Radio Catchup

An interactive segment-based radio listen again service

Christian Couch

May 2017

This dissertation is submitted for the degree of
Computer Science with Business Management (Hons)

Supervisor: **Dr. F. Nemetz**

Department of Computer Science
University of Bath

Radio Catchup: An interactive segment-based radio listen again service

Submitted by: Christian Couch

Copyright

Declaration

Abstract

The radio landscape has changed drastically over the past decade. With the availability of the Internet, broadcasters have introduced listen again radio services for users to catch up on programmes that they missed. However, these services often contain limited contextual information, which makes it difficult for users to locate and navigate to their desired content.

This dissertation presents a method to automatically segment radio programmes and deliver the contextual parts of a programme to the listeners using an interactive segmented web player. The results of the empirical evaluation suggest users prefer this segmented player over a traditional web player and that locating content is significantly faster with this player.

The work presented here has laid the foundations for many exciting opportunities. Individual content within shows could be searchable, listener personalisation could be possible, and listen again services could be vastly more user friendly in the future.

To experience the web player, it is available online: http://radio.chris.io.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to thank my supervisor, Fabio Nemetz, whose enthusiasm, advice and expertise has been so valuable throughout this dissertation. I would also like to thank my committee at University Radio Bath who tested the system, provided their shows for the user experiment and importantly kept me sane throughout this project. I must also thank the various people in the radio industry who I have spoken to and for their suggestions and recommendations.

Finally, I could not have done this dissertation without the support of my family and friends, I am extremely grateful. Special mention must be given to my parents, their partners and Jessica Craig who have given their time generously to proofread and provide feedback.

1 Introduction

"This is 2LO calling, the London station of the British Broadcasting Company calling. This is 2LO calling"

The first official broadcast from the BBC by Arthur Burrows on 14th November 1922 [1].

1.1 Background

Radio broadcasting has been around for over a century. Originally called 'radio-telephony', the first broadcast is claimed to have been performed by Reginald Fessenden in December 1906 in Massachusetts [2, 3]. Starting off small, Fessenden 'increased his range from this distance of about twelve miles to... about five hundred miles' [4]. The U.K. did not start broadcasting until 1920 when the Post Office licensed a few large companies. A couple of years later a Parliamentary Committee decided to regulate the industry, leading to the formation of the British Broadcasting Corporation (BBC) [5].

Since then radio broadcasting has developed immensely, first moving from *Amplitude Modulation* (AM) to *Frequency Modulation* (FM). This transition gave rise to better sound quality and the ability for digital data to be encoded and transmitted, through RDS[1]. In more recent years, *Digital Audio Broadcasting* (DAB) has developed this further with increased radio reception, further improved sound quality and even more supplementary information being provided to the listener [6].

Nowadays the Internet is a big platform for radio. Internet Protocol has allowed for a much larger transmission stream, consequently stations can benefit from providing as much supplementary information as they wish. Additionally, not being confined to a traditional 'one-to-many' transmission, the Internet has provided more information along with personalisation and unique content for each listener. All of these developments have occurred to improve the whole listening experience. However, although Internet radio distribution does not suffer from data limitations, the constraints of traditional broadcasting means that there is often a lack of supplementary information being provided.

[1]Radio Data System - system used throughout Europe to send station identification, now playing and other supplementary information.

1.2 Motivation and Problem Statement

Among the benefits of Internet radio is the ability to deliver on demand content in the form of listen again services. These services provide users the ability to listen back to a show that has already been broadcast, however this dissertation will address the lack of information available in listen again services and specifically how it can affect the listener experience.

Consider a listener who missed their favourite radio programme. They can use the listen again service to catch up. However, suppose the show is three hours[2] long? The listener does not have time to catch up on the whole programme, so they are likely to jump to parts of interest, for example an interview with a guest or a particular discussion topic. Current listen again services present users with a single audio progress bar that indicates which part of the programme they are listening to, rather than the content they are listening to. Listeners wanting to jump to certain content must manually search, often waiting for the content to buffer before realising that they are not at the correct point. This is extremely inefficient and degrades the user experience.

1.3 Research Questions

To help analyse this problem and to provide direction for this dissertation, the following research questions have been considered:

1. Is it possible to automatically segment a radio programme?
2. Is it possible to develop a segmented web player that is easy and intuitive to use?
3. Is it possible to develop a segmented web player that is quick for listeners to locate content within a radio programme?

1.4 Aims

This dissertation aims to develop an interactive segment-based radio listen again service, which provides users with the ability to view and locate specific content within a radio programme. One potential solution is for radio station production teams to manually tag content and timestamps as a show is broadcast, however this would be labour intensive and potentially very costly. For this reason, this dissertation will be searching for an automated solution.

If this dissertation is successful in segmenting radio programmes and presenting them to the audience, the potential is enormous. This dissertation aims to lay the foundations that would allow individual content within shows to be searchable, listener personalisation to be possible, and listen again services to be vastly more user friendly. Although not all of these features will be in scope for the dissertation, they are only possible once a radio programme has been segmented.

[2]Based on the average weekday daytime BBC Radio 1 show (http://www.bbc.co.uk/radio1/programmes/schedules)

1.5 Objectives

This dissertation will be considered a success if the following objects are achieved:

- Research the current academic literature concerning audio content analysis and audio retrieval.
- Investigate the current technologies available and review similar existing solutions.
- Devise a requirements specification from a wide breadth of sources.
- Design a system that is capable of segmenting a radio programme and displaying the segments against the audio within a web player.
- Following the design, develop an automated interactive segment-based radio listen again service.
- Evaluate the success of the developed system by conducting a user study and gathering user feedback.

1.6 Dissertation Structure

This dissertation is divided into 7 chapters.

Chapter 2	**Literature and Technology Review** - This chapter explores the state of the art, focussing on audio content analysis and retrieval methods. Initially discussing audio content analysis, the chapter considers different approaches to segmentation, comparing analysing audio elements with studio events. The second part of the chapter concentrates on retrieval methods and explores search techniques and audio players.
Chapter 3	**Requirements Elicitation and Analysis** - This chapter elicits requirements from a wide breadth of sources with existing solutions analysed, user research undertaken and use cases derived.
Chapter 4	**Design** - This chapter discusses the development methodology, reviews the design decisions taken at various levels and presents an appropriate system design. The design is presented through a system architecture, data schemas and user interface wireframes.
Chapter 5	**Implementation** - To build upon the previously defined system architecture, this chapter discusses the implementation of the system and presents a working solution.
Chapter 6	**Evaluation** - This chapter evaluates the solution presented using two methods: requirements evaluation and user experiment.

These methods are described in detail and hypotheses of expected outcomes are defined, which later in the chapter are proved significant. Full statistical analysis and results from the evaluations are also outlined.

Chapter 7 **Conclusion** - This chapter highlights the achievements and contributions this dissertation has made, namely developing an automated system that segments radio programmes and presents them to the audience through a web player. Limitations of the study are also discussed in detail and recommendations are made for possible future work.

2 Literature and Technology Review

This chapter covers the academic research undertaken and what methods and solutions exist for dissecting a radio programme. The project naturally splits into two sections: audio content analysis and retrieval methods. Thus the following review will research the two areas respectively.

Since Fessenden made his first broadcast [3], radio broadcasting has been thoroughly researched, from early day technical papers on how radio works [7], to modern day analysis of psychological impact on listeners [8]. The focus of this project is the radio broadcasting domain, however it is important to recognise that other industries are developing tools for audio content analysis and retrieval methods. This literature review will therefore make use of these other domains in order to provide an insight into how these could be relevant in the field of radio.

2.1 Introduction

Before addressing the challenges of audio content analysis and retrieval, it is important to understand its relevance in the domain of choice for this project, radio broadcasting. The BBC, the first radio broadcaster in the United Kingdom, has been broadcasting radio since 1922 [1]. In the early years radio programmes were not recorded, mostly because recording equipment was rare, cumbersome and broadcasters had a prejudice against recording shows, as it was thought reusing the recording, i.e. broadcasting a pre-recorded show, “was not real” [9]. However, by the 1960s it became common practice for broadcasts to be recorded [10] which resulted in a vast archive of broadcasts. Unfortunately, these broadcasts have been largely forgotten, with little thought to metadata that would aid content searching and retrieval. Raimond and Lowis [11, pg. 1] found that “creating this metadata is a time- and resource-expensive process; a detailed analysis of a 30 min program can take a professional archivist 8 to 9 hr”. It is clear to see that this detailed manual processing could never be applied across the complete archive. Instead a number of researchers [11, 12, 13, 14, 15] have developed automatic unsupervised approaches to identifying these semantic tags.

2.1.1 Listen Again Services

Today, the availability of the Internet and the ability to deliver specific content to different listeners mean broadcasters are able to offer listen again services. The terms *listen on demand*, *catch up service* and *listen again service* are all interchangeable; they refer to a service which provides the audience with the ability to listen to a show that has already been broadcast[1]. Full show listen again is a listen again service employed by most large scale radio stations. It gives the listener the ability to replay a previously

[1]For consistency this project will only refer to this service as a listen again service.

broadcast show. In section 3.1.1 two solutions will be discussed: BBC iPlayer Radio[2] and Radioplayer[3].

Another form of listen again service is the podcast. While podcasts are not exclusive for content from broadcasted shows, they are commonly used as another catch up service. Due to licensing restrictions, they cannot contain music and therefore are often used as a speech 'best-bits' reel. Podcasts are currently manually edited and produced with production teams curating the content. The key difference between full show listen again and podcasts is that some semantic analysis, mostly human based, has been used to identify relevant and interesting audio clips for the podcast audience, whereas full show listen again acts to fully replicate the broadcast. However, it is evident that podcasts were originally used as a means for listener time-shifting (tuning in it any time) before full show listen again services were developed.

2.1.2 Listener Experience

Listeners listen to radio or podcasts for a variety of reasons. McClung and Johnson [16], studied podcasts as they was the only readily available listen again service at the time. They found that listeners use these services for "entertainment, timeshifting, library building, and social aspects" [16, pg. 93]. Consequently understanding what listeners are looking for is vital to building effective analysis and retrieval systems.

A recent study by Nencioni et al. [17] found that users prefer serialised content. This supports the findings of BBC iPlayer press packs which suggest that the most popular listen again requests are for serialised content [18]. This indicates that users want to catch up on content they know is out there, rather than look to discover anything new. Nencioni et al. also found that users prefer short duration content (<60 minutes), although this is not the case with entertainment or music based radio programmes which can be evidenced by the average BBC Radio 1 show being three hours long[4]. Nonetheless, this could explain the popularity of podcasts within entertainment and music based radio programmes for listen again services.

An important listener habit first identified in 1948, that has become increasingly relevant in recent years, is the social aspect of media [19]. Lasswell found that people enjoy discussing programmes in a social setting, which can often lead listeners who may have missed the initial broadcast to use listen again services to catchup.

Not only do listeners want to listen back to particular shows, but according to research carried out by Skov and Lykke [20], 48% of his respondents found it important to locate a part (or section) of a radio programme. To further analyse this, researchers [21] investigated how listeners currently locate this content within an audio file and the results revealed a very manual process (techniques to improve this process are discussed in section 2.3, as it is an important feature for listeners). In order to identify and retrieve

[2]BBC iPlayer Radio: Listen to live BBC radio online and on-demand radio catch up for programmes missed. http://www.bbc.co.uk/radio

[3]Radioplayer: Non-profit partnership between the BBC and commercial radio which aims to keep radio listening simple, through providing radio web technologies. http://www.radioplayer.co.uk

[4]BBC Radio 1 Schedule: (http://www.bbc.co.uk/radio1/programmes/schedules)

parts of a radio programme, it is important to understand how they are structured. Radio programmes often follow a set format, which is determined by producers and repeatedly used each week. If these can be defined, then automated processes can utilise the known formats and structure segments, ultimately resulting in a natural breakdown of a show for listeners.

Further to this research, the BBC piloted *Responsive Radio*[5], a tool that tailored a documentary to a user's desired listening length. Responsive Radio had categorised and prioritised the segments in the documentary. Depending on the length a user selected, the tool would customise the show they heard, choosing enough segments for the desired duration. While this was very innovative and a great feature, it still required timely manual curation of segments and priorities that meant the pilot was infeasible for public release. The experiment presented a new listen again experience, highlighting a different take on providing requested content to the audience.

Overall, it is clear to see that listen again services are important and valued by listeners. Each year RAJAR[6], the national body that measures radio audiences, release a 'Measurement of Internet Delivered Audio Services' report that analyses how and when content is consumed online. Comparison of the Autumn 2015 and Autumn 2016 reports [22, 23], reveals that full show listen again and podcast usage has increased 20% in the past year. These are encouraging results for the medium and emphasise the importance of these services for listeners.

2.2 Audio Content Analysis

Audio content analysis is not a new research area. Many studies and researchers have tried to find efficient methods for extracting meaningful and relevant data from an audio document. Lycke et al. [14], Raimond et al. [15] both studied accurate tagging methods of archive material within broadcasters and proposed solutions using speech recognition and semantic keyword extraction. Once audio had been transcribed, Lu and Hanjalic [13] likened these approaches to the methods used for text document analysis and categorisation. They proposed that findings from textual semantic extraction research should be exploited for audio based content too. However, often audio documents contain more information, such as temporal and structure data, that is not extractable with these methods. To analyse an audio document sufficiently, Larson and Köhler [24] argue that structure metadata (segments) is better utilised than content metadata (transcripts). This section addresses the problem of unsupervised detection of audio segments and researches current methods and solutions.

Audio segments are parts of an audio track that can be divided naturally. They should be thought of as scenes in a film, with each one defined by a beginning and end and having a semantic meaning. It is important to note that audio can be broken down

[5]BBC Responsive Radio: http://www.bbc.co.uk/taster/projects/responsive-radio

[6]RAJAR: Radio Join Audience Research is the official body in charge of measuring radio audiences in the UK. It is jointly owned by the BBC and the Radiocentre on behalf of the commercial sector. http://www.rajar.co.uk

further than scene or segment decomposition. For speech content this could be broken down into sentences, words, or syllables, whereas music content could be into verses or notes.

Lu and Hanjalic [25], Tzanetakis and Cook [26] have developed a framework, shown in figure 2.1, that helps represent the process of understanding audio content. A bottom up approach can be used, first extracting the low-level audio features, followed by defining mid-level audio elements, grouping them as a semantic segment and finally classifying the high-level segments.

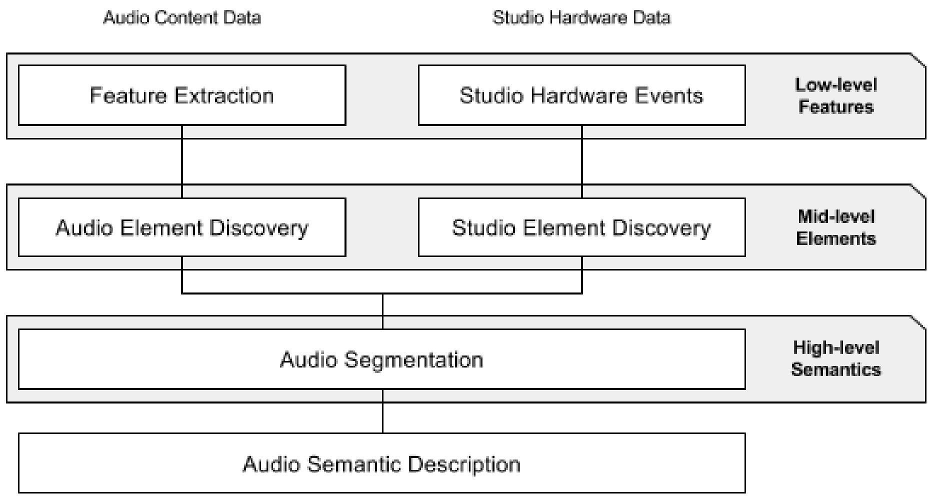


Fig. 2.1 Audio Content Analysis Framework [25, 26]

2.2.1 Feature Extraction

Pfeiffer et al. [27] recognised that audio can be analysed from two angles; physical and human cognition. Physical analysis studies quantifiable properties, such as amplitude or waveform, whereas human cognition is qualitative and varies per person (for example, how loudness or harmony is perceived). While physical properties of sound waves influence our perception of psychological features of sound (loudness, pitch and timbre), it is difficult to quantify human cognition.

Cooke [28] modelled the human auditory system to understand how it classifies certain groups of sounds, however our current understanding of the auditory system is insufficient to derive semantic meanings from physical features. Instead, a number of researchers [27, 29, 30] explored the proposal that if the input is known (physical properties of sound) and the output is known (classification of audio) then through the use of a training set, a black box process can be learnt and developed. To discuss this further it is important to understand the basics of sound.

Sound is defined as a change in air pressure generated by vibrating molecules, often modelled through a waveform composed of sinusoidal waves of different amplitude, frequency and phase. Experiments with different sounds show that phase is a perceptually insignificant (inaudible) component in complex sound, but it is evident that amplitude affects our perception of loudness and frequency affects our perception of pitch [31]. However, in order for us to analyse audio, analogue audio signals must be converted into a digital data form. This is known as sampling. Each sample takes a snapshot of an audio signal at a particular point and coverts it into an 8 or 16 bit data point. The digital representation is never as accurate as the analogue data, as it is merely sampling the signal at one point, however the greater the number of samples the higher the audio quality and the less likely human senses will be able to distinguish between the two. CD quality audio uses a sample rate of 44.1 kHz, or 44,100 samples per second.

Once the audio is in a digital form, it can then be converted into the sinusoidal waves, enabling us to identify the range of frequencies present. While human ears use the cochlea to convert time-dependent signals, such as those on a sound waveform, into signals corresponding to a frequency spectrum, computers must use a mathematical technique known as the Fourier Transform [32]. This process details what frequencies are present in the signal and in what proportion. Once this information is identified, the different sounds can be distinguished. These are described as audio features, but could be thought of as phonemes in language [33].

2.2.2 Audio Element Discovery

Once audio has been sampled and the Fourier Transform process has converted time-based signals into a frequency-based spectrum, it is then possible to derive the psychological features of sound, audio elements. Referring back to the previous analogy, as audio features can be viewed as phonemes, audio elements can be viewed as morphemes. Understanding a sequence of phonemes, helps to understand a morphemes and with that brings higher-level semantic understanding.

Element discovery does not just refer to audio, it has previously been used to identify scenes in sport videos. Huang and Tung [34] analysed baseball videos, looking at the visual features in each clip. Huang defined thresholds that if certain video features met, could confidently mark a clip as a particular scene. For example, analysing a shot for skin colour (number of skin regions and size of dominant skin region) would help classify it as a close-up player scene or audience scene.

Returning back to audio, researchers such as Pfeiffer et al. [27], Lu et al. [35], Barbosa et al. [36], recognise that these audio elements should be first classified into speech or non-speech content, although some researchers break non-speech elements down further into music, sound effects, noise or silence [25]. To achieve this, there are a number of techniques available. However, often to reduce the computational complexity of signal processing, audio features are first combined into frames. An audio frame can be considered a sample of the audio for the given timeframe, complete with time and frequency based data. The timerange can vary from 372 second frames [37], to 10 millisecond frames [38] depending on the use for the data. If we were analysing 44.1 kHz audio, then using 20ms frames would cut the number of samples to be processed down by 99.9%. Even with this sort of compression, the data given is still extremely rich with sufficient information to classify and segment audio further.

When attempting to classify audio elements there are two main methods to describe the audio from the previously extracted audio features: Mel-frequency Cepstral Coefficients (MFCC) and Zero Crossing Rate (ZCR) [35]. ZCR estimates the frequency content in an audio signal and can detect speech content by excitation of vocal tract that exhibits a low zero crossing rate, compared to non-speech content that exhibits a high zero crossing rate [39]. MFCC mimics the human auditory system measuring perception of loudness and pitch, but excluding speaker dependent characteristics such as fundamental frequency and harmonics [40].

Using these descriptors it is then possible to discriminate content types. “Silence and breathing noise is made from a simple energy-based detector (VAD) and the detection of music and noise is achieved using Guassian Mixture Models (GMMs)” [38, pg. 1].

2.2.3 Studio Element Discovery

While audio elements map directly on to the audio, it is important to note that audio elements should not be exclusively analysed to determine the segments. Often when content is produced the audio structure and metadata of each audio clip is available, however when it is stored this information is lost, as it is composed down into one audio file. An example of this is when a pre-recorded news story is edited down ready for broadcast. The audio could contain an intro, sound effects, a couple of interviews and a conclusion. In an audio editor this would appear as distinct tracks, positioned in a timeline. When this is exported the audio is mixed down into a new file, however the individual tracks and their relative position are all lost.

Object-Based Broadcasting

Object-based broadcasting (OBB), a new research area driven by the BBC [41] and a number of other European partners [42], is attempting to preserve this information within the audio files metadata. An European research group, ORPHEUS[7], have been formed by broadcasters across Europe looking generically across all forms of audio production, from live studio broadcasts to pre-recorded multi-track edits.

The Object Based Broadcasting term has only recently been used and recognises that segmentation information does not need to come from analysing audio, but can be collected from the production of the audio. This is very encouraging for segmenting audio as capturing data at the source is likely to be a lot more accurate, rather than reengineering it after it has been compiled down.

2.2.4 Segmentation Methods

Methods vary for detecting audio segments, with most using the detected audio elements to make an informed decision about a segment start and end. While we have

[7]ORPHEUS: https://orpheus-audio.eu

discussed two types of element discovery methods, audio and studio, it is worth mentioning that very little research mentions and explores using studio data to inform segmentation decisions. One possible reason for the lack of research is that this information is not publicly available, and often a lot of research is conducted separately away from the radio stations that are producing it. Although it could also be argued that radio stations might be cautious about letting new technology/techniques near studios for the fear it could cause problems. However, the hypothesis of this project is that studio elements could significantly reduce computational complexity required to accurately segment a radio show. This project has the advantage that the author is the station manager of a university radio station and therefore able to test solutions in a live environment. Due to the lack of research available about studio based segmentation, this literature review will focus on audio element based segmentation methods.

Theodorou et al. [38] defined two different approaches to segmentation; *distance-based segmentation* and *model-based segmentation.* The former can be referred to as top-down (context based) and the later as bottom-up (sensory analysis) [43].

- *Distance-based segmentation* analyses audio for the presence of a certain characteristic (described through methods such as *MFCC* and *ZCR*) and plots it against adjacent audio feature vectors, resulting in the distance curve. Adjacent features with the largest distance (characteristic change) will be the peaks to the curve. These can be considered as the audio segment boundaries. The Bayesian Information Criterian is a statistical analysis technique that performs this task [44].
- *Model-based segmentation* on the other hand classifies all the audio elements as specific sound objects (music, speech, sound effect). Each element is analysed individually and determined by a model that analyses its component frames, comparing them to training data for identification. Machine learning techniques, such as Hidden Markov Model (HMM) [45, 46] / Guassian Mixture Model (GMM) [47, 48] and Support Vector Machine (SVM) [49, 36] are the most common algorithms, but these models require a minimum number of example audio elements to train them properly [50, 51].

These two approaches ultimately come to the same conclusion, with some methods having better precision and recall than others. One approach breaks down an audio file into segments, while the other builds the segment from events [35]. However, while each segmentation approach has its benefits, Theodorou et al. [38] found that a hybrid of the above approaches were preferred in complex segmentation tasks. Three segmentation methods are now discussed:

1. *Volume analysis* - A distance-based segmentation method theorising boundary segments can be noticed by sudden volume changes [52, 53]. For example, a song fading out and a jingle suddenly starting before a presenter starts talking. The experiments found that some semantic content is attainable through volume

analysis, however this method alone is not enough to accurately segment. Other methods should be used in conjunction with volume analysis [53].

2. *Semantic Affinity Measure* - Another distance-based segmentation method [13], also known by its inverse *Audio Novelty* [54]. These methods find natural segment boundaries by analysing a range of different audio features, such as the ones extracted using MFCC. Analysing them on a distance-based graph indicates peaks where the concurrent audio features are most different or 'novel'.

3. *Pattern Recognition* - A technique following the model-based technique that can be used to detect parts of audio, often known as *audio fingerprinting* . Through a reproducible mathematical process, hash tokens are extracted from an audio element giving it a unique identifying signature known as a fingerprint. The same mathematical process is then applied to another audio file and if the unique fingerprint matches an item in the library of known fingerprints, then it can be deemed present in the audio. A very well known application using this is *Shazam*[8]. Starting in 2000, Shazam developed an algorithm that could recognise music even under unsuitable conditions, such as heavy ambient noise or subject to reverb [55].

Focusing on the *pattern recognition* segmentation method, an important factor is the reference database. Any item that has not been processed, and therefore its fingerprints are not listed in the database, will not be recognised. This method requires complete access to any audio you would like to recognise. For certain audio elements this might not be a problem, for example music and sound effects are stored within the station and could be processed if a system was put in place. Speech content on the other hand would not work. Presenters are unlikely to speak in the exact same tone even when repeating a sentence or phrase. Therefore, each time would result in unique fingerprints and no match would be detected.

Audio *pattern recognition* is an extensively researched area with numerous papers covering the advances in the technology across many domains [56, 57]. Oliveira et al. [58] developed a solution for the commercial advertising industry that could monitor tv and radio streams for given adverts. The results were impressive with Oliveira et al. achieving a 95.4% recognition rate with less than 1% of false-positives.

However, implementing this approach in a broadcasting environment requires fingerprinting all the content within a radio station's playout system, which are often very large, containing thousands of audio tracks. For fingerprinting to work, all these tracks would need to be analysed and compared against each processed entry. Burges et al. [37] developed the *RARE (Robust Audio Recognition Engine)* focussing on two key problems with previous solutions: detection of distorted audio and lookup speed within large audio fingerprint databases. Unfortunately Burges et al. did not provide any experiments between standard recognition techniques and RARE, so no quantifiable comparison can be made.

[8]Shazam: https://www.shazam.com

A slightly different take to pattern recognition is *audio recurrence* which identifies segments in programmes composed of recurrent segments that act as anchor points [59]. Rather than comparing extracted fingerprints with an existing detailed library, it compares it with itself. Unlike traditional pattern recognition, where the matched fingerprint will contain metadata describing itself, audio recurrence cannot semantically classify itself. An example of this would be a news program that uses the same sound effect between news stories. If a system detects each time it is played then it could split the programme accordingly.

2.2.5 Segmentation Classification

While it is important to detect segments, it is arguably as important to identify what the segment is. There are two reasons for this. Firstly, without it being identifiable a listener is none-the-wiser what the audio timeline contains, and secondly it would not be indexable so therefore cannot be searched and retrieved. Often the methods to classify a segment are very closely aligned to the methods used to segment the audio in the first place. This section will look at *speech recognition with semantic tagging* and *audio fingerprinting*.

Speech recognition is an extremely useful tool that can transcribe audio into a textual form and with further semantic analysis into keywords [14, 60, 15]. Once in a text form traditional semantic analysis can be performed. Dowman et al. [60] developed a prototype system that can annotate TV and radio news broadcasts using speech recognition and relevant articles on the Internet to provide semantic annotations. The system analysed the transcription and used mathematical key-phrase extractors to determine keywords from the audio. Using the keywords written articles could be found on the Internet to provide further information. Combining the temporally precise speech recognition with the conceptually accurate written articles had helped reduce the impact of speech recognition errors.

Since Dowman et al.'s research was published, there have been vast technological improvements. The solution was not a perfect one and had considerable flaws particularly around the speech recognition. Arguably only using one speech recognition technique means that transcripts cannot be compared and errors detected; this study was released in 2005 and since then technical developments have led to a myriad of speech recognition systems available.

Audio fingerprinting, on the other hand, uses an existing database to provide classification. As discussed previously in the segmentation method section, it is important that fingerprints are organised and tagged properly to help identification. The tags associated with a fingerprint can then be combined with other fingerprint tags in the same segment to derive an overall semantic description.

2.2.6 Metadata Structuring

Segmenting and classifying audio structure information is very important for it to be used further, however it must be stored in a suitable storage format. Currently popular audio file formats use metadata containers that wrap around the audio, which is encoded according to the file format. *ID3* is an example of a metadata container and is commonly used in conjunction with the *mp3* audio format. *ID3* metadata describes the audio file with properties such as title, artist and genre. The tag refers to the audio file as a whole, so if the audio contains different content then a common name and shared details must be agreed. MPEG-7 on the other hand is a multimedia content description standard that provides the ability to describe content rather than files. The specification uses XML to store the metadata and requires timecodes to correspond descriptions to the precise media content.

Metadata Production Framework (MPF) is another specification "designed as an efficient environment for generating metadata for TV programs from the viewpoint of the broadcaster" [61, 62, pg 1017.]. The framework is designed to describe portions of a video for a given time. Although this model simply adopts a subset of MPEG-7, it is modified slightly to allow relevant URL pointers for segments, something that is omitted in the MPEG-7 format which Sano et al. identifies as very useful.

2.3 Audio Content Retrieval

Once the audio segments have been detected and classified, it can lead to a myriad of uses. Audio retrieval is an important aspect to keep in mind through this project despite it not being a main focus. This project will focus on utilising existing retrieval methods, namely a web audio player, and enriching it with visualised segment information.

A further extension to this would be enabling the listener to search for specific content across a station, rather than retrieving segments through the specific broadcasted show. This would enable users to find relevant content and not be restricted by a single show or a show series. This section will briefly reflect on some existing research and solutions.

2.3.1 Search Techniques

Searching audio is another key topic benefiting from a range of research dating back to 1996 when Kubozono et al. [63] proposed an audio key-information search system. This project does not intend to develop or utilise a search technique, nonetheless it is important that metadata is stored in a suitable way. As a result, search techniques remain a popular area for future research. This section will briefly outline two approaches: textual search and content-based search [64].

1. *Textual search* uses textual indices that can be extracted from an audio file by using speech-recognition and keyword analysis. Subsequently they can be applied to traditional information retrieval techniques, such as the Document Ranking Optimisation algorithm (DROPT) [65]. Raimond and Lowis [11] used this process to tag programmes within the BBC Archive for text-based search retrieval.

2. *Content-based search* on the other hand gives the user the ability to search by physical and psychological audio elements, such as searching for a particular singer. It works using query-by-example where a fragment of the singer's voice is

submitted and compared for similarity against items in the database, as demonstrated by Tsai and Wang [66] and used by Shazam [55].

2.3.2 Audio Players

Once an audio file has been located and the file retrieved, the next step is to play it. Audio players vary from hardware players, such as tape machines or CD players, to software based players such as Windows Media Player[9] or Apple Quicktime[10]. With the rise of the internet, web based audio players have become increasingly popular. A web player gives listeners the ability to go to a website and stream the content directly, without the need to download it or use proprietary software to listen to it. Most basic media players consist of a play/pause button and an audio progress bar with a slider to indicate the current position. Understanding how users navigate content is an important detail that Hürst et al. [67], Lauer and Hürst [21], Lee [68], Carlier et al. [69] have thoroughly researched. Their findings are detailed below.

