Measuring latency

In order to accurately test the performance of the server component, a synchronous function call was implemented client-side. It works by recoding a time stamp immediately before, and immediately after the server interface is called, and taking the difference between those time stamps.
This way, there should be no impact on the performance of the search itself.
Since testing was performed on a local network, the effects of jitter are negligible.

```
    func retrieveFilteredResultsFromServer(searchText: String) {
        let startTime: NSTimeInterval = NSDate().timeIntervalSince1970
        if (searchText.characters.count > 1) {

            var searchURL: String = baseURL + "mode=search&searchString=" +
searchText.stringByAddingPercentEncodingWithAllowedCharacters(.URLHostAllowedCharac
terSet())!
            do {
                json = try getJSON(searchURL)
            } catch {
                return
            }

            let returnedData = parseJSON(json)!
            populateIndexWithResults(returnedData)
            print("Server returned \(returnedData.count) documents")

        } else {
            //populateIndexWithResults2(["Enter 3 or more characters to start
searching", ""])
        }

        let endTime: NSTimeInterval = NSDate().timeIntervalSince1970
        let timeTaken = (endTime - startTime) * 1000
        print("Search string \(searchText) took \(timeTaken)ms to return")

    }
```

4.13.4. Performance testing

Approach

The database was populated with a sample set of 14815 entries of file metadata from mp3 files.

The two files searched for were two podcasts, Outlook's "Plastic surgeon Mohammad Jawad on transforming faces of acid victims" (BBC, 2012), and Science Weekly's "A journey to the heart of the planet we made" (The Guardian, 2014).

Within the directory structure, the Outlook podcast was placed near the beginning (within the first 20 files), while the Science Weekly podcast was placed towards the end (within the last 20 files).

From each title, a substring was copied, and pasted, as a whole, into the search field of the respective application.

Both strings were then searched ten times, and the average of those searches taken as a result, with the measurement taken from the output produced by the client, as described in "Measuring latency".

While Nep-Tune provides statistics down to the millisecond, search times on OS X and Windows were taken using a stopwatch.

Size of sample set: 14815 - all results in milliseconds

Test search string "heart of the planet we made"

#	Nep-Tune Server mode (SpeedyGonzales)	Nep-Tune Server mode (simpleSearch)	Nep-Tune Client mode	Windows 8.1 Explorer	Apple 10.10 Finder
1	229.6	315.7	3,432.8	9,400	2,160
2	234.7	384.3	3,226.9	10,300	1,650
3	279.3	351.0	3,158.4	10,160	1,750
4	617.3	367.3	3,152.9	10,180	1,720
5	270.7	332.5	3,090.8	10,150	1,630
6	330.0	365.2	3,091.9	10,110	1,580
7	339.7	320.8	3,132.2	10,100	1,510
8	227.8	328.3	3,115.1	10,230	1,460
9	220.7	322.0	3,060.5	10,050	1,460
10	240.3	304.6	3,058.6	10,300	1,480
Average	299.0	339.2	3,152.0	10,098	1,640
Percentage change	100.0	113.4	1,054.2	3,377.3	548.5

Table 9: Performance measurements for search string "heart of the planet we made"

Test search string "faces of acid victims"

#	Nep-Tune Server mode (SpeedyGonzales)	Nep-Tune Server mode (simpleSearch)	Nep-Tune Client mode	Windows 8.1 Explorer	Apple 10.10 Finder
1	221.6	306.2	3,044.9	1,700	-
2	232.2	386.3	3,717.6	850	-
3	233.9	310.8	3,070.2	850	-
4	246.7	313.8	3,163.8	760	-
5	217.9	325.2	3,048.8	1,020	-
6	229.3	301.3	3,018.7	860	-
7	238.5	313.3	3,020.6	880	-
8	222.9	322.6	3,060.3	850	-
9	235.4	306.1	3,059.1	920	-
10	232.7	320.8	3,055.6	720	-
Average	231.1	320.6	3,126.0	941	-
Percentage change	100.0	138.7	1,352.6	407.2	*This search never returned

Table 10: Performance measurements for search string "faces of acid victims"

Further tests were performed with collection sizes of 14815, 17944, 22157, 41870 and 65796 to allow the projecting the scalability of Nep-Tune.

Number of documents	“heart of the planet we made”	“faces of acid victims”
14815	299	231
15000	134	128
17944	391	391
20000	343	332
22157	486	485
30000	760	739
40000	1178	1146
41870	620	655
50000	1595	1553
60000	2012	1960
65796	2584	2510
70000	2430	2368
80000	2847	2775
90000	3265	3182
100000	3682	3589

Table 11: Performance projection. 14815, 17944, 22157, 41870 and 65796 were measurements, the rest is interpolated. All measurements in SpeedyGonzales mode.

4.14. Feature suggestions

This table summarises the currently desired features for Nep-Tune, which mostly stem from discussion with James Hatton (2016). A client features are identified with a ‘C’, and server features identified with an ‘S’.

Identifier	Name	Comment	Priority
C-1	Client for the Android system	Highly desirable, but probably out of scope for the project.	medium
S-1	Recognition of binary data	In particular, the categorization of music by its content were discussed, such as the mood of the music, or matching it to certain styles of dance. This may include machine learning. Out of scope for the project.	low
S-2	Bias search towards type of information	Certain attributes should be weighted more highly than others. E.g. a string match in the “title” tag should be higher up the results list than one in “copyright notices”. Doable, but probably out of scope for the project.	medium
S-3	Recognise years in search	When a four-digit number is typed in, check if it could be a year, and if so, bias search results accordingly. Extension of S-2.	medium
S-5	Processing of “streamed” data sets, such as those resulting from big data.	An extension to S-4, the ability to take snapshots and process dynamically arriving data from a stream would be useful, but is computationally too complicated to be included at this point.	low
S-7	Parsing semantics	E.g. resolving “the composer of Lohengrin” to “Richard Wagner”. Involves AI, not feasible within the project.	low
S-8	Cross-platform compatibility	Currently, server component only works on Linux derivatives, enhance to support other OS flavors. This will be attempted at the end, or shortly after completion of the project.	medium
S-9	Implement HTTP POST	HTTP get limits queries to approximately 2048 characters, and prevents complex symbols from being sent. POST would overcome these restrictions.	high
S-10	Implement encryption	Encrypted communication is absolutely necessary for corporate deployment.	high

Table 12: Feature suggestions

4.15. Known issues

This table shows the currently known issues of Nep-Tune. Client issues are denoted with a ‘C’. Currently, there are no known issues with the server component.

Identifier	Title	Comment	Priority
C-1	Communication synchronous	To obtain accurate performance measurements, the client search was implemented synchronously. For user experience reason, search should be handed over to a GCD high priority serial queue. A semaphore should control that a maximum of two entries are out for search, ensuring that only the last user entry gets submitted to the server, besides the request that has already left.	high
C-2	Color scheme inconsistent	Navigation bar colour slightly off because of transparency, but opacity hides search.	low
C-3	Default entry sweeps off the bottom	When activating search bar, default entry sweeps to bottom of screen. Cool effect, but inappropriate here.	low
C-4	Default entry can be selected for playback	Generally, it should not be possible to select playback, unless server has returned a file.	medium
C-5	Tag ordering in supplementary row is meaningless	For returned search results, the order in which tags in the supplementary row are displayed is undefined, and rarely makes sense. Define meaningful order (see figure 13).	medium
C-6	Rows do not scroll horizontally	Information is often too large to be displayed on the width of the screen. Scrolling rows would overcome this restriction. Touchable scrolling rows would be even better.	medium
C-7	Splitview doubles up detail view in portrait mode,	In portrait mode only, the splitview controller shows the detail view in both sections of the screen (without a back button!). This must be prevented.	high

Table 13: Known issues

4.16. Project Logs

Owing to a change of tutor, no formal project logs are available dating before 24 April 2016.

