# Introduction

This sub-report documents the efforts made by myself in researching appropriate material for the design and development of the final deliverable. To show the full extent of the research performed, all of the stages involved have been included, even aspects that won’t make it into the final design of the end deliverable.

The purpose of this report is to not only to show what research has been carried out, but to also show how the research will ultimately affect the end deliverable. The main contents of this report will discuss:

* Software Evaluation
* Platforms and technologies
* Methodologies
* Academic research
* Evaluation of commercial alternatives

The project I have undertaken is fairly technical in nature, therefore, I have split the report into themes that are common in computer science and other relevant areas.

# Research Approach

For the majority of my research I follow a standard approach, a review of relevant literature and practical application. To get a better understanding of the techniques and technologies that will be implemented into my system

//Experimentation//

//..,. in the worst case that there is no deliverable, I have made the assumption that I will be evaluating my system myself, using methods that will be researched at a later date…//

# Project Planning

At the beginning of the project I detailed all of the tasks that needed to be completed in a gantt chart. I placed all of the activities that needed to completed in a sequential fashion, where one task led to the next. Research unveiled that what I had implemented was the waterfall process model. Hughes and Cotterell (2006, pp.//FIND PAGES//) explain how students can easily misjudge time allocation of tasks, simply due to the fact that they will be using tools that they are unfamiliar with. This point will play a major factor in my planning approach as most of the tools/software/techniques I will have never used.

//diagram of Gantt chart//

## Waterfall approach

The waterfall method is seen as one of the more basic models of system development. It typically follows the pattern as shown below.

//waterfall example//

As Hughes and Cotterall (2006, pp. 75-76) explain, the waterfalls main strength is also its downfall; its lack of flexibility. The clear-cut nature of the waterfall model promotes strict adherence to deadlines, however there is great importance in defining the exact deliverables early on as there is very little room for change later on in the development process.

For very straightforward projects with low risk of change, the sequential nature of the waterfall model is ideal. This project will most likely go through unpredictable iterations of changes and experimentation.

//pros, small summary//

//cons, why it isn’t suitable for my project//

## Incremental Approach

With the limitations of a //waterfall model/sequential// model in mind, I searched for a process model more suited to this type of project. The spiral model is an iterative process model. As Hughes and Cotterall (2006, pp. 76-77) explains – Aspects of a system that are hard to specify could be better developed using an incremental approach. The spiral model’s key strength is the fact that it is able to cope if various phases of the project encounter complications, or unforeseen issues that hinder development.

Incremental development is well suited to experimentation and prototyping; Sommerville (2010, pp. 32-31) states that incremental development achieves a final system implementation by progressively validating new versions. This will be ideal for my system as I intend to implement at least one prototype.

As this is such an early stage in my project it is impossible to pin down a time allocation for every single task that I will be performing. If I had greater experience in risk assessment, or the system was very clear to produce then maybe a waterfall model would be the better choice to base my system upon. The fact that an incremental approach, such as the spiral method, accommodates possible contingencies in development, this makes it an ideal approach to use.

//show example spiral process/ or my own//

An iterative/incremental approach looks almost perfect, so what are the disadvantages? Is there a limit to the amount of iterations a system can cycle through? Sommerville (2010, pp. 34-35) explains that an incremental approach may degrade a system’s structure, as functionality is constantly being ‘tacked on’, and resources can be used up by constantly documenting new implementations.

## Summary

There are a lot of factors to consider when implementing a project plan. Every process has its pros and cons, however there is flexibility when it comes to choosing. A process model is simply an abstraction for a certain application, which means it doesn’t have to be strictly adhered to. The most advantageous approach for this project is most likely incremental development. With the amount of uncertainty within this project, the most valuable asset I need in a plan is flexibility. Iterative development excels when it comes to trying out various features through prototyping – a task that will feature heavily during the development of the system.

I believe that the disadvantages of an incremental approach outweigh those of a typical waterfall based development. My project is small compared to the large projects that these process models have been based upon; there aren’t other team members to manage; no customers are involved; and I doubt that my system will grow so large that it will become unmanageable. These points alone should lessen the impact of any disadvantages.