Without bookmarks or segmentation indicators a user is required to manually navigate an audio file to identify areas of interest. Hürst et al. [67] researched how users can interactively navigate multimedia files based on audio feedback. Whereas video is made up of individual pictures (visual frames), audio is made up of individual sounds (audio frames). Visual frames can be understood independently of other frames, however independent audio frames do not provide the same semantic meaning. Multiple audio frames must be played together to provide any interpretation, but this is inconsistent with the quick and uneven movement of a slider. To overcome this issue, Hürst et al. designed two different approaches: elastic audio slider and position based slider. The first approach limited the slider movement and instead introduced a new UI tool that would not allow the user to move the slider directly but influence the speed of its movement. The second approach reduced the strict synchronisation of thumb and acoustic feedback. In the educational multimedia domain, Lauer and Hürst [21] also took the same approach to analysing multimedia documents, but unfortunately neither Hürst et al. or Lauer and Hürst presented any user testing or analysis in their papers so it is difficult to examine the success of this method. Nonetheless it is an interesting approach to the problem of locating specific audio content within a file.

However, as we have been able to extract audio segments from the audio track, an interface allowing the user to view and navigate to these segments is required. Many papers address improving media players to facilitate searching and seeking, however these are primarily directed towards video browsing. Carlier et al. [69] developed a video player with bookmark navigation and automated prefetch of predicted content. The study compared two video players, one with the bookmarks and prefetched content, and one designed as a control player has no new features. The results were promising and identified that the new user interface “can lead to up to 4x more seeks to bookmarked segments and reduce seek latency by 40%, compared to a video player interface

[9]https://support.microsoft.com/en-us/help/17615/windows-media-player-12
[10]https://support.apple.com/en-gb/HT201175

commonly used today" [69, pg. 967]. However, it is important not to infer that the inclusion of segmentation bookmarks alone increased seek rate to these sections. By prefetching content the player could have biased users by offering significantly reduced seek latency, meaning that users will not have to wait for the video to buffer.

Amalia.js is an open-source HTML5 multimedia player that Hervé et al. [70] developed to edit, view and play audio with temporal and spatial metadata. The solution is a lot more advanced than what Carlier et al. presented, offering a full object oriented broadcasting view of an audio file. The player is promising with a number of useful features. There are also a number of lessons that can be learnt from the development of this solution. For example, Hervé et al. [70] developed the media player using HTML5 as it was well supported and available across most internet capable devices, but this brought its own challenges as the player had to be tested on a large variety of browsers, including mobile devices.

2.4 Summary

In conclusion, this literature review has critically analysed the state of the art, focussing on audio content analysis and retrieval methods. A brief introduction outlined the importance of the listener experience and why listeners value a listen again service. These findings should be taken into account to ensure any system developed will meet listener expectations.

It was evident that a large amount of research has been conducted on audio content analysis, with the majority attempting to extract data, that was once available during composition, by analysing the linear audio. Instead of reverse engineering the audio, another less researched approach is to collect the data during composition. This dissertation aims to follow the latter approach and apply it to segmenting a radio programme. Specifically, collecting studio events during a radio broadcast and analysing them to detect segments in a radio programme.

One possible reason for the lack of research is that this studio event information is not publicly available, and often a lot of research is conducted separately away from radio stations that are producing it. Although it could also be argued that radio stations might be cautious about letting new technology/techniques near studios for the fear it could cause problems. However, this project is able to take advantage of the university radio station and test solutions in a live environment.

3 Requirements Elicitation and Analysis

Chapter 2 explored a number of segmentation methods, however at the end of the chapter it was decided that this dissertation will use studio events to segment a radio programme. This chapter discusses how requirements were collected from a number of different sources, including analysis of similar existing solutions and conducting user research. A full requirements specification is presented at the end of the chapter

3.1 Establishing Requirements

This project is experimental in nature, therefore stakeholders have not been used to define requirements. Instead, requirements for this project have been gathered from a wide breadth of sources. Existing solutions have been analysed, user research has been undertaken and use cases have been derived.

3.1.1 Existing Solutions

To establish requirements for this project, it was important to understand the pitfalls and advantages of existing solutions to ensure the project learns from them. Although no automated segmented listen again service has been implemented, based on research of radio stations, there remains variations that are available to be analysed. For this section two types of existing solutions will be analysed; catchup radio players and podcasts.

Catchup Radio Players

A number of broadcasters have utilised catchup on-demand services for years. These services give users the ability to replay shows that have already been broadcasted, but unfortunately they are difficult to navigate to specific content. The BBC's *iPlayer* (Fig. 3.1) is considered by many the most advanced catchup player, but other broadcasters have followed suit.

Originally *Radioplayer*[1] (Fig. 3.2) was designed as a simple and consistent web player for online UK radio listening. It was formed by the BBC and commercial stations and became widely used by many, including student/community stations. *Radioplayer's* primary focus is to deliver live radio streams to listeners, however recently audio on-demand has been integrated into the web players. Unfortunately, the audio on-demand functionality is limited and can only handle an *Rich Site Summary (RSS)* feed that provides a title, description and links to the audio location. Many stations, such as

[1]Radioplayer, http://www.radioplayer.co.uk

Global Radio's stations[2], use the limited functionality on-demand tools *Radioplayer* provides. Contrastingly, the BBC have integrated audio on-demand into *iPlayer*, an existing in-house on-demand application. Despite the BBC's use of a separate catchup service, the user interface is very similar to *Radioplayer*.

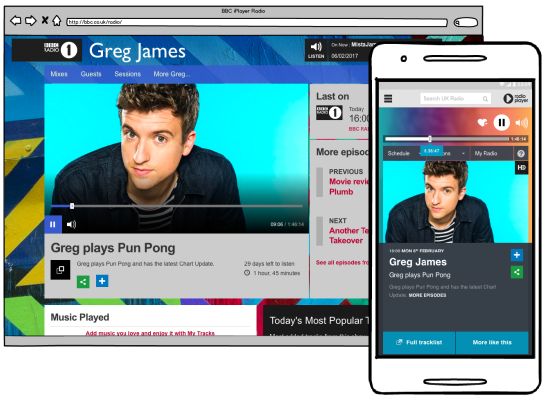


Fig. 3.1 BBC iPlayer Radio

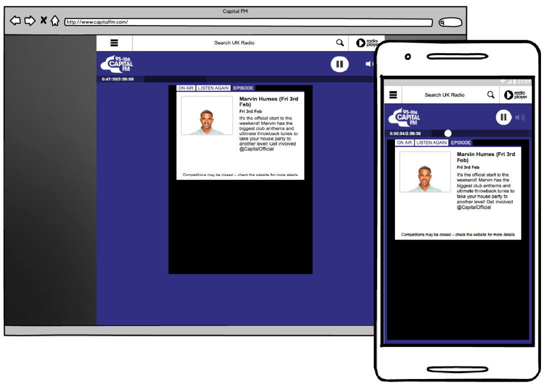


Fig. 3.2 Capital FM Listen Again

Comparing BBC *iPlayer* and *Captial FM's* web player (which uses *Radioplayer*) reveals a number of notable features:

- *Audio Progress Bar* - Each web player uses an audio progress bar (Fig. 3.3) to indicate the duration and relative position of the playing audio. A horizontal line usually represents the audio duration, while a cursor on the line indicates the current position. Hovering over the progress bar at any given point displays a popup with the time of that audio. Clicking on the progress bar begins audio from that point, this allows users to jump to content if given a specific time. Alternatively, a user can seek through audio by jumping to 'random' points until desired content is found. The time popup, that is displayed when you hover over the progress bar, can be used to navigate to an exact moment if the movement of the cursor along the progress bar is not clear enough. However, seeking along the progress bar can be more problematic on smaller devices where movement of the cursor over a short distance is difficult, such as a smartphone.

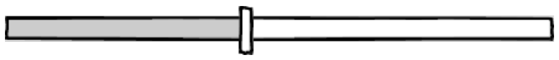


Fig. 3.3 Audio Progress Design

- *Audio Controls* - Users control the audio through a number of buttons (Fig. 3.4). Each web player analysed had a play/pause button and volume controls, which gives the user the ability to increase/decrease volume or mute the audio entirely. Notable controls missing are rewind and fast-forward buttons, however this could be to simplify the controls as the audio progress bar can easily be used to seek

[2]Global is a British media company which owns a number of large radio stations in the UK, such as Capital FM, Heart, Classical FM and LBC. www.global.com

through content, as described above. Another important observation are the standard icons used for controls, such as ▶, which creates intuitive players.

Fig. 3.4 Audio Control Design

- *Programme Information* - To give context to the user, each player displays the title and description of the content being played. Although there is no relation between the text and the audio on the player, it gives users a brief summary of what the audio is. For example, it might say 'Greg plays Pun Pong', but it does not give any indication what time the game was played and therefore a user must manually seek through the audio to find it. The BBC's catchup player displays more information, such as listing the music played, yet the player makes no attempt to correspond this information to the audio progress bar.
- *Mobile Optimised* - According to RAJAR[3] 27% of adults listen to the radio via smartphone or tablet. Therefore web players have been designed to be responsive to smartphone and tablet users, such that each device receives an appropriate interface. Although, as previously mentioned, the audio progress bar is a vital component which does not scale down well for smaller devices.

Podcasts

A podcast is an episodic series of audio files that users can subscribe to, made available via the Internet. Once subscribed, the user will then receive any new episodes published for a given podcast. The content of podcasts vary, yet primarily contains speech, as due to licensing reasons they cannot contain music. Podcasts can be split into three categories:

1. *Original Content* - The podcast uses content that was recorded specifically for the podcast.
2. *Republished Content* - The podcast has no new content and simply recycles content that was recorded for another purpose, such as a radio show.
3. *Hybrid Content* - The podcast mixes original content and republished content.

Podcasts have long been used by broadcasters as a radio catchup service, before full show listen again services were technically feasible. By removing the songs and leaving the 'best bits' the podcast is a manageable amount of audio for a listener to catchup on and for stations to offer. Similarly to full show catchup, podcasts are produced as linear audio and therefore currently face the same problems. Nonetheless, podcasts tend to be shorter and therefore seeking for specific content is not as time intensive.

[3]Radio Joint Audience Research: Official body in charge of measuring radio audiences in the UK. http://www.rajar.co.uk/index.php

Republished content and *hybrid content* podcasts may reuse parts of a broadcasted show. However, these parts have no link between the main show and a podcast. A listener to the *TED Radio Hour*[4] might hear part of a TED talk and be interested to listen to the rest, but this can sometimes be time consuming to find. Even though the audio has been sourced and condensed from the original talk, the linear form of a podcast removes a clip's individual metadata, making any link between a segment in one audio file and a segment in another audio file extremely difficult.

For example, *BBC Radio 1's Scott Mills Daily*[5] is a podcast produced after every Scott Mills show, which is broadcast weekday afternoons between 1pm-4pm on *BBC Radio 1*. The podcast describes itself as 'all of Scott's show, without the music'. It provides listeners with a quick and easy way to catchup if they miss the show.

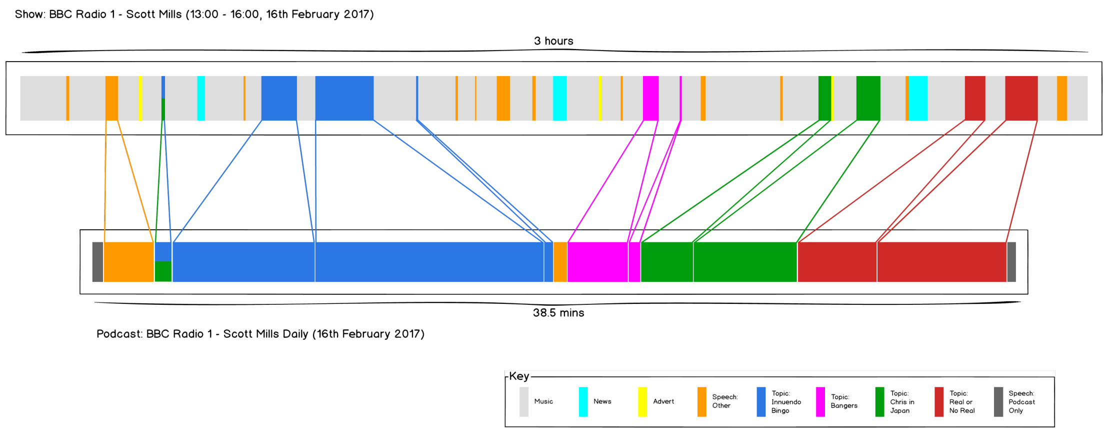


Fig. 3.5 Comparison between Scott Mills radio show and podcast.
Broadcast Show: 16th February 2017

Scott's daily podcast is unique. Having a single episode mapped to a single podcast can help identify the 'best bits' at a granular level. However, most other radio shows with a podcast release an episode weekly, taking content from a number of different days and combining them into one 'best bits' podcast. Figure 3.5 illustrates the direct one show to one podcast mapping, used in Scott's daily podcast. It is possible to further analyse this relationship and focus on speech segments. Figure 3.6 displays the likelihood of a speech segment being included within a podcast, based on the duration of segment. Arguably, this analysis is limited as only one show and corresponding podcast has been studied. Nonetheless, analysis clearly shows a trend that the longer the segment the more likely it would be included in a curated 'best bits' podcast. If a 'best bits' segment pattern can be identified then perhaps an automated process could deliver this to listeners, without any manual interaction.

Another interesting observation are the topics discussed in Scott's show. 71% of speech segments could be classified into one or more of the four show topics; Innuendo Bingo, Bangers, Chris in Japan, and Real or No Real. The topics and the frequency a topic

[4]The TED Radio Hour is hosted by Guy Raz, and is a co-production of NPR & TED. http://www.npr.org/programs/ted-radio-hour/

[5]BBC's Scott Mills Daily Podcast http://www.bbc.co.uk/programmes/p02nrv1j

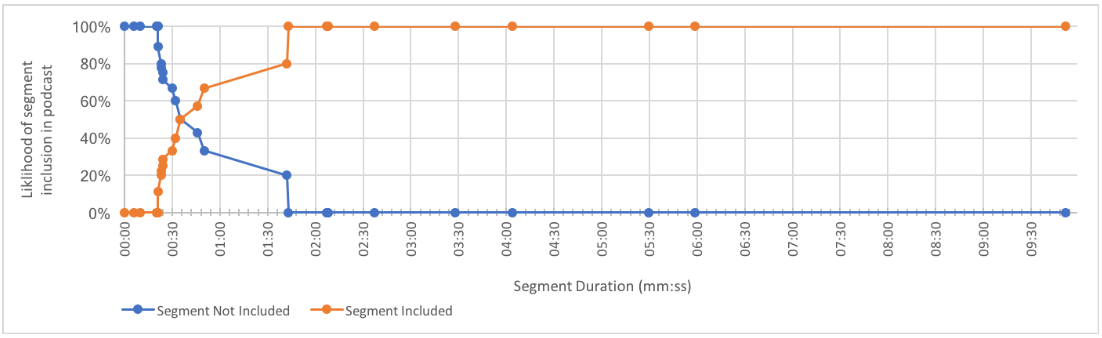


Fig. 3.6 Comparing speech segments from the Scott Mills Radio Show included and not included in the Scott Mills Daily Podcast. *Broadcast Show: 16th February 2017*

was discussed are displayed in Table 3.1. This observation highlights that segments should not be treated independently and instead could have a ordered nature.

For example, looking at appendix A.3, which is an itemised list of segments from a Scott Mills show and podcast, it is clear to see that item *52* and *54* are linked, despite the game 'Real or No Real' being split by music. If a listener missed the game and wanted to listen back, it is clear that item 52 must be played before 54. Highlighting this relation and the required listening order is important for user experience.

Table 3.1 Topics and frequency discussed in Scott Mills' show. *Broadcast Show: 16th February 2017*

Topic	**Description**	**Frequency**
Innuendo Bingo	A popular visualised game between a BBC presenter and a celebrity.	4
Bangers	Each presenter chooses a song and a listener decides between them.	3
Chris in Japan	Co-presenter Chris is in Japan and phones the show.	8
Real or No Real	A quiz asking true or false questions to guests.	2
Other	All other speech segments not discussing above topics.	7

3.1.2 User Research

Listener Focused Research

Another research technique utilised, to elicit requirements by this project, was a questionnaire. It was designed to gauge listener habits, reasons for listening and colour association with certain show elements (e.g. music, news or speech). The questionnaire and full results are available in appendix A.1.1. The main findings can be summarised as:

- *Respondents* - The questionnaire received 45 responses made up of 52% male, 44% female and 4% other gender respondents. The ages of respondents ranged from 19 to 74 ($M = 24.4, SD = 10.6$), however the majority of the responses came from 21 and 22 year olds in their final year studying a undergraduate degree. The questionnaire was widely distributed online, however the majority of responses were friends and peers, causing the demographics to be very similar to the University population[6]. Therefore we can theorise that the results are indicative of students and can not be directly extrapolated to the wider U.K. population.

- *Listening Preferences* - Live radio is popular, with 85% of respondents listening to radio on a typical weekday and 77% on a typical weekend. When comparing these figures with listen again services the results are very different with only 33% and 23% using BBC or a commercial radio station listen again service, respectively. Instead, music seems to be favoured with a huge 33% listening to 13 hours or more a week and only 11% not listening to streaming music services or downloaded music at all. A key consideration of this dissertation is to understand why live radio appeals to listeners far more than catchup radio. By using segmentation, this dissertation theorises that making catchup content more accessible will appeal to a wider audience.

- *Listening Times* - Similar to the U.K. RAJAR figures [23], respondents are most likely to listen to live radio weekday primetime (morning and drivetime) and weekend daytime. Listeners of catchup radio on the otherhand prefer a weekday evening slot, which could be explained by locations that the users listen to catchup radio as discussed below.

- *Listening Locations and Devices Used* - Listening locations differ depending on listening to live radio or catchup radio. Listening at home is favoured by both options, but far fewer listeners commuting will listen to catchup radio compared with live radio. Although, this could simply be due to data network restrictions and poor signal while travelling. The most popular device used for catchup radio is a smartphone, compared to live radio listeners who use AM/FM and DAB radios.

- *Listening Reasons* - People listen to radio for various reasons, but the questionnaire results indicate that presenters and music choice were the two main reasons. Interestingly these were the most popular options across both live radio and listen again radio. A reason for this could be that listening to presenters or music content live or on catchup does not affect the individuals listening experience as the content does not vary, whereas with news programmes content quickly becomes outdated. This is supported by the number of listeners for news coverage dropping from 24.4% on live radio to 7.7% on catchup radio. Another notable reason for listening included using live radio as a daily alarm clock.

[6]University of Bath Student Population Stats: http://bath.ac.uk/about/organisation/facts-figures

- *Design Research* - To aid with the design stage, a quick fire round of questions were devised. Respondents were given a number of categories, such as music and news, and asked to note the first colour they would associate with it. The results would then be used to design the user interface. However on analysis respondents could not agree on category colours, with the exception of news, where 69% preferred the colour red. It cannot be ascertained as to if the diverse results are due to individuals having no significant colour preferences or if respondents did indeed feel strongly about the colour they chose. As a result, the design stage will not use these findings as initially intended.

Expert Research

To further understand the current problems facing listen again services, it was important to speak to an industry expert. A senior technologist at the BBC discussed a number of considerations that should be made, highlighting some of the challenges they currently face:

- *Programme Information* - The BBC stores a vast amount of data about programmes that are never published. Whenever a radio programme is commissioned, a show plan must be submitted detailing the parts of the programme. A challenge for them is identifying how this data can be extracted and checked against a live show to ensure it was broadcast and at what time. Unfortunately as the accuracy of the information cannot be guaranteed, this additional information cannot be published. Data quality is extremely important for the BBC.

- *Programme Start Time* - Identifying the start of a radio programme may appear trivial, however it is extremely different. This is often because radio programmes have songs between shows and do not start exactly on the their scheduled time slot. It is a challenge for *iPlayer* as a listener may replay a show and find the audio does not start at the beginning. This can be confusing and annoying for the user.

- *Archival* - One of the biggest challenges for the BBC is archiving the immense quantity of content produced by the broadcaster and maintaining an efficient search index. As discussed in the literature review, chapter 2, it takes a significant amount of time for an archivist to describe and tag every programme so that it is easily searchable. Providing a mechanism to automatically segment and describe programmes would be extremely useful and provide huge efficiency savings.

- *Rights Management* - Another issue is tracking the legal rights of a programme. Often legal contracts are drawn up indicating the allowed use of a clip, programme or series. This becomes problematic when contracts may only apply to certain parts of the programme or for a specified amount of time. Every time content is reused an individual must search through the original contracts and check the conditions. This is extremely inefficient and leads to a lot of manual overhead.

> In a radio programme this does not just apply to speech content, music is licensed through the company *PPL*[7] who enforce strict rules, such as requiring an additional license to play music online in listen again services and restricting the duration this content can be available for.

The BBC are working with *ORPHEUS: Object-Based Broadcasting*, an European research group tasked at improving the management of audio content [71], to remediate a number of these issues and revolutionise the way data and audio is stored. This research group aims to develop future technologies and standards that will eventually become common place to describe and present media content.

3.1.3 Use Case

To assist in developing the requirements specification, some analysis has been performed on the various system use cases, and expanded into user stories pertaining to the system usage. Figure 3.7 illustrates a use case for the system being developed. It highlights the need for manual interaction, should the segment extractor need corrections, and the need for show segments to be generated before a listener can listen back to a radio show.

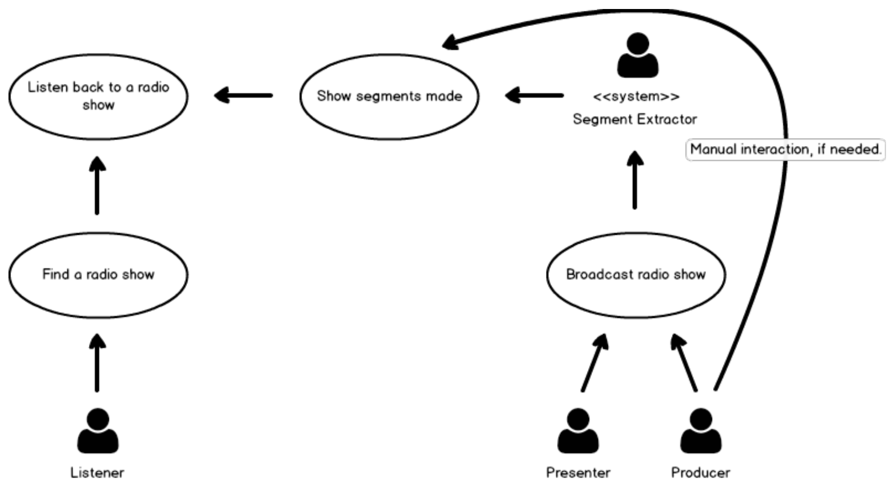


Fig. 3.7 System Use Case

3.2 Requirements Specification

Consolidating the analysis performed in the previous two sections, it is possible to derive a suitable requirements specification for this project. The requirements are split into two functional areas:

- Back-End: Detecting show segments and associated metadata
- Front-End: Interactive HTML-based listen again web player with segment-based metadata

[7]PPL - Phonographic Performance Limited: http://www.ppluk.com

3.2.1 Back-End: Detecting show segments and associated metadata

Functional Requirements

- *D.1* **Log Show Elements** (*Priority: High*): The system must identify and log the start and end time of show elements.
  - *D.1.1* **Music** (*Priority: High*): Identify and log songs being played during a show.
  - *D.1.2* **Speech** (*Priority: Medium*): Identify and log presenters talking on air.
  - *D.1.3* **Adverts** (*Priority: Low*): Identify and log an advert being played during a show.
  - *D.1.4* **News** (*Priority: Low*): Identify and log presenters talking on air.
- *D.2* **Analyse Show Elements** (*Priority: Medium* The system should analyse logged show elements and extract relevant information that'll describe and enhance the original element.
  - *D.2.1* **Speech Topic Detection** (*Priority: Low*): Perform speech recognition and frequency analysis on a presenter's link to attempt to identify keywords and topic tags.
  - *D.2.2* **Music Metadata** (*Priority: Medium*): Retrieve artist, song and other relevant information.
  - *D.2.3* **Segment Information** (*Priority: Low*): Retrieve information about segments based on data stored internally within the radio station's database, including segment legal rights and allowed usage.
  - *D.2.4* **Segment Ranking** (*Priority: Low*): Analyse a segment and calculate its perceived popularity. This may be based on preset rules or a learning algorithm that identifies popular content.
- *D.3* **Automatic Podcast Generator** (*Priority: Low* The system should automatically generate a 'best bits' podcast based on segment ranking or preset rules. *Dependencies: D.2.4*
- *D.4* **Management System** (*Priority: Low*) The system should have the ability to manually correct any automated processing error, such as an incorrect segment tag, or add additional information that could not be automatically generated, such as a show description.

Non-functional Requirements

- *D.5* **System Modularisation** (*Priority: Low*): The system should be built to enable additional detection techniques and segment analysis tools to be added in future.

D.6 **Performance** (*Priority: Medium*): The system should process and tag segments of a show within 1.5x it's duration. Podcasts should be generated within 2x the show's duration.

D.7 **Data Integrity** (*Priority: Medium*): The system should identify 90% of segments of the type music and speech.

3.2.2 Front-End: Interactive HTML-based listen again web player with segment-based metadata

Functional Requirements

P.1 **Visualise Show Segments** (*Priority: High*): The front-end player should visualise the different segments of a show with each segment type a predetermined colour.

P.2 **Audio Progress Bar** (*Priority: Medium*): The front-end player should have an audio progress bar which can be used to seek through a show.

P.3 **Audio Controls** (*Priority: Medium*): The front-end player should have simple and intuitive controls.

- *P.3.1* **Play/Pause Button** (*Priority: Medium*): A button that pauses the audio if it is playing, or plays it if it is paused.
- *P.3.2* **Volume Controls** (*Priority: Low*): A set of controls to give the user the ability to customise the audio of the player, including mute/un-mute button.
- *P.3.3* **Skip Button** (*Priority: Low*): A button to allow the listener to skip the current segment.

P.4 **Listen Back to a Show** (*Priority: High*): The front-end player must be able to operate as a listen again service and playback a specific show on request, within licensing agreements.

P.5 **Mobile Optimised** (*Priority: High*): The front-end player should be responsive to device screen sizes, resulting in a positive experience for mobile users.

Non-functional Requirements

P.6 **Performance** (*Priority: Medium*): The front-end player should load within 10 seconds and start playing a show within 30 seconds, based on an average U.K. internet connection[8].

P.7 **Usability** (*Priority: Medium*): The front-end player should be easy and intuitive to use, based on a design feedback focus group.

[8]Ofcom 2016 Report: 28.9Mbps

4 Design

Chapter 3 elicitated requirements from a breadth of sources with existing solutions analysed, user research undertaken and use cases derived. The full requirements specification was presented at the end of the chapter.

This next chapter will cover how the system was designed. In line with the development methodology, the following discussion begins with the high level system architecture and then moves down the layers to examine the granular system components.

4.1 Development Methodology

Development methodologies are regularly used in the development of software. They provide a structured approach focussed around the delivery of solutions and a lifecycle to aid project management. Agile methodology was chosen for this project as it allows more flexibility to respond to changes and less process overhead, meaning the focus is on solutions rather than process. As the client and the developer are the same person in this project, this methodology is well suited.

4.1.1 Iterations

Following the agile methodology, the system was developed in iterations. Each iteration (*I.1* - *I.3*) added some functionality, yet maintained a 'shippable product' after each iteration.

I.1 Firstly a manual event collection system is developed, which logs events to a database. A process, with limited rules, then converts the events into a format for the web player. This web player is then built as an event-driven player that displays the segmentation and plays the show audio.

I.2 Next, using automatically detected studio events, a more thorough analysis is performed to generate the segments. Following this, connecting with the radio station's database allows access to programme data which can then dynamically display programme information on the player's web page.

I.3 Finally, additional event collector processes will be implemented, such as recording the microphones on cue. Next, connecting to a number of third-party services, it is possible to get more information and provide further analysis of the contents of a segment, including rating and ranking them. Using this, an automated podcast generator can be developed to provide a 'best bits' catchup service. Additionally the web player will be enhanced to provide the ability to skip segments.

4.1.2 Risk Management

A simple preventative risk management strategy was applied to identify potential risks of the project. The potential risks, outlined below, have been assessed and given an impact and probability rating. Due care and attention has been taken throughout each stage of this project and the three risks below have been assigned remediation guidance.

R.1 Data Loss | Prob: Low | Imp: High
Remediation Guidance: Regularly backup reports and code on two different third-party cloud hosting providers.

R.2 Serious Illness | Prob: Low | Imp: High
Remediation Guidance: Project plan has accounted for contingency time.

R.3 Project too Ambitious/Has Technical Problems | Prob: Medium | Imp: Medium
Remediation Guidance: Each iteration will be developed to a ‘ready to ship’ state meaning should complications arise, the project can revert to the previous iteration.

4.2 System Architecture

This section discusses a high level abstraction of the system. A system architecture manifests the earliest design decisions which then become the foundations of development. The design decisions made here are the hardest to change and therefore are the most critical.

Firstly, a high level architecture was defined in order to create an intellectually graspable model (Fig 4.1). It is illustrative of the various processes and interactions required between the different databases, servers and third party services. The architecture is divided into three stages:

Stage 1 *Event Collection* - Live collection of events in the radio studio and storing them to a database.

Stage 2 *Segmentation Analysis* - Interpreting studio events to form segments; pulling in supplementary data from various third-party services. Two examples of available supplementary data are album artwork and speech recognition with keyword analysis.

Stage 3 *Web Player* - Interactive listen again web player, displaying a programme split into segments.

Splitting the architecture into three stages follows a modular approach which aids maintainability and reliability. For example, if new equipment was installed in the studio, the event collection process may have to be adjusted/redeveloped, however this would not impact the next process. To guarantee this, consideration must be made on the inputs and outputs of each stage, which are illustrated in figure 4.1. Each component is shown bordering either the back-end server, front-end server or both. The specific data and schemas stored and passed between components will be discussed in section 4.5. It is important to note that the servers act as the borders between stages.

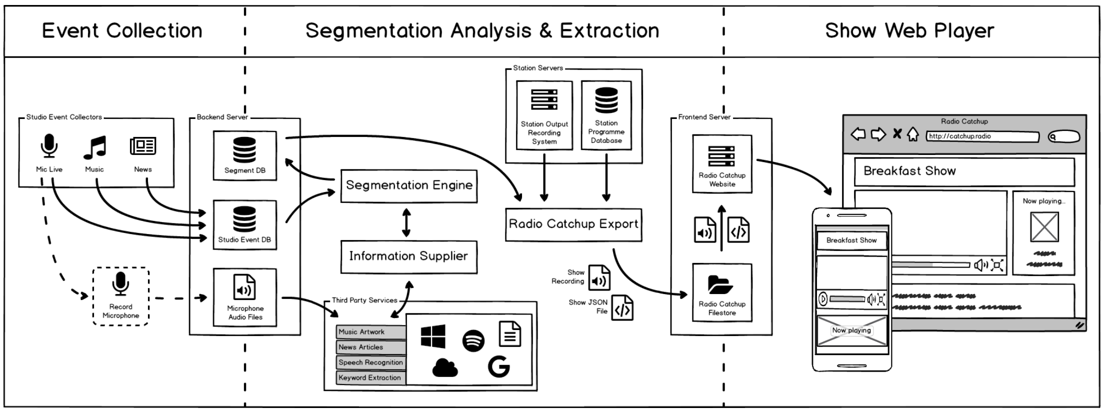


Fig. 4.1 System Architecture

The expected interactions and operability between each stage and the key components are outlined below.