<table>
<tr><td>LOG SHEET
No: 1</td><td>SESSION No:</td><td>TMA No:
03</td><td>DATE:
5 - 24
April 16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session: Establish client-server communication, and implement filter function.</td></tr>
<tr><td>Work Completed:

Established communication between server and client component. Representation of search results in client user interface.</td><td colspan="2">Problems:

Formatting of the URL proved initially unsatisfactory. Stackoverflow helped on both end of the communications link.</td><td colspan="2">Comments:

Progress in the last two weeks has been a bit slower than anticipated, mostly due to TM354 needing some of my attention, and some social engagements I placed into the week of the TMA02 submission. However, the integration of the com.sun.net http library was uneventful, which means I managed to stay on schedule nonetheless.
The client is now able to search the server for content, but is not able to retrieve that content yet.</td></tr>
<tr><td colspan="3">Data Files Used:</td><td colspan="2">Program Files Used:</td></tr>
<tr><td colspan="5">References Used:

Oracle documentation:
https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpServer.html
OU M362 Unit 8 module materials, in particular section 2.4 “Processing GET and POST requests”.
http://stackoverflow.com/questions/15235400/java-url-param-replace-20-with-space
http://stackoverflow.com/questions/24551816/swift-encode-url</td></tr>
<tr><td colspan="5">Objective: Largely / ~~Not Quite / Not really~~ achieved</td></tr>
</table>

Next work planned:	Comments:
The next step will be to test responsiveness of the search function with a larger dataset, and possibly take action if it turns out to be too slow. Also in the session up to the next project log, I will work on implementing HTTP Live streaming, so the client will actually receive a file, which it can play back.	HTTP Live streaming is a much newer technology than the ones used up to now, and less documentation is available. For this reason, it will be more challenging. The M362 will help me further with some generic elements of connection handling, for the actual implementation of the protocol will likely be a challenge. Note also that the TM354 exam is in early June, so I have scheduled a two-week break from TM470 for the purpose of revision. In practice, I am likely to continue with some of the programming for TM470, but much less than I otherwise would.

LOG SHEET No: 2	SESSION No: 2	TMA No: 3	DATE: 25/4 – 8/5 2016	TIME SPENT:
Objective of Session: Complete core functionality				
Work Completed: Multithreaded optimised search (SpeedyGonzales class). Additional client search (to be used to constrain search fields). Media file playback in client. Ability to store settings in user interface of App Options for HLS evaluated.	Problems: *The implementation of HTTP Live streaming (HLS) is proving more complicated than expected. Almost all libraries I found are client side for 'decoding' a stream. FFMPEG seems the only viable solution (Apple provides a set of tools, but those won't run on Linux), but unfortunately, the x264 library fails to build on my server. Implementing reading from the App settings file caused some head scratching, as it has changed significantly from Swift 1.*	Comments: That the libraries I previously had my eye on for encoding HLS turned all out to be client-side proved a blow. Aside from commercial stand-alone software (which defies the point of my open-source project, e.g. Wowza), FFMPEG seems to be the only viable cross-platform solution for transcoding file to the Apple-approved format. In principle, this should work well, but in practice, failure to build the required x264 library is a big rock in my path. As a workaround, I now use the unaltered source files, which works, but would not be approved on submission to the Apple store. In the worst case scenario, this will be an acceptable set-back for the TM470 project, but is not very satisfying. With literature review and the TM354 exam on my schedule for the next two weeks, I will have limited time to find a solution. For usability, it was important to make the client app settings user configurable (server already is, client settings were previously hardcoded). I implemented reading from the default iOS settings file, which can be accessed via the phone settings.	
Data Files Used:		Program Files Used:		

References Used:

https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html#//apple_ref/doc/uid/TP40008332-CH102-SW5

https://trac.ffmpeg.org/wiki/CompilationGuide/Centos (compiling ffmpeg on CentOS)

http://www.nsprogrammer.com/2013/08/http-live-streaming-videos-on-demand.html#more (some HLS approaches)

http://stackoverflow.com/questions/1281353/use-java-ffmpeg-wrapper-or-simply-use-java-runtime-to-execute-ffmpeg (FFMPEG wrapper for Java)

OU M362 Unit 5 module materials: Multithreading and the chat room in section 6, on which the refined search logic is based.

https://www.wowza.com (commercial encoding software)

https://gist.github.com/cmittendorf/a427e5ae1e390bdced9b (solution for settings)

Objective: Largely / ~~Not Quite / Not really~~ achieved

Despite the HLS set-back, phase 2 is now feature-complete, and all core functionality of the software is present.
Given that music streaming is mainly a proof-of-concept with respect to the underlying server technology, I think it is fair to say the objective was largely achieved.

Next work planned:	Comments:
The focus during the next two weeks will be on the comparison of available commercial software, and starting a draft of TMA03. Time permitting, I will further try to get the x264 library to build, which is essential for FFMPEG to work, which in turn is needed to create an Apple-compliant media stream.	The M362 module materials were a very helpful resource this time, the multi-threaded search logic is entirely based on M362 sample code. The same cannot be said for the iOS developer documentation; the way to interact with the persistent settings file is not obvious, and a fair amount of online research was needed before I found some code that pointed me into the right direction.

<table>
<tr><td>LOG SHEET No: 3</td><td>SESSION No:
3</td><td>TMA No:
3</td><td>DATE:
9/5/16 – 22/5/16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session:</td></tr>
<tr><td>Work Completed:

Moved server project to Ubuntu server

Enhanced client settings interface

Basic implementation of HLS

Wrote installation script

Started evaluation against commercially available solutions, such as DocStar and MaxxVault</td><td colspan="2">Problems:

Failure to build libx264 (a required component for HLS) prompted me to change from CentOS to Ubuntu.

Facing problems with evaluation; most systems do not state whether archived files are hold in database, or left in their original location (the USP of my project).</td><td colspan="2">Comments:

Frustration with failure to build libx264 meant a change of operating system was the path of least resistance, having exhausted all possibilities other than rewriting elements of libx264 itself.

This had two positive side effects: firstly it went some way towards proving that my solution will work on a number of platforms, and secondly, it gave me an opportunity to write an installation script, which I had not planned until later.

Implementing the settings interface for the client app means some comparative tasks are now possible without changing the code (e.g. I can switch from server search to client search in the client settings).

Evaluation against commercial (and free) software is proving somewhat difficult, since few details about the implementation of the actual document management process are available.</td></tr>
<tr><td colspan="2">Data Files Used:</td><td colspan="3">Program Files Used:</td></tr>
</table>

References Used:

http://stackoverflow.com/questions/10971039/http-live-streaming-the-linux-nightmare

https://ffmpeg.org/general.html

https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/index.html#//apple_ref/occ/instm/NSUserDefaults/registerDefaults:

http://www.ffmpeg.org/ffmpeg-formats.html

Objective: ~~Largely~~ / Not Quite / ~~Not really~~ achieved

I was hoping to have made further progress with my evaluation against commercially available and free software, a task that proved difficult, not only because such software seems to be a rather niche product, but also details of particular implementations are hard to come by.

Development objectives have been fully achieved.
I remain on schedule.

Next work planned:	Comments:
Besides the schedule break for TM354, focus will now shift towards TMA03, and completing the evaluation task.	From a development perspective, I found myself again mostly in M362 materials for web interaction techniques, and the Apple developer library for HLS and other client-specific queries. From an evaluation perspective, Google was my main source of information, together with snippets from the OU library, which didn't turn out to be useful in the end.

