DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Final Year Research Project 2009, Final Report

Project Title:	**Record, Mix, Play, Share: A web-based collaborative music creation environment**
Project Number:	**103**
Supervisor:	**Assoc.Prof. Gillian Dobbie**
Second Examiner:	**Assoc.Prof. John Morris**
Author:	**Danver J. Braganza**
UID:	**4316790**
Project Partner:	**Joseph Hobbs**
Date Submitted:	**Monday, 14th September 2009**

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

Signed: [Danver J. Braganza]

Record, Mix, Play, Share: A Browser-based Collaborative Music Composition Environment

Danver J. Braganza

Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand
dbra072@aucklanduni.ac.nz

Abstract

This report details the final year project on which I worked. The aim of this project was to develop a web application, Mixa, which would support collaborative music composition.

Music composition is a creative activity that could benefit from a collaborative approach. Similar websites and applications that attempt to deliver collaborative composition exist, but have a number of differences in their goals and implementations.

The key features that were successfully implemented by Mixa are client-side audio capture, browser-based composition and version control of songs. Other features include support for sharing, streaming audio online and tagging clips.

As Mixa is a web application, the implementation is subject to constraints and required the integration of a number of technologies. Due to the limitations of the project with respect to time, an extensive evaluation not possible. A usability evaluation was performed upon the composer module of the system. Response to the evaluation was strongly positive.

The prototype that was created by the conclusion of the project is judged to be functional and fulfil most of the requirements that were set. Due to the ambitious nature of the project, there is a large body of potential work to be addressed.

1. Introduction

The final year of the Software Engineering Degree at the University of Auckland requires the completion of a research and implementation project that lasts the entire academic year. This report details my project, which is centred on the development of a web application that provides support for co-operative music composition between its users. This application has been named 'Mixa'.

1.1. Project Aims

The aim of this project was to create a prototype web application (Mixa), which would enable musicians and other interested users to co-operate in the composition of music. This application would support the recording, mixing and playing of music. In addition, it would possess features that support the formation of online communities, and promote co-operation and sharing between their members. It is expected that this will improve the quality of music produced as well as providing an enjoyable experience composing and listening to music.

The key innovative feature of this application was to preserve the inner representation of music to store the songs in a way which can be modified. Storing an unflattened representation of the data enables this application to support editing or re-sequencing of a song. Another key innovation that this application would offer is the ability to record music from the standard audio input devices on a client machine within the browser itself. Recording would take place without requiring the user to download or manually install a platform-specific application or a browser-specific plug-in.

The music created in Mixa should be available in a variety of formats to support the ability for the music to be played a variety of portable devices which do not support codec installation.

Some of the more ambitious goals of this project, such as the creation of a collaborating community of musicians, were not achievable within the time frame that was afforded, but are instead suggestive of a long-term approach. One of the aims, therefore, was for the prototype application to demonstrate the potential of collaborative composition on a small scale.

Once this proof of the feasibility of Mixa was available, the application could be re-engineered to scale, extended and improved. For this reason it was also important that Mixa be implemented in a way that is extensible and supportive of change.

1.2. Overview

The rest of this report aims to provide a comprehensive understanding of what was accomplished during the project.

The following section will detail the background to this project, especially focusing on the motivations and work in related areas. Certain similar applications are highlighted and contrasted with Mixa.

The next section will detail the features which were implemented within Mixa. These features are further subdivided into user functions, administrative functions, and non-functional requirements.

The next section will deal with the details of implementation. This section will begin with an overview of the entire architecture, and then examine each interesting component in turn. A broad level of detail will be aimed for, and this section should be considered introductory to, and not replacement for, proper documentation to the application.

The next section will concern itself with a discussion about evaluation, and certain limitations of this project will be examined. This section also contains the summary of a usability evaluation which was conducted on one of the modules of Mixa.

The final two sections of this report are the conclusion and the future work.

2. Background and related work

Since the invention of the World Wide Web in 1989, user adoption has been swift. Currently, there are an estimated 1.5 bil-

lion users on the Internet, and this number is growing daily. By 2012, it is expected that 1.9 billion people, or 30% of the world's population, will have access to the World Wide Web.[1]

2.1. Rich Internet applications

Since its inception, the technologies underlying the World Wide Web have been improving. One trend that can be observed is the increase in the interactivity of Web pages.

JavaScript, also called ECMAScript, enables client-side interactivity over the World Wide Web. Web page developers are now able to easily write mobile code that runs on the client machine. An advantage of this code running locally is that there is increased liveness as the interface does not have spend time in communication with the server. The improved response times afford the creation of more interactive interfaces. Java applets, and true rich media plug-ins such as Adobe®Flash and Microsoft®Silverlight are now popular platforms for creating rich interactive user interfaces.