- *Event Collection* - Event collectors are tasked with collecting studio data, usually in the form of events, and logging them to a database. Figure 4.1 lists three collectors: mic going live, music being played and news going out. This is by no means a definitive list of events, however they have been chosen as they form the most common show segments. If an event can be detected and a collector defined, then it can be added with minimal effort. The methods of collecting these events may differ, although the events will all feed into a single event database which will be analysed by the next process, segmentation analysis. Event collectors may run additional tasks, such as recording the microphones for further analysis in future stages. The priority is logging events to the database, so if the additional tasks are time intensive, a separate process can be spawned and run in parallel.
- *Segmentation Analysis* - This stage has a number of components. Firstly, segments are determined using logic rules and the previously detected events. This component is referred to as the segmentation engine stage, it retrieves additional information, from the information supplier, and incorporates it into the segment metadata. For example the segmentation engine stage would retrieve music metadata, such as album artwork. Arguably this component could be contained within the event collection stage, yet it is not as event collection revolves around live data capture, whereas segmentation analysis is run post-broadcast. Should another information supplier be added, a simple re-run of the segmentation analysis stage would incorporate the new information, without having to re-produce the show. The final component, Radio Catchup Export, is responsible for exporting the segments, combining show metadata (not to be confused with segment metadata from the information supplier) and packaging it into a suitable format.

- *Show Web Player* - The show web player stage has limited architecture defined as it is at the end of the process and provides a simple view. The focus of this stage revolves around delivering audio and segment data from the front-end server to a listeners device. Embedded on a single page and run on an existing radio station's website means a site page hierarchy or detailed web server architecture are not required.

4.3 Development Tools

4.3.1 Development Workflow

The development workflow, illustrated in figure 4.2, was defined to ensure that risks were mitigated and good practice was followed. Remediation guidance for risk *R.1* required that data is stored with two different cloud hosting providers, chosen to be *Github* and *Google Drive*. A project repository was setup on Github[1] with changes committed following development and Google Drive constantly backing up files after every change. Pycharm[2] was choosen as the IDE as it provided a powerful editor that could handle Python and shell scripts for the back-end systems and HTML, CSS and JS for the front-end web player.

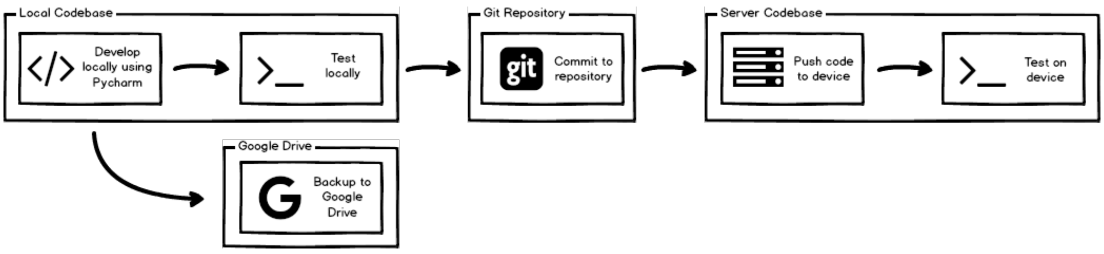


Fig. 4.2 Development Workflow

4.3.2 Software Libraries

This project does not aim to reinvent the wheel, instead it will reuse a number of existing third party software libraries. This should reduce the development workload, while also increasing performance and reliability of the developed solution. Larger and more experienced teams have spent far longer studying and developing open-source tools, thus it is sensible to use these as a basis for this project. This should also avoid the complexity cost which Gale [72] states you accrue by implementing complicating features or technology internally. While sometimes this is not avoidable, it must be managed. A good solution is to delegate functionality to tried and tested third-party systems, instead of attempting to increase the project scope and keep it all in-house.

One such case where the complexity cost is not required, is the web player. Requirement *P.4* states that a show must be able to be listened back online through a web browser.

[1]Github: https://github.com

[2]Pycharm by Jetbrains: https://www.jetbrains.com/pycharm/

There are a vast number of web players available online, including HTML5's built in media player and the player discussed previously in section 2.3.2, *Amalia.js.* Although *Amalia.js* provided an object orientated player, on closer analysis it was too complex for the project and other players were found to be more suitable. This project heavily used two front-end JavaScript libraries for the web player:

- *jPlayer* - Developed as an open source media library for JavaScript by Happyworm[3], *jPlayer*[4] is a light weight, customisable web player. Having been built over the past 8 years, jPlayer has an active and growing community. The player is well used, well supported and consequently very reliable. The player can be easily skinned and is very simple to setup. For these reasons *jPlayer* is the web player of choice for this project.

- *Popcorn.js* - The audio player must be interactive and have the capability to add bookmarks or segments. While *jPlayer* provides a seekbar so listeners can jump to content within the audio file, it does not provide a mechanism to tag the content. *Popcorn.js*[5], a Mozzila[6] project, is a media library providing an event system for audio players. The architecture is built around a core library that provides low level implementation of an event-driven system connected to the web player. Plugins can be used to further extend the capability and allow the development custom functions, which would all run based on events within the web player.

jPlayer and *Popcorn.js* have been used together frequently in the past, with Happyworm providing a detailed technical guide with a working demo. Utilising both of these well documented and supported libraries should lead to a reliable and seamless web player, which is a vital requirement of this project *P.6*.

Complexity cost can further be shed in the back-end development. The event collectors log to a database and the segment processors will retrieve this data, further analyse it using a number of third party services. The following stable Python libraries will be used to reduce the development overhead:

- *mySQLdb* - MySQL database connection library
- *pyodbc* - MSSQL database connection library
- *pyaudio* & *wave* - Audio recording and processing libraries
- *spotipy* - Spotify API connector library
- *speech_recognition* - Speech recognition library integrating with various API's

[3]Happyworm: http://www.happyworm.com
[4]jPlayer: http://jplayer.org
[5]Popcorn.js: http://popcornjs.org
[6]Mozilla Foundation: https://www.mozilla.org

- *nltk* & *textblob* - Natural language text processing libraries

The implementation and use of these software libraries will be discussed in greater detail later in this chapter.

4.4 User Interface Design

The interface is an important part of the project. Although event collection and segmentation analysis are automated processes with limited user interaction, the segmentation extraction process is taking the form of a web player and consequently will require a user front end.

Using the analysis from section 3.1.1, high-level sketches and wireframes were produced. The designs took inspiration from existing catchup players as these have already been studied, evaluated and revised in terms of user experience. The key challenge was to seamlessly integrate the standard web player design with segmentation.

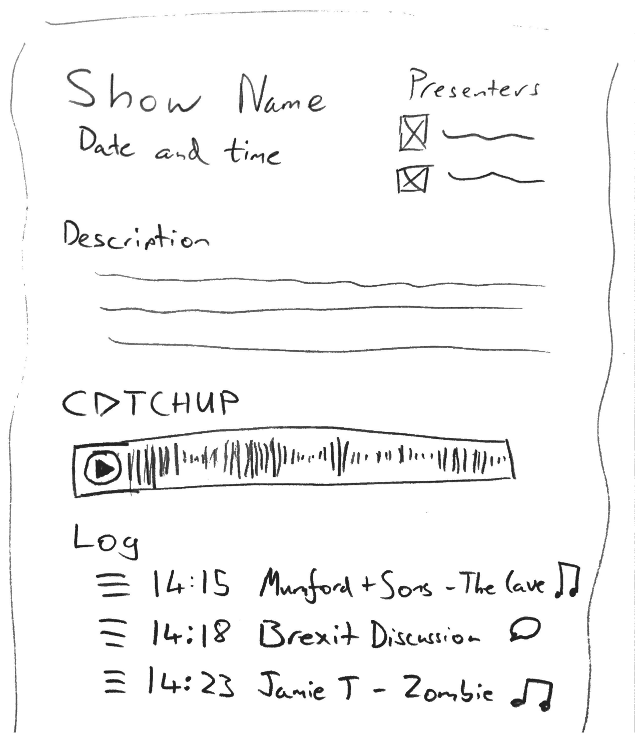
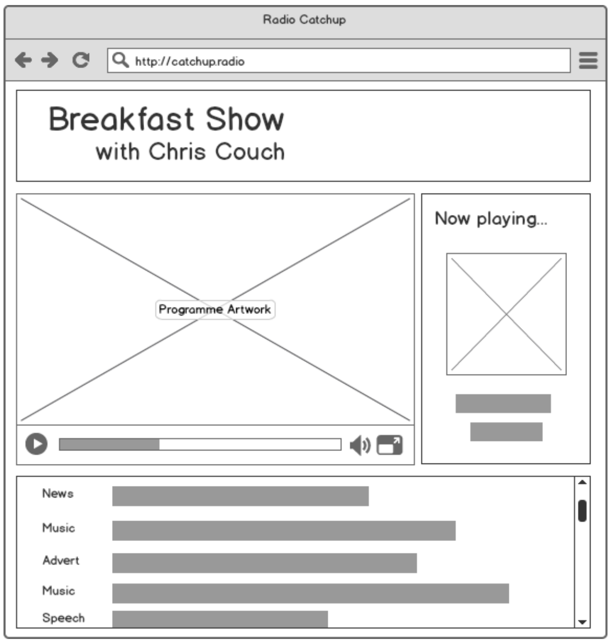


Fig. 4.3 Sketch - Web Player

The first approach, shown in figure 4.3, illustrates the initial design for the desktop webpage. A simple analysis of the design identified that the location of the web player may be inefficient and therefore improvements could be made. Having a large body of text above could require users to scroll down the page to start the player. This is not effective as the primary function of the web page is the web player, so this should be more prominent and available without scrolling or clicking. However, the log, sketched at the bottom of figure 4.3, illustrated a simple yet intuitive vertical timeline with segment elements listed.



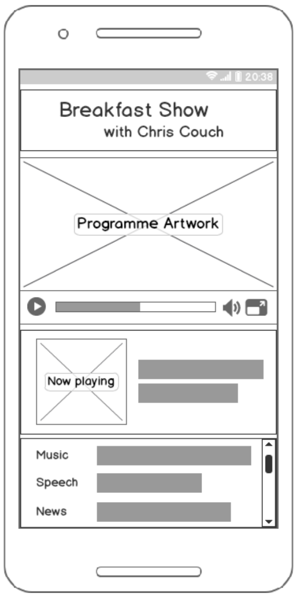


Fig. 4.4 Wireframe - Web Player (Desktop) Fig. 4.5 Wireframe - Web Player (Mobile)

The second design, shown in figures 4.4 and 4.5, addressed the feedback from the initial design. The second design also focused on a mobile-friendly design as per requirement *P.5*. A responsive design was envisioned, with each component on the page rendering for the specific device. The page components identified were:

- Programme Information
- Audio Player
- Currently Playing Information
- Programme Timeline

The wireframes show how, although the display may differ when viewed on a desktop or a mobile phone, the content remains the same. This graphical view of the proposed system and its responsive nature will be invaluable when developing the web player further. The implementation of this design and the challenges faced will be discussed in section 5.3.

4.5 Data and Schemas

As previously mentioned, it is important to define the data that is expected at each stage of the architecture. Figure 4.1 illustrates two boundaries between stages; the back-end server with an event database, and the front-end server with an audio and

JSON filestore. This section will define the schemas that each stage will follow, whether that is inserting, editing or reading data.

4.5.1 Event Collection Database

The event collection database is a MySQL[7] database which is responsible for storing studio events that have been detected. Defining the database structure now will help ensure both components in stage 1 and stage 2 can be built to effectively interface with the database and prevent any reprogramming, which would be required if the design changed later. Figure 4.6 is an entity relationship diagram for this database.

Each event is stored within the *catchup_event* table and is given an event type from the *catchup_event_type* table. Device names and network locations are stored to provide a record of the event collector which logged the event. Some events, such as mic live detection, may provide more data than simply event and timestamp. The *speech_keywords* table is an example of such an occasion. Keywords are extracted, given a score by a parallel process and stored within the database for retrieval during the Segment Analysis stage.

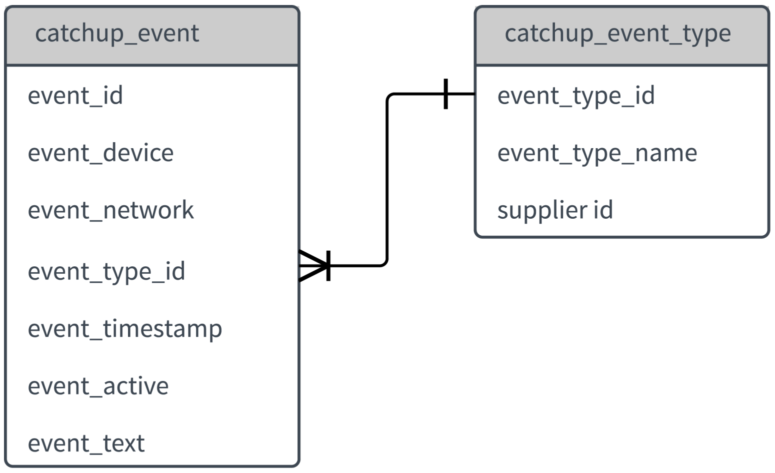


Fig. 4.6 Event Collection Database - Entity Relationship Diagram

4.5.2 JSON Schema

JSON, JavaScript Object Notation, is a light-weight data interchange consisting of objects, arrays and values. This project uses JSON as it is extremely fast and simple to parse. Stage 2, *Segment Analysis*, retrieves a programme's audio logging and generates a list of segments. These segments are then exported to a JSON file, with the programme information and audio location listed.

Although a database could have been used to store this information, it became apparent this would be inefficient as once the segment analysis stage has finished, the data becomes static and unlikely to change. Simply this means that the data should be 'hard-coded' to the specific audio file, with segment start and end times referring to

[7]MySQL: https://www.mysql.com

their location within the audio file. Should a programme and its segments wish to be shared, both audio and data files would have to be passed on. As discussed in 2.2.6, a more suitable storage mechanism would be using a single file that holds the audio and segment metadata, however currently format does not exist. For this reason two separate files are used and data integrity between the two files must be maintained. An mp3 audio file will be used for the show recording and a JSON data file for programme and segment metadata.

A full schema has been defined to ensure stage 2's segment export and stage 3's web player can interface. Using an example from the show 'The Hangover Cure', illustrated in listings 4.1, 4.2 and 4.3, the JSON schema will be outlined below. A full JSON schema can be found in appendix B.

Media Properties (Listing 4.1) The media properties object refers to the audio file of the programme recording. The *mp3* field stores the audio file location, where as the *poster* field stores the image location for the holding graphic, displayed in the web player.

Listing 4.1 Catchup Audio JSON File - Media Properties

```json
"media": {
    "mp3": "data/shows/brookes.mp3",
    "poster": "img/default-poster.png"
},
```

Programme Properties (Listing 4.2) Similar to the metadata stored within an mp3 file, the programme object holds the metadata of the programme as a whole. It includes broadcast start and end date and time, series information and episode information, which is populated from the station's scheduling system.

Listing 4.2 Catchup Audio JSON File - Programme Properties

```json
"programme": {
    "broadcastStart": 1490881500.0,
    "broadcastEnd": 1490883300.0,
    "seriesName": "Reverb",
    "presenters": "Dean Breed",
    "episodeName": "Fall Out Boy Special",
    "description": "Dean goes back through the years with his
   favourite band, Fall Out Boy."
}
```

Plugins (Listing 4.3) Segments are stored as an array within the plugins object. This structure and format follows *Popcorn.js*. A simple built-in function takes JSON data and loads it into the event-driven audio player. Following the same schema means little to no conversion is necessary by the web player.

Listing 4.3 Catchup Audio JSON File - Segment Properties

```
"plugins": [
    {
        "pluginName": "segment",
        "options": {
            "body": "",
            "displayKeywords": "Pop punk",
            "displayPicture": "https://i.scdn.co/image/
   e0cff5...",
            "displayEnd": "15:16:09",
            "end": 1869,
            "displayStart": "15:13:16",
            "title": "Fall Out Boy",
            "disabled": false,
            "start": 1696,
            "skipItem": false,
            "subtitle": "Dance Dance",
            "type": "music"
        }
  },
  ...
]
```

4.6 Summary

This chapter began by discussing a development methodology and went on to defining the system architecture and data schema. Relevant software libraries were also found and a suitable web player interface has been designed. In the chapter that follows we present the developed system, which has been built using these resources.

5 Implementation

So far this dissertation has researched and designed a system to segment and display a segmented radio programme. This chapter will now take a detailed look at the implementation of said system. Although development followed an iterative approach, details of each iteration will not be discussed, allowing the focus to be on the final solution and challenges faced during implementation. The chapter will follow the system architecture, defined in section 4.2, starting at the Event Collection stage and moving towards the Show Web Player stage.

5.1 Event Collection

The event collection stage (Fig. 5.1) is made up a number of components, each collecting different types of studio events. Each collector must be specific to the studio and setup to interface with the existing equipment. This project is using the *University Radio Bath (URB)* studios and focussing on two main collection processes; playout system audio and mic live detection.

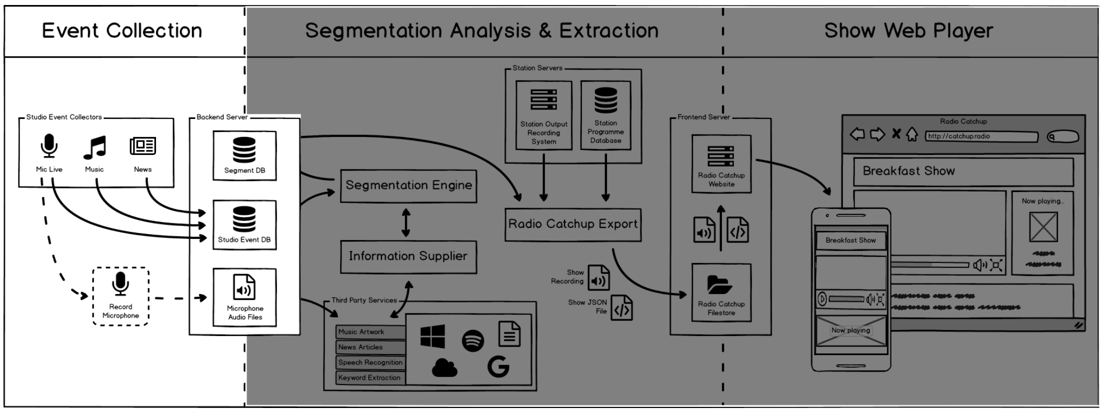
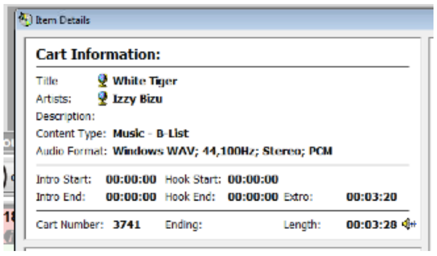


Fig. 5.1 System Architecture - Event Collection

5.1.1 Playout System Audio Collection

Technology has come along way since Vinyls and CDs were used to play audio on-air. Today the playout system is the core of a radio station and is usually made up of a number of computers, databases and fileservers. The system stores the station's music, adverts, jingles and any other pre-produced audio. In effect the system is responsible for all content broadcast, except live speech. A playout computer is fitted in each studio and gives the presenter access and control of this content, allowing them to cue and play audio at the touch of a button. *URB* uses a leading industry playout system - Myriad

Playout[1]. In order to understand how this content can be detected, it is important to understand how a presenter would operate the playout system.



Item Details

Cart Information:

Title	White Tiger
Artists:	Izzy Bizu
Description:	
Content Type:	Music - B-List
Audio Format:	Windows WAV; 44,100Hz; Stereo; PCM

Intro Start: 00:00:00 Hook Start: 00:00:00
Intro End: 00:00:00 Hook End: 00:00:00 Extro: 00:03:20
Cart Number: 3741 Ending: Length: 00:03:28

Fig. 5.2 *Myriad Playout* - Item Metadata

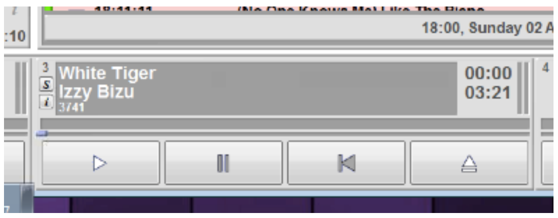
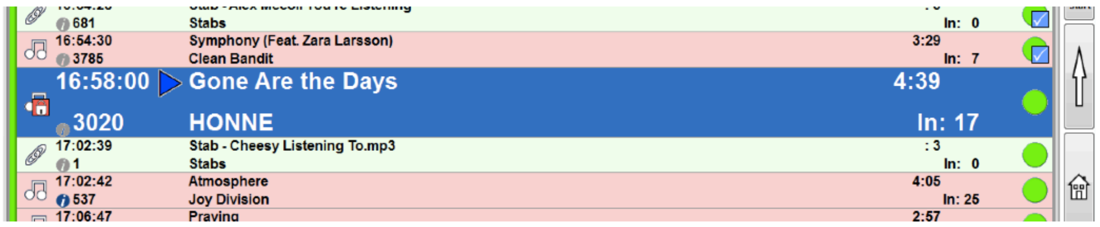


Fig. 5.3 *Myriad Playout* - Play Item

Firstly, audio must be added to the playout system. Although audio will be added by different teams, such as music by the music team and adverts by the marketing team, it is pooled into one repository and tagged (Fig. 5.2) accordingly to its content and purpose. This information is stored within a database and used when searching or playing content. Once audio has been added, a log is built (Fig. 5.4) which contains all the audio that will be played out during the show. Each log item's metadata, added when the audio is first imported into the playout system, is then displayed within the log to give the presenter context. During the show, the presenter simply presses the play button (Fig. 5.3) on the computer and the audio will start playing.



```
681        Stabs                                   In:  0
16:54:30   Symphony (Feat. Zara Larsson)     3:29
3785       Clean Bandit                            In:  7
16:58:00   Gone Are the Days                 4:39
3020       HONNE                                   In: 17
17:02:39   Stab - Cheesy Listening To.mp3    :3
1          Stabs                                   In:  0
17:02:42   Atmosphere                        4:05
537        Joy Division                            In: 25
17:06:47   Praying                           2:57
```

Fig. 5.4 *Myriad Playout* - Log

Table 5.1 *Myriad Playout* Database - *Log* Table Excerpt

Played Date	**Played Time**	**Played Length**	**Item Type**	**Artist Name**	**Item Title**
...					
42775	47851.508	256.59766	7	Ed Sheeran	Castle On The Hill
42775	48108.105	2.3554688	3	Sweeper	Sweep - Spring King You're Listening
42775	48110.46	222.94922	7	Bruno Mars	24K Magic
42775	48333.41	3.8867188	3	Stabs	Stab - You're Listening to UniRadioBath.mp3
42775	48337.297	213.27344	7	David Bowie	Life On Mars?
...					

[1]Myriad Playout: http://www.psquared.net/software/myriad-playout

Myriad Playout is intelligent and offers a number of useful features, such as generating now/next information to an online stream. The event collector must capture when certain audio content is played, such as songs, and log the start and end time. After careful analysis of the playout software, it was discovered that show logs are stored within a database to enable the software to record when and how much of the content was played (Table 5.1). The log table has a large number of columns, however most of them are not relevant as they contain empty fields. The *item type* field is used as a filter, ensuring only music (7) and advert (4) items are processed. The other items, such as a stab (3), do not constitute as a segment so are ignored.

5.1.2 Mic Live Collection

Whenever a presenter goes live on air, from the radio studio, they must turn the microphone on. A fader on the mixing desk in the studio (Fig. 5.5) operates each microphone. When the fader is faded up, the mixing desk puts the microphone on air and emits a signal which in turn activates the mic live light (Fig. 5.6). This light is traditionally used to warn presenters and guests when the microphone is live and consequently is only turned on when a presenter intends to talk on air. Therefore, if we capture and log the signal being detected, we have a record of the microphones going live and hence can reasonably detect a speech segment.

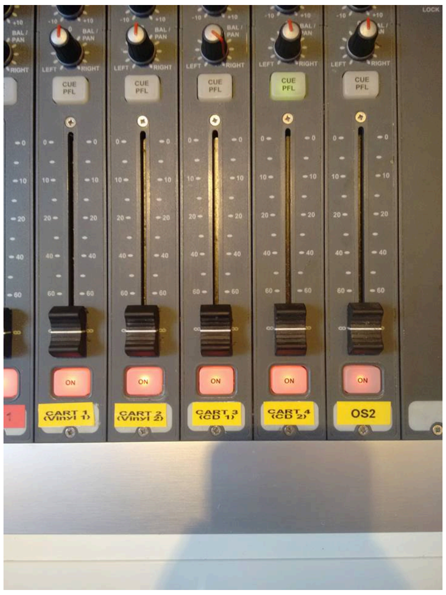


Fig. 5.5 *URB* - Studio Mixing Desk

Fig. 5.6 *URB* - Studio Mic Live Light

To capture the signal, a Raspberry Pi 2 Model B was used for this project. The Raspberry Pi is suitable as it is a low power, reliable and inexpensive micro-computer designed for simple tasks. The Pi runs Raspbian (Debian based OS), has network connectivity (using an Ethernet port) and is capable of interfacing with physical inputs through the bi-directional I/O pins. Using a simple electrical circuit and a voltage regulator (Fig. 5.7) the mic live signal emitted from the mixer (12V) is converted into a signal supported by the Raspberry Pi (3.3V). A Python script (Listing. 5.1), running on the Pi, assigns an event handler to the GPIO port which fires every time a change in signal is detected, i.e. when the microphone is turned on or off. The event function assigned then logs the timestamp and event type (mic on or off) to the database.

Listing 5.1 Raspberry Pi - Mic Live Signal Handler

```python
GPIO.setmode(GPIO.BCM)
GPIO.setup(event_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.add_event_detect(event_pin, GPIO.BOTH, callback=event_detected,
    bouncetime=event_wait)

def event_detected(channel):
    state = GPIO.input(channel)
    timestamp = int(time.time())

    if state and channel is event_pin:
        mic_live_active(timestamp)
    elif not state and channel is event_pin:
        mic_live_not_active(timestamp)
    else:
        pass
```

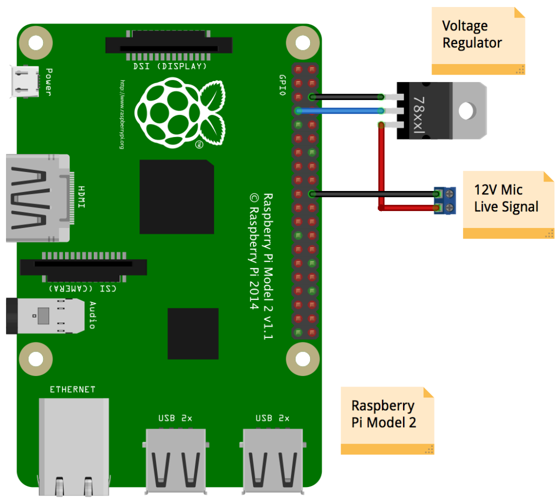


Fig. 5.7 Mic Live Event Collector - Circuit Diagram

While capturing the mic live event is crucial in order to detect a speech segment, it does not provide the full picture. This method, unlike the *playout system audio collection*, does not provide any segment metadata as only a segment start and end can be given. Instead the contents of a speech segment must be determined through other means, such as using speech recognition and keyword extraction. However these are not without challenges, during speech segments presenters often speak over songs and beds[2] leading to difficulties with speech recognition when analysing the combined station output. To overcome this, the microphone audio channel had to be analysed separately.

[2]A production element, usually instrumental music, played in the background for a presenter to talk over.

On splitting the microphone channel an audio feed could be run directly to the Raspberry Pi's 3.5mm audio input without affecting the existing studio infrastructure. Using this input the microphones could then be recorded on the Pi in isolation. Unfortunately due to the nature of the studio architecture, the three microphones could not be recorded individually but only collectively. While recognition did improve by removing songs and beds, with multiple presenters talking at once it remains difficult to transcribe. Furthermore, modifying the Python GPIO event handler enabled the start and stop recording to be automated every time the microphone was turned on or off. To ensure performance is not compromised logging the mic live events, further processing is passed off to another server. The Pi's final task after logging and recording the microphones, is to move the recording to the *back-end server* where it is stored for analysis. The audio file is then removed from the Pi to ensure there is space for future recordings.

Stage 1, *Event Collection*, leaves studio events logged in the database and microphone recordings stored ready for recognition on the *back-end server*. To capture these events the collectors must be running constantly. Therefore, much care and attention has been taken to ensure reliability and improve error handling. Should an error occur, log files are generated for manual inspection and cron jobs (ie. scheduled tasks) monitoring the event collectors will restart the process.

5.2 Segment Analysis

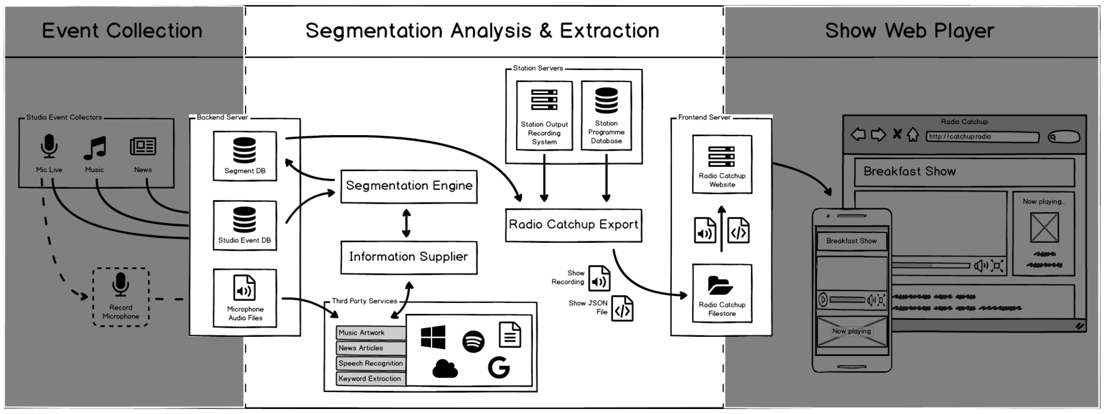


Fig. 5.8 System Architecture - Segmentation Analysis

Stage 2 (Fig. 5.8) performs logic on the events, which are detected in stage 1, and generates segments with relevant metadata. Whereas stage 1 must be running constantly to detect studio events, stage 2 has a different process. The nature of collating events and analysing them together to form segments, means it makes sense to batch process at the end of a show. Requirement *D.6* states that segments should be processed and available within 1.5x the programme's duration. Consideration must be made for performance and speed to ensure the process is completed within the required timeframe.