LOG SHEET No: 4	SESSION No:	TMA No: 3	DATE: 6/6/16 – 18/6/16	TIME SPENT:
Objective of Session: Mostly satisfying academic requirements, such as feature comparison, performance evaluation, and beginning the project report.				
Work Completed: Feature comparison against commercial software Demonstration of project to James Hatton Initial performance testing Logo and User interface refinements Start draft of TMA03	Problems: *I found it quite difficult to draw up a feature comparison with commercial software, since there is no software available with quite the same purpose in mind. There seem to be either document management software packages, or multimedia applications, both of which share some common elements with my project.*	Comments: The feature comparison shed new light on the validity of my project, as I was unable to find any software that quite matches what mine does. Also the task of validating which solutions I could sensibly compare my project with, and which features to evaluate helps in establishing "unique selling points". The demonstration of the software to James Hatton was very helpful. We explored its application in his field of expertise – vehicle management – and he also provided feedback on user experience. Initial performance testing results are promising. It turns out that some of the applications I wanted to test against (e.g. Airplayer) do not actually support searching the directory tree, so a true performance test could only be performed against desktop software (and mine always wins). I find TMA03 extraordinarily challenging, since I struggle to understand its requirements.		

Data Files Used:	Program Files Used:
Test set of 14815 mp3 files.	Windows Explorer and OS X Finder, in addition to my project software.

References Used:

Apple (2016h) *HTTP Live Streaming* [Online]. Available at https://developer.apple.com/streaming (Accessed 12 June 2016).

Brooks, C. (2016) 'Best Document Management Software and Systems' *Business News Daily*, 24 March 2016 [Online]. Available at http://www.businessnewsdaily.com/8038-best-document-management-software.html (Accessed 5 June 2016).

FFmpeg (2016) 'A complete, cross-platform solution to record, convert and stream audio and video', *FFmpeg* [Online]. Available at https://ffmpeg.org (Accessed 12 June 2016).

FFmpeg Java (2016) 'bramp/ffmpeg-cli-wrapper', *Github* [Online]. Available at https://github.com/bramp/ffmpeg-cli-wrapper (Accessed 12 June 2016).

Hatton, J. (2016) Online meeting with Toby Scholz, 13 June 2016

Lixandroiu, R., Maican, C. (2016) A system architecture based on open source enterprise content management systems for supporting educational institutions, *International Journal of Information Management*, vol 36, no 2, pp 207 – 214 [Online]. DOI: 10.1016/j.ijinfomgt.2015.11.003 (Accessed 5 June 2016)

The Open University (2016b) 'From domain to analysis models', *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739476&section=4 (Accessed 12 June 2016).

The Open University (2016c) 'Model-view-controller pattern', *TM354 Software engineering* [Online]. Available at https://learn2.open.ac.uk/mod/oucontent/view.php?id=739465&section=5.4 (Accessed 12 June 2016).

Oracle (2016f) 'Interface Executor Service', *Java Platform* [Online]. Available at https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ExecutorService.html (Accessed 12 June 2016).

Oard, D. W., Baron, J. R., Hedin, B., Lewis, D. D., & Tomlinson, S. (2010). 'Evaluation of information retrieval for E-discovery' Artificial Intelligence and Law, vol 18(4) pp347-386 [Online[. DOI: 10.1007/s10506-010-9093-9 (Accessed 18 June 2016).

As well as the websites of the various software packages for the feature comparison.

Objective: Largely /~~ Not Quite / Not really achieved~~

Next work planned:	Comments:
Completing TMA03, which is due in just over two weeks from now.	

LOG SHEET No: 5	SESSION No:	TMA No: 3	DATE: 19/6/16 – 3/7/16	TIME SPENT:
Objective of Session:				

Work Completed:	Problems:	Comments:
Completed performance testing for TMA03 Reviewed Bellini's article on ECLAP and evaluating Neumann's article on New ways of indexing av material. Completed TMA03	*Unsure about the structure of TMA03. It seems like many items should appear several times, namely in the project report, in the report on work completed since the last TMA, and in reflection in the review. Requirements are extraordinarily unclear from the question paper.*	Performance testing went smoothly, and yielded results much better than anticipated. It's great to see software working, after only speculating how it will behave for several months. Bellini's article on ECLAP was a goldmine, because ECLAP's search is spectacularly slow. This may be because of the hardware it runs on, or other restrictions, but in any case, it places my project software nicely into context. TMA03 was very challenging. It feels more compact (fewer words!) and structured than TMA02, but I cannot help feeling that the question paper just asks me to repeat myself over and over again. I tried to avoid this, which means that sections one might expect in the report can be found in the review, those one might expect in the account of project work in the report, and those one might expect in the review in the appendix. Very confusing.

Data Files Used:	Program Files Used:
Test set of 14815 mp3 files.	Project software, ECLAP.

References Used:

Neumann, J., Plank, M. (2014) 'TIB's Portal for audiovisual media: New ways of indexing and retrieval', *IFLA Journal*, 3 March (40.1), pp 17-23 [Online]. DOI: http://dx.doi.org/10.1177/0340035214526531 (Accessed 28 June 2016).

Basili, V. R., Turner, A. J. (1975) Iterative enhancement: A practical technique for software development, *IEEE Transactions on Software Engineering*, Vol.SE-1(4), pp.390-396. [Online] DOI: 10.1109/TSE.1975.6312870 (Accessed 6 February 2016).

Objective: Largely / ~~Not Quite / Not really~~ achieved

Next work planned:	Comments:
Refinements on software. File change monitoring is up next, followed by some client enhancements and bug fixes.	After TMA03, it will be nice to ease back into development. TMA03 seems to have a bit of an identity crises, being half a draft report, and half a mixture of things that could also be a draft report, but are not supposed to be.

<table>
<tr><td>LOG SHEET No:
5</td><td>SESSION No:</td><td>TMA No:
EMA</td><td>DATE:
4/7/16 – 17/7/16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session:</td></tr>
<tr><td>Work Completed:

Added support for new file types

Started work on polling mechanism for changes</td><td colspan="2">Problems:

Investigation shows that there is no mechanism to notify clients of changes in the file structure using the samba protocol. As a consequence, a polling mechanism needs to be devised.</td><td colspan="2">Comments:

Because file change monitoring requires polling, some refactoring was required to be able to hash files, allowing for a quick comparison.
Since this is done in the same section of code as metadata parsing, I used to opportunity to add support for more file types, leveraging Apache’s Tika library.

Other than parsing metadata, Tika also detects the file mime type, which is handy, as I can record it with the file data, and use it for the standardised interface.</td></tr>
<tr><td colspan="2">Data Files Used:</td><td colspan="3">Program Files Used:

Apache tika library</td></tr>
</table>

References Used:

https://technet.microsoft.com/en-us/library/cc939973.aspx

http://tika.apache.org

https://msdn.microsoft.com/en-us/library/cc246231.aspx

Objective: Largely /~~ Not Quite / Not really achieved~~

Next work planned:	Comments:
Completing the implementation of the file change polling mechanism. Start EMA.	Supported file types now include mp3, wav, aiff, mp4, flac, pdf, class, doc, docx, epub, html, gif, bmp, jpg, jpeg, png, psd, tiff, matlab, 3gpp, ogg, vorbis, opus, and others.