The web as a platform has a number of key characteristics that make it attractive for developing applications on. Web applications are distributed, easy-to-access, easily separable/loosely coupled, platform independent, and installation-free.[2]

The growing multimedia capability on the browser was accompanied by another trend developing in server-side technologies. Servers had already grown more dynamic in their ability to handle user requests and respond to them. Instead of delivering a static web page from a repository, web servers began to service HTTP requests by generating web pages using CGI (Common Gateway Interface) scripts.

Using AJAX (Asynchronous Java and XML (eXtensible Markup Language)), developers were able to establish communication between the client machine and the server without the user having to refresh an entire page. This took web applications further away from the submit-response style of interaction that had existed before, and closer to web pages that behaved more like desktop applications. [2]

The integration of media delivery with asynchronous communication protocols has lead to the development of the rich Internet application. [3]

It is predicted that soon Web interaction will be on par with desktop applications.

2.2. User-created content

Alongside the development of the Rich Internet Application, the web had experienced a large paradigm shift in the underlying content creation model. The focus began to move away from provider-centred data, and towards a more user-centric model.

Another hallmark in what is now called the Web 2.0 revolution was the shift in the source of content, from developer to user. There was an explosion of user-created content caused by open authoring and revising which eroded the distinction between authors and readers.[4] Users have a far richer experience when they collaborate than when they react passively in isolation, as the generation, evaluation and validation of ideas is self-reinforcing.

User-created content is now a large driver of Web usage. Users are more sensitive to their own requirements than the developers. When the user's needs did not coincide with the format of the data provided by the developer, the loose coupling between service providers and the open network facilitates users manipulating information themselves to improve it.

Once information has been created, there are three identified ways that Web 2.0 supports the management of information. These three ways are aggregation, projection and cross-production, and together they can be called the Web 2.0 information management model. Web content combined is more than the sum of its parts. [5]

2.3. Music

The commercial music that is readily available today is generally produced to be consumed. The format in which it is produced is inflexible. The listeners of this music do not interact with it creatively. The inflexible format does not encourage the creative efforts of the listeners.

After listening to a song track an a Web site today, a user may be prompted to review, rate or tag the musical piece. The user may even be able to email the artists, and get in touch with them through a social networking site. However, the music itself is not easily available for manipulation. The music is not provided in a format that readily allows for modification without special knowledge and technology. The user in unable to explore the music.

In contrast, this project relies on the conception of malleable music: music which is provided in a format which is readily manipulated. Such music would let listeners transcend their roles as passive listeners and encourage them to creatively interact with the pieces they listened to. [6] Such music would enable listeners to study the it, to modify it to suit their own personal tastes, and would let them reuse components of the compositions in their own music.

Music composition is a creative activity that could benefit from a collaborative approach, but there is no popular collaborative composing tool with widespread adoption. The perceived lack of collaborative tools for music composition is what motivated this project.

2.4. Related work

There have been a number of similar applications that have attempted to approach the same problem domain as Mixa.

2.4.1. The Audiotool by Hobnox

The Audiotool is designed to enable users to create their own electronic music within their own browser.[7].

The Audiotool is built in Flash, and has a slick and attractive interface. Learning to use the Audiotool is easy, and creating music with it is entertaining. The ability to perform sound synthesis within Flash is extremely impressive.

However, the Audiotool does not allow for complete creative freedom in music generation. It is limited to only one genre of music and there is no way to record new sound clips and import them into projects.

There is no support for collaborative composing, or for sharing partially completed projects, although fully rendered sound files can be saved to the user's on-line directory.

While the Audiotool has is similar to Mixa, its intended purpose is quite different.

2.5. Myna by Aviary

Aviary is a company that aims to make creation of digital content accessible to artists of all genres.[8] Their website intends to provide a suite of tools on-line through the browser that will achieve this aim.

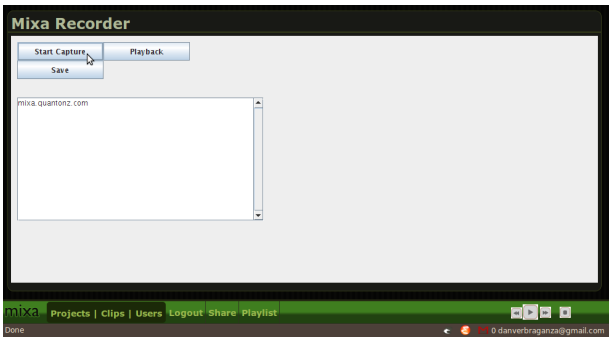

Figure 1: The recording module

Figure 2: The composer module in use

Although the Myna sound editing tool is still in its pre-alpha phase, the other tools from Aviary are extremely well designed to a high standard. The Aviary user interfaces were built in Flash, and are all very attractive.