The original architecture defined the *segmentation engine* component, to process events and output segments to another database. Whereas the *RadioCatchup export* component would extract the data and pass it on in a suitable form alongside the programme metadata and audio recording. However, during implementation this approach seemed unnecessarily complex. Instead for this project, the *segmentation engine* takes the events, analyses them and is responsible for exportation to the front-end server for the web player. This simplifies the process and results in a Python script responsible for the whole stage. Any modification, such as on-boarding a new event collector or third-party information supplier, the script can be re-run and consequently update the audio and JSON files, without having to edit the segment information in the database or call a separate component to export.

The Python script (Appendix E.2) requires three arguments: programme start, programme end and filename. Using these arguments the script will build a segmented show JSON file and retrieve the audio for the given start and end times, following the steps below:

1. Programme information is retrieved from the URB database
2. Show audio recording is downloaded from URB system
3. Programme data is generated
4. Media data is generated
5. Segment data is generated
6. Data is exported and saved to a JSON file

5.2.1 Station Data Retrieval

Steps 1 and 2 interface with the station's scheduling and audio logging systems, respectively. Using the start and end time the scheduling database is queried for programmes scheduled on-air. If more than one result is returned then it is possible the timings may cross multiple shows, or there is a scheduling error. In either case a warning is flagged, but the first programme is used. A more thorough implementation might handle the case of multiple shows, however as this script is run automatically after a programme using its schedule start and end time, this case should never happen. Retrieving the show recording is relatively simple using a HTTP request to the logging server.

Once the audio and programme information has been retrieved, steps 3 and 4 store it in an array with a similar structure to the JSON schema defined in section 4.5. The array is later passed onto step 6 for the JSON file.

5.2.2 Segmentation Engine

Step 5, segment generation, is the crux of this project. Each segment type, with the exception of music and adverts, will be generated differently and based on different events. Therefore, as demonstrated in listing 5.2, separate functions are called.

Listing 5.2 Segment Data Generation - Python Function

```python
def generate_segment_data(start_date, end_date):
    segments = []

    speech_segments = mic_live_retriever.retrieve_segments(start_date,
    end_date)
    segments.extend(speech_segments)
    # playout_audio_segments includes music and advert segments
    playout_audio_segments = play_log_retriever.retrieve_segments(start_date
    , end_date)
    segments.extend(playout_audio_segments)

    # Sorts the segments based on their start time
    segments = sorted(segments, key=lambda x: x['options']['start'], reverse
    =False)

    return segments
```

Playout Audio Segment

The play_log_retriever module accesses the playout system and gets the audio type, start time and played length, as described in section 5.1.1. These results are parsed and depending on the audio type, are passed to a specific function to construct the segment. Listing 5.3 gives an abridged view of the music segment function. It was observed that the segment start and end did not match the audio, thus it was necessary to introduce an offset variable. After some testing it was found that using a seven second delay, i.e. adding seven seconds to the play log start time, re-aligned the segments with the audio. The delay is likely caused by unsynchronised clocks and the station's transmission chain. As per the schema specification, the music segment metadata is added to the JavaScript options object and is later returned through the segment object.

Listing 5.3 Play Log Retriever - generate_music_segment(...)

```python
def __generate_music_segment(start, length, end, intro_end, artist, song,
    genre, notes, start_programme_datetime, end_programme_datetime):

    start = start + timedelta(seconds=log_time_offset)
    end = end + timedelta(seconds=log_time_offset)

    # JSON DATA SCHEMA FORMAT
    options = dict()
    options["title"] = artist
    options["subtitle"] = song
    ...
    options["start"] = (start - start_programme_datetime).seconds
    options["end"] = (end - start_programme_datetime).seconds
    options["type"] = "music"
    options['displayPicture'] = spotify_information.get_album_artwork(artist
    , song)
    ...
    return segment
```

Mic Live Segment

The mic_live_retriever module is similar to the previously described play log function, except that it must associate two events together, mic on and mic off, to produce a segment. The mic live events are read chronologically with a mic on event and mic off event setting the speech segments start and end time, respectively. Although this segment now describes the period of the microphone being on, it does not necessarily refer to when a presenter is speaking. Often to prepare for their live broadcast, presenters turn on the microphone several seconds before talking. Further development could determine this by analysing the microphone levels in the recordings from stage 1 (Section 5.1.2), however pinpoint accuracy of segments was not a priority for this project. Once a start and end time is attributed to a segment, additional metadata is added (Listing 5.4). Interestingly, unlike the music segments, the speech segments did not require any timing delay between show recording and event timestamp.

Listing 5.4 Mic Live Retriever - generate_speech_segment(...)

```python
def __generate_speech_segment(start, end, start_programme_datetime, keywords
    ):
    options = dict()
    options["title"] = "Presenter Link"
    options["subtitle"] = ""
    ...
    options["displayKeywords"] = __get_speech_keywords(time.mktime(start.
    timetuple()), keywords)
    options["start"] = (start - start_programme_datetime).seconds
    options["end"] = (end - start_programme_datetime).seconds
    options["type"] = "speech"
    ...

    return segment
```

Information Suppliers

Segmentation alone is not enough to provide semantic understanding, often supplementary information is needed. Take a filesystem for example, locating a pdf or image file is useful, but unless you open the file or remember what is stored at that location you cannot not know what the file contains. Although artist and song information could be extracted from the playout software for music segments, other information could not, as a result *information suppliers* were introduced. The suppliers connect to third-party services and provide additional information about the segments, such as album artwork for songs or associated news articles for news segments. As previously discussed (section 4.3.2), using third-party services reduces our complexity cost and likely leads to better results than implementing it ourselves.

Spotify[3] is a popular music streaming service with a large music catalog that is searchable through a simple REST web API. Using the artist and song title to search the cata-

[3]Spotify: http://www.spotify.com

log, Spotify returns a number of matching songs ordered by search relevance. Therefore, selecting the first result is usually appropriate. The vast amount of information about individual songs that Spotify holds is accessible through the API. Currently only album artwork url is extracted, yet further development could extract information such as artist profiles or release dates, which could be interesting for listeners using the web player.

On the other hand, context for speech segments are difficult to provide. There is no title or artist information to collect, only a start and end timestamp. In stage 1 (Section 5.1.2) the microphones were recorded and the audio sent to the *back-end server* for analysis. Next, using speech recognition services, such as Google Speech Recognition[4], it was possible to transcribe the speech within a segment. Further to this, natural language processing was applied to extract keywords in order to provide some context to the speech segment. Listing 5.5 highlights the keyword extraction function. Simply put, a *term frequency-inverse document frequency* (TF-IDF) strategy was applied, using previous transcripts as a corpus. This strategy resulted in each keyword being scored; the higher the score the higher the relevance of the keyword. The Python libraries *TextBlob* and *NLTK* were used to perform natural language processing on the speech segment transcripts.

Listing 5.5 Text to keywords function

```python
def __text_to_keywords(transcript_file, transcript_library, library_size):
    print ("TextToKeywords.py: Extracting keywords from - " + transcript_file)
    recent_transcripts = __generate_recent_transcript_library(transcript_file, transcript_library, library_size)
    transcript_text = tb(__get_text(transcript_file))

    print ("TextToKeywords.py: Scoring words - " + transcript_file)
    scores = {word: __tfidf(word, transcript_text, recent_transcripts) for word in transcript_text.noun_phrases if word not in stopwords.words('english')}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)

    for word, score in sorted_words[:5]:
        print("Word: {}, TF-IDF: {}".format(word, round(score, 5)))

    return sorted_words
```

As an information supplier, the architecture positions the speech-keyword extraction process during the segment analysis stage. However it was found that this process had a significant runtime overhead. As a result, the speech-keyword extraction process begins immediately after the microphone recording has been moved to the server. At the end of the process, the keywords are exported to a database where they are retrieved during the generate_speech_segment function (listing 5.4). Only the top 5 keywords, based on the TF-IDF score, are included and stored within the 'displayKeywords' variable.

[4]Google Speech Recognition: https://cloud.google.com/speech/

5.2.3 Data and Audio Export

Finally, the segments are sorted into ascending order of start times. This ensures when the segments are exported into JSON and read by the web player they will be displayed in chronological order. The segments array object is returned to the main function which passes the object along with the program and media data, from steps 3 and 4, to the JSON dump function. This function then writes the data to a JSON file in a given directory. The JSON file stores the location to the downloaded audio, thus there is no requirement for the JSON and audio to be located in the same directory. Nonetheless the web player reads both files, therefore they must be stored in a publicly accessible filestore.

5.3 Web Player

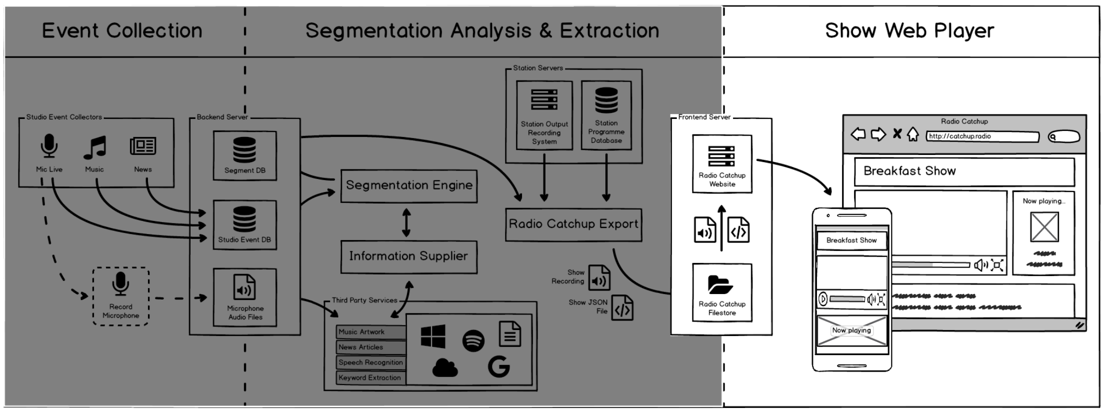


Fig. 5.9 System Architecture - Web Player

The last stage (Fig 5.3) is a web player which plays the audio and visualises the segments which were extracted in the previous stage. The web player is built using the latest web technologies: HTML5, JavaScript and CSS. Although Adobe Flash was previously popular for media players, now. A number of mobile devices do not support flash and the ones that do suffer from performance issues. As a result, this project will use HTML5 and JavaScript instead. There exists a plethora of media players available, however as discussed in section 4.3.2 this project uses *jPlayer* and *Popcorn.js*.

5.3.1 Media Player

The most important functionality of the web player (Fig. 5.10) is the ability to listen to programmes. HTML5 provides built-in support to play audio, however the functionality is relatively basic and any advanced functionality requires interfacing directly with the browsers. *jPlayer* removes this complexity and instead provides a simple API to load and play media content. One of the main advantages this library offers is the ability to skin and customise the player interface.

The first challenge was implementing the design, discussed in section 4.4, using the jPlayer media library and integrating University Radio Bath's branding. Although the wireframes had shown volume and full screen controls, these were not implemented. Alternatively, audio rewind/skip buttons were introduced as research found that it was difficult to precisely navigate using the seek bar for long duration media. This problem would have been exacerbated on mobile devices where the seek bar is smaller and therefore harder to move. The new audio controls, a simple ±15 seconds option shown in the bottom right of figure 5.10, mitigate this.

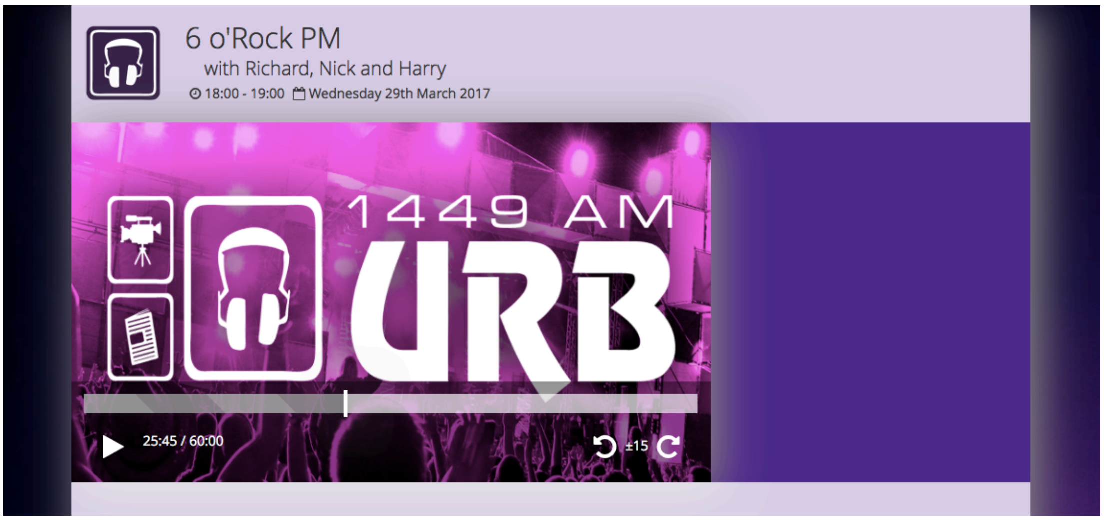


Fig. 5.10 Web Player - Initial Design

Utilising *Popcorn.js* events were added to the jPlayer media. Originally this library was designed to display subtitles or particular content at a given point in the media. However its rich plugin architecture allows for developers to build their own functionality into it. This project takes advantage of this by developing a custom 'segment' plugin for the purpose of displaying segments interactively. This is achieved using three different UI components on the player: progress bar, current segment, and timeline. The plugin runs on page load for setup, but also on the segment start and end, which allows the interface to the current playing segment to be dynamically updated.

Listing 5.6 Web Player - Setup Player (JavaScript)

```javascript
$(document).ready(function(){
  player = setupCatchupPlayer("#jquery_jplayer_1", "timeline-container", "
    seekbar-container", "data/shows/grace.json", {});
});
```

On page load, the *setupCatchupPlayer(...)* function is called from a script within the main HTML web page (Listing 5.6). The function takes a number of arguments, including the HTML id for the player and the JSON file containing audio and segment information. The web page contains the basic HTML elements, whereas the content is customised using the JSON file. The programme metadata populates the webpage header, while the media metadata that loads the audio and poster image into *jPlayer*. The segments are then passed to *Popcorn.js* which generates the segment UI elements and sets up an event handler for when a segment is played (Listing 5.7). The segment data, generated in section 5.2, is stored within *Popcorn.js* event objects so they can be retrieved for use within other functions. For example, Listing 5.7 is called every time a segment is played and decides whether it can be played or not. The function retrieves the disabled & skipItem attribute, which were defined in the JSON file and passed through to the options object.

Listing 5.7 Web Player - Segment Start Function (JavaScript)

```javascript
start: function( event, options ) {
  // 1. Check item can be played
  if (options.disabled == true || options.skipItem == true){
    this.currentTime(nextSegmentTime(this));
  } else {
    $(timelineElement).addClass('segment-playing');
  }

  // 2. Update segment sidebar UI
  refreshSegmentSidebar(segmentPlayingContainer, options);

  // 3. Update segment timeline UI
  centreCurrentSegmentTimeline(timelineElement, timelineContainer, options);
},
```

5.3.2 Segmented Progress Bar

The progress bar (Fig. 5.11) is an integral part of the player. It displays the current position of the media and is used to navigate through the content. Previous analysis of existing systems (Section 3.1.1) found these to be commonly used and beneficial to listeners. As the progress bar represents the duration of audio, it is possible using HTML *divs* to overlay the segments using their start and end timings. On page load each segment is added to a container *segment-container div* as a child *div* element and given a width and left position according to the segments location within the programme. The segment *divs* can be personalised further, such as setting the segment type as a class to give it specific styling or using Bootstraps tooltip to display the segment title on mouse hover over. Most importantly, the segment overlay does not interfere with the ability to seek using the progress bar. Listing 5.8 is an excerpt of the segmented progress bar HTML generated on page load, while Figure 5.11 is how it looks to a user.

Listing 5.8 Web Player - Progress Bar Elements (HTML)

```html
<div id="seekbar-container" class="jp-seek-elements">
  ...
  <div id="SB-segment1491753815487" class="jp-seek-element segment-speech"
    data-toggle="tooltop" style="width: 4.304%; left: 41.711%;" data-
    original-title="Presenter Link"></div>
  <div id="SB-segment1491753815489" class="jp-seek-element segment-music"
    data-toggle="tooltop" style="width: 8.192%; left: 45.794%;" data-
    original-title="Prince - Sign O The Times"></div>
  ...
</div>
```

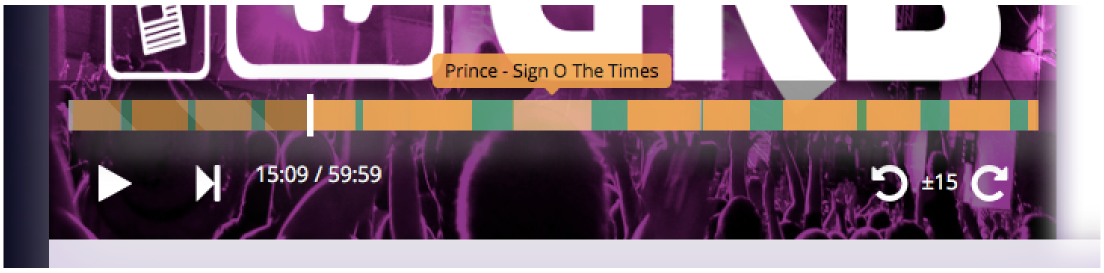


Fig. 5.11 Web Player - Segment Seekbar

5.3.3 Segment Timeline

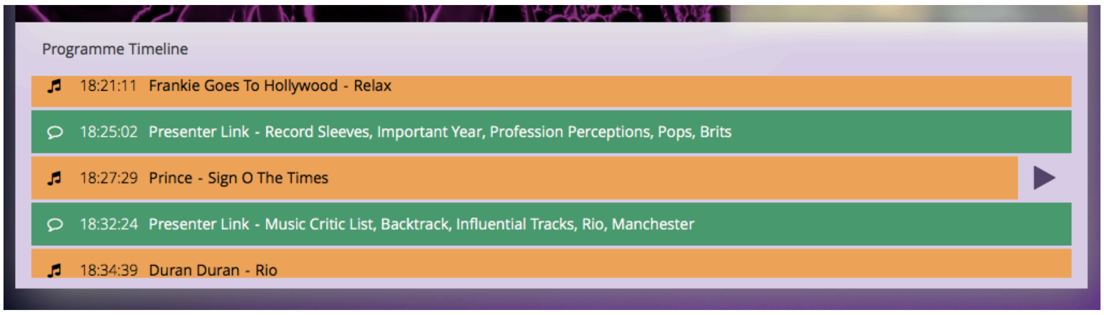
The timeline (Fig. 5.12) displays all the segments for a given show vertically in chronological order. As a result it can be used to view the contents of a programme and to navigate quickly to a chosen segment. Similar to the progress bar, the timeline is setup on page load by adding each segment as an *a* element to a parent container. However, the timeline displays more information than the progress bar, which has start time, keywords and displays a play symbol next to the current playing segment. A JavaScript *onclick()* function also allows a user to click a segment to set the audio player to that position. Listing 5.9 is an excerpt of the timeline HTML generated on page load, while Figure 5.12 is how it looks to a user.

Listing 5.9 Web Player - Timeline Elements (HTML)

```html
<div id="timeline-container" class="timeline-container">
  ...
  <a id="TL-segment1491757608630" class="list-group-item timeline-segment
    skipped timeline-speech">
    <span class="timeline-display-time">18:25:02</span>
    <span class="timeline-title">Presenter Link</span>
    <span class="timeline-subtitle">Record Sleeves, Important Year,
    Profession Perceptions, Pops, Brits</span>
    <i class="timeline-play-icon fa fa-fw"></i>
  </a>
  <a id="TL-segment1491753815489" class="list-group-item timeline-segment
    timeline-music">
    <span class="timeline-display-time">18:27:29</span>
    <span class="timeline-title">Prince</span>
    <span class="timeline-subtitle">Sign O The Times</span>
    <i class="timeline-play-icon fa fa-fw"></i>
  </a>
  ...
</div>
```

5.3.4 Current Playing Segment Sidebar

The current playing segment sidebar is there to display what audio is currently playing in the web player. Unlike the timeline and progress bar items which are generated on page load, the sidebar information is generated by the *Popcorn.js* ‘segment’ plugin, using the event objects, every time a segment is played. This happens regardless of when



Programme Timeline

18:21:11 Frankie Goes To Hollywood - Relax

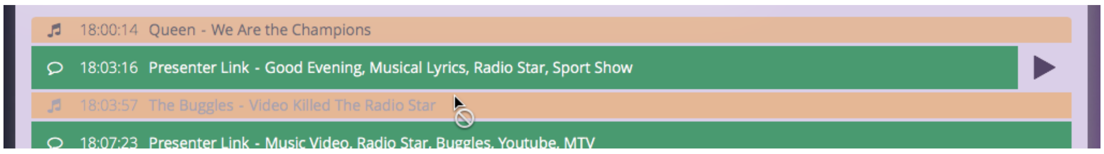
18:25:02 Presenter Link - Record Sleeves, Important Year, Profession Perceptions, Pops, Brits

18:27:29 Prince - Sign O The Times

18:32:24 Presenter Link - Music Critic List, Backtrack, Influential Tracks, Rio, Manchester

18:34:39 Duran Duran - Rio

Fig. 5.12 Web Player - Segment Timeline



18:00:14 Queen - We Are the Champions

18:03:16 Presenter Link - Good Evening, Musical Lyrics, Radio Star, Sport Show

18:03:57 The Buggles - Video Killed The Radio Star

18:07:23 Presenter Link - Music Video, Radio Star, Buggles, Youtube, MTV

Fig. 5.13 Web Player - Segment Timeline (Segment Disabled)

the segment starts playing, whether that is at the beginning or mid way through. The information displayed will depend on the metadata available for the given segment. For example, if album artwork has been given, then the URB station logo and background will be replaced with the artwork. Figure 5.14 displays the sidebar for a given music segment, whereas Figure 5.15 shows the sidebar for a speech segment.

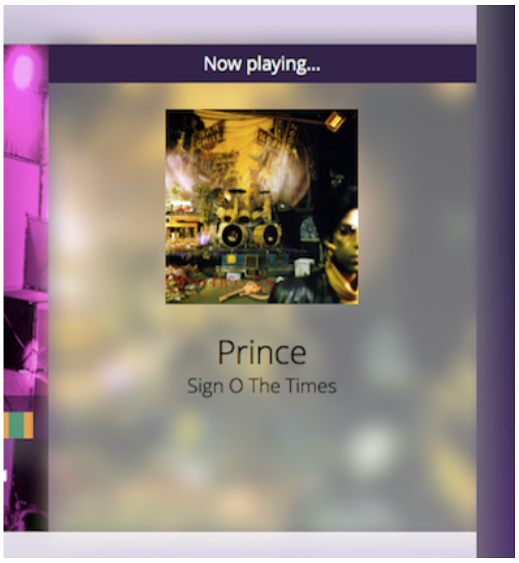


Fig. 5.14 Web Player - Music Segment

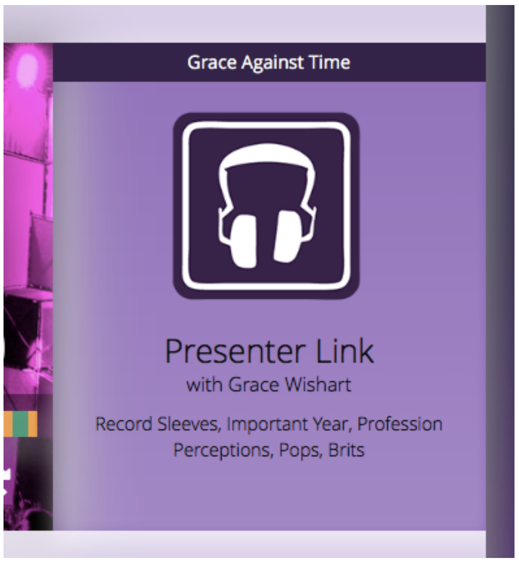
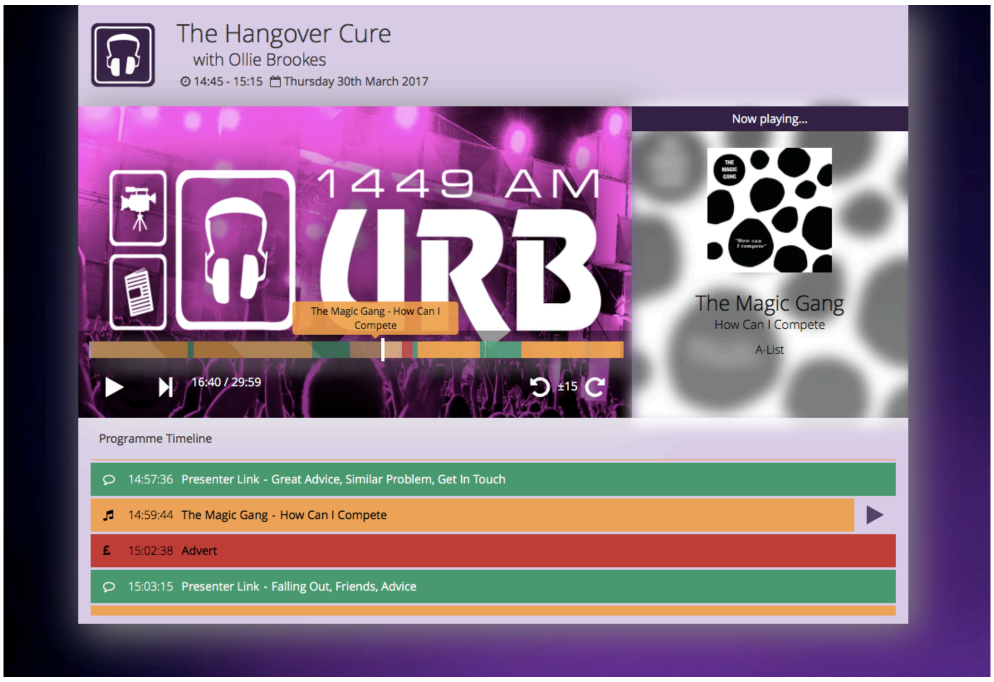

Grace Against Time
Presenter Link
with Grace Wishart
Record Sleeves, Important Year, Profession Perceptions, Pops, Brits

Fig. 5.15 Web Player - Speech Segment

UI elements work together to visualise a programme's segments and provide interactive controls, in order to create an easy to use web player in which content is readily accessible. Another notable implemented feature is segment rights management; the web player can disable certain segments if legally they are not allowed to be played. The segment will appear disabled (Figure 5.13), and any attempt to play it will result in the audio skipping to the next 'allowed' segment. This feature ensures stations can operate a radio listen again service within the legal boundaries.



14:57:36 Presenter Link - Great Advice, Similar Problem, Get In Touch
14:59:44 The Magic Gang - How Can I Compete
£ 15:02:38 Advert
15:03:15 Presenter Link - Falling Out, Friends, Advice

Fig. 5.16 Web Player

The web player is presented in full in figure 5.16. The player's HTML web page is included in appendix E.3 and the *Popcorn.js* 'segment' plugin is included in appendix E.4.

5.4 Summary

This chapter has given an overview of the final developed system and discussed the features implemented. Following the system architecture, defined in section 4.2, the chapter started by describing the *Event Collection* stage, highlighting the hardware to capture studio events and the software to record and store them. Subsequently the *Segmentation Analysis* stage was discussed, focussing on the methods to analyse the previously detected events and generate segments. The section also explained how data was extracted from the station database and third party information suppliers to produce segment metadata. Finally, the chapter concluded presenting the *Web Player*. The chapter concluded by discussing each component of the web player before presenting the *Web Player* in full in in figure 5.16.

To experience the web player, it is available online: **http://radio.chris.io**.

6 Evaluation

Having discussed the implementation, the next chapter evaluates the system which was produced. Although some formative evaluation has been carried out throughout the design and implementation stage, many of the recommendations were not able to be made due to insufficient resources at the time. Instead, two different summative methods proposed by Baxter et al. [73] will be evaluated. One will follow a criteria based assessment, where the system produced will be compared against the system requirements written in Section 3.1. The second method is a tutorial-based approach and involves conducting a user experiment to compare and contrast user behaviour between an existing web player and the newly developed web player.

6.1 Requirements Evaluation

Following a criteria based assessment, the system was analysed using requirements as the criteria. This was felt appropriate given the short nature of this project and the lack of resources to follow the full assessment criteria. The requirements were originally devised through numerous methods, with the aim to build an interactive radio listen again service that would provide listeners with the ability to locate and listen to specific content. Arguably this means if the requirements are satisfied, then the developed system should achieve its aim. Each requirement was asked "Was it implemented? Provide supporting comments if warranted." Appendix C contains the full analysis, but the following section will summarise the findings.

Table 6.1 lists the number of high, medium and low requirements and how many of them were successfully implemented. 80% of requirements were implemented, all high and medium priority requirements were met. Three requirements that were not met have been briefly outlined and the consequent impact on the solution discussed below.

Requirement Priority	**Number of Requirements**	**% Implemented**
High	5	100%
Medium	11	100%
Low	9	50%
All	*25*	*80%*

Table 6.1 Requirements Evaluation - Requirements Implemented

6.1.1 RadioCatchup Management System

The station management system was included as a requirement to fix and improve the output of the automated segmentation process (*D.4*). During initial testing, it was found that segment recognition was very accurate and satisfied the 90% data integrity requirement (Req. *D.7*). However, it was soon apparent that some of the segment metadata produced was not always accurate, especially the speech topic detection. At times, when the system was tested in situ in the radio studio, the metadata had to be manually edited. As there was no management system implemented, this required editing the raw data within the database and JSON files.

The lack of a suitable management system does raise concerns. Stations are less likely to implement and use this segmented system if they do not have control over the segments and metadata that is displayed in the player. This evaluation has identified that the initial requirement (*D.4*) should have been given a higher priority, which would have ensured resources were allocated to implementing this management system.

6.1.2 Segment Ranking & Podcast Generator

Although these were two separate requirements (Req. *D.2.4* & *D.3*), the podcast generator was dependent on segment ranking. The podcast generator had been planned to export the top n segments and combine them into a single audio file, but unfortunately segment ranking was found to be more challenging than expected. Analysis in Section 3.1.1 identified a simple trend for speech segments, the longer the duration the more likely it would be a 'best bit' and included in highlights or podcasts. However, this alone was not enough to develop an accurate ranking system. This feature was therefore halted, but it still has potential and could be implemented in a future product cycle. The omission of these features was not detriment to the developed system.

6.1.3 Summary

Overall the system produced has satisfied the majority of the requirements and importantly met all the high and medium requirements. The pitfall of this project is the lack of a management system, to enable stations to have control over the segments and metadata produced and displayed within the web player. Although this had not been implemented due to constraints and the requirement's priority, this has highlighted that potentially a stronger requirements analysis could have been employed during the requirements stage of the project.