<table>
<tr><td>LOG SHEET No:
6</td><td>SESSION No:</td><td>TMA No:
EMA</td><td>DATE:
18/7/16 – 31/7/16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session:</td></tr>
<tr><td>Work Completed:

Polling and file maintenance mechanism completed

HLS null pointer errors addressed

Improved error handling

EMA outline started</td><td>Problems:

There is limited support for detecting whether a file is open for writing by another program in Java (the cause of the HLS problems), which I understand to be a Linux limitation. Further, Java provides no support for accessing a file’s last accessed time stamp.

A back injury unfortunately limited the amount of work I was able to do this weekend, throwing me off schedule.</td><td colspan="3">Comments:

The polling and file maintenance mechanism was implemented using a simple maintenance class, in combination with a hashing algorithm based on a file’s length and last modified time. It is a thread that runs periodically checking whether the actual file structure, or individual files have changed, and updates the index accordingly.

The HLS problems described in TMA03 stemmed from Java trying to send files that had not been completely written yet (more specifically opening files for writing although FFmpeg also had them open for writing). This has been addressed by checking a file’s last modified time before sending it.
In the process, I have also started improving general error handling.

A start on the outline of the EMA was made.</td></tr>
<tr><td colspan="2">Data Files Used:</td><td colspan="3">Program Files Used:

None.</td></tr>
</table>

References Used:	
None.	
Objective: ~~Largely~~ / Not Quite ~~/ Not really~~ achieved	
Next work planned: Formally testing file maintenance. Further improve error handling. Complete structure of EMA, and start writing its contents.	Comments:

<table>
<tr><td>LOG SHEET No:
7</td><td>SESSION No:</td><td>TMA No:
EMA</td><td>DATE:
1/8/16 – 14/8/16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session:</td></tr>
<tr><td>Work Completed:

Formally tested and recorded the maintenance feature

Server can now pass errors to the client (and client handles them)

Progress on EMA, particularly revision of the literature review, which received poor marks in TMA03</td><td>Problems:

I’m still behind schedule, and think I won’t be able to fix all bugs and add all proposed features to the client before submission.</td><td colspan="3">Comments:

Testing of the maintenance feature went smoothly, and error handling is much improved with the latest iteration of the software.

Following the feedback from TMA03, I completely tore up my previous literature review, and started from scratch – a surprisingly difficult task. However, it has now taken reasonable shape.

Most of the testing / evaluation / analysis tasks have now moved to the appendix, significantly reducing word count in the main body.

Have not yet started the project review yet, which should have been completed last week.

Given the situation, it is unlikely I will be able to implement all proposed features for the client component prior to submission, but at the same time, the server component is now feature complete, and has no known issues.</td></tr>
<tr><td colspan="2">Data Files Used:</td><td colspan="3">Program Files Used:

None.</td></tr>
</table>

References Used:	
None.	
Objective: ~~Largely / Not Quite~~ / Not really achieved	
Next work planned: Complete draft of EMA, work through remaining client component bugs. If sufficient time, I should tidy server code, and make Bitbucket repository public.	Comments:

<table>
<tr><td>LOG SHEET No:
8</td><td>SESSION No:</td><td>TMA No:
EMA</td><td>DATE:
15/8/16 – 28/8/16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session:</td></tr>
<tr><td>Work Completed:

Performance projection for multicore deployments

Draft of the EMA</td><td>Problems:

I’m still not confident I will have time to implement all client features.</td><td colspan="3">Comments:

I found it important to give some indication of which performance can be expected for multicore deployments of the software, and evaluated against Amdahl’s, Gustafson’s, and Hill’s projections of multicore performance.
In the end, I only included Amdahl, with an eye on word count.</td></tr>
<tr><td colspan="2">Data Files Used:</td><td colspan="3">Program Files Used:
None.</td></tr>
<tr><td colspan="5">References Used:

Amdahl, G. (2013) Computer Architecture and Amdahl’s Law, Computer, Vol 46(12) pp. 38-46 [Online]. DOI: 10.1109/MC.2013.418 (Accessed 27 August 2016)

Hill, M., Marty, R. (2008) ‘Amdahl’s Law in the Multicore Era’, Computer, vol 41(7), pp 33-8 [Online]. DOI: 10.1109/MC.2008.209 (Accessed 27 August 2016).

Gustafson, J. (1988) ‘Reevaluating Amdahl’s Law’, Communications of the ACM, vol. 31, no. 5, pp. 532-3.</td></tr>
</table>

Objective:	Largely / ~~Not Quite / Not really~~ achieved
Next work planned: Complete EMA, hopefully discuss feedback with Charly. Continue work on client features.	Comments:

<table>
<tr><td>LOG SHEET No:
9</td><td>SESSION No:</td><td>TMA No:
EMA</td><td>DATE:
29/8/16 – 11/89/16</td><td>TIME SPENT:</td></tr>
<tr><td colspan="5">Objective of Session:</td></tr>
<tr><td>Work Completed:

EMA completed

Playback enhancements to the client component

Server component encryption</td><td>Problems:

None</td><td colspan="3">Comments:

Very happy to have completed the EMA, and particularly relieved it’s within the OU word limit (if citations, captions, and code excerpts are excluded).

Using the iOS AV player framework allowed me to implement phase 3.2.a, which was the penultimate scheduled phase. 3.2.b sadly was not achieved, as a result of unexpectedly having to implement HLS.

I started exploring the implementation of encryption, and am currently gathering the required documentation. Hopefully, this should be complete by the end of September.</td></tr>
<tr><td colspan="2">Data Files Used:</td><td colspan="3">Program Files Used:

None.</td></tr>
<tr><td colspan="5">References Used:</td></tr>
<tr><td colspan="5">Objective: Largely / ~~Not Quite / Not really~~ achieved</td></tr>
</table>

Next work planned:	Comments:
Complete encryption feature. Address outstanding client component issues.	

4.17. Selected email correspondence with Charly Lowndes

4 April 2016

Dear Toby,

thanks for all this. I have not read the code - all I need to know is (1) does it work, (2) what did you test and (3) were the test results interesting?

Problem description: the explanation of searchability is a little wordy. The complexity discussion is a level 2 one: you can feel free to assume your reader is familiar with concepts such as Big O.

You don't really start discussing the specific problem until page 6, and even there you do not state what the problem actually is. This arrives on page 8. Is it rather personal to you? Does this mean that the solution has a value only for you?

You describe the issue in terms of music, but then test by searching for an image. This needs some explanation and justification.

How far have you looked at readily available solutions for general content search? What candidates are there and what are their limitations?

Evaluation: you description of Hegarty (2015) on p 25 looks good.

Code snippets: these mostly make sense (I have only skimmed your report) - be ready to comment further with anything complicated. Remember that I am not a programmer: I understand stuff as long as it is explained clearly.

"Page" references: if there's an easy way to make the location findable, use it. eg section numbers. Don't spend huge effort on this.

Use the first author on the list, regardless of who is main author.

Plan: I'd leave it all in there for now.

Likewise the other bits: I'll read it all fully when I assess it and can advise you better then how to organise the EMA.

Copyright: the regulations take their power and legitimacy from: In these Regulations “the Act” means the Copyright, Designs and Patents Act 1988 - so cite that.

I hope this all helps.

I recommend that you consider the Big Questions of: What Problem Will This Solve? Why Is It Worth Doing? more than the fine details of referencing and organising: remember that TMA02 is about the progress you have made, and TMA03 is the draft. It is after TMA03 that you should worry about making everything tidy.

Best wishes,

Charly

Charly Lowndes

Tutor, Open University

TM129 TT284 T215 T320 T324 TM470

Tel: 01684 573604

Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 02 April 2016 22:14
To: C.A.M.Lowndes
Subject: TMA02 - draft

Dear Charly,

thank you for the tutorial the other evening.
I have done a little bit of work on TMA02 since we spoke, and as discussed, please find a draft attached - I would be very grateful to receive your feedback.