# System Overview

My main objective is to develop a working prototype of the system. As stated above, the focus at this current time is to develop server-side environment that provides collaborative functionality to multiple client devices. This section discusses the purpose of the system and what the system’s functionality is. This is by no means the final layout of the system as continued research and experimentation continue to show other avenues of development.

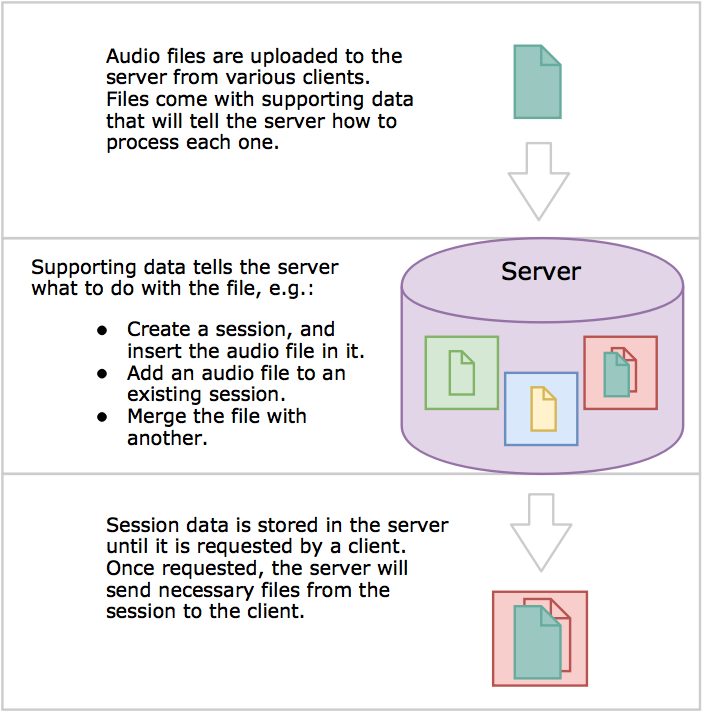
The system requirements haven’t changed drastically during my research, apart from the focus changing to a server-side system. My research was based off of fulfilling the requirements outlined below.

## Initial Proposal

The first proposal for the system specified the development of an environment that would allow users to upload and interact with each other’s content via a //remote// server. The initial proposal was heavily centred on a client/mobile device, with the mentality that the GUI and //network programming// would be the main technical hurdle to overcome. However as I continued my research, I found the opposite to be true. My research unearthed that there are many //frameworks/open source aids// that can create a basic client application for you on many different platforms with ease. I found that servers on the other hand, or at least frameworks for collaborative audio specifically, were virtually non-existent. For this reason the main focus was shifted to researching how the server will operate.

### Initial Server Setup

This diagram was created at a very early in my research stage to better define what was being created. It shows, very basically, what a client will typically upload/download and how the server will process these files.



### Use Case

Below is a very basic use case diagram outlining the interactions of the system with its environment. I used this diagram to help construct the system requirements.

//Use case diagram//

## 2.2. Basic System Requirements

The system requirements haven’t changed drastically during my research, apart from the focus changing to a server-side system. My research was based off of fulfilling the requirements outlined below.

### Functional requirements

Functional requirements are determined by the outcome of individual functionalities within the system itself. From an early point it was established what actions the system would need to perform. Functional requirements are typically generated from a use case diagram. The functional requirements of my system are as follows:

### Non-functional requirements

A non-functional requirement can be outlined as a characteristic that affects the broader elements of the system. Sommerville (2010, pp. 87-91) states that the non-functional aspects of a system are often the most critical. A user can typically avoid an encounter with abnormal system functionality, whereas a single non-functional feature that isn’t meeting requirements can often cripple the entire system. The non-functional requirements that affect my system include:

#### Implementation

As the system currently stands, all programming for the server will be carried out with the Python programming language, and will be executed on a Linux-based platform. The client prototype will be implemented using Python’s TkInter GUI library. If the project is then ahead of schedule then a final client for mobile will be developed using Intel’s XDK platform using JavaScript, HTML5 and CSS.