6.2 Empirical Evaluation

Evaluating how successful the solution is based on requirements implemented can only reveal so much. Already we have found the initial requirements were not faultless, thus a second evaluation method follows. Using a tutorial-based assessment [73] a pragmatic evaluation of the usability of the system was possible. As the majority of the system is automated and has little user interaction, this evaluation concentrated on the web player.

6.2.1 Introduction

A user study was conducted to evaluate the performance of the web player, inspired by Carlier et al. [69] and Schoeffmann and Boeszoermenyi [74]. In this study the newly developed web player's performance was compared to a standard web player. To ensure a fair test was conducted, the standard player used the same interface as the new player, however several UI elements that displayed segments were disabled (segmented seekbar, timeline and current segment). This made the standard player look and work similar to the industry web players which were analysed in section 3.1.1. Participants were given a number of tasks, to locate content within a radio programme and they were timed on how long it took them. Each participant was given the opportunity to use both the segmented web player and the non-segmented player. The full experiment is detailed below, however first it is important to state our expectations and what this project's hypotheses were.

6.2.2 Hypotheses

Following on from the research questions, derived in section 1.3, a hypothesis-driven experiment was defined. The aim was to prove or disprove the following hypotheses:

HT.1 The segmented web player will be quicker to locate specific content than the non-segmented web player.

HT.2 The longer the programme duration, the longer it will take to locate specific content within it.

HT.3 Users prefer using the segmented web player rather than the non-segmented web player.

The hypotheses will be tested by collecting relevant and reliable data during a user experiment with feedback through questionnaires. The first two hypotheses will be measured based on time taken to complete a task, whereas the third hypothesis will be measured by a system usability scale (SUS) [75]. Feedback received in the questionnaires and observations made during the experiments will be used when discussing the hypotheses.

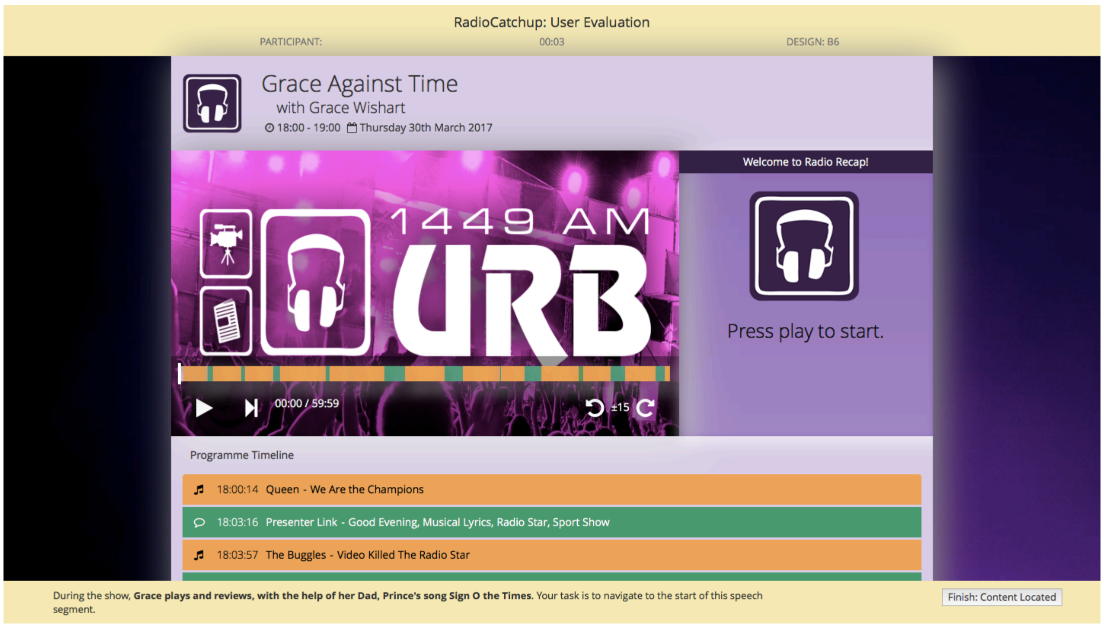
6.2.3 Method

Participants

Ten undergraduates (7 males and 3 females) studying various degrees at the University of Bath were recruited through adverts in the Computer Science and University Radio Bath members Facebook group. The participants mean age was 21.7 years and ranged from 21 to 23. Participation in this study was on a voluntary basis and no compensation was offered. All participants were naive to the purpose of the experiment.

Apparatus

An experiment web app was developed to guide participants through the experiment, reducing the need for interrupting the participant with verbal instructions. The web app was setup for a given participant and took them through the experiment task by task, pausing for questionnaires between players. During each task the app displayed the relevant web player and collected usage information while recording the time taken to complete the task (Figure 6.1). This information was then uploaded and stored confidentially in a Google Form. The app had no impact on web player user interaction or performance. The web app was run on a computer that each participant was provided with and a pair of headphones was offered to listen to the audio content.


RadioCatchup: User Evaluation

PARTICIPANT: 00:03 DESIGN: B6

Grace Against Time
with Grace Wishart
18:00 - 19:00 Thursday 30th March 2017

Welcome to Radio Recap!

Press play to start.

00:00 / 59:59 ±15

Programme Timeline

- 18:00:14 Queen - We Are the Champions
- 18:03:16 Presenter Link - Good Evening, Musical Lyrics, Radio Star, Sport Show
- 18:03:57 The Buggles - Video Killed The Radio Star

During the show, **Grace plays and reviews, with the help of her Dad, Prince's song Sign O the Times.** Your task is to navigate to the start of this speech segment.

Finish: Content Located

Fig. 6.1 Experiment - Web App

Materials

Consent Forms and Experiment Guide - Informed consent forms (Appendix D.1) were used containing information about the experiment and how it would be run. There was also a section for participants to state any requests they have for the experiment and reminded them they may withdraw from the study at any point. Furthermore a how-to guide was provided (Appendix D.2), to give participants exact instructions on how to use the evaluation web app and complete each task, however it did not give details on how to use the web player or where the content may be located.

Web Player Design - Two web players were used: the project's segmented player, and a control player with the segmentation function disabled to emulate a standard player. The reason for using the same player, just with the segmentation functionality disabled, was that it ensured any findings could be attributed to segmentation, rather than a change of design.

Radio Programmes and Content to Locate - The tasks (Appendix D.3) required participants to locate content within a number of radio programmes. The experiment used content from specialist music and entertainment programmes broadcast on University Radio Bath. The shows chosen for the tasks were decided based on station scheduling; only shows that had been broadcast after the segmentation system had been deployed were able to be used. Six shows from this list were used and permission was specifically sought from the production teams of these shows. All shows agreed the recordings may be used for the experiments. The content that each task asked participants to locate was determined so that there would be an even split across the programme, some content in the first half and others in the second half. The content was also picked to be unambiguous, so that it is clear to participants listening which part of the programme the task is referring to.

Web Player Questionnaire - The web player response questionnaire, hosted on a Google Form, (Appendix D.4) consisted of a system usability scale (SUS), a simple ten-item scale giving a quantititive view of subjective assessments of usability. Derived from Brooke et al. [75], eight out of the original ten of questions were included and asked on the questionnaire, with two additional questions (Q9 & Q10) used to evaluate the specific segmented nature of the web player. A free-text box was given at the end of the questionnaire to capture comments about the player.

Final Questionnaire - Another Google Form questionnaire was given to participants (Appendix D.5) which they responded to regarding their web player preferences, reliance on segment metadata and thoughts on future development possibilities.

Design

A within groups design was used to evaluate the web players. There were two independent variables (the web player used and the programme duration) and two dependent measures (the time taken by the participant to locate a specific piece of content and the SUS usability scale). However, it is important to note the dependent usability scale measure is only recorded against one independent variable. The SUS scale is measured within a questionnaire that is only given once per design, not after each task. As there were two different web player designs and we were conducting a within groups experiment, counterbalancing was required. Each participant was randomly allocated to either group one or group two. Group one used the non-segmented design for the first three tasks and the segmented design for the last three tasks, whereas group two used the segmented design for the first three tasks and the non-segmented design for the last three tasks. The experiment workflow is illustrated in Figure 6.2. The task order was not changed.

Two statistical tests were used: a two-way analysis of variance (ANOVA) and a Wilcoxon signed rank test. The two-way ANOVA examines the influence of the two independent variables on the continuous dependent variable, time taken to locate content. As mentioned, it would not be suitable to include the SUS usability scale in this statistical analysis as it was not influenced by the task's programme duration. Therefore, the non-parametric Wilcoxon signed rank test was used. We compared the two usability scores between participants and assessed whether their mean ranks differed. The rejection level for both statistical tests was set at $p = .05$.

Finally qualitative data was collected through the free-text fields on the questionnaires, the evaluator's observations, and the verbal debrief at the end of the experiment with

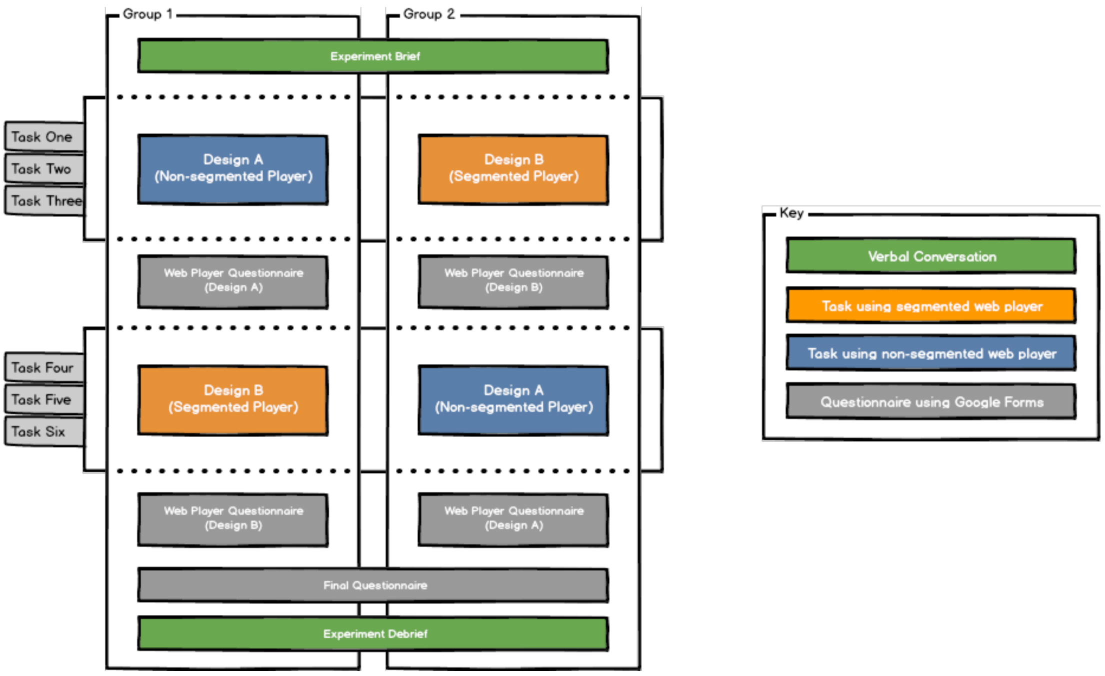

Group 1
Group 2
Experiment Brief
Task One
Task Two
Task Three
Design A (Non-segmented Player)
Design B (Segmented Player)
Web Player Questionnaire (Design A)
Web Player Questionnaire (Design B)
Task Four
Task Five
Task Six
Design B (Segmented Player)
Design A (Non-segmented Player)
Web Player Questionnaire (Design B)
Web Player Questionnaire (Design A)
Final Questionnaire
Experiment Debrief
Key
Verbal Conversation
Task using segmented web player
Task using non-segmented web player
Questionnaire using Google Forms

Fig. 6.2 Experiment - Workflow

the participant. Thematic analysis, devised by Braun and Clarke [76], was utilised to highlight patterns and identify "themes" from the qualitative data. For this analysis a 'bottom up' approach was followed, meaning that rather than looking for evidence of themes determined at the start, related points were sought out in order to make these the themes found.

Procedure

The experiment took place under lab conditions within a room that was reasonably distraction-free. Participants were first briefed on the nature of the experiment, highlighting there will be six tasks to complete and 3 questionnaires to fill in; one after each design and one at the end. After the brief, participants were given a consent form and asked to read and sign it. The participants were then given a computer and headphones to use. The computer had a desktop screen recorder recording and the evaluation web app ready to go with the relevant task and web player open, depending on the participant's experiment group. Figure 6.3 illustrates the experiment workstation setup, showing a participant, computer, headphones and materials needed to complete the experiment.

The evaluator then took a step back and left the participant to read the full task brief. Once they had read it, they were ready to proceed and the participant clicked the 'Start Task' button. During each task the evaluator made a note of the search strategy employed by each participant and any other interesting observations. The participant had 5 minutes to complete each task. Each task was completed by the participant clicking 'Finish: Content Located' and the evaluation app then led the participant to

Fig. 6.3 Experiment - Workstation Setup

the next task brief. If participants went over 5 minutes, the evaluator was required to step in and manually move the participant to the next task. The same process was followed again in the next task, with the evaluator overseeing and taking notes, but most importantly not interrupting or distracting the participant. After the third task the participant was required to complete the 'Web Player Questionnaire'. There was no action for the evaluator during the questionnaire.

After the questionnaire, another three tasks were completed by the participant with the other design. The process was identical to the first three tasks, with the evaluator app leading the participant through and the evaluator noting down observations. Finally after the second set of three tasks, the 'Web Player Questionnaire' and 'Final Questionnaire' were completed by the participant. Again, no action was required by the evaluator during the questionnaires. After completing the tasks and questionnaires, the participants were debriefed and asked for verbal feedback on the web players they used and the tasks they had just completed. They were also given given a chance to ask any final questions.

6.2.4 Results

The user study revealed a number of interesting findings. The raw task results and questionnaire responses can be found in Appendix D.7, however this section will briefly highlight the findings and results from the tasks, statistical analysis and qualitative data. The statistical analysis aims to prove or disprove our hypotheses, defined in section 6.2.2.

Results Adjustment

All 10 participants attempted six tasks and three surveys, however five tasks were not completed within the allotted five minutes and 1 task was done incorrectly. All surveys were correctly filled in. It was decided that participants who did not complete the task within 5 minutes, would have their time capped at 5 minutes. Although arguably this may make the task seem easier, as the results appear that a participant can complete it within 5 minutes, it was felt that the maximum score would already penalise the task enough. Whereas, had the results been invalidated, then the mean score would be lower and that would be indicative that the task was easier, even though many participants could not complete it. In the case of this experiment, all five overrun tasks were on Design A with a sixty minute programme (task three or task six, depending on experiment group).

Table 6.2 Participant 3 - Adjusted Task Result

Task	**Participant Score**	**Mean**	**Difference**
Player A - 10 Mins	64	73	0.88
Player A - 60 Mins	132	238	0.55
		Average Difference	**0.71**

Correcting for the incorrect task was more complicated. The task was incorrectly done, using Design A with the 60 minute programme, by Participant 3. With a counterbalanced sample, excluding the incorrect task score would result in the exclusion of the participant and this in turn would require a counterbalancing participant's data removed as well. Following this action would result in 20% of the collected data being removed for a single datapoint (1.6% of the task completion time data) being incorrect. It would also require a process to decide which counterbalancing participant would be removed and this was thought to be more problematic than the erroneous result. To overcome this issue it was decided to analyse the participant's other task results. Table 6.2 shows the participants average difference from the mean during both successfully completed tasks using design A (non-segmented player). The average difference of 0.71 can be applied to design A 60 minute programme task: to give the participants adjusted task completion time $= 102 \times 0.71 = 73$.

Hypothesis 1 and Hypothesis 2

HT.1 The segmented web player will be quicker to locate specific content than the non-segmented web player.

HT.2 The longer the programme duration, the longer it will take to locate specific content within it.

With the adjusted participant score, Figure 6.4 illustrates the mean time, taken by each experiment group and overall group, to complete the tasks. Trend lines have been added to interpolate between the set of known task completion time values. The graph shows that the segmented player was faster to locate content than the non-segmented player. In addition it illustrates the seemingly exponential trend between time to locate content and length of the audio with the non-segmented player, compared to the flat lined segmented-player.

A two way 2 (*web player design*: segmented or non-segmented) × 3 (*programme duration*: 10 mins vs. 30 mins vs. 60 mins) repeated measures ANOVA was conducted on the *task completion time*. This revealed a significant main effect of *web player design*, $F_{1,9} = 125.0, p < .001$, indicating that the *task completion time* was significantly higher when using the non-segmented player ($M = 136.9, SD = 9.70$) over the segmented player ($M = 40.33, SD = 3.49$).

There was also a significant main effect of *programme duration*, $F_{1,9} = 26.60, p < .001$, indicating that *task competition time* was statistically higher the longer the *programme duration*, 60 minute programme ($M = 137.9, SD = 14.5$) vs. 10 minute programme ($M = 58.15, SD = 4.61$).

The *web player design* × *programme duration* interaction was also statistically significant, $F_{1,9} = 30.43, p < .001$, indicating that the difference in *task completion time* due to *programme duration* was present in the non-segmented design but not the segmented design.

All three effects were statistically significant at the .05 significance level and Figure 6.4 supports these findings. Therefore it is sufficient to say that Hypothesis *HT.1* and *HT.2* have been supported. The full statistical analysis can be found in appendix D.7.2

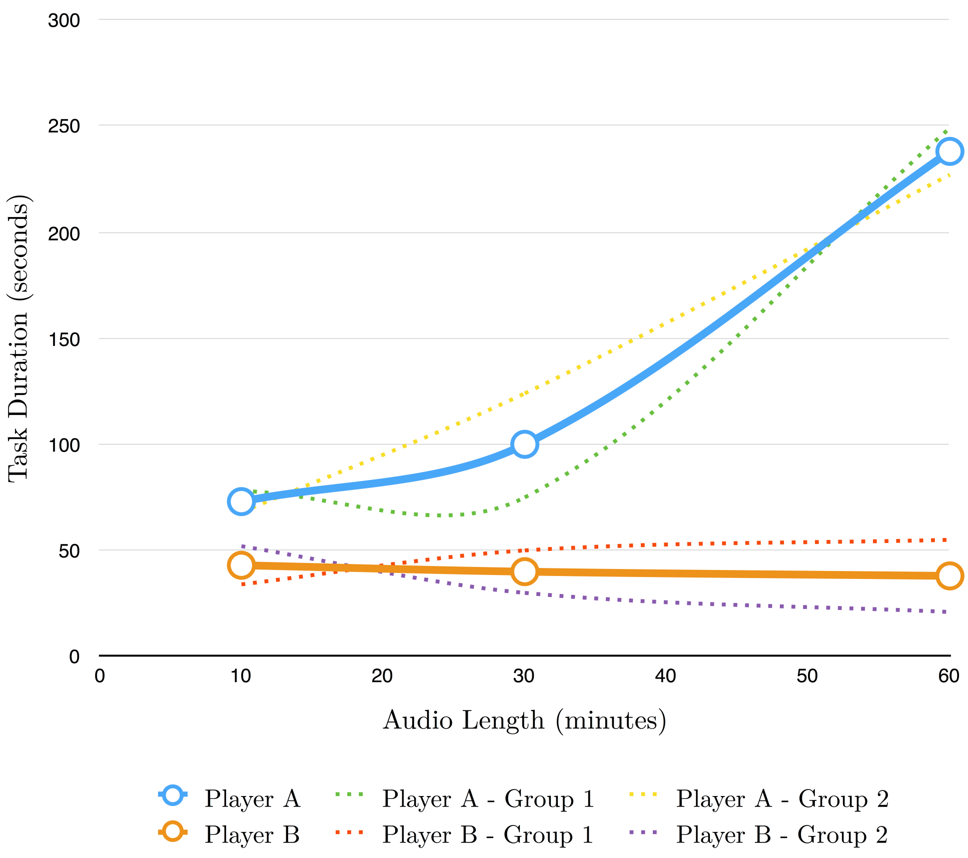


Fig. 6.4 Time taken to locate a specific speech segment compared with the length of the audio

Hypothesis 3

HT.3 Users prefer using the segmented web player rather than the non-segmented web player.

At the end of the three tasks for each design, participants were asked to fill out a questionnaire with some standardised questions, following the System Usability Score (SUS) [75]. A Wilcoxon signed-rank test showed that using a segmented web player ($M = 90.8, SD = 4.02$) elicited a statistically significant improvement in usability scores, $Z = -2.807, p = 0.005$, over the non-segmented web player ($M = 63.2, SD = 13.5$). This is sufficient to say that Hypothesis *HT.3* has been proved. Full results can be found in appendix D.6.2 & D.7.3.

Questionnaires & Thematic Analysis

Participants responded to a total of three questionnaires, based on segmented player, non-segmented player and final thoughts. The responses can be found in Appendix D.6. All participants indicated that they preferred the segmented player, when asked "Which web player would you rather use?" A follow on qualitative question asked why they had made that choice, using these results with thematic analysis the trends were identified. The themes identified were 'easy to use', 'breakdown of show content' and 'new listening potentials'.

Easy to Use - An overwhelming number of responses described the player as being "easy to learn and understand functionality". The specific functionality mode participants were referring to was the ability to find and skip specific segments. Participant 1 stated that it was "so much easier to know where talk segments started/ended without skipping through the songs".

Breakdown of Show Content - Many participants found the web player's breakdown of show content to be extremely valuable. Participant 4 "prefered design B [segmented web player] because it meant that [they] could see what [they were] looking for and where it came in the programme relative to other segments". Other participants valued the breakdown as made the searching for content extremely quick, removing the guesswork.

New Listening Potential - This theme encapsulates a number of the participants' responses where they indicated they would be able to listen to radio programmes in new ways, with this web player. Participant 7 was excited by the potential that this could be a music discovery tool, whereas participant 8 would use it to test the show by "flick[ing] through a show to see if it is worth listening to the whole thing or not". A common point made though was the ability to jump and skip irrelevant or uninteresting parts.

These results and the findings from the requirements evaluation will be discussed in the following section. The 'Future Development' questions and responses, put to participants in the final questionnaire, will be discussed later in section 7.3.

Participant's Search Strategies

During the experiments, evaluators monitored participants, watching how they used the player and noting search strategies employed to locate the content. Participants generally fitted into one of two categories; a slow methodical user, and a fast impatient user. The former tended to go through the programme in chronological order until the segment was located, while the latter would skip randomly until the segment is detected. No correlation was found between the search strategy used and the time taken to locate content.

6.3 Discussion

The requirements evaluation and user experiment have analysed the system developed and led to numerous findings which will be discussed in this section. The findings have been split into two categories: *segmentation process* and *web player*.

6.3.1 Segmentation Process

Segment Detection

The requirements evaluation identified that automatic programme segmentation had been implemented, however the user study did not specifically evaluate the process and results. Nonetheless, the produced segments were analysed indirectly as the web player, which uses the generated segments, was analysed in the user study.

In the user experiment, the automated segment processor was used to extract segments from each of the radio programmes for the 6 user tasks. The segments for each task were manually checked to ensure they were accurate for the experiment and as a result no changes were made. Each segment had been correctly detected and the segment metadata was sufficient. Unfortunately, apart from this manual check, no conclusive analysis had been conducted. As a result the accuracy of the segments produced by the system cannot be validated.

Segment Metadata

During the study participants were asked the importance of the metadata that accompanies a segment. The user study found all participants thought segment metadata was important or very important, indicating that identifying segments is not enough as classification and segment metadata are also vital. Consider an individual wanting to listen back to an interview in a programme, without segment metadata they would be required to manually seek each segment. Although the segments may be listed and a start and end time displayed, the listener would still be expected to search each segment and they would struggle to gain a high-level understanding of the programme quickly.

The solution developed incorporated metadata from music segments and speech segments using various information suppliers. For speech segments it was particularly beneficial taking advantage of the various improvements to speech recognition and natural language processing. Many users found this particularly useful and even credited the keywords extracted as the reason for finding the content in the user study so quickly.

6.3.2 Web Player

Locating Content

The hypotheses (*HT.1* & *HT.2*) were confirmed through statistical analysis and showed that the speed to locate content within a programme was dependent on the web player design and the programme duration. It was interesting to note that when using the segmented player the programme duration had little effect on the time taken to locate content. Instead, the mean task duration marginally decreased when searching longer programmes, which may be due to the 'practice effect'. Unlike the non-segmented player, which is an industry standard web player that many participants would have previously used, the segmented player had numerous unfamiliar features. As the tasks were ordered by programme duration, starting with 10 minutes and ending with 60 minutes, users may have learnt how to use the segmented player by the later tasks and therefore became quicker at locating content despite the increased duration.

Player Usability

Studies found that the segmented player was more usable, according to the SUS usability scale, and that participants preferred it over the non-segmented design, proving hypothesis *HT.3*. Unsurprisingly, as the non-segmented players are standard practice and participants would have been familiar using them, the non-segmented player still received a positive SUS score. Interesting, when analysing the individual SUS questions, it was found that the segmented player scored much higher on usability questions regarding finding content, confidence using the player and appeal of player. Contrastingly, the scores on questions regarding ease of use and need for technical support were similar across the segmented and non-segmented player. Considering the SUS scores indicate the segmented player adds functionality while not reducing operability, this is extremely promising for the success of the player. For the web player to be commercially viable it must be intuitive and easy to use, otherwise users may struggle, loose interest and leave the service, all which reflect badly on the radio station or catchup service provider.

6.3.3 Summary

Within this chapter the developed solution has been evaluated using a requirements evaluation and a user study. Based upon the research questions in section 1.3, three hypotheses were defined and later proved significant using statistical analysis on the user experiment. Conclusive results indicate that the segmented player was quicker to navigate to specific content, more user friendly and preferred by participants. Thus the developed system can be deemed successful.

7 Conclusion

This final chapter discusses the contributions this dissertation has made, analyses the limitations of the study and suggests possible improvements. Areas for future research are also proposed, illustrating a number of interesting topics that could be investigated.

In chapter 1, a number of objectives were devised which would be used to determine the success of the project. The objectives have been reproduced below and evidence is given illustrating how they have been met.

- *Research the current academic literature concerning audio content analysis and audio retrieval* - Chapter 2 explored the state of the art and discussed the large amount of literature that has been published on the subject of audio content analysis and retrieval.
- *Investigate the current technologies available and review similar existing solutions* - Current technologies are discussed within the literature and technology review in chapter 2 and a number of existing solutions, such as *BBC iPlayer Radio* and *Radioplayer* are analysed in chapter 3.
- *Devise a requirements specification from a wide breadth of sources* - Chapter 3 discusses requirements gathered from a wide breadth of sources with existing solutions analysed, user research undertaken and use cases derived. The chapter concludes with the full requirements specification.
- *Design a system that is capable of segmenting a radio programme and displaying the segments against the audio within a web player* - Chapter 4 reviewed the design decisions taken and presented the system designed, illustrated through a system architecture, data schemas and user interface wireframes.
- *Following the design, develop an automated interactive segment-based radio listen again service* - Chapter 5 discussed the implementation of the system and presented a working prototype of the automated interactive segment-based listen again service.
- *Evaluate the success of the developed system by conducting a user study and gathering user feedback* - The developed system was evaluated using two methods, including a user study, in chapter 6. The developed system was proved to be a success.

Chapter 1 also considered 3 research questions, which were used to provide direction to the dissertation. They have been reproduced below and their findings summarised:

1. *Is it possible to automatically segment a radio programme?* - Yes, this dissertation has developed a system to automatically segment a radio programme based on studio events. (Chapter 5)
2. *Is it possible to develop a segmented web player that is easy and intuitive to use?* - Yes, a web player was developed in chapter 5 and it was demonstrated to be easy and intuitive in the user study in chapter 6.
3. *Is it possible to develop a segmented web player that is quick for listeners to locate content within a radio programme?* - Yes, the empirical evaluation in chapter 6 found the segmented web player to be far quicker to locate content, than a non-segmented player.

7.1 Key Contributions

This dissertation has made a number of key contributions, both in terms of systems produced and research conducted.

7.1.1 System Contributions

A number of key system contributions have been made to the field. To the best of our knowledge, this is the first complete implementation of an automated segment-based radio listen again service. The scale of this contribution can be recognised by listing the individual contributions that have been made:

- Proposed a system architecture that automatically detects studio events, analyses the events to generate segments and exports the segments to a radio catchup web player.
- Proposed a segmented web player user interface that utilises existing web player design standards to reduce the learning complexity to users.
- Developed hardware to capture studio events and software to record and store them.
- Developed a method to analyse the studio detected events and generate segments, using data extracted from the station database and third party information suppliers to produce segment metadata.
- Developed an interactive segment-based web player that provides the user the ability to listen, navigate and interpret the content within a radio programme.

7.1.2 Research Contributions

Research contribution to community

This dissertation has also made many research contributions to the field:

- Provided a thorough literature and technology review summarising the field and technologies available. To date, a considerable body of research has sought to analyse audio based on the physical audio properties, however this dissertation has investigated analysing audio based on studio events, an area struggling from limited research. For this reason it can be concluded that this dissertation provides a major contribution to the field and supports the initial research of ORPHEUS, the European research group dedicated to improving the management of audio content to create new user experiences [71].
- Existing listen again solutions were analysed and designs reviewed. Common design traits were identified and recommended for future web players, to reduce the learning required by users of a new segment-based player.
- User research was conducted to gauge listening habits between live and catchup radio. The findings were similar to the U.K. RAJAR research [23], suggesting their validity, however the research conducted in this dissertation went into more detail, collecting reasons for listening to live and catchup radio.
- Designed and conducted a user study to evaluate the usability of a segmented web player and the affect on locating content. Our study indicated that users prefer a segmented player and that it brought significant improvements, both in terms of speed to locate content and providing users with more functionality, such as identifying now playing segments.

This methods can be used for future research The research conducted has been extremely promising and provides a significant contribution to the field.

7.2 Limitations of the Study

The user study and requirements evaluation provided an insight into the success of the project, however some limitations were identified. The project had two challenges; automatically segmenting a radio programme and building a web player to view them. Although the web player was evaluated thoroughly through the user study, less consideration was given to analysing the segmentation process. The web player will only be as good as the data it is provided, meaning that if the segments generated are not correct, then the player would be redundant. A simple requirements evaluation concluded segments were successfully generated, however it would have been insightful to run a study comparing the segments generated automatically and manually by a member of the radio station.

Additionally, no study has been conducted on the appeal and usefulness of the developed system to radio stations. Nonetheless both the requirements evaluation and user study revealed limitations. The requirements evaluation highlighted a key potential problem: a segment metadata management system has not been developed. It was hypothesised that this omission may restrict the appeal to stations, as they would have no manual override to correct or add segment metadata. As the segmented catchup system requires installation within a radio station, it needs station participation. Ensuring that the system would be suitable and appealing to stations would produce an important analysis.