Following our discussion, I have tried to re-phrase my description of the problem. Is the new problem description more suitable than the previous one (I see it as a different take on the same problem)?

There are a few points I am uncertain about:
- my list of literature has grown a little bit, but I am not confident that I "critically evaluate" each piece adequately. I fear explicitly assessing each item according to PROMPT would rather explode the size of the TMA. Could you please give feedback on the level of evaluation in the draft, and whether each item should be explicitly evaluated?
- is the type and level of evidence supplied appropriate? I opted for code snippets instead of screenshots to keep size down, and I have removed things like try...catch statements from the code snippets given. Is this ok, or should I always include everything?
- referencing: The OU guide states to include page numbers with the in-line citation. I could see how including the URL would be useful for citing online sources (e.g. the Apple developer API documentation). Is there an analogous way to do this for online sources?
- For *Survey on NoSQL Database* (DOI**:** 10.1109/ICPCA.2011.6106531), it seem that Jing Han is the main author, but Jian Du comes first in the alphabet. Do I reference this as (Du et al., 2011), or (Han et al., 2011)?
- As per your advice on the forum, I have included a project plan, modelled on the RISE project plan. It seems that some of the information in it is possibly surplus to requirements, and can be left out. Could you please advise is any sections can be cut for the submission?
- I followed your advise in the initial tutorial (from before I was your student), and have included some additional elements, such as test cases, code snippets, a class diagram, etc. Would some of these be better of in the appendix?
- I'm unsure how to quote these copyright regulations, since they're not an act of parliament, it seems to make more sense to quote the court, as opposed to the parliament?

For now, I have attached the code in a zip file, as it inevitably will change before submission.

For the submission, I will copy it into the appendix.
I have not forgotten our discussion on including references in the code itself; will work on that tomorrow - wanted to get an email out tonight.

thanks for all you help, and kind regards,

Toby Scholz
TM470-16B
tfs34 / B6960379

5 April 2016

Dear Toby,

you probably have enough to read, but I think I said I's share this:

Grolinger, K., Higashino, W.A., Tiwari, A. and Capretz, M.A., (2013). Data management in cloud environments: NoSQL and NewSQL data stores.*Journal of Cloud Computing: advances, systems and applications*, *2*(1), p.22.
http://dx.doi.org/10.1186/2192-113x-2-22

Feel free to ignore it!

Best wishes,

Charly Lowndes

Tutor, Open University

TM129 TT284 T215 T320 T324 TM470

Tel: 01684 573604

Skype charly.lowndes

26 April 2016

Dear Toby,

Chatting with another TM470 student, we digressed into efficient search methods for incomplete metadata and he mentioned his several terabytes of MP3 music files, which he has yet to index or tag effectively. I thought of your project, and wondered if the two of you might find it amusing to swap ideas. His project is about something completely different.

He is James Hatton, and is happy for me to give you his email, which is

james.hatton@my.open.ac.uk

Feel free to contact him, or not.

All the best,

Charly

Charly Lowndes

Tutor, Open University

TM129 TT284 T215 T320 T324 TM470

Tel: 01684 573604

Skype charly.lowndes

29 April 2016

Dear Toby -

If performance is being measured by time to complete a task, then any device that measures time consistently is appropriate. If the intervals are more than a second or so, then a stopwatch should be fine. Consider any source of error, and the implications of the error on your conclusions.

Differences by a factor of ten tell a story which may not be affected by a few hundred milliseconds one way or the other.

Best wishes,

Charly

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 28 April 2016 20:11
To: C.A.M.Lowndes
Subject: Advice needed - performance testing

Dear Charly,

as alluded to in our conversation on Monday, I do face a problem with comparative performance testing against other solutions, which I have been thinking about for a while.

For my own iPhone app, it is perfectly possible to record and recall precise time stamps for starting a search request, and receiving the result, making it very easy to get accurate test results.
In fact, it is even possible to logs incoming requests server-side (my LOG class already does this), so I am fully covered.

For third-party applications however, the closed nature of the iOS platform means I have no access to any performance statistics, let alone a possibility of adding specific code to enable performance measurement.

Server-side, I may be able to measure when a samba connection was opened and closed again (bearing in mind a 3rd-party app will access the samba directly, not using my server daemon), but I do not think this is a good approach, since I have no way of telling what causes the connection to be closed.

The best solution to solve this dilemma I can think of so far is to use an external stop watch.
I think this will give measurable results, since already now (without any optimisations in place), my software is quicker by a factor of about 10 than the Airplayer app, the main commercial solution I propose to compare myself against. But the resulting measurements would of course be quite crude.

Could you please advise what a best-practice approach would be to get valid measurements? Is the stop-watch approach acceptable, or is it entirely unsuitable to use in my paper?

kind regards,
Toby Scholz
tfs34
TM470

5 June 2016

Dear Toby, yes, cite as you wish.

We can do a tutorial before or after TMA03, as you wish; I am away from tomorrow until the 20th June so it would be after then.

Thinking of TMA03 as a draft of the report for the EMA is exactly right. So the headings should all be in place, although some aspects will await completion.

Take it in small steps and make notes of what you have left to do as you go along.

Have fun!

Charly

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 04 June 2016 15:14
To: C.A.M.Lowndes
Subject: Citing previous TMAs?

Dear Charly,

I hope you are well.

For TMA03 and the EMA, is it possible to cite previous TMAs, and your marking? My main reason for asking is that the ordinary reader will presumably not have access to those, since they are not published in any way, so it would be impossible to check any further details that may be found?

Also, will there be a tutorial for TMA03 / EMA?
I must admit I'm feeling a little bit lost with TMA03 at the moment, so I'm wondering whether I should approach it as the EMA and strip it down.

kind regards,
Toby Scholz
TM470
tfs34

12 June 2016

Dear Toby,

I suggest you discuss the changes and the reasons, and include a summary of what has been done, and your plans for what has not.

Best wishes

Charly Lowndes

Tutor, Open University

TM129 TT284 T215 T320 T324 TM470

Tel: 01684 573604

Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 11 June 2016 20:21:28
To: C.A.M.Lowndes
Subject: Schedules in the report for TM470

Hi Charly,

the schedule (of tasks) has now changed somewhat, due to moving operating system.

In the final report (i.e. TMA03 and EMA), should I present the original schedule and acknowledge there were changes, the altered schedule and point out it has changed, or both?
If TMA03 followed the structure of TMA01 / 02, I would discuss the changes made, and then include the revised schedule in the EMA. Since TMA03 is a draft of the EMA however, I am not sure if this approach is appropriate, since the report should presumably present the original schedule, and then acknowledge the changes made?

kind regards,
Toby Scholz
TM470
tfs34

15 June 2016

Dear Toby,

Sorry for late reply: this hid in junk mail for some reason. There are two parts to the EMA. The report on your solution is a technical report, so impersonal and more passive than active. The account of how you developed the solution can be less formal, but don't be afraid of using a passive approach. I will not be counting.

Best wishes, Charly.

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 04 June 2016 20:55:28
To: C.A.M.Lowndes
Subject: Active vs passive

Hi Charly,

I've glossed over this topic a bit in past submissions, but with a view to the EMA, what ratio of active to passive voice should I be aiming for?

The Writing, Structuring, Styling and Editing Reports paper on one hand cites George Orwell saying "never use the passive where you can use the active", on the other states passive should be used for "writing technical reports for publication".