#### Performance

The nature of the system will mean that the server will have significant overhead, especially if there are many clients interacting with the server at once. This will mean techniques, such as concurrent programming, may need to be employed to achieve a more efficient system. //Users could potentially be equally distributed in a multithreaded system.//

The client implementation will most likely require slightly less computation. Audio file formats will need to be used efficiently in order to get quick transmission between client and server.

# Evaluation of Existing Technologies & Techniques

The initial steps of my research involved finding similar systems that already existed and how they achieved a collaborative environment. This research was vital as it uncovered solutions to implementation issues that I was facing. I will discuss existing solutions and technologies related to my end deliverable. More specifically I will look into how they handle multiple users, how they implement a collaborative environment, and the various tools and techniques used.

## Collaborative Services

### Kompoz

Kompoz (//ref//) could be defined as collaboration as a service, it allows musicians to upload projects from their computer and download other user’s projects.

#### Collaborative Environment

To collaborate with other musicians, the user selects a session from the site’s ‘Collaborations’ page. A user can then submit a recording to the session by uploading it from their computer. However the creator of a session must accept a submission for it to become part of the final track, otherwise it is labelled as an ‘idea’.

In terms of version control, this implementation may be the least technical in the list. Essentially a ‘creator’ user controls all submissions, so there is no risk of conflicting versions as only a single user makes decisions about the tracks.

### Splice

Splice (//ref//) is essentially another cloud collaboration service. The main target audience seems to be musicians creating mainly electronic dance music, this is assumed due to the content on the site, however it would seem there is nothing stopping other genres from using the service. A Splice client must be downloaded to the user’s computer in order to upload and download projects. Projects must be made in a compatible digital audio workstation (DAW), such as Ableton Live 8 or Logic Pro X //refs?//.

#### Collaborative Environment

The service’s main feature is providing access to other user’s projects, allowing you to create alterations via your personal DAW. Once a user has made adjustments to the project and uploads it, it becomes a new ‘version’ alongside the original project.

//show picture of me editing tiesto’s track//

In this case no merging of files actually happen unless you consider the operations that the user performs in their DAW. This software highlights a different approach to collaboration: all users can create their own versions; this effectively makes version control unnecessary.

### Jamly

Jamly is another musical collaboration service, however all functionality is provided via its website. Jamly uses JavaScript within the browser’s website to record both audio and video from the computer’s default camera and microphone. To collaborate with other musicians, the user must select a session from the home page. The user is then allowed to watch the selected session. If they like what they hear, the user can go on to record their own section within the session. Once on the recording page, the user records their device whilst listening to the original version to keep in time. Finally the user can listen/watch a preview of the session, correct any latency in the recording, and upload.

//show picture of Jamly recording interface//

#### Collaborative Environment

Jamly operates quite differently to the other two services listed. Firstly, the service records your audio directly, whereas the other two services required you to upload a file. Secondly, it records video, the benefits of this are debatable; it may be beneficial as a musician to observe other musicians for visual cues when recording (e.g. watching a drummer to help you keep in sync), However the musicians within the video are quite small, and the video is often of poor quality depending on each user’s capture device.

Jamly’s approach to version control is also unique in this list. I tested the version control system within Jamly by creating my own session, and then recording two sections simultaneously on separate tabs within the browser.

|  |  |
| --- | --- |
| Macintosh HD:Users:jackholmes:Documents:Development:Resources:Images:No Conflict.tiff | Macintosh HD:Users:jackholmes:Documents:Development:Resources:Images:Conflict.tiff |
| Default  A typical merge operation – a recording is made and saved to the session. | conflict  The session forks if a concurrent recording is made during another recording. |

Jamly effectively forks a session in two if a version conflict is detected. This implementation of version control avoids the need for a ‘controlling’ user to manually choose what to do with the separate versions. To record a new section, a user must select which version that they want to collaborate with.

|  |
| --- |
| Macintosh HD:Users:jackholmes:Documents:Development:Resources:Images:Post Conflict.tiff |
| Post-conflict  Additional recordings can be added to either end of the fork. |

Any new user can now collaborate with the new version or the old version. One criticism of this system however is that there is no warning to any user that a concurrent recording is being made. Concurrency issues are discussed later in the report.