Without knowing the specifications of Myna, it is impossible to tell exactly how similar Myna is to Mixa. It is likely that Myna too will support recording of music from the client's hardware, when it is completed.

The key difference between Myna and Mixa will be the collaborative support built into Mixa.

2.6. F@ust Music On-line

FMOL is a system for real-time music collaboration on the Web, that allows several composers to work collectively on a single piece using a common interface.[9]

FMOL was an advanced on-line music creation system that was developed in the Pompeu Fabra University in Spain. Just like the other two applications, it was a thin client that ran in a browser with a plug-in. Users composed music by selecting fom an array of switches. FMOL was an academic project that had a lot of work go into it, and it had some extremely attractive features.

One such feature was user profiling, where it would attempt to match users with certain composing styles with co-composers they would like. The initial version of FMOL did not have a full-duplex 'jam' capability, but it was eventually added.

It is exciting to see that FMOL was successful, and the success of FMOL validates some of the ideas from Mixa.

One key area where FMOL differs from Mixa is the lack of recording in FMOL. Also, the FMOL was a self-contained project, and did not support the interchange of information with other applications.

3. Features

3.1. User functionality

The primary driver of this project was user functionality. The features described below were prioritized and chosen for implementation by the way in which they supported the aims of the project.

3.1.1. Recording

Mixa allows users to directly capture audio from the recording devices they have available on their client machine. Recorded clips are uploaded to the server machine, where they are stored persistently.

There is currently no support for deleting uploaded clips as this would have cascade effects on the musical projects the clips were a part of.

3.1.2. Composing

Mixa provides a basic song sequencing interface where songs can be composed from a list of music clips. The composer module (see figure 2) is the interface where users can customise existing songs and create new ones.

The composer provides users with a view of the 'Library', (the lower-right hand panel in figure 2) where they can browse a selection of all clips in the database. At present this has been implemented by a list, which is sufficient as long as the number of clips remains below 30. Directly above the library is the 'Palette', which shows all those clips that have currently been downloaded locally for use in the song. Clips can be added to the 'Canvas' by dragging and dropping. The canvas represents a song stored in something like piano-roll notation.

The composer module is expected to be the region where most of the user's work would take place. For this reason it was a requirement that this module be attractive, intuitive and efficient.

3.1.3. Version control

Mixa allows users to modify the songs that others have created. This means that a form of version control is required, so that users are not inconvenienced by losing their music. The form of version control implemented in Mixa relies on duplicating the resource when it is accessed by another user. This allows the application to ensure that both the original creator and the new editor have full freedom to interact with the music.

Collaboration is encouraged by informing the original artist that her work has been duplicated by someone else. To load all the changes the other composer has made, all she needs to do is load the new version, at which point she will end up with a duplicated copy that contains her original work in addition to the changes that were made by the other composer.

3.2. Rendered playback

Mixa stores the songs users create in a relational database, and represents them with a specialised format. This format is appropriate for communicating with the composer application, but is not supported by external applications or hardware devices. For this reason Mixa supports the rendering of song versions to widely supported fast-playback formats.

Rendering to a high-compression format is also ideal for

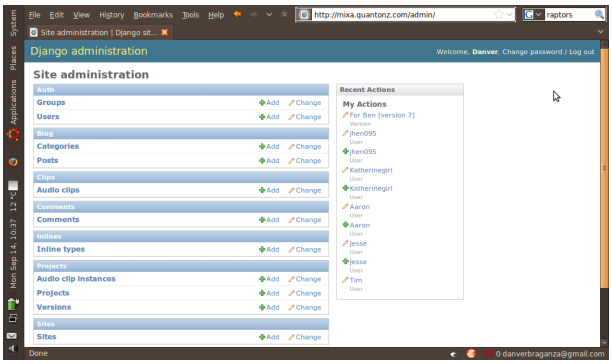

Figure 3: The Django admin panel

supporting streaming audio. For this purpose, lossy audio is considered appropriate. In contrast, when composing full quality audio is delivered, as the quality might impact the composition process negatively.

3.3. Sharing

Mixa makes it easy to announce details about a song project on any social bookmarking site that features sharing of URLs. It achieves this by assigning each instance within its data model a unique permanent URL which then can be used by anyone, anywhere, to refer to that object. To facilitate the easy sharing of resources, Mixa features a set of links available on every page which automatically submit the current resource to one out of a selection of social bookmarking sites when a user requests it.