I'm not sure how far the EMA is considered a "technical report for publication", but I imagine certainly the review bit will become quite ugly when using passive voice only.
In view of this, would a 40/60 ratio (active/passive) be acceptable, or would it be advisable to move more into one (or the other) direction?

kind regards,
Toby Scholz
tfs34
TM470

20 June 2016

Dear Toby,

I think that for the EMA the trick is to think of this as telling the story.

What were the risks as you perceived them when you started?

How did you deal with them?

Did anything change?

Looking back, how do you think your assessment worked out?

So for TMA03, you can give the story as it is now.

Remember that there are two parts to your project: the development, and the description and reflection on it.

I hope this helps,

Charly

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 19 June 2016 16:44
To: C.A.M.Lowndes
Subject: Risks

Hi Charly,

a quick question about risks if I may.

For TMA03, and indeed the EMA, is it expected to give a full account of all risks, and how they were mitigated?

My reason for asking is that the TMA03 paper says "Identify any significant risks to project completion", whereas in my notes from our feedback session, I wrote "reflect on how risks were mitigated, rather than just stating they are no longer risks".

With regards to risks to project completion, it's looking quite thin out there now. From a development perspective, the project is complete, with the exception of some beautifications and bug fixes scheduled for July. So the biggest risks to project completion are indeed that I don't know how to show risks in the project report, or getting run over by a bus (about 1 in 6.5m apparently, in case you wondered).

On the other hand, what I have written is unfortunately starting to look quite hefty again, looking for any opportunity to cut word count.

kind regards,
Toby Scholz
TM470
tfs34

23 June 2016

Dear Toby. This is looking pretty good. I have added a few comments, which I hope are useful. One area worth thinking about is the functional testing: it seems a little thin to me.

Do let me know if anything fails to make sense,

Best wishes,

Charly

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 21 June 2016 07:25:01
To: C.A.M.Lowndes
Subject: TMA03 - draft | TM470

Good morning Charly,

please find attached a (very rough) draft of what I imagine TMA03 might look like.

If at all possible, I would be extremely grateful if you could glance over it quickly during the course of this week, hoping you might be able to give me some pointers as to what aspects I might be best advised to work on.
Items in red generally indicate I am aware something needs to be done, figures and tables aren't numbered yet, and the reference list is a mess.

I have one specific question regarding the account of related literature:
Much of the literature currently weaves into (and is mentioned at or near) the account of the project work, and its outcomes.
For example, I used the TM354 module materials to structure my domain analysis. Is the expectation that I then provide an account in a separate section, (e.g. "I have read the TM354 module materials, and find them lacking in quality somewhat as they violate the OMG OCL specifications with regards to object navigation [insert reference]"), or would it be better to elaborate on literature as it appears?

Below, I have made a feeble attempt to assess my own work against the learning outcomes. Please do take it with a pinch of salt (Luft and Ingham taught us how little one knows about oneself); my main reason for including this self-assessment is to indicate what I see myself working on over the next two weeks (with which you will probably disagree. :)

many thanks, and kind regards,
Toby Scholz
TM470
tfs34

LO2: The nature and goals of the project are clear and consistent, and a valiant attempt was made to describe an extremely complex problem to an informed reader. However, it is less clear what impact this system would have on the world in a best-case scenario.

LO4: The wealth of information acquired demonstrates relevance to the project and has been applied to solve problems at hand. However, neither the credibility of the sources, nor their relative impact on decisions has been well accounted for at this point.

LO1: It is clear that the project benefits from an excellent understanding of the fundamental technical concepts and relevant principles, even if, in some cases, it could be clearer what specific decision led to an implementation in software.

LO9: The project plan and organisation were suitably minimal to warrant an excellent mark for the application of agile development principles. The records are witness to a very small amount of changes applied to the initially proposed schedule, demonstrating a good understanding of project management skills taught and tested in M364.

LO3: Particularly compared with previous TMAs, the account of activities carried out and skills acquired is less satisfying, presumably because of an effort to reduce word count.

LO8: The successful acquisition of a new programming language during the course of the project, as well as building on the foundations laid by previous OU modules are evidence that an great amount of independent learning has taken place. However, its process and progress could be documented better.

28 July 2016

Try putting this into Google Scholar:

perception 100ms delay

Look at refs [1] and [13] in Jansen, J., & Bulterman, D. C. (2013, February). User-centric video delay measurements. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (pp. 37-42). ACM.

Also ask yourself how critical this issue is and how much research time is justified.

And ...

Have fun!

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 28 July 2016 20:44
To: C.A.M.Lowndes
Subject: Quantification of the perception of immediacy of call and response

Dear Charly,

thank you very much for the review session earlier.

We touched on software performance in the session, and there is actually one key question, which I don't quite know how to address, or where to look for the answer:

As a rule of thumb, I have always worked on the principle that human beings do not perceive a delay when call and response happen within less than 100ms (in my experience, this applies to VoIP, where participants seem to detect a delay when round-trip latency is larger than this figure).

However, I very much struggle to find academic literature to support this claim.
I remember that Trevor Cox talks about shouting at a wall at various distances, and experimenting when we stop perceiving an echo, and start merging reflections with direct emissions in his boob *Sonic Wonderland*.
Using this example does not seem feasible, since I would have to derive latency figures from his estimated distances using the speed of sound, not what I would call solid academic evidence.

A Google Scholar search for *Perception of Immediacy* brings about many interesting articles (e.g. "Measurements of perceived nonverbal immediacy", which explores what makes us feel comfortable in somebody's company), but few (none?) that gives any concrete figures that would allow me to justify my 300ms target latency.

How might I find a relevant article?
Is *immediacy* the wrong word?
Should I restrict my search to certain areas of research (e.g. neurology)?

Much appreciate your help.

kind regards,
Toby Scholz
tfs34
TM470

15 August 2016

Dear Toby,

Details are important when they explain or provide evidence for your decisions. So your choice of what to include is part of the assessment of your communication skill.

Hyperlinks to the appendices make for easy reading.

I hope that helps -

Best wishes,

Charly

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 13 August 2016 19:57
To: C.A.M.Lowndes
Subject: Including the plumbing

Dear Charly,

in the interest of achieving a short EMA, is it worth including what you might call "plumbing"?

For example, the maintenance class I wrote last week does not really help address the underlying problem, nor does it employ any particular algorithms or libraries which are particular to this project. Nonetheless, it is rather important to producing usable software, and including it would show I have not sat idle in the last few weeks before submission.
Does it deserve a place in the body of the EMA?

Also, when I move the (functional / performance) test cases to the appendix, how should they be referred to in the EMA body?
Briefly, in the sense of "I tested it, it works (see appendix)", or more elaborately, in sense of describing a the nature and outcome of each test, only leaving the details in the appendix?

many thanks,
Toby

21 August 2016

No. You might add a comment about having published. You cannot plagiarise yourself. Turnitin is an alert system, not a decision system.

Best wishes,

Charly Lowndes
Tutor, Open University
TM129 TT284 T215 T320 T324 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 21 August 2016 15:29
To: C.A.M.Lowndes
Subject: Publishing the software

Dear Charly,

in line with my proposal to make the server component open source, I was just in the process of making the repository public.
However, it occurred to me that the question paper advises not to include auto-generated code, as the Turnitin software may detect it as plagiarism due to other software on the internet using the same auto-generated code.

So my question is:
If I make the repository public, do I then run the risk that the Open University will

accuse me of plagiarising my own work if the Turnitin software compares my paper with the repository?

kind regards,
Toby Scholz
TM470
tfs34

28 August 2016

Dear Toby -

Version 3 contains the most informattion.