Another note to make about this system is that it allows you to compensate for recording latency and change the mixing volume for the recording. This highlights some possible issues with this type of system. Firstly all client devices could be different, this will mean that they will all have different recording equipment, different processing power (latency issues), and they could all output a mixture of audio formats.

## Version Control Systems

Due to the collaborative nature of this system, I looked for collaborative services or open-source frameworks that may provide some insight to how to deal with multiple users uploading/editing shared content. There are many systems that exist for version control, however most of them are built for software development. They all deal with the same issue: providing a system to manage change in documents or files between multiple users. This section highlights the techniques used by these systems.

### Conflict Resolution

There are typically two methods to avoiding conflicts: locking and merging.

#### Merging

Git is a version control system. The main feature of Git that is most relevant to this project is how it handles conflict resolution. Git uses a merging concurrency model; this basically means that a user has to physically manage how data from two files will create a new file. Users are allowed to edit the same document as each other at a local level,

#### Locking

Locking may be considered an old-fashioned technique, however it is still a valid option to consider. Locking is still used by version control systems however in most cases it’s used in conjunction with a merging approach. In systems where users are unable to communicate, a locking mechanism may be necessary, as //[ref]// states, that binary files, such as audio, are often impossible to merge back into each other.

However //[ref]// does mention that locking doesn’t prevent conflicts. Where merging allows many users to work on the same file, locking explicitly allows only a single user to make changes at a time. This indicates that a hybrid approach may be the best approach for my system.

## Open-Source Audio Tools

A key technicality in my system is manipulating audio files. My search started with open-source solutions. Using an open source implementation was beneficial as it could show me the necessary techniques involved in editing audio files.

### Audacity

Audacity //[ref]// is a program that allows users to record and edit audio files. An advantage of Audacity is that it’s cross-platform, and the source code is available for download. I downloaded the source code and opened the project in xCode//[ref]//. My initial reaction to the source code was that it was far too large to gain any useful information from. The majority of Audacity is written in C++, trying to understand such a large program would consume far too much of my time. By inspecting the various libraries that Audacity used I discovered some libraries that looked promising – ‘libsoxr’ and ‘WxWidgets’.

### Sound Exchange – SoX

Within Audacity, ‘libsoxr’ is a library used by SoX[], an audio file manipulation program. SoX is a command line utility, this was also a benefit, as I knew I could access the functionality of the program without delving into the ‘libsoxr’ library . SoX is actually constructed from a collection of other open source libraries, giving it a huge amount of functionality. SoX is also open-source and is written in C, making it cross-platform, ideal for development prototyping.

How I intend on implementing SoX

..following some experimentation…

## Summary

As seen above, collaboration can be implemented using many different techniques. In terms of collaborative functionality between users, two patterns seem to emerge in all of these services and tools.

#### User-Dependent Collaboration

Within Kompoz and GitHub is typically a hierarchical structure of users that governs how files/content are authorised into a final/master ‘mix’. These collaborative services keep the collaboration process ‘on rails’ - a linear approach, however one that could be deemed suitable for its application. Github, is slightly different to Kompoz, as it gives users full reign of what they can change and upload, however, if conflicting files have been made, or multiple users try to upload to the same branch there can be great complications when trying to merge it back into the original ‘Master branch’. Git’s strength and weakness is that it needs micro-management of every single change.

#### User-Independent Collaboration

Services such as Splice and Jamly use a system where users don’t have to control how collaboration flows. A single user doesn’t control iterations of collaboration; instead whenever a user uploads an idea, a new ‘session’ is effectively created. Jamly manages conflicts by simply ‘forking’ into two separate versions. Splice doesn’t even consider version control, it simply saves files as new ‘versions’ for each user.

# Server Development

The server will perform tasks essential to the operations of my system. It will have to manage clients and will have to perform data transfer tasks.

### Prototyping

To prototype my system I will need access to a server environment. This will allow me to test both the client and server aspects together. I will be using XAMPP //[ref//, which is built upon standard server tools, such as Apache HTTP server and MySQL Database. Apache HTTP server is the most popular web server in the world. It is typically used with a UNIX/Linux OS this is due to it being free and multi-user oriented. This is useful as all of my development will be in OSX, a derivative of UNIX.