Furthermore, focusing on the user study, three further limitations were revealed:

Participants

The study could have benefited from more participants and from a wider demographic. The ten participants involved in the study were all young adults studying at the University of Bath. There is a possibility, although unlikely, that the segmented system favoured young university educated individuals. The small participant size could make the results inconclusive, thus further testing should be conducted with a larger group. However, considering the high probability significance of the results, a larger group would not be expected to affect the conclusions.

Programmes Assessed

The experiment tasks required participants to locate specific content within a radio programme. The programmes used were chosen based on station scheduling and the content to locate was decided to ensure an even split across the programme and that it was unambiguous. Table 7.1 lists the mean task completion times between groups. The groups swapped player designs, but the audio order was not changed. Therefore, if the programmes and difficulty to locate the content was similar, the difference between task completion means for each group would be expected to be similar. While every effort was made to ensure task difficulty was similar, the results show that it was not achieved perfectly, as analysis of table 7.1 reveals the difficultly of the two 30 minute programmes may have differed.

		Player A			Player B		
Participant	**Group**	**Task - 10 Min**	**Task - 30 Min**	**Task - 60 Min**	**Task - 10 Min**	**Task - 30 Min**	**Task - 60 Min**
Average	*1*	*78*	*75*	*249*	*34*	*50*	*55*
Average	*2*	*68*	*124*	*227*	*52*	*30*	*21*

Table 7.1 Comparison between experiment groups and time taken to complete each task.

Task Permutations

The tasks were structured and carried out so that each participant located content in a 10 minute, then 30 minute and finally 60 minute programme using one of the web player designs. The process was then repeated with the other web player design.

While counterbalancing was applied, so that two different groups used the players in different orders, the task order remained the same. It is unclear from the results whether finding content within a shorter programme first helped the participants to learn the web player, which the experience was then used on the longer programme tasks. Had another group performed the tasks in reverse order, this could have been analysed.

7.3 Future Work

The solution presented in this dissertation leads to many interesting and exciting opportunities, which in turn suggests specific directions for future research. This project has laid the foundations that now allows content within shows to be searchable, programme personalisation to be possible and listen again services to be vastly more interactive.

It is recommended that further research be undertaken to take advantage of these new areas. As with the rest of the dissertation, the recommendations are naturally grouped into two sections: *segmentation process* and *web player*.

7.3.1 Segmentation Process

The research and solutions in this paper presented a basic approach to segmentation, however more research is needed to better understand whether the segments being produced are accurate, how they could be more accurate and how additional segments could be identified.

Segmentation Methods

The literature review researched a number of different segmentation methods, using both audio element discovery and studio element discovery. This project focussed on using studio elements to detect segments, however many of the audio element segmentation methods also proved popular. Further work could establish whether this is the most accurate segmentation method for the radio studio environment, and possibly discover if other methods would be complementary or supplementary.

Further to this, some audio element segmentation methods use machine learning to analyse what certain content may be, based on training data and similarities between audio features. This approach could be extended to studio elements, where a large number of event collectors could be analysed to identify trends, such as detecting segments or even studio interaction behaviour. Additionally, new event collectors could be investigated via analysis of studio equipment to ascertain whether useful event information could be extracted.

Information Suppliers

The user study identified the importance of segment metadata to users, however the requirements specification prioritised the segment information provided as low (Req. *D.2.1* & *D.2.3*) and music as medium priority (Req. *D.2.2*). Further studies should research segment classification and means to retrieve metadata. In this project, information suppliers were used to provide additional information about segments, such as album artwork or speech topic detection. Utilising other third party services could provide users with more information to describe or further inform them about segments. Simple extensions could include providing news segments with associated news articles on the web for further reading or listening.

Segment Management System

A segment management system was listed within the requirements specification, however due to various reasons this was not implemented. While further work should implement this functionality, it would be interesting to assess the popularity and use of it. Theoretically, the better the segmentation and classification processes, the less likely manual interaction should be needed. Determining if this hypothesis is correct and by extrapolation whether manual interaction will always be required during the segmentation and information collection stage, would be an interesting future research area.

7.3.2 Web Player

The web player was used as a content retrieval method to play and display the segments extracted during the segmentation process. As discussed in the literature review (section 2.3), other retrieval methods exist and these could be explored with segmented content, however focussing on the web player there already exists a large number of research recommendations, so only these will be discussed below.

Development Opportunities

During the user study in section 6.2, a number of questions were asked to participants regarding what future features they would be interested in. The responses are summarised below and could be used to guide future development. The full responses can be found in appendix D.5.

- *Live Segmentation* (100% participants interested) - Providing a web player that displays current and past segments when listening live to a radio programme. This feature was extremely popular with many participants highlighting the benefit if joining during the middle of a programme. A listener could catchup on missed segments and then return back to the live show.
- *Personalised Programming* (80% participants interested) - Providing a web player that automatically gets segments from a variety of programmes and creates a personalised radio programme for you. Interest was still high for this development opportunity. Many participants expressed concern over the accuracy of recommending content to users, however there is a large amount of research available in this field, which would help to develop a suitable recommendation system. Participants valued the mix of speech and music segments from various shows, and some participants suggested this could rival their listenership to music streaming

services. Considering the research conducted in section 1.3, which concluded that music streaming services are far more popular than radio catchup services, this could lead to a number of new listeners.

- *Catchup TV* (60% participants interested) - Providing a similar segmented web player for catchup television. Participants did not feel as passionate about this development prospect. Some worried the segments would reveal spoilers, while others thought it would not work with a lot of programming. It could be applied to news and sports programming which inherently has highlights and distinct content, however for chronological programmes, such as drams, there would be little benefit.

User Behaviour

The user study's focus was analysing how participants locate content within a programme and the time taken, however this did not cover user behaviour when listening to a programme. Further investigation and experimentation into general user behaviour, when listening to programmes, is strongly recommended. The findings should in turn feedback into the design process and be used to improve the player and the associated user experience.

New Listening Potential

Another recommendation involves studying the new listening potential, available with a segmented radio programme. The original user study investigated interest in *personalised programming* and received some positive responses, however this could be explored in more detail. It would be interesting to explore how segments can be shared, especially through social media, while retaining integrity between the segment and its parent programme to allow listeners to follow clips back to their source.

7.4 Concluding Remarks

This dissertation has developed *Radio Catchup*, an automated solution to segment radio programmes and provide an interactive radio listen again service. To the best of our knowledge, this is the first complete implementation of such a system, which makes the positive results from user experiments even more promising. The solution presented significantly extends the capabilities of current listen again services and could be commercialised in the future. Additionally a number of key contributions have been made, both in terms of systems developed and research conducted. Notably the contribution to the European research group ORPHEUS and the interest received by broadcasters, such as the BBC, during presentations of *Radio Catchup*.

To experience the web player, it is available online: http://radio.chris.io.

Bibliography

[1] L. Symones. "'2LO calling'". In: *IEE Review* 44.4 (July 1, 1998), pp. 178–182. ISSN: 0953-5683. DOI: 10.1049/ir:19980413.

[2] George Gascoigne G. Blake and Repr. *History of radio telegraphy and telephony.* New York: Ayer Co Pub, Aug. 1976. ISBN: 9780405060342.

[3] J.E. Brittain. "Reginald A. Fessenden and the origins of radio [Scanning the Past]". In: *Proceedings of the IEEE* 84.12 (Dec. 1996), p. 1852. ISSN: 0018-9219. DOI: 10.1109/jproc.1996.546441.

[4] John V. L. Hogan. *The Outline of Radio.* Boston : Little, Brown, and Co., 1923. URL: https://archive.org/details/outlineofradio01hoga (visited on 10/27/2016).

[5] Keith Geddes. *Broadcasting in Britain, 1922-1972: A brief account of its engineering aspects.* H.M. Stationery Off., 1972. URL: https://books.google.co.uk/books?id=bL8MAQAAIAAJ (visited on 10/27/2016).

[6] J. Stott. "Digital radio Mondiale: Key technical features". In: *Electronics & Communication Engineering Journal* 14.1 (Feb. 1, 2002), pp. 4–14. ISSN: 0954-0695. DOI: 10.1049/ecej:20020101.

[7] F. Conrad. "Short-wave radio broadcasting". In: *Proceedings of the IRE* 12.6 (Dec. 1924), pp. 723–738. ISSN: 0096-8390. DOI: 10.1109/jrproc.1924.220005.

[8] A. Pinkerton and K. Dodds. "Radio geopolitics: Broadcasting, listening and the struggle for acoustic spaces". In: *Progress in Human Geography* 33.1 (Feb. 1, 2009), pp. 10–27. ISSN: 0309-1325. DOI: 10.1177/0309132508090978.

[9] BBC. *BBC archive - the BBC sound archive.* Aug. 28, 2009. URL: http://www.bbc.co.uk/archive/sound%5C_archive.shtml?chapter=2 (visited on 11/23/2016).

[10] British Library - Sound Collection. *Radio broadcast recordings.* The British Library. Jan. 21, 2015. URL: http://www.bl.uk/collection-guides/radio-broadcast-recordings (visited on 11/23/2016).

[11] Yves Raimond and Chris Lowis. "Automated interlinking of speech radio archives". In: (). URL: http://events.linkeddata.org/ldow2012/papers/ldow2012-paper-11.pdf (visited on 10/31/2016).

[12] Rui Cai, Lie Lu, and Alan Hanjalic. "Unsupervised content discovery in composite audio". In: *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05* (2005). DOI: 10.1145/1101149.1101292. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/unsupervisedaudiodiscovery%5C_mm05.pdf (visited on 11/22/2016).

[13] Lie Lu and A. Hanjalic. "Text-like segmentation of general audio for content-based retrieval". In: *IEEE Transactions on Multimedia* 11.4 (June 2009), pp. 658–669. ISSN: 1520-9210. DOI: 10.1109/tmm.2009.2017607.

[14] M. Lycke, M. Matton, and L. Overmeire. "An automated tagging approach to improve search and retrieval in a radio archive". In: *SMPTE Motion Imaging Journal* 124.8 (Nov. 2015), pp. 25–32. ISSN: 1545-0279. DOI: 10.5594/j18635.

[15] Y. Raimond, C. Lowis, R. Hodgson, and D. Tinley. "Automated Metadata enrichment of large speech radio archives". In: *SMPTE Motion Imaging Journal* 123.1 (Jan. 2014), pp. 35–41. ISSN: 1545-0279. DOI: 10.5594/j18370xy.

[16] Steven McClung and Kristine Johnson. "Examining the motives of Podcast users". In: *Journal of Radio & Audio Media* 17.1 (May 6, 2010), pp. 82–95. ISSN: 1937-6529. DOI: 10.1080/19376521003719391.

[17] Gianfranco Nencioni, Nishanth Sastry, Gareth Tyson, Vijay Badrinarayanan, Dmytro Karamshuk, Jigna Chandaria, and Jon Crowcroft. "SCORE: Exploiting global broadcasts to create Offline personal channels for on-demand access". In: *IEEE/ACM Transactions on Networking* 24.4 (Aug. 2016), pp. 2429–2442. ISSN: 1063-6692. DOI: 10.1109/tnet.2015.2456186.

[18] Mimmi Andersson. *Monthly Performance Pack - April, May & June 2016*. Tech. rep. July 26, 2016. URL: http://downloads.bbc.co.uk/aboutthebbc/insidethebbc/mediacentre/iplayer/performancepackaprmayjun2016.pdf (visited on 11/23/2016).

[19] Harold D Lasswell. "The structure and function of communication in society". In: *The communication of ideas* 37 (1948), pp. 215–228.

[20] Mette Skov and Marianne Lykke. "Unlocking radio broadcasts". In: (Aug. 21, 2012), pp. 298–301. DOI: 10.1145/2362724.2362779. URL: http://dl.acm.org/citation.cfm?id=2362779%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[21] Tobias Lauer and Wolfgang Hürst. "Audio-based methods for navigating and browsing educational multimedia documents". In: (Sept. 28, 2007), pp. 123–124. DOI: 10.1145/1290144.1290166. URL: http://dl.acm.org/citation.cfm?id=1290166%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[22] RAJAR. *MIDAS Measurement of Internet Delivered Audio Services*. Tech. rep. Oct. 2015. URL: http://www.rajar.co.uk/docs/news/MIDASAutumn2015LFFinal.pdf (visited on 11/23/2016).

[23] RAJAR. *MIDAS Measurement of Internet Delivered Audio Services*. Tech. rep. Oct. 2016. URL: http://www.rajar.co.uk/docs/news/MIDAS%5C_Autumn%5C_2016.pdf (visited on 11/23/2016).

[24] Martha Larson and Joachim Köhler. "Structured audio player". In: (May 30, 2007), pp. 268–273. URL: http://dl.acm.org/citation.cfm?id=1931416%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[25] Lie Lu and Alan Hanjalic. "Audio Keywords discovery for text-like audio content analysis and retrieval". In: *IEEE Transactions on Multimedia* 10.1 (Jan. 2008), pp. 74–85. ISSN: 1520-9210. DOI: 10.1109/tmm.2007.911304.

[26] G. Tzanetakis and F. Cook. "A framework for audio analysis based on classification and temporal segmentation". In: *Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium* (1999). DOI: 10.1109/eurmic.1999.794763.

[27] Silvia Pfeiffer, Stephan Fischer, and Wolfgang Effelsberg. "Automatic audio content analysis". In: (Jan. 2, 1997), pp. 21–30. DOI: 10.1145/244130.244139. URL: http://dl.acm.org/citation.cfm?id=244139%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[28] Martin Cooke. *Modelling Auditory processing and Organisation*. Cambridge University Press, Feb. 17, 2005. URL: https://books.google.co.uk/books?id=nP4WEcEByVcC (visited on 11/20/2016).

[29] D. Howard and J. Angus. *Acoustics and Psychoacoustics*. Taylor & Francis, 2013. ISBN: 9781136121586. URL: https://books.google.co.uk/books?id=e8cqBgAAQBAJ.

[30] Richard Parncutt. *Harmony: A Psychoacoustical Approach*. 1st ed. Springer-Verlag Berlin An, 2012.

[31] Juan G Roederer. *Introduction to the physics and psychophysics of music.* 2nd ed. New York: Springer-Verlag, 1975. 2d ed, May 17, 1978. ISBN: 9780387901169.

[32] Randall J. LeVeque, Charles S. Peskin, and Peter D. Lax. "Solution of a two-dimensional cochlea model using transform techniques." In: *SIAM Journal on Applied Mathematics* 45.3 (June 1985), pp. 450–464. ISSN: 0036-1399.

[33] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. "Unsupervised feature learning for audio classification using convolutional deep belief networks". In: *Advances in Neural Information Processing Systems 22.* Ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta. Curran Associates, Inc., 2009, pp. 1096–1104. URL: http://papers.nips.cc/paper/3674-unsupervised-feature-learning-for-audio-classification-using-convolutional-deep-belief-networks.pdf.

[34] Yin-Fu Huang and Lien-Hung Tung. "Semantic scene detection system for baseball videos based on the MPEG-7 specification". In: (Mar. 22, 2010), pp. 941–947. DOI: 10.1145/1774088.1774285. URL: http://dl.acm.org/citation.cfm?id=1774285%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[35] Lie Lu, Rui Cai, and A. Hanjalic. "Towards A unified framework for content-based audio analysis". In: *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* (2005). DOI: 10.1109/icassp.2005.1415593.

[36] V. Barbosa, T. Pellegrini, M. Bugalho, and I. Trancoso. "Browsing videos by automatically detected audio events". In: *2011 IEEE EUROCON - International Conference on Computer as a Tool* (Apr. 2011). DOI: 10.1109/eurocon.2011.5929358.

[37] Chris J C Burges, John C Platt, and Jonathan Goldstein. "Identifying audio clips with RARE". In: (Feb. 11, 2003), pp. 444–445. DOI: 10.1145/957013.957104. URL: http://dl.acm.org/citation.cfm?id=957104%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[38] Theodoros Theodorou, Iosif Mporas, and Nikos Fakotakis. "An overview of automatic audio segmentation". In: *International Journal of Information Technology and Computer Science* 6.11 (Oct. 8, 2014), pp. 1–9. ISSN: 2074-9007. DOI: 10.5815/ijitcs.2014.11.01.

[39] RG Bachu, S Kopparthi, B Adapa, and BD Barkana. "Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal". In: *American Society for Engineering Education (ASEE) Zone Conference Proceedings* (2008), pp. 1–7.

[40] Michael Lutter. "Mel-Frequency Cepstral coefficients". In: (Nov. 25, 2014). URL: http://recognize-speech.com/feature-extraction/mfcc (visited on 11/24/2016).

[41] BBC. *Future Content Experiences: The First Steps For Object-Based Broadcasting.* Mar. 13, 2015. URL: http://www.bbc.co.uk/rd/blog/2015-03-future-content-experiences-the-first-steps-for-object-based-broadcasting (visited on 10/22/2016).

[42] Michael Weitnauer. *Object-based broadcasting – for European leadership in next generation audio experiences D2.1: Initial reference architecture specification report.* Tech. rep. 2016. URL: https://orpheus-audio.eu/wp-content/uploads/2016/09/Orpheus-D2.1%5C_Initial-Reference-Architecture-Specification-Report%5C_v1.0.pdf (visited on 11/25/2016).

[43] Hynek Hermansky. "Multistream recognition of speech: Dealing with unknown unknowns". In: *Proceedings of the IEEE* 101.5 (May 2013), pp. 1076–1088. ISSN: 0018-9219. DOI: 10.1109/jproc.2012.2236871.

[44] Bowen Zhou and John HL Hansen. "Unsupervised audio stream segmentation and clustering via the Bayesian information criterion." In: *INTERSPEECH*. 2000, pp. 714–717.

[45] Wen-Huang Cheng, Wei-Ta Chu, and Ja-Ling Wu. "Semantic context detection based on hierarchical audio models". In: *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval - MIR '03* (2003). DOI: 10.1145/973264.973282.

[46] Rui Cai, Lie Lu, Hong-Jiang Zhang, and Lian-Hong Cai. "Highlight sound effects detection in audio stream". In: (June 7, 2003), pp. 37–40. URL: http://dl.acm.org/citation.cfm?id=1171723 (visited on 11/24/2016).

[47] Jun Huang, Yuan Dong, Jiqing Liu, Chengyu Dong, and Haila Wang. "Sports audio segmentation and classification". In: *2009 IEEE International Conference on Network Infrastructure and Digital Content* (2009). DOI: 10.1109/icnidc.2009.5360872. URL: https://www.lrde.epita.fr/%5Ctextasciitidlereda/cours/speech/speakerDiarization/5360872.pdf (visited on 11/24/2016).

[48] Rongqing Huang and J.H.L. Hansen. "Advances in unsupervised audio classification and segmentation for the broadcast news and NGSW corpora". In: *IEEE Transactions on Audio, Speech and Language Processing* 14.3 (2006), pp. 907–919. DOI: 10.1109/tsa.2005.858057. URL: http://crss.utdallas.edu/Publications/Huang2006.pdf (visited on 11/24/2016).

[49] Zhouyu Fu. "Improving feature aggregation for semantic music retrieval". In: *Proceedings of the 23rd ACM international conference on Multimedia - MM '15* (2015). DOI: 10.1145/2733373.2806391.

[50] Wei-Ta Chu, Wen-Huang Cheng, Ja-Ling Wu, and J. Yung-jen Hsu. "A study of semantic context detection by using SVM and GMM approaches". In: *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)* (2004). DOI: 10.1109/icme.2004.1394553.

[51] Guodong Guo and S.Z. Li. "Content-based audio classification and retrieval by support vector machines". In: *IEEE Transactions on Neural Networks* 14.1 (2003), pp. 209–215. DOI: 10.1109/tnn.2002.806626. URL: http://www.ee.columbia.edu/%5Ctextasciitidlesfchang/course/spr-F05/papers/guo-li-svm-audio00.pdf (visited on 11/24/2016).

[52] Zhu Liu, Yao Wang, and Tsuhan Chen. In: *The Journal of VLSI Signal Processing* 20.1/2 (1998), pp. 61–79. ISSN: 0922-5773. DOI: 10.1023/a:1008066223044.

[53] Silvia Pfeiffer. "Pause concepts for audio segmentation at different semantic levels". In: *Proceedings of the ninth ACM international conference on Multimedia - MULTIMEDIA '01* (2001). DOI: 10.1145/500141.500171.

[54] J. Foote. "Automatic audio segmentation using a measure of audio novelty". In: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)* (2000). DOI: 10.1109/icme.2000.869637.

[55] Avery Li-chun Wang. "An industrial-strength audio search algorithm". In: *Proceedings of the 4 th International Conference on Music Information Retrieval* (2003). URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.217.8882 (visited on 11/21/2016).

[56] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. “A review of audio fingerprinting”. In: *Journal of VLSI signal processing systems for signal, image and video technology* 41.3 (2005), pp. 271–284.

[57] Jaap Haitsma and Ton Kalker. “A highly robust audio fingerprinting system.” In: *Ismir*. Vol. 2002. 2002, pp. 107–115.

[58] Bruno Oliveira, Alexandre Crivellaro, César, and Roberto M. “Audio-based radio and TV broadcast monitoring”. In: (May 12, 2005), pp. 1–3. DOI: 10.1145/1114223.1114238. URL: http://dl.acm.org/citation.cfm?id=1114238%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[59] Alina Elma Abduraman, Sid-Ahmed Berrani, Jean-Bernard Rault, and Olivier Le Blouch. “From audio recurrences to TV program structuring”. In: *Proceedings of the 2011 ACM international workshop on Automated media analysis and production for novel TV services - AIEMPro ’11* (2011). DOI: 10.1145/2072552.2072556.

[60] Mike Dowman, Valentin Tablan, Hamish Cunningham, and Borislav Popov. “Web-assisted annotation, semantic indexing and search of television and radio news”. In: (Oct. 5, 2005), pp. 225–234. DOI: 10.1145/1060745.1060781. URL: http://dl.acm.org/citation.cfm?id=1060781%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[61] Masanori Sano, Yoshihiko Kawai, Hideki Sumiyoshi, and Nobuyuki Yagi. “Metadata production framework and metadata editor”. In: *Proceedings of the 14th annual ACM international conference on Multimedia - MULTIMEDIA ’06* (2006). DOI: 10.1145/1180639.1180810. URL: https://www.nhk.or.jp/strl/mpf/english/pdf/MPFandME.pdf (visited on 10/31/2016).

[62] Masanori Sano, Hideki Sumiyoshi, Masahiro Shibata, and Nobuyuki Yagi. “Metadata production framework (MPF) version 2.0”. In: (Oct. 23, 2009), pp. 1017–1018. DOI: 10.1145/1631272.1631497. URL: http://dl.acm.org/citation.cfm?id=1631497%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[63] R. Kubozono, K. Gomi, S. Kinohara, M. Inagaki, and Y. Matsuura. “Browse search using audio key-information for multimedia on-demand systems”. In: *IEEE Transactions on Consumer Electronics* 42.4 (1996), pp. 900–906. ISSN: 0098-3063. DOI: 10.1109/30.555770.

[64] G. Tzanetakis and P. Cook. “Multifeature audio segmentation for browsing and annotation”. In: *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA’99 (Cat. No.99TH8452)* (). DOI: 10.1109/aspaa.1999.810860.

[65] K. K. Agbele, E. F. Ayetiran, K. D. Aruleba, and D. O. Ekong. “Algorithm for Information Retrieval optimization”. In: *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (Oct. 2016), pp. 1–8. DOI: 10.1109/IEMCON.2016.7746242.

[66] Wei-Ho Tsai and Hsin-Min Wang. “A query-by-example framework to retrieve music documents by singer”. In: *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)* (2004). DOI: 10.1109/icme.2004.1394621.

[67] Wolfgang Hürst, Tobias Lauer, and Robert Kaschuba. “Interfaces for interactive audio-visual media browsing”. In: (Oct. 23, 2006), pp. 807–808. DOI: 10.1145/1180639.1180819. URL: http://dl.acm.org/citation.cfm?id=1180819%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[68] Eric Lee. "Towards a quantitative analysis of audio scrolling interfaces". In: (Apr. 28, 2007), pp. 2213–2218. DOI: 10.1145/1240866.1240982. URL: http://dl.acm.org/citation.cfm?id=1240982%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[69] Axel Carlier, Vincent Charvillat, and Wei Tsang. "A video Timeline with Bookmarks and Prefetch state for faster video browsing". In: (Oct. 13, 2015), pp. 967–970. DOI: 10.1145/2733373.2806376. URL: http://dl.acm.org/citation.cfm?id=2806376%5C&CFID=859577222%5C&CFTOKEN=95035629 (visited on 10/31/2016).

[70] Nicolas Hervé, Pierre Letessier, Mathieu Derval, and Hakim Nabi. "Amalia.js". In: *Proceedings of the 23rd ACM international conference on Multimedia - MM '15* (2015). DOI: 10.1145/2733373.2807406.

[71] *Project summary.* Aug. 2, 2016. URL: http://orpheus-audio.eu/project-summary/ (visited on 10/27/2016).

[72] Kris Gale. *The One Cost Engineers and Product Managers Don't Consider.* URL: http://firstround.com/review/The-one-cost-engineers-and-product-managers-dont-consider/.

[73] Rob Baxter, Mike Jackson, and Steve Crouch. "Software Evaluation: Criteria-based Assessment". In: (2011). URL: https://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf.

[74] K. Schoeffmann and L. Boeszoermenyi. "Video Browsing Using Interactive Navigation Summaries". In: *2009 Seventh International Workshop on Content-Based Multimedia Indexing.* June 2009, pp. 243–248. DOI: 10.1109/CBMI.2009.40.

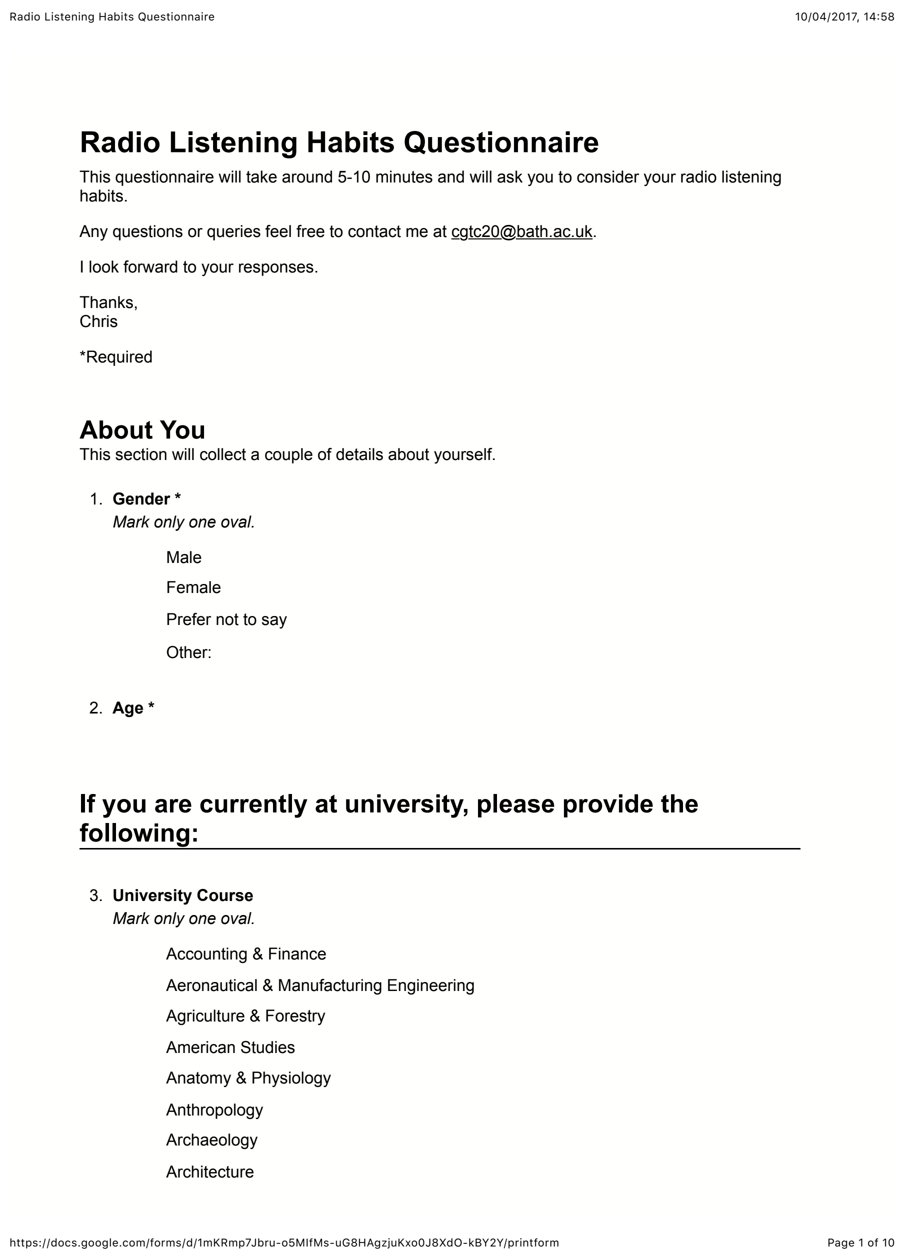
[75] John Brooke et al. "SUS-A quick and dirty usability scale". In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.

[76] Virginia Braun and Victoria Clarke. "Using thematic analysis in psychology". In: *Qualitative Research in Psychology* 3.2 (2006), pp. 77–101. DOI: 10.1191/1478088706qp063oa. URL: http://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa.

A Requirements Research

A.1 Research - Questionnaire

A.1.1 Questionnaire



Radio Listening Habits Questionnaire 10/04/2017, 14:58

Radio Listening Habits Questionnaire

This questionnaire will take around 5-10 minutes and will ask you to consider your radio listening habits.

Any questions or queries feel free to contact me at cgtc20@bath.ac.uk.

I look forward to your responses.

Thanks,
Chris

*Required

About You

This section will collect a couple of details about yourself.