Bear in mind that it is unlikely that the marker will read, as opposed to briefly check the existence of, an appendix. You will be assessed on the evidence given in the main body of the report.

Best wishes,

Charly

Charly Lowndes
Tutor, Open University
TM129 M269 TT284 T215 TM352 TM355 TM470
Tel: 01684 573604
Skype charly.lowndes

From: Toby F Scholz <tobythetenor@gmail.com>
Sent: 28 August 2016 15:07
To: C.A.M.Lowndes
Subject: Hyperlinks and Code comments

Dear Charly,

you previously encouraged me to hyper-link to appendices (,references, and glossaries) in the body of the EMA.

Is it then still necessary to refer to appendices by index?
E.g. should the main body read:

1) Refer to the test results for details...
2) Refer to the test results (see appendix) for details...
3) Refer to the test results (appendix 1.2.3) for details...
?

Secondly, I am providing some code excerpts in the appendix. These currently exclude details, such as in-code comments, imports, log entries, try..catch etc. Would it be advisable to comment within those code excerpts to highlight sections of particular importance (these are typically already shown in the main body), or would it be better to avoid this, lest I unnecessarily increase the size of the paper?

kind regards,
Toby Scholz
tfs34
TM470

4.18. LGPL license

See GNU (1999) for details.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is

linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the

ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

4.19. Apache license 2.0

See (Apache, 2004) for details

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You

institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!) The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

4.20. Oracle developer license

See Oracle (2014) for details.

Oracle Technology Network License Agreement

Oracle is willing to authorize Your access to software associated with this License Agreement (“Agreement”) only upon the condition that You accept that this Agreement governs Your use of the software. By selecting the “Accept License Agreement” button or box (or the equivalent) or installing or using the Programs You indicate Your acceptance of this Agreement and Your agreement, as an authorized representative of Your company or organization (if being acquired for use by an entity) or as an individual, to comply with the license terms that apply to the software that You wish to download and access. If You are not willing to be bound by this Agreement, do not select the “Accept License Agreement” button or box (or the equivalent) and do not download or access the software.

Definitions

"Oracle" refers to Oracle America, Inc. "You" and "Your" refers to (a) a company or organization (each an “Entity”) accessing the Programs, if use of the Programs will be on behalf of such Entity; or (b) an individual accessing the Programs, if use of the Programs will not be on behalf of an Entity. “Contractors” refers to Your agents and contractors (including, without limitation, outsourcers). "Program(s)" refers to Oracle software provided by Oracle pursuant to this Agreement and any updates, error corrections, and/or Program Documentation provided by Oracle. “Program Documentation” refers to Program user manuals and Program installation manuals, if any. If available, Program Documentation may be delivered with the Programs and/or may be accessed from www.oracle.com/documentation. “Separate Terms” refers to separate license terms that are specified in the Program Documentation, readmes or notice files and that apply to Separately Licensed Third Party Technology. “Separately Licensed Third Party Technology” refers to third party technology that is licensed under Separate Terms and not under the terms of this Agreement.

License Rights and Restrictions

Oracle grants You a nonexclusive, nontransferable, limited license to internally use the Programs, subject to the restrictions stated in this Agreement, only for the purpose of developing, testing, prototyping, and demonstrating Your application and only as long as Your application has not been used for any data processing, business, commercial, or production purposes, and not for any other purpose. You may allow Your Contractor(s) to use the Programs, provided they are acting on Your behalf to exercise license rights granted in this Agreement and further provided that You are responsible for their compliance with this Agreement in such use. You will have a written agreement with Your Contractor(s) that strictly limits their right to use the Programs and that otherwise protects Oracle’s intellectual property rights to the same extent as this Agreement. You may make copies of the Programs to the extent reasonably necessary to exercise the license rights granted in this Agreement. You may make one copy of the Programs for backup purposes.

Further, You may not:

a remove or modify any Program markings or any notice of Oracle’s or a licensor’s proprietary rights;

b make the Programs available in any manner to any third party (other than Contractors acting on Your behalf as set forth in this Agreement);

c use the Programs to provide third party training;

d assign this Agreement or distribute, give, or transfer the Programs or an interest in them to any third party, except as expressly permitted in this Agreement for Contractors (the foregoing shall not be construed to limit the rights You may otherwise have with respect to Separately Licensed Third Party Technology);

e cause or permit reverse engineering (unless required by law for interoperability), disassembly or decompilation of the Programs; and

f disclose results of any Program benchmark tests without Oracle’s prior consent.

The Programs may contain source code that, unless expressly licensed in this Agreement for other purposes (for example, licensed under an open source license), is provided solely for reference purposes pursuant to the terms of this Agreement and may not be modified.

All rights not expressly granted in this Agreement are reserved by Oracle. If You want to use the Programs or Your application for any purpose other than as expressly permitted under this Agreement, You must obtain from Oracle or an Oracle reseller a valid Programs license under a separate agreement permitting such use. However, You acknowledge that the Programs may not be intended for production use and/or Oracle may not make a version of the Programs available for production or other purposes; any development or other work You undertake with the Programs is at Your sole risk.

Ownership

Oracle or its licensors retain all ownership and intellectual property rights to the Programs.

Third-Party Technology

The Programs may contain or require the use of third party technology that is provided with the Programs. Oracle may provide certain notices to You in Program Documentation, readmes or notice files in connection with such third party technology. Third party technology will be licensed to You either under the terms of this Agreement or, if specified in the Program Documentation, readmes or notice files, under Separate Terms. Your rights to use Separately Licensed Third Party Technology under Separate Terms are not restricted in any way by this Agreement. However, for clarity, notwithstanding the existence of a notice, third party technology that is not Separately Licensed Third Party Technology shall be deemed part of the Programs and is licensed to You under the terms of this Agreement.

Source Code for Open Source Software

For software that you receive from Oracle in binary form that is licensed under an open source license that gives you the right to receive the source code for that binary, you can obtain a copy of the applicable source code from https://oss.oracle.com/sources/ or http://www.oracle.com/goto/opensourcecode. If the source code for such software was not provided to you with the binary, you can also receive a copy of the source code on physical media by submitting a written request pursuant to the instructions in the "Written Offer for Source Code" section of the latter website.

Export Controls

Export laws and regulations of the United States and any other relevant local export laws and regulations apply to the Programs . You agree that such export control laws govern Your use of the Programs (including technical data) and any services deliverables provided under this agreement, and You agree to comply with all such export laws and regulations (including "deemed export" and "deemed re-export" regulations). You agree that no data, information, program and/or materials resulting from Programs or services (or direct products thereof) will be exported, directly or indirectly, in violation of these laws, or will be used for any purpose prohibited by these laws including, without limitation, nuclear, chemical, or biological weapons proliferation, or development of missile technology. Accordingly, You confirm:

- You will not download, provide, make available or otherwise export or re-export the Programs, directly or indirectly, to countries prohibited by applicable laws and regulations nor to citizens, nationals or residents of those countries.
- You are not listed on the United States Department of Treasury lists of Specially Designated Nationals and Blocked Persons, Specially Designated Terrorists, and Specially Designated Narcotic Traffickers, nor are You listed on the United States Department of Commerce Table of Denial Orders.
- You will not download or otherwise export or re-export the Programs, directly or indirectly, to persons on the above mentioned lists.
- You will not use the Programs for, and will not allow the Programs to be used for, any purposes prohibited by applicable law, including, without limitation, for the development, design, manufacture or production of nuclear, chemical or biological weapons of mass destruction.

Information Collection

The Programs' installation and/or auto-update processes, if any, may transmit a limited amount of data to Oracle or its service provider about those processes to help Oracle understand and optimize them. Oracle does not associate the data with personally identifiable information. Refer to Oracle's Privacy Policy at www.oracle.com/privacy.