### Server-side Languages

In order for a language to be considered for the development of a server, it needs to be able to interact with key aspects of typical server functionality, such as sockets and protocols. As already stated, my project will most likely change as I experiment with various features and implementations. This means that the language I pick will need to be flexible, cross-platform, and high-level //[ref]//. The two languages that I decided to focus on were PHP and Python. This was a decision based upon research findings and personal familiarity. I could have written my system in C, however low-level program development is known for complexity and time consumption //[ref]//.

When initially experimenting with code I was naturally drawn to PHP. It’s known for web-based applications, and I was already rather familiar with it. PHP is a general-purpose programming language, like C, however it is considered to be high-level//[ref]//, and is interpreted. PHP’s main purpose is for developing dynamic web pages //[ref]//, however its also known for its flexibility in being able to develop a much broader scope of applications.

Python is again another high-level interpreted language. Its main purpose is for improving productivity //[ref]//. Python’s syntax and code structure are centred on human-readable code making it ideal for beginners. Python is also highly flexible due to a large standard library.

Both languages seemed ideal for my application, however only a single language would be used in the final deliverable. By carrying out various experiments with both languages I discovered issues with PHP that made less suited for this project.

### Python

I decided to choose python as the primary server-side language. Python is ideal for this project. The language is centred on fast deployment and supports iterative development through its great support for extendibility. I came to my decision by putting the Python language against the most important requirements I would need for the development of my system.

#### Flexibility

Flexibility is one of the most important characteristics that my project will require. Flexibility will ensure that the language can adapt to the changing needs of the project specification. Python’s main strength, or perhaps its weakness, is its large standard library //[ref]//. Python makes use of modules; these are basically libraries, which can extend the functionality of a program. If a user wants to program socket functionality, they can, by simply using the import keyword.

If the project uses prototyping, then Python is ideal. Python itself can run quite quickly at runtime compared to other interpreted languages //[ref]//. If extra functionality, or optimisation is still needed, Python supports extension through C and C++ code. C/C++ code can be written as modules and simply imported into a Python script.

Python’s ability to balance high-level and low-level functionality is what makes it ideal for this project.

#### Usability

One of the most noticeable features of a Python program compared to other languages, is its clarity. Python was created with readability of code as one of its main features. Rossum (1996) states that this was done to make code more reusable. Python’s clarity also helps in learning the program. I have very limited experience with PHP, however I already feel I am far more competent with Python, considering my small time in using the language. As stated at the beginning of the report, learning any tools being is important to consider, as it will have a knock-on affect with any set deadlines. //[ref] Python is easier to learn//.

//show python and php programs side by side//

These are short programs that I wrote, both create the same outcome, however notice the syntactical difference between them. PHP uses C style syntax, which can differentiate between each coder. Python’s code structure means there is usually only a single ‘standard way of laying out code. Below is a prime example of what Python avoids.

//show C program and how curly bracketing can vary between users.//

#### Efficiency

Due to Python’s high-level nature, it can often achieve the same outcome in fewer lines of code than a similar program written in a low-level language. //[ref] shows that code written in Python can sometimes be up to 5-10 times shorter than the same implementation in a lower level language, such as C++//

Efficiency in writing short concise code shouldn’t be confused with efficient execution of code. Python runs relatively slowly compared to a low-level programming language such as C. Speed and efficiency rely greatly on how code is implemented, so judging python on these factors is rather subjective. /[ref] study shows that for certain tasks, such as text handling, Python was …

As said earlier, optimisation can be added later in development through module extension. This just shows that Python doesn’t excel when developing performance sensitive applications, however, it is capable of very fast deployment. //[]how long it takes to write a program in python vs other languages.// Python is a great ‘glue’ language; it provides the basic structure of the program, but provides faster execution by including, modular low-level code.

### Python IDE’s

#### IDLE

Python on mac comes equipped with its default IDE – ‘IDLE’ //[ref]//, A simple application that provides access to the Python shell. It gives users all of the standard features of an IDE, such as a debugger and syntax highlighting. This application is suitable for smaller applications //[ref]//. I decided to look for other platforms that could help me obtain maximum productivity when developing Python code.

