Research will include aspects of:

* evaluating software,
* platforms,
* methodologies,
* academic research,
* evaluation of commercial alternatives, etc.

If a project possesses a final application as a deliverable then it is expected that the student will have commenced their consideration of the expected s/w development platform.

**STRUCTURE**

**Introduction**

System will be implemented as a prototype in Python, low-level optimisation can be added later.

Aspects of system:

Server, client, other(data exchange formats)

Brief overview of system

**Similar Technologies**

* What are their client/server operations.

<http://www.kompoz.com/music/home>

a web site, makes use of a Ideas -> accepted process. This would solve concurrency issues, however one user has all power.

<https://splice.com/>

Collaboration through midi files and audio samples. Pros: especially good for electronic music (many synths) and high–end productions. Cons: needs to be compatible with existing DAW software, potentially very large project files.

<http://www.jamly.co/>

Collaboration through video files pros: good for demoing, cons: larger files as it is video being uploaded, gui is java based(less compatibility, only available through browser).

Audacity

Accessed Audacity[ref] source code and soon realised C/C++ code was far too complex for my timeframe.

Audacity led to discovery of the Sound Exchange Library[ref] and other open source libraries(SoundTouch[ref], PortAudio[ref]).

**Colaborative Tech**

Github -

**CLIENT**

**Development**

Platform Considerations (native mobile, mobile web app, mobile hybrid app, Desktop)

What will it need to do?

Desktop or Mobile – most suitable for prototyping?

Native vs Web vs Hybrid – most suitable for prototyping?

HTML5 – allows more features, however still not fluid

Native – gives access to all features of device

**Cordova**

What functionality does it allow

PhoneGap/Intel XDK

**GUI**

WxWidgets[need ref]

Tkinter[need ref]

**HCI**

**SERVER**

Identify:

* What needs to be stored.
* Why does it need to be stored Data Objects (Structures).
* How will this data be processed (algorithms).
* Issues:
  + Multi user system (asynchronous, concurrent?)
    - Semaphores
    - Locking Strategies <http://www.blackwasp.co.uk/optimisticpessimistic.aspx>
      * Make timeline example, as shown in link above
  + Access rights

What will it need to do?

* Handle audio files (receive/download)
* Deal with multiple user input to a single session

**Storage**

Compression?

Audio files – how big are certain formats? Compression for transfer?

File structure?

**Python**

Overview :

* Highly adaptive(works with other languages)
* Who uses Python (google, nasa…)
* Interpreted, syntax, speed, compatibility
* Object Oriented/Classes?
* Does it work with data exchange formats
* python methodology(optimise later): pros + cons

Trends of use in web servers - <http://w3techs.com/technologies/details/pl-python/all/all>

Strengths

* Easy to understand syntax
* Large Library
* High level

Cons

* Large Library
* High-level

vs PHP

https://wiki.python.org/moin/PythonVsPhp

**CGI**

Part of http

What does it achieve

Advantages

disadvantages

<http://webdesign.about.com/od/cgi/a/aa021599.htm>

**OTHER**

**Scripting language vs Compiled language**

**SOX (SOund eXchange)**

Handling audio files

Client audio formats

Where will mixing/latency compensation come into the system (client or server?)

**JSON, XML - Data exchange formats**

Simply textual information

JSON new standard – widely supported(python natural choice as it basically uses the same syntax(maybe show diagram)) simple, easy to work with(human readable)

XML dated - complicated

**HTTP/FTP**

Upload to server

Download from server

**Development Environment**

Github

My github https://github.com/jackholmes1992/ProjectRepo

IDLE

XAMPP – Apache Web Server

**RESEARCH/EVALUATION METHODS**

**DESIGN**

**Client**

As soon as the user opens a session for collaboration, the client will send a notification to the server to create an active session.

**Server**

Active Session Class

Whenever a user starts interacting with the application or data on the server, python will create an active session object. This object will handle user input (audio files), and manage who is able to make changes to the Session. The session class will also implement locking mechanisms/semaphores to stop concurrency issues.

The activeSession class will make use of processes/threads to manage multiple user requests. Mutex locks will ensure critical/sensitive sections aren’t accessed at the wrong time. This will improve UX.