Disclaimer of Warranties; Limitation of Liability

THE PROGRAMS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ORACLE FURTHER DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT .

IN NO EVENT WILL ORACLE BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR DATA USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, EVEN IF ORACLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. ORACLE'S ENTIRE LIABILITY FOR DAMAGES UNDER THIS AGREEMENT SHALL IN NO EVENT EXCEED ONE THOUSAND DOLLARS (U.S. $1,000) .

No Technical Support

Unless Oracle support for the Programs, if any, is expressly included in a separate, current support agreement between You and Oracle, Oracle's technical support organization will not provide technical support, phone support, or updates to You for the Programs provided under this Agreement.

Audit; Termination

Oracle may audit Your use of the Programs. You may terminate this Agreement by destroying all copies of the Programs. This Agreement shall automatically terminate without notice if You fail to comply with any of the terms of this Agreement, in which case You shall promptly destroy all copies of the Programs.

Relationship Between the Parties

Oracle is an independent contractor and we agree that no partnership, joint venture, or agency relationship exists between us. We each will be responsible for paying our own employees, including employment related taxes and insurance.. Nothing in this agreement shall be construed to limit either party's right to independently develop or distribute software that is functionally similar to the other party's products, so long as proprietary information of the other party is not included in such software.

Entire Agreement; Governing Law

You agree that this Agreement is the complete agreement for the Programs and this Agreement supersedes all prior or contemporaneous agreements or representations, including any clickwrap, shrinkwrap or similar licenses, or license agreements for prior versions of the Programs. This Agreement may not be modified and the rights and restrictions may not be altered or waived except in a writing signed by authorized representatives of You and of Oracle. If any term of this Agreement is found to be invalid or unenforceable, the remaining provisions will remain effective.

This Agreement is governed by the substantive and procedural laws of the State of California, USA, and You and Oracle agree to submit to the exclusive jurisdiction of, and venue in, the courts of San Francisco or Santa Clara counties in California in any dispute arising out of or relating to this Agreement.

Notices

Should you have any questions concerning this License Agreement, or if you desire to contact Oracle for any reason, please write:

Oracle America, Inc.
500 Oracle Parkway
Redwood City, CA 94065

Oracle Employees : Under no circumstances are Oracle Employees authorized to download software for the purpose of distributing it to customers. Oracle products are available to employees for internal use or demonstration purposes only. In keeping with Oracle's trade compliance obligations under U.S. and applicable multilateral law, failure to comply with this policy could result in disciplinary action up to and including termination.

Last updated: 09 September 2014

4.21. Excerpt from TMA02 – Choosing MongoDB

The following excerpt was taken from Scholz (2016a, p. 47), based on recommendation by Charly Lowndes (Lowndes, 2016, 5 June 2016c)

> *In their paper Survey on NoSQL Database, the authors introduce two implementations of document-oriented databases, MongoDB (MongoDB, 2016), and Apache's CouchDB (Apache, 2016b). They praise MongoDB's support for "bjson data structure to store complex data types", a "powerful query language", and "High-speed access to mass data", which stands in stark contrast to the limitations of CouchDB, which include "only providing an interface based on HTTP REST, concurrent read and write performance is not ideal and so on" (Du et al, 2011, p. 366), making MongoDB the clear winner in this comparison.*
>
> *Grolinger at al. elaborate the MongoDB model, and find that MongoDB allows "either multiple readers to access data, or a single writer to modify them" (Grolinger et al., 2013, p. 15), a policy well-suited to our use case, where we may have multiple server components running for large organisations wishing to service clients in multiple locations from a single database.*
>
> *In her entertaining (if of dubious quality) blog History of MongoDB, Kristina Chodorow (2010) points out one of the Achilles' heels of MongoDB, which is that it is a relatively young project. Founded in 2007, MongoDB has neither the user base, nor the size of expertise behind it that larger competitors, such as Apache, have. Nonetheless, MongoDB ranks as the fourth most popular database (DB-Engines Ranking, 2016), and the feature set and performance of MongoDB far outweigh concerns about its future, in my opinion.*
>
> *In addition, MongoDB is the database solution used at my place of work, meaning I have expertise at hand, should I require help with any aspect of dealing with MongoDB.*

PROJECT PROPOSALS

Below are listed some project proposals that were also considered as a subject for this dissertation, in no particular order. This section is not part of the submitted paper.

Parkinson's disease – early diagnosis

Proposed was a collaboration with the Parkinson's Voice Initiative, to produce a small device that can be placed in the waiting rooms of surgeries. The device would allow patients to give a voice sample, which would be sent securely to the Parkinson's Voice Initiative for analysis. Following analysis, the physician, rather than the patient, would (confidentially) receive an indication whether a full diagnosis for the patient in question should be considered.

Ping pong counter

A device that can be placed near a ping pong table for the purpose of keeping score. The device (e.g. a Raspberry Pi) would use auditory and visual information to keep track of the movements of the ping pong ball, and record scores in a database connected via a network for further processing and recall.

Environmental soundscape analysis

A web-based platform that leverages microphones in our environments (CCTV, phones, weather monitoring, etc.) to statistically analyse the soundscape by types of sound, and produce an accessible set of data that allows an insight into the qualitative and quantitative aspects of our noise environment, and, over time, helps build a data set that lets tracking changes in the acoustic environment.

Sexual harassment / assault tracker

A mobile app that allows users to anonymously report incidents of sexual harassment or assault to a central database. Incidents would be added to a map, that gradually can build up an index of hotspots for assaults by frequency and time past.
This would help people who are afraid to report assault / harassment to the police (for example because the perpetrator is a family member) to report the assault anonymously, creating a record of the incident without the fear of being identified.
Law enforcement can use the map to target problem areas without linking it to any specific case, and use the additional data available in specific ongoing investigations.

Concurrent system for finding strongly connected components within a directed graph

A concurrent implementation of Tarjan's algorithm to build a directed graph of strongly connected components of otherwise discrete systems.

Collaborative online drawing platform

An online drawing platform that allows multiple people to draw on the same canvass simultaneously.

Transport tracking system

The London Rail and Underground are effectively two separate transport systems, and although often a combination of both is the best way to travel, the journey planner for the underground will only return journeys solely made by underground, and the journey planner for rail will only return journeys made solely by rail respectively.
The new system could combine information from both underground and rail (leveraging the public API both offer) to calculate quicker journeys combining both modes of travel.

Supermarket self-checkout next available till predictor

In front of large self-checkout installations, lines often bunch up at one point, and customers fail to spot free checkouts that may be further away, or out of sight.
Based on artificial learning, a prediction algorithm could be created that will tell customer which checkout will be the free next, allowing checkouts to be used more efficiently, thereby reducing the overall waiting time.

Security of IPv6

Evaluation of network security of IPv6 compared to IPv4, looking, among other topics, at NAT-free environments, random source addresses, and built-in IPSec encryption of IPv6 versus the respective technologies in IPv4.

Monitoring tool for WAN connections

Centralized tool to monitor WAN connections in an organization, gathers statistics, and sends alerts in case of problems.

App for my local community association

A mobile app for my local community association that integrates documents, emails and calendars of the association for all members to easily view / edit (based on permission levels).

Household application tracker

Web platform that allows gathering manuals, receipts, warranties etc. for household products in any one household, and links in with manufacturers and third-party suppliers to quickly process warranty claims, and locate spare parts easily when necessary.

Outlook / Jira integration

Add-on for Microsoft Outlook to identify relating Atlassian JIRA cases and log time.