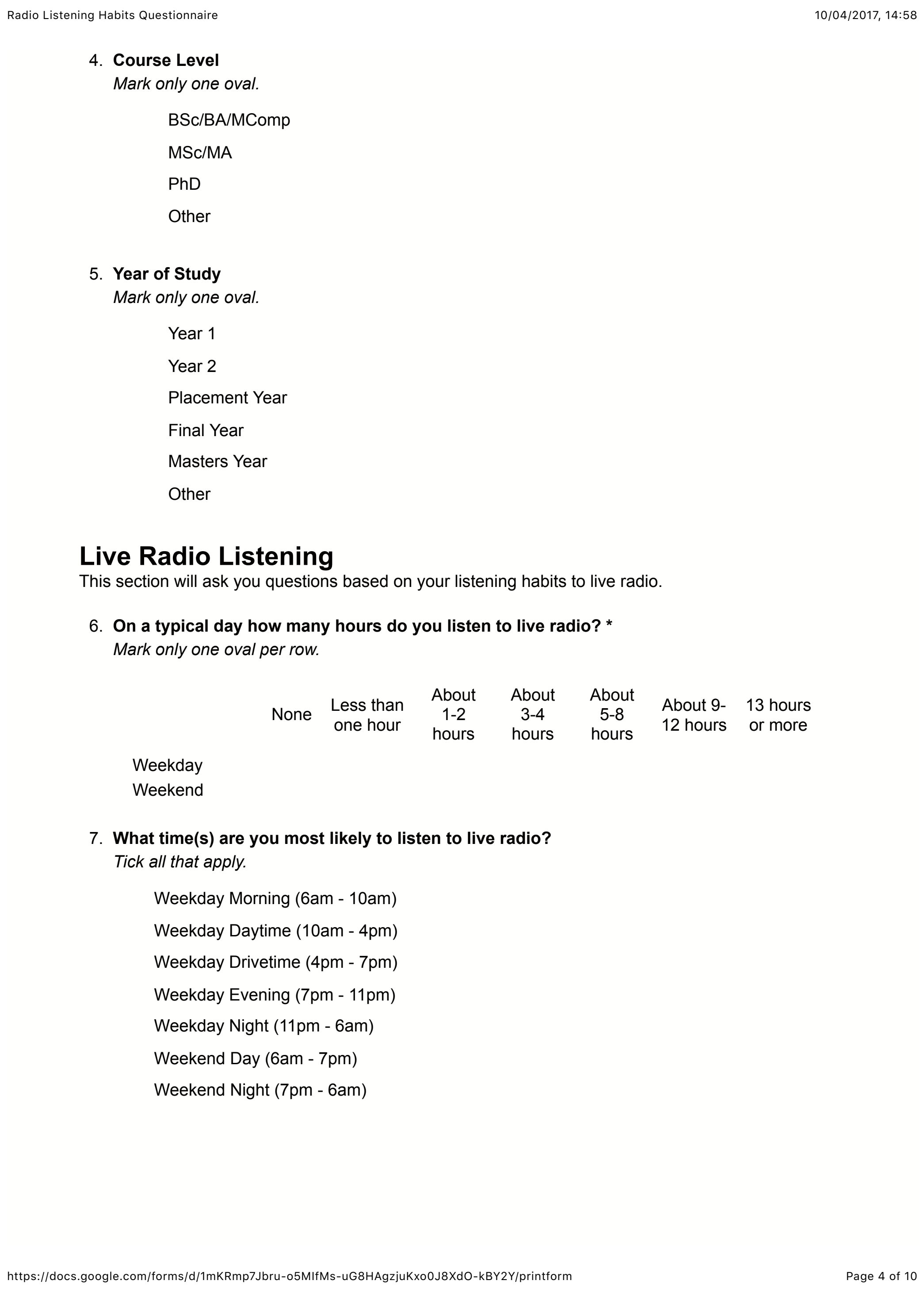
1. **Gender ***
   Mark only one oval.
   - Male
   - Female
   - Prefer not to say
   - Other:

2. **Age ***

If you are currently at university, please provide the following:

3. **University Course**
   Mark only one oval.
   - Accounting & Finance
   - Aeronautical & Manufacturing Engineering
   - Agriculture & Forestry
   - American Studies
   - Anatomy & Physiology
   - Anthropology
   - Archaeology
   - Architecture

https://docs.google.com/forms/d/1mKRmp7Jbru-o5MIfMs-uG8HAgzjuKxo0J8XdO-kBY2Y/printform Page 1 of 10



Radio Listening Habits Questionnaire 10/04/2017, 14:58

4. **Course Level**
   Mark only one oval.
   - BSc/BA/MComp
   - MSc/MA
   - PhD
   - Other

5. **Year of Study**
   Mark only one oval.
   - Year 1
   - Year 2
   - Placement Year
   - Final Year
   - Masters Year
   - Other

Live Radio Listening

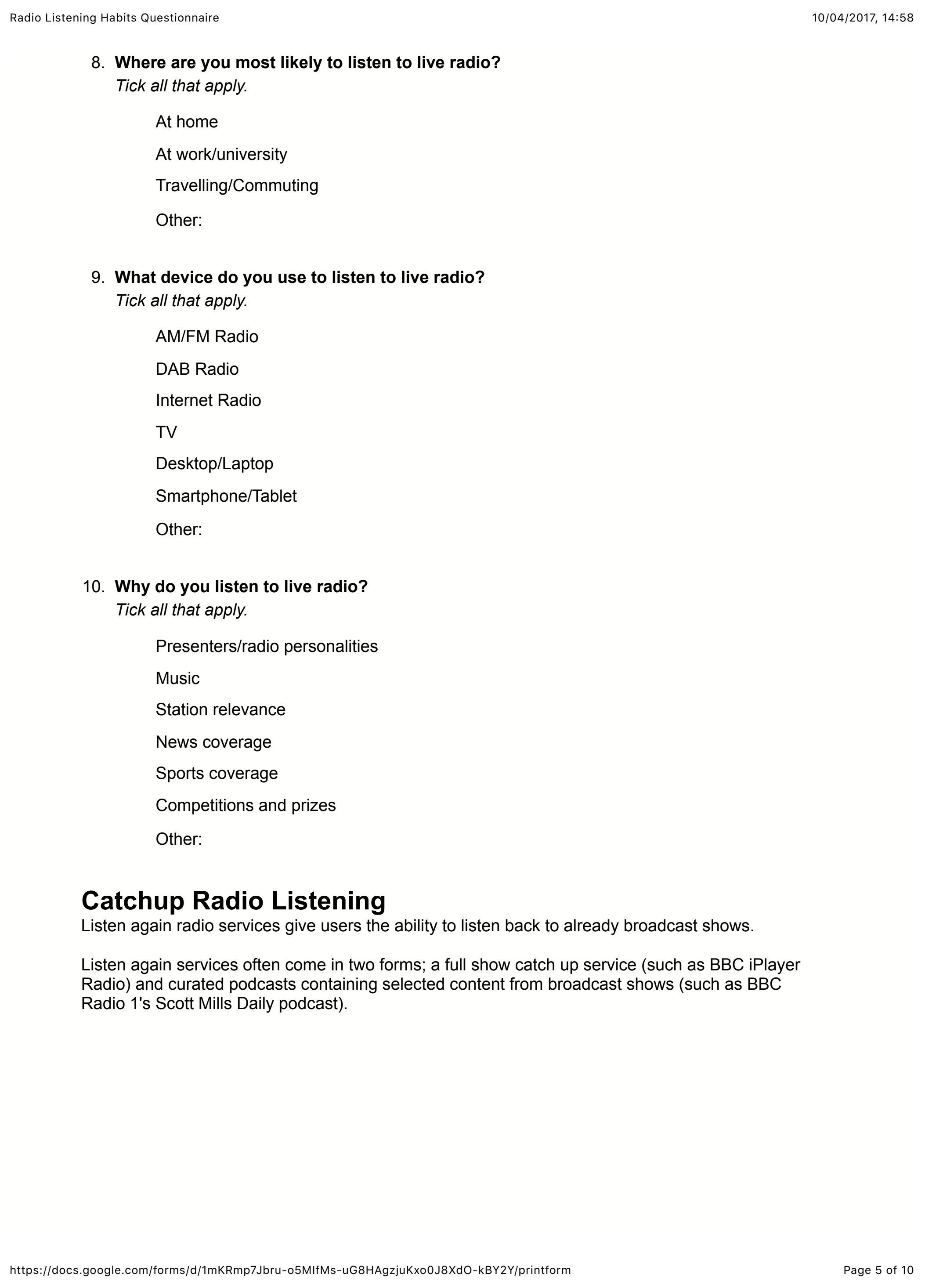
This section will ask you questions based on your listening habits to live radio.

6. **On a typical day how many hours do you listen to live radio? ***
   Mark only one oval per row.

	None	Less than one hour	About 1-2 hours	About 3-4 hours	About 5-8 hours	About 9-12 hours	13 hours or more
Weekday							
Weekend							

7. **What time(s) are you most likely to listen to live radio?**
   Tick all that apply.
   - Weekday Morning (6am - 10am)
   - Weekday Daytime (10am - 4pm)
   - Weekday Drivetime (4pm - 7pm)
   - Weekday Evening (7pm - 11pm)
   - Weekday Night (11pm - 6am)
   - Weekend Day (6am - 7pm)
   - Weekend Night (7pm - 6am)

https://docs.google.com/forms/d/1mKRmp7Jbru-o5MIfMs-uG8HAgzjuKxo0J8XdO-kBY2Y/printform Page 4 of 10



8. **Where are you most likely to listen to live radio?**
   Tick all that apply.

   At home

   At work/university

   Travelling/Commuting

   Other:

9. **What device do you use to listen to live radio?**
   Tick all that apply.

   AM/FM Radio

   DAB Radio

   Internet Radio

   TV

   Desktop/Laptop

   Smartphone/Tablet

   Other:

10. **Why do you listen to live radio?**
    Tick all that apply.

    Presenters/radio personalities

    Music

    Station relevance

    News coverage

    Sports coverage

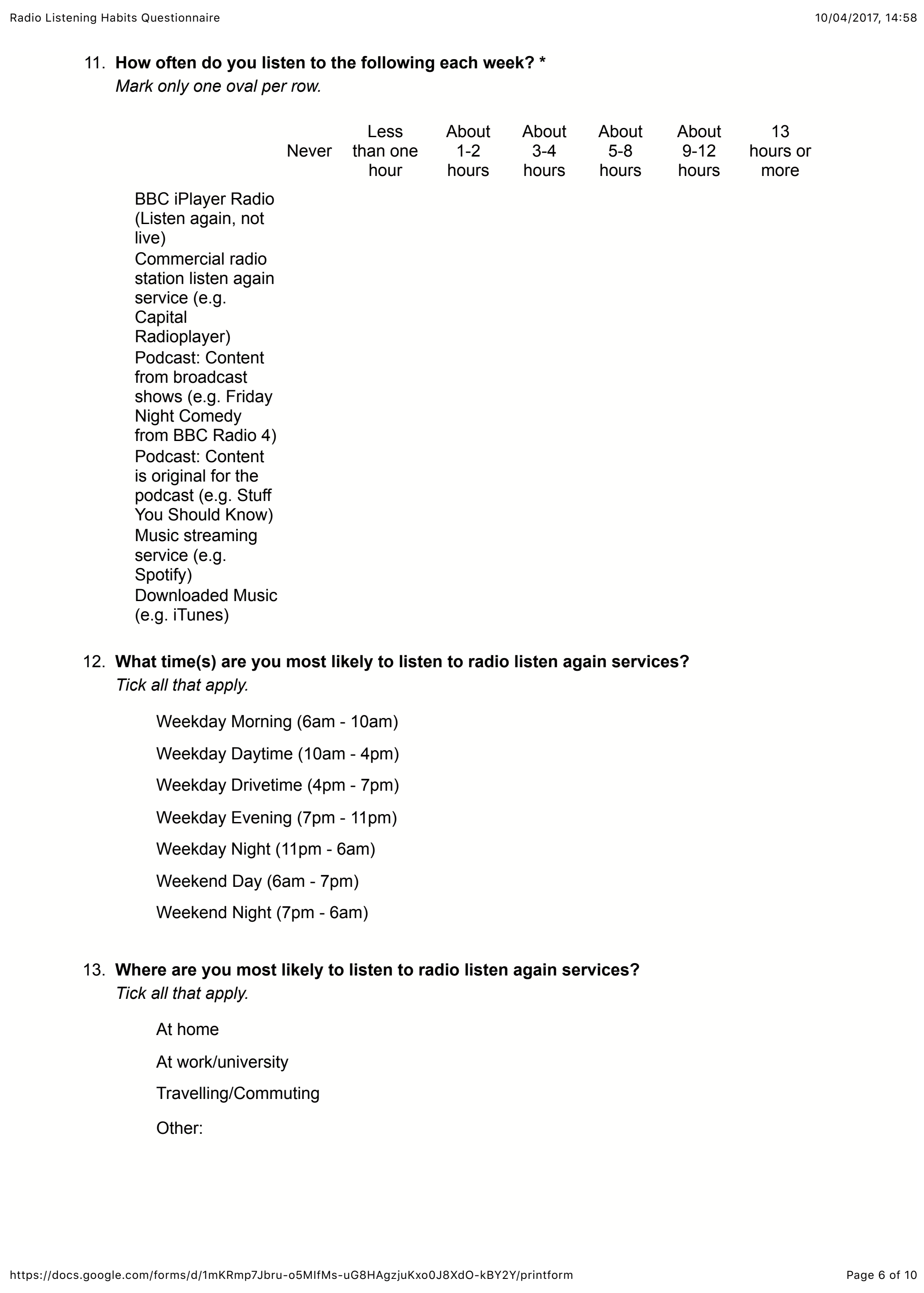
    Competitions and prizes

    Other:

Catchup Radio Listening

Listen again radio services give users the ability to listen back to already broadcast shows.

Listen again services often come in two forms; a full show catch up service (such as BBC iPlayer Radio) and curated podcasts containing selected content from broadcast shows (such as BBC Radio 1's Scott Mills Daily podcast).



Radio Listening Habits Questionnaire 10/04/2017, 14:58

11. **How often do you listen to the following each week? ***
Mark only one oval per row.

	Never	Less than one hour	About 1-2 hours	About 3-4 hours	About 5-8 hours	About 9-12 hours	13 hours or more
BBC iPlayer Radio (Listen again, not live)							
Commercial radio station listen again service (e.g. Capital Radioplayer)							
Podcast: Content from broadcast shows (e.g. Friday Night Comedy from BBC Radio 4)							
Podcast: Content is original for the podcast (e.g. Stuff You Should Know)							
Music streaming service (e.g. Spotify)							
Downloaded Music (e.g. iTunes)							

12. **What time(s) are you most likely to listen to radio listen again services?**
Tick all that apply.

Weekday Morning (6am - 10am)

Weekday Daytime (10am - 4pm)

Weekday Drivetime (4pm - 7pm)

Weekday Evening (7pm - 11pm)

Weekday Night (11pm - 6am)

Weekend Day (6am - 7pm)

Weekend Night (7pm - 6am)

13. **Where are you most likely to listen to radio listen again services?**
Tick all that apply.

At home

At work/university

Travelling/Commuting

Other:

https://docs.google.com/forms/d/1mKRmp7Jbru-o5MIfMs-uG8HAgzjuKxo0J8XdO-kBY2Y/printform Page 6 of 10

14. **What device do you use to listen to radio listen again services?**
Tick all that apply.

Internet Radio
TV
Desktop/Laptop
Smartphone
Other:

15. **Why do you listen to radio listen again services?**
Tick all that apply.

Presenters/Radio personalities
Music choice
Station relevance
News coverage
Sports coverage
Competitions and prizes
Other:

Design Research

The following questions are designed to be quick fire. Please select the first response that comes into your head. Do not worry if some colours are used twice.

16. **What colour do you most associate with music? ***
Mark only one oval.

Red
Orange
Purple
Blue
Green
Yellow
White
Grey
Black

17. **What colour do you most associate with news? ***
    Mark only one oval.

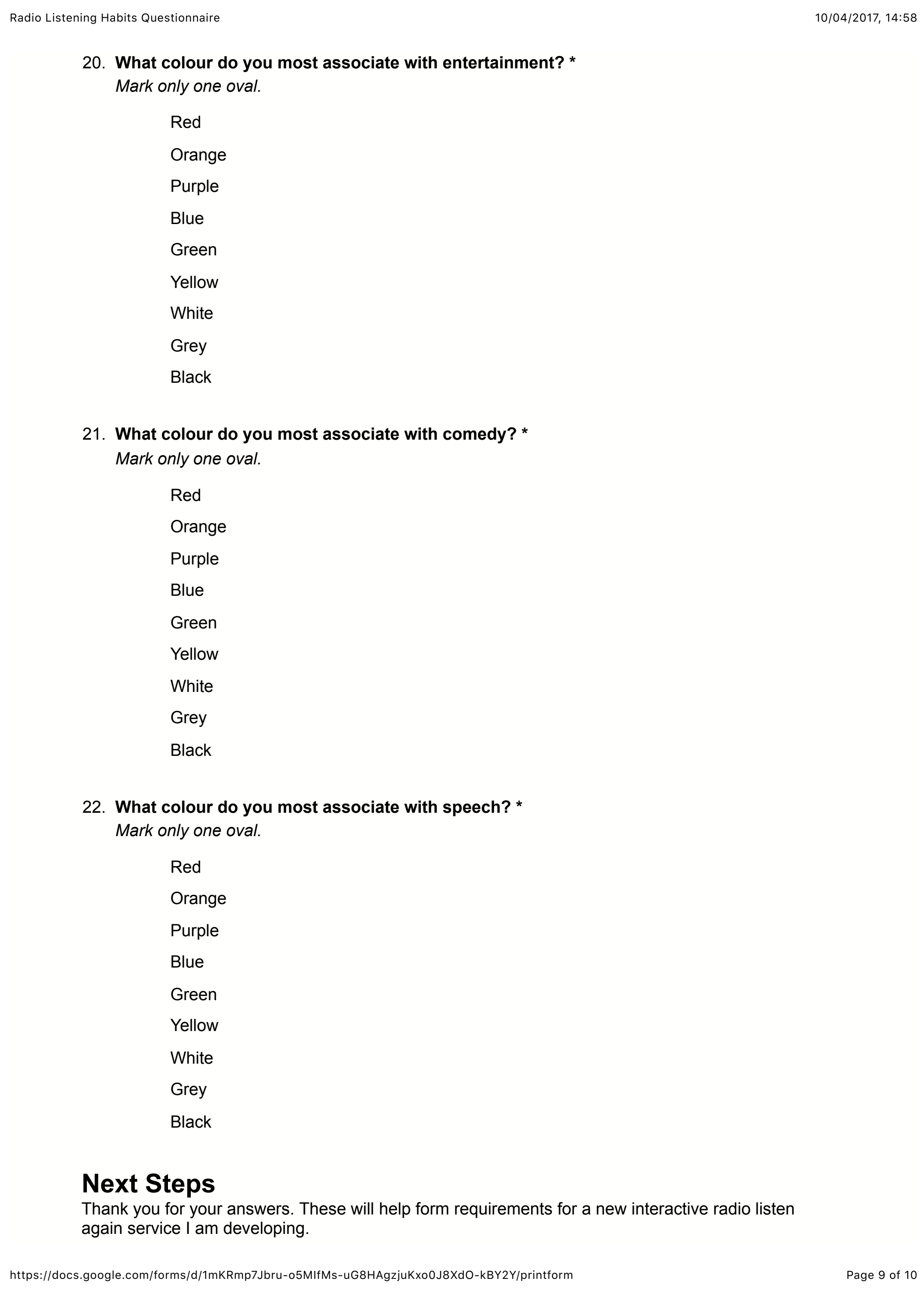
    - Red
    - Orange
    - Purple
    - Blue
    - Green
    - Yellow
    - White
    - Grey
    - Black

18. **What colour do you most associate with sport? ***
    Mark only one oval.

    - Red
    - Orange
    - Purple
    - Blue
    - Green
    - Yellow
    - White
    - Grey
    - Black

19. **What colour do you most associate with drama? ***
    Mark only one oval.

    - Red
    - Orange
    - Purple
    - Blue
    - Green
    - Yellow
    - White
    - Grey
    - Black



Radio Listening Habits Questionnaire 10/04/2017, 14:58

20. **What colour do you most associate with entertainment? ***
Mark only one oval.

Red
Orange
Purple
Blue
Green
Yellow
White
Grey
Black

21. **What colour do you most associate with comedy? ***
Mark only one oval.

Red
Orange
Purple
Blue
Green
Yellow
White
Grey
Black

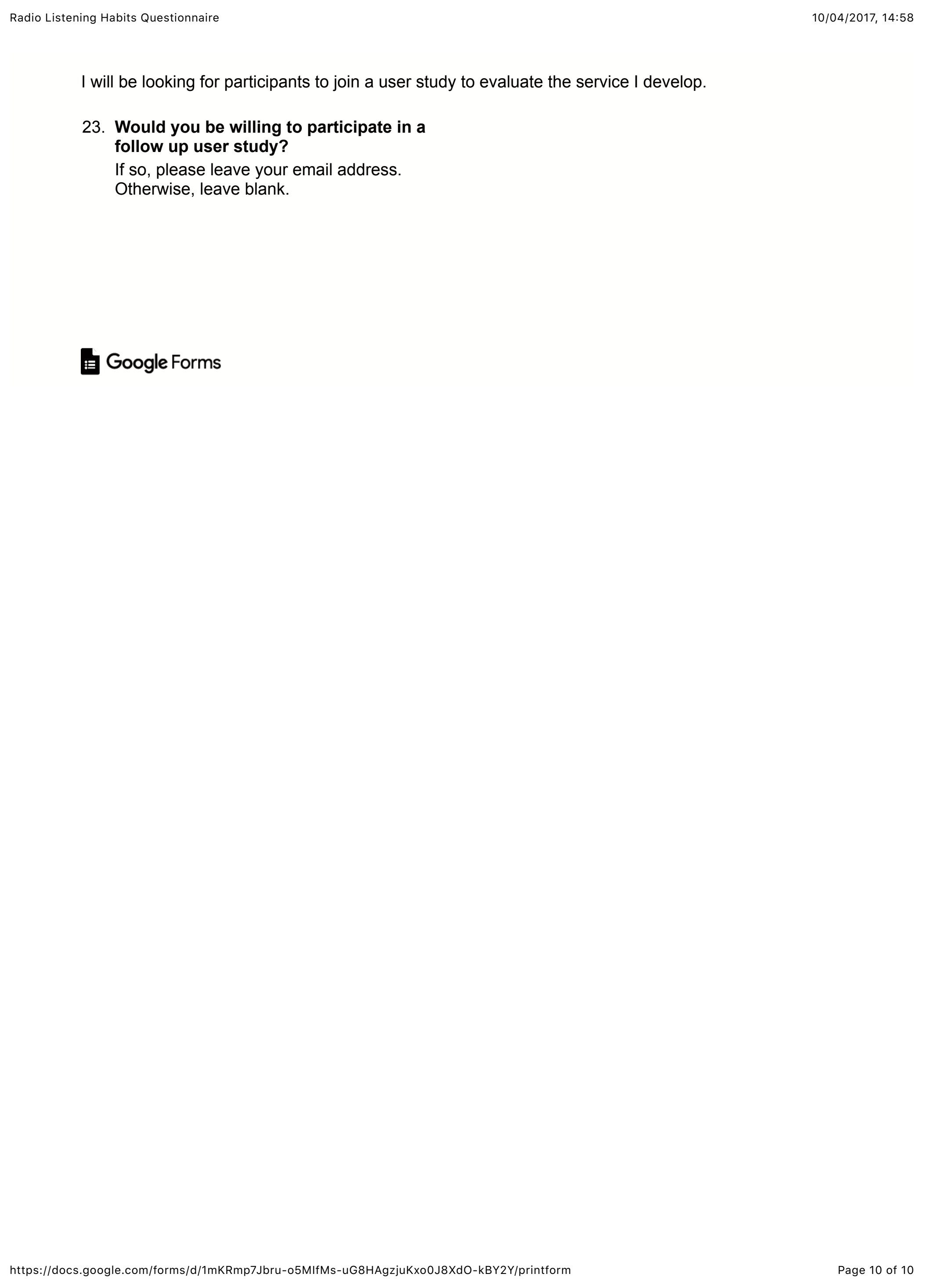
22. **What colour do you most associate with speech? ***
Mark only one oval.

Red
Orange
Purple
Blue
Green
Yellow
White
Grey
Black

Next Steps

Thank you for your answers. These will help form requirements for a new interactive radio listen again service I am developing.

https://docs.google.com/forms/d/1mKRmp7Jbru-o5MIfMs-uG8HAgzjuKxo0J8XdO-kBY2Y/printform Page 9 of 10



Radio Listening Habits Questionnaire 10/04/2017, 14:58

I will be looking for participants to join a user study to evaluate the service I develop.

23. **Would you be willing to participate in a follow up user study?**
    If so, please leave your email address.
    Otherwise, leave blank.

Google Forms

https://docs.google.com/forms/d/1mKRmp7Jbru-o5MIfMs-uG8HAgzjuKxo0J8XdO-kBY2Y/printform Page 10 of 10

A.1.2 Responses

Table A.1 Research Questionnaire: Respondent Gender

Gender	Count
Male	26
Female	22
Prefer not to say	0
Other	2

Table A.2 Research Questionnaire: Respondent Age

Age	Count
19	1
20	4
21	16
22	13
23	7
24	2
28	1
29	1
30	1
35	1
49	1
64	1
74	1

Table A.3 Research Questionnaire: Respondent University Course

University Course	Count
Computer Science	7
Iberian Languages/Hispanic Studies	4
Biological Sciences	3
Mechanical Engineering	3
Physics and Astronomy	3
Mathematics	2
Politics	2
Business & Management Studies	1
Chemical Engineering	1
Economics	1
Education	1
Law	1
Medicine	1
Pharmacology & Pharmacy	1
Sports Science	1

Table A.4 Research Questionnaire: Respondent Course Level

Course Level	Count
BSc/BA/MComp	26
MSc/MA	5
PhD	0
Other	1

Table A.5 Research Questionnaire: Respondent Year of Study

Year of Study	Count
Year 1	0
Year 2	4
Placement Year	1
Final Year	25
Masters Year	1
Other	1

Table A.6 Research Questionnaire: On a typical day how many hours do you listen to live radio?

	None	Less than one hour	About 1-2 hours	About 3-4 hours	About 5-8 hours	About 9-12 hours	13 hours or more
Weekday	7	15	19	7	2	0	0
Weekend	11	18	9	10	2	0	0

Table A.7 Research Questionnaire: What time(s) are you most likely to listen to live radio?

Time	Count
Weekday Morning (6am - 10am)	32
Weekday Daytime (10am - 4pm)	13
Weekday Drivetime (4pm - 7pm)	20
Weekday Evening (7pm - 11pm)	12
Weekday Night (11pm - 6am)	2
Weekend Day (6am - 7pm)	22
Weekend Night (7pm - 6am)	6

Table A.8 Research Questionnaire: Where are you most likely to listen to live radio?

Location	Count
At home	28
At work/university	8
Travelling/Commuting	28
Other	0

Table A.9 Research Questionnaire: What device do you use to listen to live radio?

Device	Count
AM/FM Radio	19
DAB Radio	18
Internet Radio	3
TV	1
Desktop/Laptop	11
Smartphone/Tablet	13
Other	10

Table A.10 Research Questionnaire: Why do you listen to live radio?

Reason	Count
Presenters/radio personalities	26
Music	40
Station relevance	6
News coverage	11
Sports coverage	4
Competitions and prizes	2
Other	3

Table A.11 Research Questionnaire: How often do you listen to the following each week?

	Time spent listening						
	Never	**Less than one hour**	**About 1-2 hours**	**About 3-4 hours**	**About 5-8 hours**	**About 9-12 hours**	**13 hours or more**
BBC iPlayer Radio (Listen again, not live)	32	13	4	1	0	0	0
Commercial radio station listen again service (e.g. Capital Radioplayer)	38	8	2	1	1	0	0
Podcast: Content from broadcast shows (e.g. Friday Night Comedy from BBC Radio 4)	31	8	9	2	0	0	0
Podcast: Content is original for the podcast (e.g. Stuff You Should Know)	32	5	6	5	2	0	0
Music streaming service (e.g. Spotify)	10	3	4	4	5	9	15
Downloaded Music (e.g. iTunes)	17	11	4	4	4	8	2

Table A.12 Research Questionnaire: What time(s) are you most likely to listen to radio listen again services?

Time	**Count**
Weekday Morning (6am - 10am)	9
Weekday Daytime (10am - 4pm)	12
Weekday Drivetime (4pm - 7pm)	6
Weekday Evening (7pm - 11pm)	21
Weekday Night (11pm - 6am)	5
Weekend Day (6am - 7pm)	12
Weekend Night (7pm - 6am)	11

Table A.13 Research Questionnaire: Where are you most likely to listen to radio listen again services?

Device	**Count**
Internet Radio	7
TV	1
Desktop/Laptop	15
Smartphone	20
Other	16

Table A.14 Research Questionnaire: What device do you use to listen to radio listen again services?

Location	**Count**
At home	29
At work/university	17
Travelling/Commuting	15
Other	0

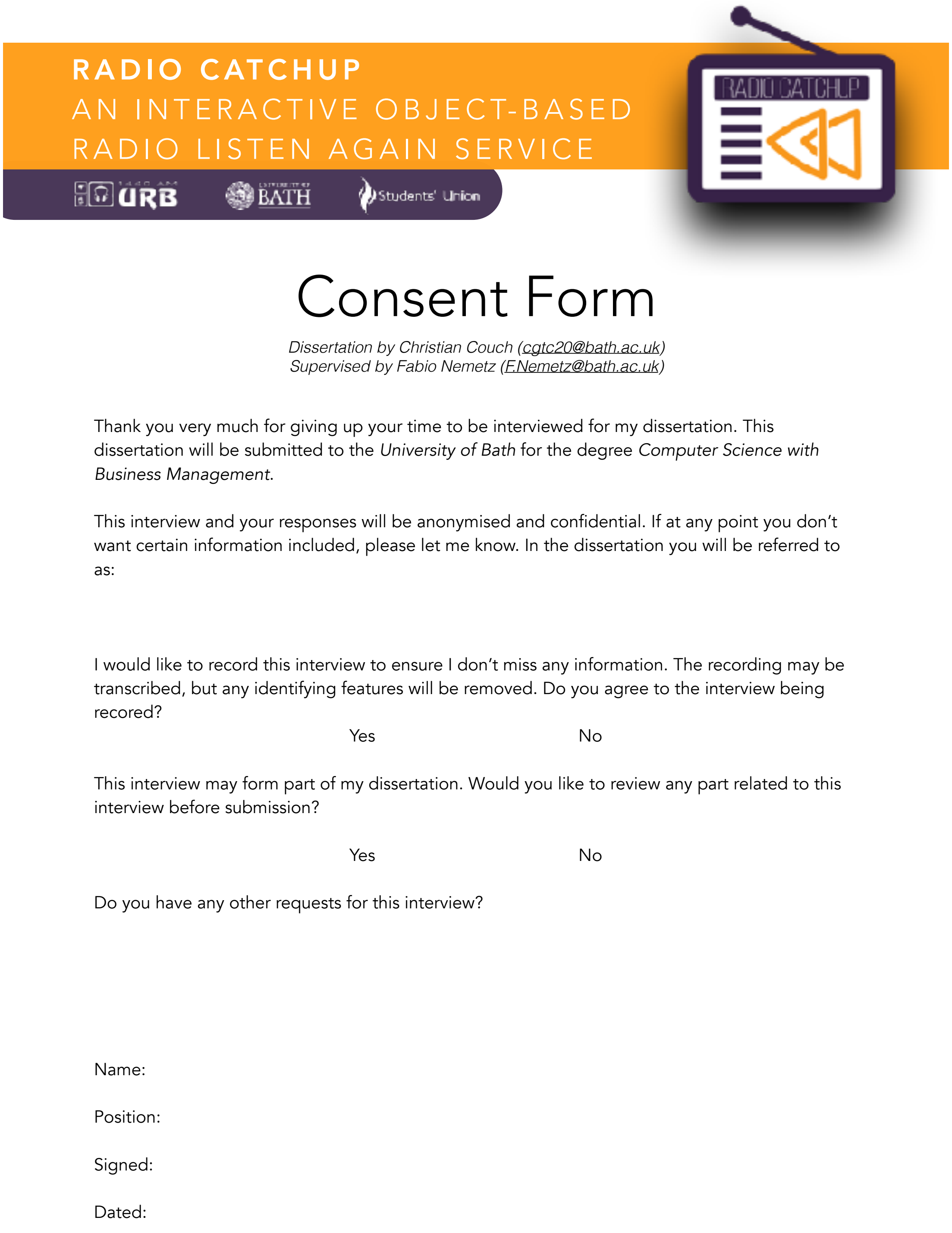
Table A.15 Research Questionnaire: Why do you listen to radio listen again services?