Mixa also provides the support for Web feeds, to enable other users to register for updates from a single source. Web feeds are a simple pull-type notification system whereby an application known as an aggregator is configured to periodically check a given URL for updates published in a particular format.

3.4. Administrative features

By leveraging the features provided by the component systems of the application, it was possible to provide several administrative features without having to develop them. This was especially beneficial as the requirements of the administrative interface to Mixa was the least specified. The Django admin panel (see figure 3)is an example of a administrative feature which came with a component system.

When the system needed to perform tasks that were outside the capabilities of the included admin interfaces, administrators had to resort to writing and running script files, and editing configuration files by hand. This procedure would be prone to error, and for this reason an administrative control panel just for Mixa would be a nice addition.

3.5. Non-functional

Due to the project's limited scope, most non-functional requirements of the application were extremely relaxed. Requirements such as security, scalability and performance were not considered for this prototype.

3.5.1. Modifiability

Mixa was developed using agile development practices. This required frequent changes of the code during iterations. In addition, the ultimate quality of Mixa is likely to be limited by the

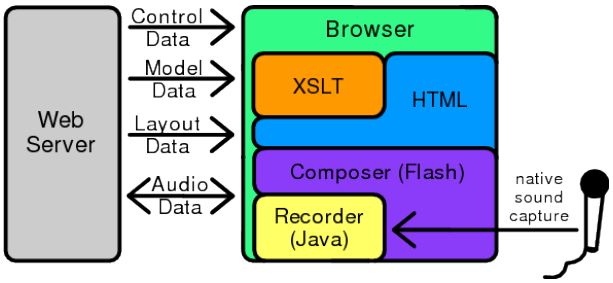

Figure 4: Architecture of Mixa–Focus: Client

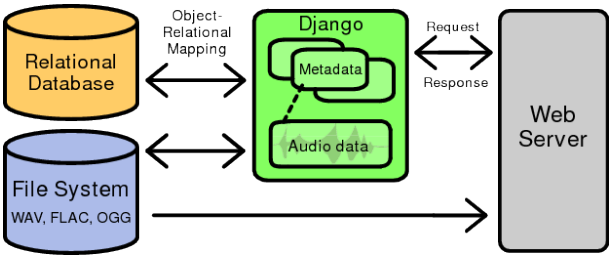

Figure 5: Architecture of Mixa–Focus: Server

time constraints of this project. For these reasons, the modifiability of the project is extremely important.

Although no formal procedures were put into place to ensure the extensibility of the application, aggressive refactorisation, good coding practices in general and steady adherence to the DRY (Don't Repeat Yourself) Principle ensured that the system has a high degree of modifiability.

3.5.2. Buildability

Buildability was a large concern during the implementation of this project. It was a criterion that any designs generated were evaluated against quite stringently. It is very important that care is taken to work within the limitations of this project limitations. Any effort spent implementing a feature which eventually turns out to have a low buildability is as good as wasted.

4. Architecture and Implementation Details

As Mixa is a Web application, the implementation is subject to a several constraints and requires the integration of a number of technologies

The overall architecture of Mixa follows the standard client–server pattern of a Web Application. Mixa is deployed using a three-tier architecture. Presentation is handled by the browser running on the client machine, which communicates to the main Web server via HTTP requests (see figure 4). Presentation was achieved by a combination of HTML 5, JavaScript, Java and Flash. The business logic is handled on the main server in the middle tier. The last tier is used for persistence, which includes a relational database management system and the file system of the computer (see figure 5).

Due to budget constraints, the persistence layer and the business logic were housed on the same machine. This was beneficial when considering testability and ease of deployment.

4.1. Client

4.1.1. Flash

The interface to Mixa's composer module was implemented in Flash due to its superior graphics capability and user-friendliness. The Flex framework was used to support the development of this module. Flex enables the specification of a user interface in a declarative language based on XML. The interface is then scripted using JavaScript to add interactivity and business logic.

This is a much more attractive proposition in terms of buildability than the alternative: building the module in Java using the Swing windowing toolkit.

4.1.2. Java Applet

Mixa is designed to gain access to the recording devices on the client machine using a signed Java applet with sufficient permission. The reason an applet is needed is because browsers do not yet support native audio capture, and the restrictive sandbox within which Flash programs run does not allow for the user to access the raw audio data.

After recording, the applet uploads the audio to the server via a HTTP POST request.