//it is good for running small bits of code, or testing small features//

#### PyCharm 5

After some basic research, I found that PyCharm IDE was given high praise for its productiveness; it is also one of the most popular Python IDE platforms //[ref?]//. PyCharm 5 allows you to create ‘projects’, similar to other IDE’s, such as Visual Studio. A project is a collection of code resources that you can assign a name to. In IDLE the user has to manually create directory structures themselves in order to make sure all of their code can co-interact. PyCharm 5 ensures that all the correct resources are available, and provides easy navigational access to them.

Another feature that make PyCharm 5 favourable over IDLE is the fact it has code completion. PyCharm 5 will display suggestions as you type, increasing productivity even further. As concurrency may feature in my system, PyCharm 5 can visualise the states of threads running in the program, this will aid greatly in my development, as developing concurrent systems can often be quite a complex task.

### Handling Clients

Concurrency issues of my system… …changing session data poses a race condition... …high level...

Two concurrency issues: handling clients & handling how clients access and write to the same file.

Semaphores: djikstra, Locking

# Client Development

The client side of development will involve implementing support for key elements, such as: file transfer, audio recording, audio playback, GUI, user keyboard input, and …// // I have considered tools and techniques for both the prototype and the final implementation.

The first implementation of my system will be a prototype; this will require a different approach to a full-fledged implementation. Prototype development is centred on quick deployment, so that the core functionality of a system can be tested. This means the tools I pick to develop the system will need to reflect this.

## Audio Recording & Playback

I will be using PyAudio to perform the core functionality of the client. PyAudio is a cross-platform audio I/O library, based on the PortAudio library. I will use It in conjunction with wxPython to implement the client functionality.

## GUI

My default choice was going to be Tkinter, a GUI package that is part of the Python standard library. The main issue however is its number of widgets, which are somewhat limited compared to other GUI libraries. Due to this reason I chose another GUI library – wxWidgets. I wanted to avoid dedicating to Tkinter and then realising that it cant implement the features that I need.

I first encountered the wxWidgets library when I was searching through the source code for Audacity. wxWidgets is a cross-platform GUI library. The implementation for Python that I’ll be using is called wxPython. This implementation isn’t part of the Python standard library, however it has more widgets than Tkinter, making it far more flexible. A major downside to Tkinter is the fact that…

Existing audio tools have already been developed using wxPython and after viewing the source code it was evident that it is relatively low-hassle to implement.

//show picture of wxPython audio player//

## Mobile Application Development

Originally, the client side of the system was originally planned to be built on a mobile platform, however as shown in this report, research has changed the path of development. If time permits then a mobile deliverable may also be developed. Here is the research I undertook, exploring this area.

### Web, Native or Hybrid

Applications for mobile can be developed using a variety of methods, each with its own strengths and weaknesses.

Web applications are the easiest type to develop out of the three. All smart mobile devices now feature a web browsing functionality so that they can display web pages. Web applications use a devices web functionality to display information in the same way. Web apps are essentially web pages packaged as an application. By using CSS and HTML, web apps are very quick to develop, using web development packages such as Dreamweaver. The downside to web applications is they have very limited access to the devices native functionality (microphone, camera, file management, etc). With the emergence of HTML5, web applications continue to grow more and more powerful.

Native Applications are on the other end of the spectrum of mobile application development. These applications require much more time to develop and require an IDE such as Visual studio or xCode to develop. Development of a native app gives the developer much more control. All of the device’s functionality is available through a native API, giving access to low-level features. Native development requires much more specialised knowledge on specific mobile architecture.

Hybrid applications obtain the best features of both web and native development. Using a development tool such as Intel XDK or PhoneGap, you can achieve native functionality, whilst implementing the GUI using a web-based approach.

### Intel XDK

Intel XDK is an IDE for hybrid mobile app development. It uses the Cordova API to encapsulate the native functionality of major phone OS’s. Effectively letting you write an application once that will work on multiple devices (This is impossible in native applications development). Intel XDK uses JavaScript, HTML, and CSS, yet still is able to deliver an app almost as flexible as one built natively.