4.1.3. Display

The main display was rendered using HTML 5. HTML 5 is the proposed next standard of the Internet, which is already being supported by some browsers. It has support for an array of new semantic tags and a looser more relaxed syntax which enables it to be more likely to be validated

Most data delivered from the server was sent across in eXtensible Markup Language (XML) format. This was the unformatted data representing the resource the client requested. The browser used eXtensible Stylesheet Language Transformations (XSLT) to process the data and convert it into HTML 5.

One of the advantages of this arrangement is the decoupling between the client and the server. Because the server delivers a standard unformatted data representation in response to a request, the behaviour of the client is more customisable and the behaviour of the server supports other clients.

4.1.4. JQuery

JQuery is a JavaScript library which implements commonly-used functions to enable the rapid development of rich Internet applications. [10] Of special importance for this project was the ability to easily code animating divisions, which JQuery simplifies. The inbuilt support for Asynchronous browser requests also was handy for rendering and displaying XML quickly.

4.2. Web Server and Scripting

Apache Web Server was chosen as the Web server because of familiarity with the system, and because it was known to work with Django. Another point in its favour is the fact that it is free, and well supported through online forums.

Python was the language of choice in which to implement the back-end. Python is a high-level, dynamically typed object-oriented language with support for some functional programming constructs.[11]. The main reason for this is because it was felt it would be good to learn Python before graduation, and learning is achieved best by doing.

Another reason for Python was that it allows for rapid, agile development. As an object-oriented language with dynamic typing Python supports the rapid mock-up of applications. The quality and amount of auxiliary tools for developing a web application with Python were very appealing too.

Django is a web framework implemented in Python that encourages rapid development and clean design. One of the core tenets of the Django framework is that it is delivered with 'Batteries Included.' This means that the most often-used features that designers of web applications want are included out of the box. [12]

This included features such as the administration panel, user authentication and session management. Django also provides support for template-based generation of web pages. to ensure that separation of content from presentation is consistently upheld.

Django provides an object-relational mapping to simplify the database accesses on the server. This allows the persistence layer to be largely transparent during design and development. The data model could be specified solely in object-oriented Python code, and the Data Access Layer would be generated automatically. This hugely freed up development time, and reduced the potential for error.

Another key feature of the Django framework is the decomposition of a web service into several loosely-coupled 'apps'. An Django app is composed of a set of models, a set of views which represent and manipulate those models, and a binding between URLs and views. This pattern of designing web applications is loosely based on the model-view-controller design pattern, and it greatly increase the understandability of the system.

Following the Django style, Mixa was decomposed into the following apps: main, users, projects, tagging. The *main* app was concerned with delivering to the client the JavaScript, XSLT and HTML pages required for the effective rendering of the content, which was supplied via the other apps. *Users* is a placeholder app which was installed in case any special user-centric data needed to be stored that did not belong in any other app. *Projects* is the most important app for Mixa, and contains within it all the models needed to support composition and version control. *Tagging* is an app which was installed from Google Code. It provides support for organising and retrieving models in other apps according to the tags which users associate with them.

A virtual host was set up to serve media files from a separate address to the main application URLs. Media files are files such as images, audio and Cascading Style Sheets (CSS), which do not need to go through Django to be delivered. Separating the two is good practice for reasons of security. Serving media files from the same name host as the applications introduces the possibility of name collisions between the URL of media files and application URLs.

PyAMF was used to implement the handling of Action Message Format(AMF) messages on the server. The Adobe®Flash Player uses AMF to communicate between a client and a remote server. The desirable feature of AMF is that it enables remote procedural calls to be invoked over a HTTP connection. PyAMF was used because it sped up the development of a communication channel between the server and Flash. The alternative solution would be to implement the communication channel by exchanging messages in an XML format. This solution would have the advantage of being consistent with the rest of the interface, but would take effort to implement whereas using AMF is a much simpler solution.

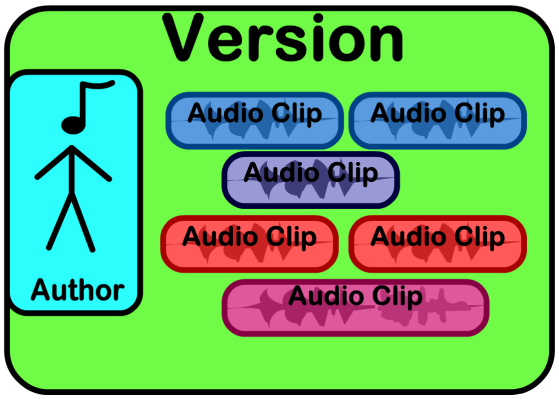

Figure 6: Detail of the version control model in Mixa

4.3. Data storage and representation

Django's Object-Relational Mapping was used to interface with an instance of MySQL server running on the same machine. All of the data corresponding to Mixa was stored in the database in this way, except for audio data. The audio data, in the form of binary files, was stored in the file system instead. Some challenges faced with the arrangement of the Object Relational Mapping were caused by the database becoming inconsistent with the model whenever the model changed.

4.3.1. Projects and Versions

The fundamental compositional unit in Mixa is the song *Version* (see figure 6). A Version consists of a collection of audio clips and associated start times, and represents a single snapshot of a song. At present, Versions are only allowed to have one composer.

Upon the receipt of a request to edit a Version of a song, a user is provided with their own copy of it. This ensures that both the original creator and the new composer have complete freedom of access to their own Version. To incorporate the changes made by another user, the original creator has to request the edited song version. The changes are then available in a new copy.

For ease of browsing, all the Versions of a song are grouped together into a Project, and are arranged into a natural tree structure, with derived Versions being assigned as child nodes to the Versions they derived from. In figure 7 such a tree structure within a Project can be seen.

A limitation of this version control system is that it does not support simultaneous collaboration.

4.3.2. Audio data

It was decided to store audio data on the disc because large binary objects (BLObs) are not particularly useful in databases. The essential operations which databases can accomplish are meaningless with this sort of data. On the other hand, having access to the files meant that they were available without having to go through a database. This was an important feature when it came to writing scripts to convert audio data from different bit-rates and formats.

For consistency, audio quality in Mixa has been set to be 16-bit, stereo, at 44100kHz. There are currently three different

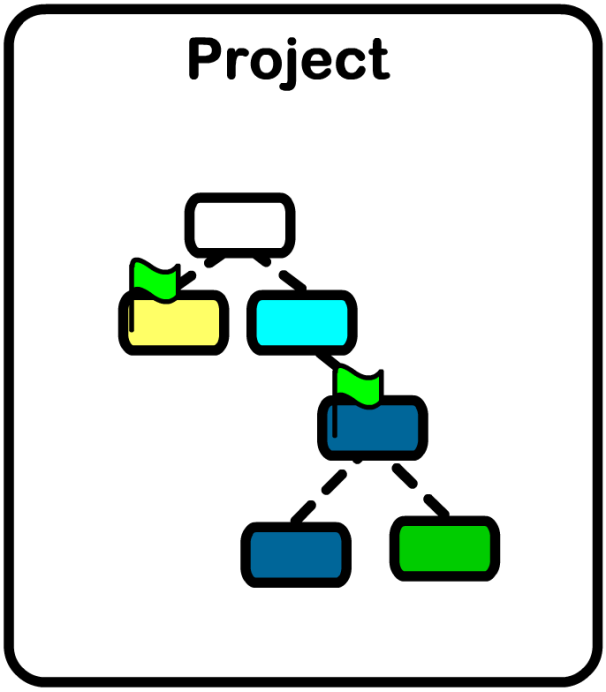

Figure 7: A tree of versions stored within a project

audio formats that Mixa makes use of.

WAV is the name given to uncompressed raw audio data. It is of full quality, but it also has the drawback of being extremely large.

The Free Lossless Audio Codec (FLAC) is high-quality audio format that is reasonably compressed. In addition, audio converted to FLAC remains at full quality.

OGG Vorbis is a lossy compression system which achieves smaller files than FLAC, at the expense of some sound quality. However, studies have shown that in practice, Vorbis is perceptibly different from lossless audio to only a minority of people [13].

The initial design called for the system to use only FLAC and Vorbis. However, due to time constraints the use of WAV files were not completely eradicated. WAV is used in uploads of recorded clips from the client machine, and in audio storage and processing on the server. In fact, currently audio is stored in all three formats on the server. This is a liability because the size of WAV files. An ideal solution would use a Java FLAC library to compress the audio within the recording applet before uploading recordings, store the audio as FLAC, and convert to OGG intelligently on demand.

The reason a more popular format such as MP3 was not used was because of the licensing issues involved with applying MP3. The MP3 format is patented by the Moving Picture Experts Group, and access to encoders is restricted. Using free formats such as Vorbis and FLAC sidesteps this restriction entirely.

5. Evaluation

5.1. Comprehensive evaluation

A comprehensive evaluation of the system would involve a live user base and a trial lasting a long period of time. This would be

the best way to gauge whether the design of the web application is effective, and if the quality of music composition is being increased.

Due to the limitations of this project, such an evaluation was not feasible. It needs to be addressed in future work.

5.2. Limitations

At the conclusion of the project, there are a number of limitations and bugs that still exist.