//picture of XDK in action

# File Formats & Storage

//By using the use case diagram as guidance,// we know that data that is going to have be stored on both the client and the server. At some point some of this data will need to be transferred over a network. This section focuses on file related issues that may affect the operations of the system.

## Audio File formats

The largest type of file that my system will have to handle will be audio. Audio formats can range in size quite drastically and can be either lossless or lossy in their compression. When choosing which audio format the system was most likely to use, I considered four main points:

* How will it be created?
* What operations will be performed on it?
* What is its purpose?
* How will it be stored?

Audio files are going to be created on the client device, already, this raises the question: What formats does the client support? The prototype will be developed on a typical desktop operating system(most likely OSX), these systems generally support a very large range of audio formats by default. However if I were to develop for a more specialised platform such as mobile, then certain distributions will have limitations on what formats they support. SoX supports an extensive amount of formats. If client audio files differ too much then SoX will be able convert the audio into a different format once they’re uploaded to the server.

Audio files will undergo modification in the system, such as merge operations. These operations will typically take an audio file, convert it into a modifiable format, modify it, and then reformat it into the original format. If this is applied again and again to a lossy format then quality of the recording will degrade with every modification. To avoid this situation a lossless audio type such as .WAV should be used.

For the prototype I will be using PyAudio to record and output the actual audio files. This uses the standard python audio modules. Python supports a small range of audio formats, including .WAV. .WAV files can be included in a program by simply using ‘import wave’. Other formats can be imported into Python, however my focus it simply implement the core functionality in the form of a prototype, so .WAV will be sufficient. If I was developing a final version of the system, where handling audio files was time critical, then I may have picked a lossy format such as .MP3.

The system is being implemented to aid collaboration through audio, meaning that recording quality needs to be guarded to an extent. I am also focussing on a prototype implementation, so .WAV is ideal for my development.

## Data Storage

There two types of data that my system will have to store: text and audio. This section discusses the techniques and tools that help implement effective storage solutions.

My system is going to handle multiple users, each user will have an account, they may have a session associated with them, they may be involved in another user’s session, they will have recordings associated to them, etc. Complex and relational data is best stored in a database. A database, such as MySQL sets up the initial infrastructure so that data can simply be organised into tables. User details, session data, associated audio file locations can all be accessed in a structured manner.

Depending upon the implementation of my prototype there may be no need for a database as the amount of users may be so low. In this case data could simply read in from text-based formats such as JSON.

In either case the server will need to also store audio files. The method I will use will make use of the default file system. This makes sense as the file system’s found on modern operating systems have already been extensively optimised.

## Data Interchange Formats

Data interchange formats provide a structured method of reading and writing data. If structured collection of information needs to be passed around a system then ideally a data interchange format will be used. A data interchange format, such as JSON, essentially provides a standard method of structuring data. Simple textual data could be used to pass simple information, however problems could soon arise if the amount of information grows.

Consider this scenario in a system such as this one: A client device wishes to view or download a ‘session’ (a collection of recordings made by various users) – This will mean that some form of information will need to be passed from the server to the client device. Information such as the session name, duration of the session, users in the session, etc, will need to be transferred. If we implemented a system where plain textual information is exchanged between the client and the server, then we have to establish a common layout of information so that A) the server can create it, and B) the client can read it.

Data interchange formats provide a scheme, so that textual data can be written and read efficiently.

### Python & JSON

As I will be using Python, it makes sense to use JSON. JSON data handling is part of the standard Python library, giving me access to essential decode and encode functionality. JSON is extremely readable, especially compared to other data interchange formats//[ref?]//. Another reason why using JSON makes sense is because it basically follows the same syntax as some of the data structures used within Python.//[ref] state that JSON syntax is very similar// .JSON entries can easily be accessed as Python lists and dictionaries.

//show example JSON implementation in python//

## Python & FTP

As stated earlier XAMPP provides an FTP server//?//. It makes sense to use this as my default method of transferring files within my system. Python comes with a FTP library that allows you to perform a variety of operations. Below shows how I transferred a file via FTP.

-file transfer

-tcp

-Python can be extended to program sockets

# Concurrency Considerations

Dining philosopher

Multiple clents could possibly write to master track within application

Possible solutions…

# Conclusion