The application works correctly only within Firefox or Safari. On Internet Explorer 8, the application works so poorly as to be deemed incompatible. This behaviour is understandable when the major compatibility errors between browsers is considered. However, this conflicts with the motivations of the project which state that a web application is browser independent.

Another limitation of this project is that the clean and open API intended to encourage other developers to utilise Mixa in their mash-ups has not yet been developed. The Mixa API has grown in an ad-hoc fashion as deadlines approached, and while it is mostly consistent, it is possibly not complete.

5.3. User study

An informal user study motivated by the need for a demonstration of this project's feasibility was performed. Another motivating reason for this study was preparation for the final year project exhibition, when live crowds will interact with the project. The study was performed on the largest client-side module, the composer.

Improvements suggested by this study will be implemented in the final exhibition, if they are sufficiently easy. This study also highlighted those features which users found most attractive. The study provided the knowledge needed to tune existing features and highlight enjoyable ones. In this way the study has helped to improve the overall quality of Mixa's exhibition.

Results of the study were generally positive. Every subject found composing to be enjoyable, and reacted positively to the tasks set. The interface was considered to be quite intuitive and easy-to-grasp.

Problems identified were subjects trying to specify actions which the module did not support, such as dragging out of window to scroll and double-clicking to seek to. Some subjects also found the interface to be inconsistent. The invisibility of system state in a few situations was identified as a problem, with the subject having no idea what was going on.

For more information, please see appendix I.

6. Conclusion

The Mixa Web application represents a successful implementation of a prototype collaborative composition environment. Contending against technical challenges, it integrates many different technologies together to deliver the functionality required of it.

A limited study Collaborative composition has been shown to be enjoyable.

More work needs to be done to demonstrate that Mixa actually improves the quality of music composed collaboratively.

There is great potential for future work left in this project.

6.1. Future work

At the conclusion of this project, there are a number of different ways in which Mixa can continue to be extended.

The evaluation performed on this project was limited. A more comprehensive evaluation would illuminate the status of the Mixa application and would be instrumental in determining how well Mixa supports the aims it espoused.

There is a lot of scope for major improvements to be made to the user interface. These improvement would focus in two major areas: usability and performance. Site design was kept minimalistic to facilitate easy development, and a number of components would benefit from better design to make them more intuitive, efficient and effective. Performance improvements are also possible within the applet and composer.

Scalability is a challenge that is extremely likely to arise if this web application experiences significant growth. The initial prototype implementation was not expected to have a high degree of scalability. Scalability is an important feature motivated by the fact that resources at this point are not sufficient to build the system as big as eventual demand might require. Scalability is a need which remains to be addressed in future work.

Security is another area where the service could be improved. One especial limitation of the application is its weakness to denial of service attacks by spamming high-cost operations such as rendering.

The usefulness of this web application could be improved by adapting it to work with other music composition applications. This is a natural next step which will allow users who are already familiar with their favourite composing software to use Mixa easily.

7. Acknowledgments

I take this opportunity to acknowledge Joseph Maxwell Hobbs for the work he has put into making this project a success. The images used in figures 4 and 5 in this report were designed in collaboration with Joseph.

I also acknowledge Professor Gill Dobbie of the Department of Computer Science at the University of Auckland for her extensive support and guidance during this project. Thanks are due to Associate Professor John Morris, also of the Department of Computer Science at the University of Auckland.

Further thanks are due to John Trengrove, for introducing me to Django, and to Chris Haden for the use of his server.

8. References

[1] I. D. Corporation, "Idc finds more of the world's population connecting to the internet in new ways and embracing web 2.0 activities," June 2008. [Online]. Available: http://www.idc.com/getdoc.jsp? containerId=prUS21303808

[2] T. V. Raman, "Toward 2w, beyond web 2.0," *Commun. ACM*, vol. 52, no. 2, pp. 52–59, 2009.

[3] R. Reinhardt, "Building communities with rich internet applications," in *SIGGRAPH '03: ACM SIGGRAPH 2003 Web Graphics*. New York, NY, USA: ACM, 2003, pp. 1–1.

[4] D. E. Millard and M. Ross, "Web 2.0: hypertext by any other name?" in *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*. New York, NY, USA: ACM, 2006, pp. 27–30.

[5] C. Ullrich, K. Borau, H. Luo, X. Tan, L. Shen, and R. Shen, "Why web 2.0 is good for learning and for research: principles and prototypes," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 705–714. [Online]. Available: http://dx.doi.org/10.1145/1367497.1367593

[6] A. Tanaka, "Malleable mobile music," in *Adjunct Proc. of Ubicomp*, 2004.

[7] "Hobnox - audiotool," http://www.hobnox.com/ audiotool.1046.en.html, Hobnox, May 2009.

[8] "Aviary - phoenix," http://aviary.com/tools/phoenix, Aviary, May 2009.

[9] S. Jord and O. Wst, "A system for collaborative music composition over the web."

[10] J. Chaffer and K. Swedberg, "Learning jquery: better interaction design and web development with simple javascript techniques," 2007.

[11] G. Lindstrom, "Programming with python," *IT Professional*, vol. 7, no. 5, pp. 10–16, Sept.-Oct. 2005.

[12] D. Documentation, "Django: Python Web Framework,[Online], Django Software Foundation," 2005.

[13] J. Moffitt, "Ogg vorbis—open, free audio—set your media free," *Linux J.*, p. 9.

A. Plan for a Usability Study of the Composer in Mixa

A.1. Motivation

An evaluation of Mixa is important for completeness. Performing such an evaluation will provide a better understanding of the project's characteristics, and inform presentations as well as providing direction for future work.

The research project also involves an exhibition where the application designed is demonstrated to the public. Improvements suggested by this study will be implemented in the final demo, if they are sufficiently easy. This study will also highlight those features which are most attractive, which will then lead to a more enjoyable presentation. In this way the study will help improve the overall quality of the exhibition by providing the knowledge needed to tune existing features and highlight enjoyable ones.

A.2. Aims

The aim of this study is to serve as a form of evaluation of the project with respect to the enjoyability of the experience of a casual user. The aim is also to elicit feedback about implemented features and potential improvements.

A.3. User profile

The profile of subjects chosen for this study were restricted by two constraints. On the one hand it was desirable to have test subjects who are as similar to those expected on the exhibition day. On the other, the potential pool of subjects within easy access was limited.

Therefore, the profile of the user adopted by this study called for a user in their late teens or early twenties. This subject would be familiar with music concepts and familiar with computers. Users were selected of varying levels of musical aptitude, to better match the exhibition day crowds.

A.4. Methodology

The methodology adopted was a guided path through the tasks of the study. These tasks provide a way to introduce the subject to features of the system, and are not timed for efficiency. *Perceived* hardness of tasks is of greater importance to this study than actual time spent.

A.4.1. Script

1. **Introduction** Greet the subject. Welcome them to the test environment. Tell them our names. Make them feel welcome.
2. **Description of the system** Introduce the system to the subject, and tell them about its motivation, aims and features.
3. **Test paperwork** Request the subject sign the waiver/questionnaire.
4. **Tasks** Proceed through the tasks at a pace the subject is comfortable with. At this point observations of the subject behaviour will be noted.
5. **Interview**

A.4.2. Test environment

Due to the loud nature of the tasks that involve the composition of music, the tests will need to be completed in a room with restricted access and sound controls. This introduces a limitation as the project exhibition environment is likely to be noisy, which will have a significant impact on the usability of the system. Attempting to run a test in a noisy environment, however, is not feasible at this point.

Sufficient lighting will be made available to ensure the subject can see the screen, and they will be comfortably seated.

A.4.3. User tasks

The tasks were presented as a series of suggestion to the subject to provide them with a path through the features of the program.

1. **Log in**
2. **Browse projects**
3. **Branch a version**
4. **Edit a version**
5. **Delete a clip**
6. **Add a new clip**
7. **Duplicate a clip**
8. **Edit the version until satisfied**
9. **Save the version**

A.4.4. Questionnaire

After the subject has completed all the tasks and is finished playing with the system, the user was presented with the questionnaire available on-line at `http://spreadsheets.google.com/viewform?formkey=dHo3bnlUWDliVmVPTGRDN1dqR0VPMEE6MA..`

A.4.5. Results

Results of the study were generally positive. Every subject found composing to be enjoyable, and reacted positively to the tasks set. The interface was considered to be quite intuitive and easy-to-grasp.

Problems identified were subjects trying to specify actions which the module did not support, such as dragging out of window to scroll and double-clicking to seek to a location. Some subjects also found the interface to be inconsistent in places. The invisibility of system state in a few situations was identified as a problem, with the subject having no idea what was going on. One example was when the entire module was loading, there was a complete lack of feedback to the user, leading to most of them wondering if it had crashed.

Common suggestions for improvements included previewing sounds before adding to project, allowing for negative selections of clips, and displaying some form of timing indication.

A.4.6. Conclusion

The major trend of results is positive. This is an extremely promising result, and does show that the composer is an effective prototype.

However, users were able to also identify those features which were counter-intuitive. A number of minor improvements can be made within the system.