Reason	**Count**
Presenters/Radio personalities	23
Music choice	23
Station relevance	7
News coverage	3
Sports coverage	1
Competitions and prizes	1
Other	5

Table A.16 Research Questionnaire: What colour do you most associate with ...

	Colours								
	Red	**Orange**	**Purple**	**Blue**	**Green**	**Yellow**	**White**	**Grey**	**Black**
Music	4	3	13	20	4	2	1	1	2
News	34	0	0	4	1	0	1	4	6
Sport	4	1	1	15	15	12	1	0	1
Drama	13	2	17	5	2	3	0	2	6
Entertainment	9	6	16	4	5	8	1	0	1
Comedy	4	13	4	7	3	17	1	1	0
Speech	4	2	2	14	1	2	11	5	8

A.2 Research - Interviews Consent Form



RADIO CATCHUP
AN INTERACTIVE OBJECT-BASED
RADIO LISTEN AGAIN SERVICE

Consent Form

Dissertation by Christian Couch (cgtc20@bath.ac.uk)
Supervised by Fabio Nemetz (F.Nemetz@bath.ac.uk)

Thank you very much for giving up your time to be interviewed for my dissertation. This dissertation will be submitted to the *University of Bath* for the degree *Computer Science with Business Management*.

This interview and your responses will be anonymised and confidential. If at any point you don't want certain information included, please let me know. In the dissertation you will be referred to as:

I would like to record this interview to ensure I don't miss any information. The recording may be transcribed, but any identifying features will be removed. Do you agree to the interview being recored?

Yes No

This interview may form part of my dissertation. Would you like to review any part related to this interview before submission?

Yes No

Do you have any other requests for this interview?

Name:

Position:

Signed:

Dated:

A.3 Podcast Analysis

Table A.17 Scott Mills Show: Itemised Breakdown - 16/02/2017

Item	Show Time (hh:mm:ss)	Duration (mm:ss)	Type	Title	Included in Podcast
1	00:00:00	07:46	Music		No
2	00:07:46	00:24	Speech	Show Intro	No
3	00:08:10	06:12	Music		No
4	00:14:22	02:07	Speech	Tom Hardy, CBBC Bed Time Stories	Yes
5	00:16:29	03:29	Music		No
6	00:19:58	00:06	Speech	Music Segue	No
7	00:20:04	00:24	Advert	Advert: Dance Anthems	No
8	00:20:28	03:09	Music		No
9	00:23:37	00:46	Speech	Chris Stark in Japan and Innuendo Bingo Tease	Yes
10	00:24:23	05:27	Music		No
11	00:29:50	01:19	News		No
12	00:31:09	06:29	Music		No
13	00:37:38	00:20	Speech	Chris Stark in Japan Tease	No
14	00:37:58	02:41	Music		No
15	00:40:39	05:59	Speech	Innuendo Bingo: Joey Essex Interview	Yes
16	00:46:38	03:07	Music		No
17	00:49:45	09:52	Speech	Innuendo Bingo: Game	Yes
18	00:59:37	07:08	Music		No
19	01:06:45	00:21	Speech	Innuendo Bingo: Recap	Yes
20	01:07:06	06:19	Music		No
21	01:13:25	00:24	Speech	Chris Stark in Japan Tease and Song Introduction	No
22	01:13:49	02:54	Music		No
23	01:16:43	00:10	Speech	Music Segue	No
24	01:16:53	03:26	Music		No
25	01:20:19	00:35	Speech	Lettuce Crisis	Yes
26	01:20:54	01:43	Speech	Brit Awards Competition Promo	No
27	01:22:37	03:45	Music		No
28	01:26:22	00:32	Speech	Chris Stark in Japan & Bangers Tease	No
29	01:26:54	02:56	Music		No
30	01:29:50	02:20	News		No
31	01:32:10	05:23	Music		No
32	01:37:33	00:27	Advert	Advert: Radio 1 Breakfast Show	No
33	01:38:00	03:14	Music		No
34	01:41:14	00:23	Speech	Bangers Tease	No
35	01:41:37	03:24	Music		No

36	01:45:01	02:37	Speech	Bangers: Game	Yes
37	01:47:38	03:32	Music		No
38	01:51:10	00:23	Speech	Bangers: Recap	Yes
39	01:51:33	03:11	Music		No
40	01:54:44	00:50	Speech	Track of the Day	No
41	01:55:34	12:37	Music		No
42	02:08:11	00:21	Speech	Chris Stark in Japan: Tease	No
43	02:08:32	06:04	Music		No
44	02:14:36	02:08	Speech	Chris Stark in Japan: Intro	Yes
45	02:16:44	00:24	Advert	Advert: BBC Radio 1 New Music	No
46	02:17:08	03:52	Music		No
47	02:21:00	04:04	Speech	Chris Stark in Japan: Korean Billy & Chris Phone Call	Yes
48	02:25:04	04:16	Music		No
49	02:29:20	00:30	Speech	Chris Stark in Japan: Recap	No
50	02:29:50	03:10	News		No
51	02:33:00	06:17	Music		No
52	02:39:17	03:28	Speech	Real or No Real: Game (Part 1)	Yes
53	02:42:45	03:20	Music		No
54	02:46:05	05:30	Speech	Real or No Real: Game (Part 2)	Yes
55	02:51:35	03:16	Music		No
56	02:54:51	01:42	Speech	Show Recap	No
57	02:56:33	03:27	Music		No

B JSON Data Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "http://catchup.radio/schema.json",
    "properties": {
        "media": {
            "additionalProperties": false,
            "id": "http://catchup.radio/schema.json/properties/media",
            "properties": {
                "mp3": {
                    "id": "http://catchup.radio/schema.json/properties/media/properties/mp3",
                    "type": "string"
                },
                "poster": {
                    "id": "http://catchup.radio/schema.json/properties/media/properties/poster",
                    "type": "string"
                }
            },
            "type": "object"
        },
        "programme": {
            "additionalProperties": false,
            "id": "http://catchup.radio/schema.json/properties/programme",
            "properties": {
                "broadcastEnd": {
                    "id": "http://catchup.radio/schema.json/properties/programme/properties/broadcastEnd",
                    "type": "integer"
                },
                "broadcastStart": {
                    "id": "http://catchup.radio/schema.json/properties/programme/properties/broadcastStart",
                    "type": "integer"
                },
                "description": {
                    "id": "http://catchup.radio/schema.json/properties/programme/properties/description",
                    "type": "string"
                },
                "episodeName": {
                    "id": "http://catchup.radio/schema.json/properties/programme/properties/episodeName",
                    "type": "string"
                },
                "presenters": {
                    "id": "http://catchup.radio/schema.json/properties/programme/properties/presenters",
                    "type": "string"
                },
                "seriesName": {
                    "id": "http://catchup.radio/schema.json/properties/programme/properties/seriesName",
                    "type": "string"
                }
            },
            "type": "object"
        },
        "plugins": {
            "additionalItems": false,
            "id": "http://catchup.radio/schema.json/properties/plugins",
            "items": {
                "additionalProperties": false,
                "id": "http://catchup.radio/schema.json/properties/plugins/items",
                "properties": {
                    "options": {
                        "additionalProperties": false,
                        "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options",
                        "properties": {
                            "body": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/body",
                                "type": "string"
                            },
                            "disabled": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/disabled",
                                "type": "boolean"
                            },
                            "displayEnd": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/displayEnd",
                                "type": "string"
                            },
                            "displayKeywords": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/displayKeywords",
                                "type": "string"
                            },
                            "displayPicture": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/displayPicture",
                                "type": "string"
                            },
                            "displayStart": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/displayStart",
                                "type": "string"
                            },
                            "end": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/end",
                                "type": "integer"
                            },
                            "skipItem": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/skipItem",
                                "type": "boolean"
                            },
                            "start": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/start",
                                "type": "integer"
                            },
                            "subtitle": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/subtitle",
                                "type": "string"
                            },
                            "title": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/title",
                                "type": "string"
                            },
                            "type": {
                                "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/options/properties/type",
                                "type": "string"
                            }
                        },
                        "type": "object"
                    },
                    "pluginName": {
                        "id": "http://catchup.radio/schema.json/properties/plugins/items/properties/pluginName",
                        "type": "string"
                    }
                },
                "type": "object"
            },
            "type": "array"
        }
    },
    "type": "object"
}
```

C Requirements Evaluation

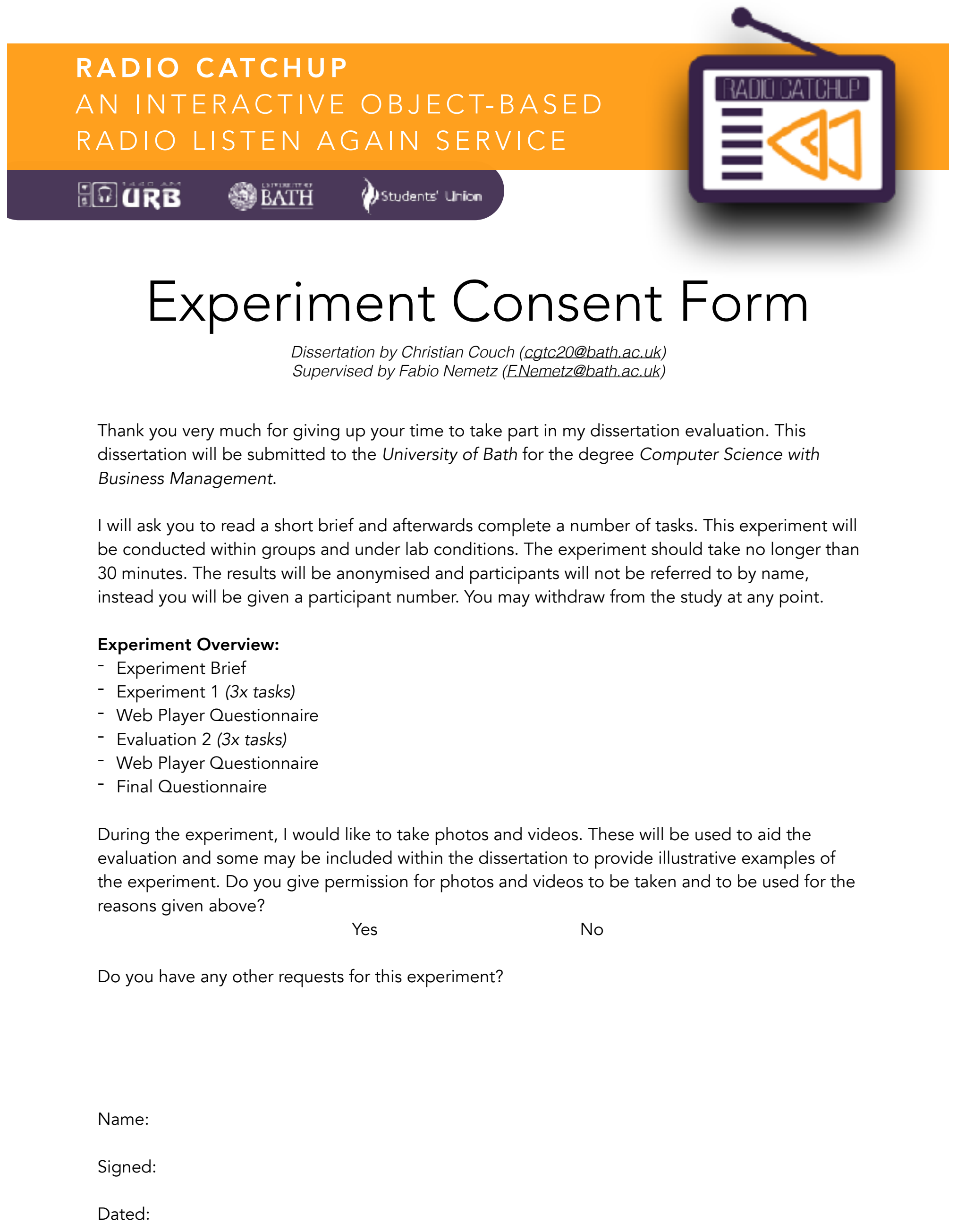
Table C.1 Criteria-based Assessment

ID	Requirement	Priority	Met?	Supporting comments.
D.1	Log Show Elements	High	Yes	Although not all segments could be logged, the important ones were.
D.1.1	Music	High	Yes	See section 5.1.1
D.1.2	Speech	Medium	Yes	See section 5.1.2
D.1.3	Adverts	Low	Yes	See section 5.1.1
D.1.4	News	Low	No	URB's news service was not working at time of implementation, so no logging method could be devised.
D.2	Analyse Show Elements	Medium	Yes	Although not all sub-requirements were implemented, without segment ranking the segment analysis is still sufficient.
D.2.1	— Speech Topic Detection	Low	Yes	See section 5.2.2
D.2.2	Music Metadata	Medium	Yes	See section 5.2.2
D.2.3	Segment Information	Low	Yes	See section 5.2.2.
D.2.4	Segment Ranking	Low	No	An accurate ranking system ended up being technically too complex
D.3	Automatic Podcast Generator	Low	No	Dependent on Req. *D.2.4* which was not implemented.
D.4	Management System	Low	No	Big workload for a low priority requirement, however it has been noticed that this requirement has a far greater impact to the usefulness of the system to stations. The priority should have been higher to reflect this.
D.5	System Modularisation	Low	Yes	Evidence of this can be seen within the system architecture.
D.6	Performance	Medium	Yes	Segmentation happens within allowed timeframes.
D.7	Data Integrity	Medium	Yes	Initial testing showed a high level of segment accuracy, far greater than requirement.
P.1	Visualise Show Segments	High	Yes	See section 5.3
P.2	Audio Progress Bar	Medium	Yes	See section 5.3.1
P.3	Audio Controls	Medium	Yes	The majority of audio controls were implemented and the ones that were not don't have a significant impact on this requirement.

P.3.1	Play/Pause Button	Medium	Yes	See section 5.3.1
P.3.2	Volume Controls	Low	No	Insufficient resources.
P.3.3	Skip Button	Low	Yes	See section 5.3.1
P.4	Listen Back to a Show	High	Yes	See section 5.3.1
P.5	Mobile Optimised	High	Yes	See section 5.3
P.6	Performance	Medium	Yes	Initial testing showed the player loading and playing within the allowed timeframes.
P.7	Usability	Medium	Yes	A user experiment concluded that the segmented web player was easy and intuitive to use. See section 6.2

D User Experiment

D.1 Consent Form



RADIO CATCHUP
AN INTERACTIVE OBJECT-BASED
RADIO LISTEN AGAIN SERVICE

Experiment Consent Form

Dissertation by Christian Couch (cgtc20@bath.ac.uk)
Supervised by Fabio Nemetz (F.Nemetz@bath.ac.uk)

Thank you very much for giving up your time to take part in my dissertation evaluation. This dissertation will be submitted to the *University of Bath* for the degree *Computer Science with Business Management.*

I will ask you to read a short brief and afterwards complete a number of tasks. This experiment will be conducted within groups and under lab conditions. The experiment should take no longer than 30 minutes. The results will be anonymised and participants will not be referred to by name, instead you will be given a participant number. You may withdraw from the study at any point.

Experiment Overview:
- Experiment Brief
- Experiment 1 *(3x tasks)*
- Web Player Questionnaire
- Evaluation 2 *(3x tasks)*
- Web Player Questionnaire
- Final Questionnaire

During the experiment, I would like to take photos and videos. These will be used to aid the evaluation and some may be included within the dissertation to provide illustrative examples of the experiment. Do you give permission for photos and videos to be taken and to be used for the reasons given above?

Yes No

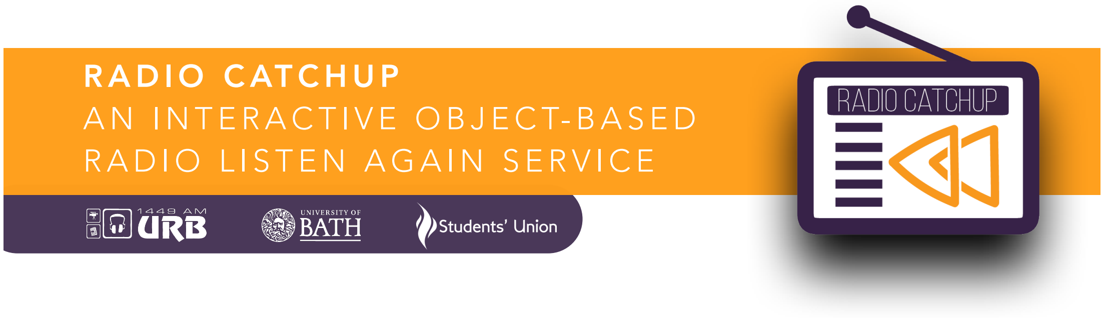
Do you have any other requests for this experiment?

Name:

Signed:

Dated:

D.2 Guide



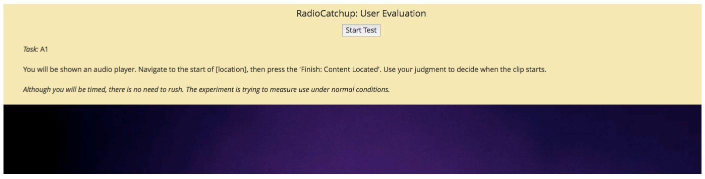
Experiment Guide

Dissertation by Christian Couch (cgtc20@bath.ac.uk)
Supervised by Fabio Nemetz (F.Nemetz@bath.ac.uk)

You will be given 3 tasks, per web player. Each one will ask you to navigate to a specific piece of content within a radio programme. Although you will be timed, there is no need to rush. The experiment is trying to measure use under normal conditions.

HOW WILL THE TASKS WORK?

1. Go to task's web address and confirm the task ID match.
2. Read the brief and when ready click 'Start Task'.


RadioCatchup: User Evaluation

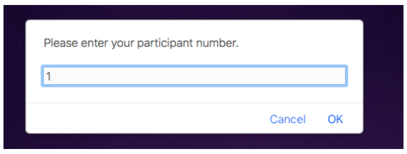
Start Test

Task: A1

You will be shown an audio player. Navigate to the start of [location], then press the 'Finish: Content Located'. Use your judgment to decide when the clip starts.

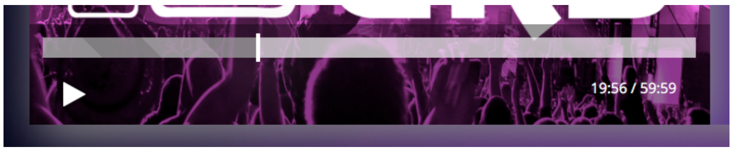
Although you will be timed, there is no need to rush. The experiment is trying to measure use under normal conditions.

3. A prompt will ask for your participant number, please enter it.


Please enter your participant number.

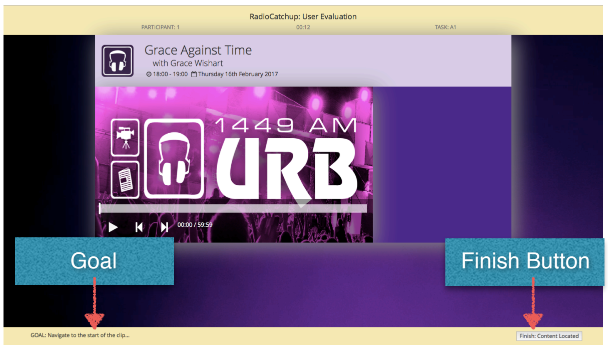
1

4. The audio player will then show.
5. Navigate to the start of the content you have been asked to find. The seekbar will indicate the players current position.

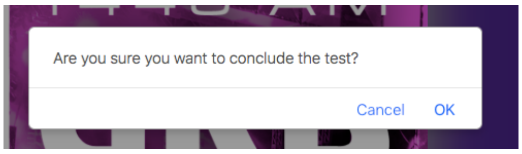


RADIO CATCHUP
AN INTERACTIVE OBJECT-BASED RADIO LISTEN AGAIN SERVICE

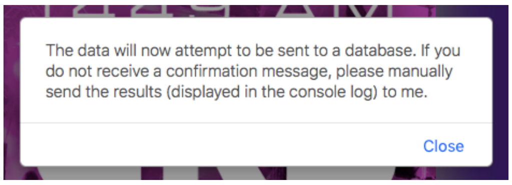
6. Once you're satisfied the audio player is at the requested location, click 'FINISH: Content Located' in the bottom right hand corner.


RadioCatchup: User Evaluation
PARTICIPANT: 1 00:12 TASK: A1
Grace Against Time
with Grace Wishart
18:00 - 19:00 Thursday 16th February 2017
00:00 / 59:59
GOAL: Navigate to the start of the clip...
Finish: Content Located

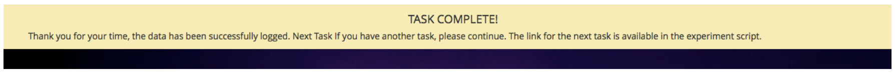
7. A confirmation window will appear. Click 'ok' to finish the task.


Are you sure you want to conclude the test?

8. A warning message will appear, reminding you to check the results have been successfully submitted. Click 'close'.

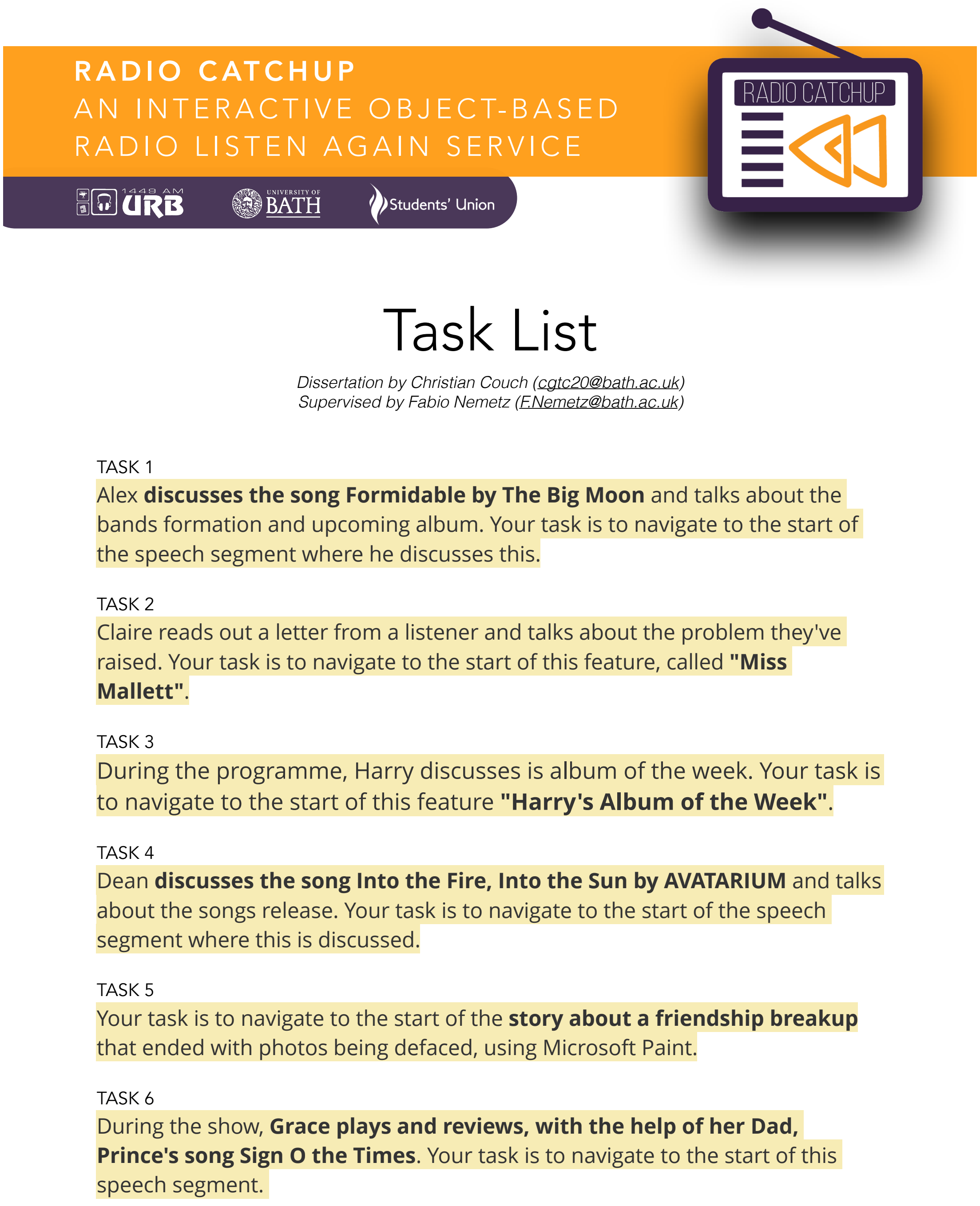

The data will now attempt to be sent to a database. If you do not receive a confirmation message, please manually send the results (displayed in the console log) to me.

9. If the results have been successfully submitted, you will be directed to a confirmation page. If nothing happens, PLEASE MANUALLY SEND THE RESULTS, displayed in the console log of the web browser.


TASK COMPLETE!
Thank you for your time, the data has been successfully logged. Next Task If you have another task, please continue. The link for the next task is available in the experiment script.

TASK COMPLETE!

D.3 Task List



RADIO CATCHUP
AN INTERACTIVE OBJECT-BASED
RADIO LISTEN AGAIN SERVICE

1449 AM URB UNIVERSITY OF BATH Students' Union

RADIO CATCHUP

Task List

Dissertation by Christian Couch (cgtc20@bath.ac.uk)
Supervised by Fabio Nemetz (F.Nemetz@bath.ac.uk)

TASK 1
Alex **discusses the song Formidable by The Big Moon** and talks about the bands formation and upcoming album. Your task is to navigate to the start of the speech segment where he discusses this.

TASK 2
Claire reads out a letter from a listener and talks about the problem they've raised. Your task is to navigate to the start of this feature, called **"Miss Mallett"**.

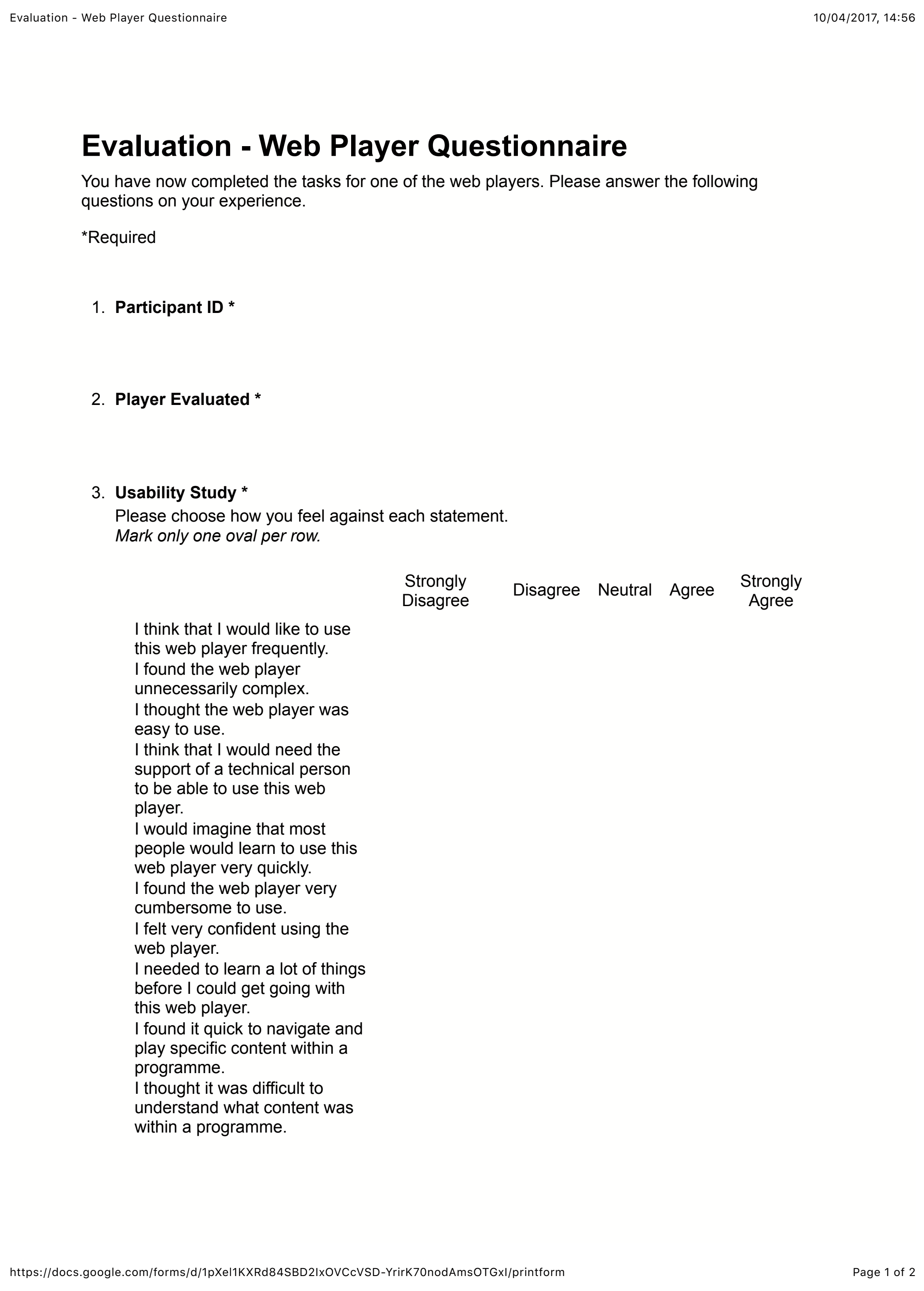
TASK 3
During the programme, Harry discusses is album of the week. Your task is to navigate to the start of this feature **"Harry's Album of the Week"**.

TASK 4
Dean **discusses the song Into the Fire, Into the Sun by AVATARIUM** and talks about the songs release. Your task is to navigate to the start of the speech segment where this is discussed.

TASK 5
Your task is to navigate to the start of the **story about a friendship breakup** that ended with photos being defaced, using Microsoft Paint.

TASK 6
During the show, **Grace plays and reviews, with the help of her Dad, Prince's song Sign O the Times**. Your task is to navigate to the start of this speech segment.

D.4 Web Player Questionnaire



Evaluation - Web Player Questionnaire

You have now completed the tasks for one of the web players. Please answer the following questions on your experience.

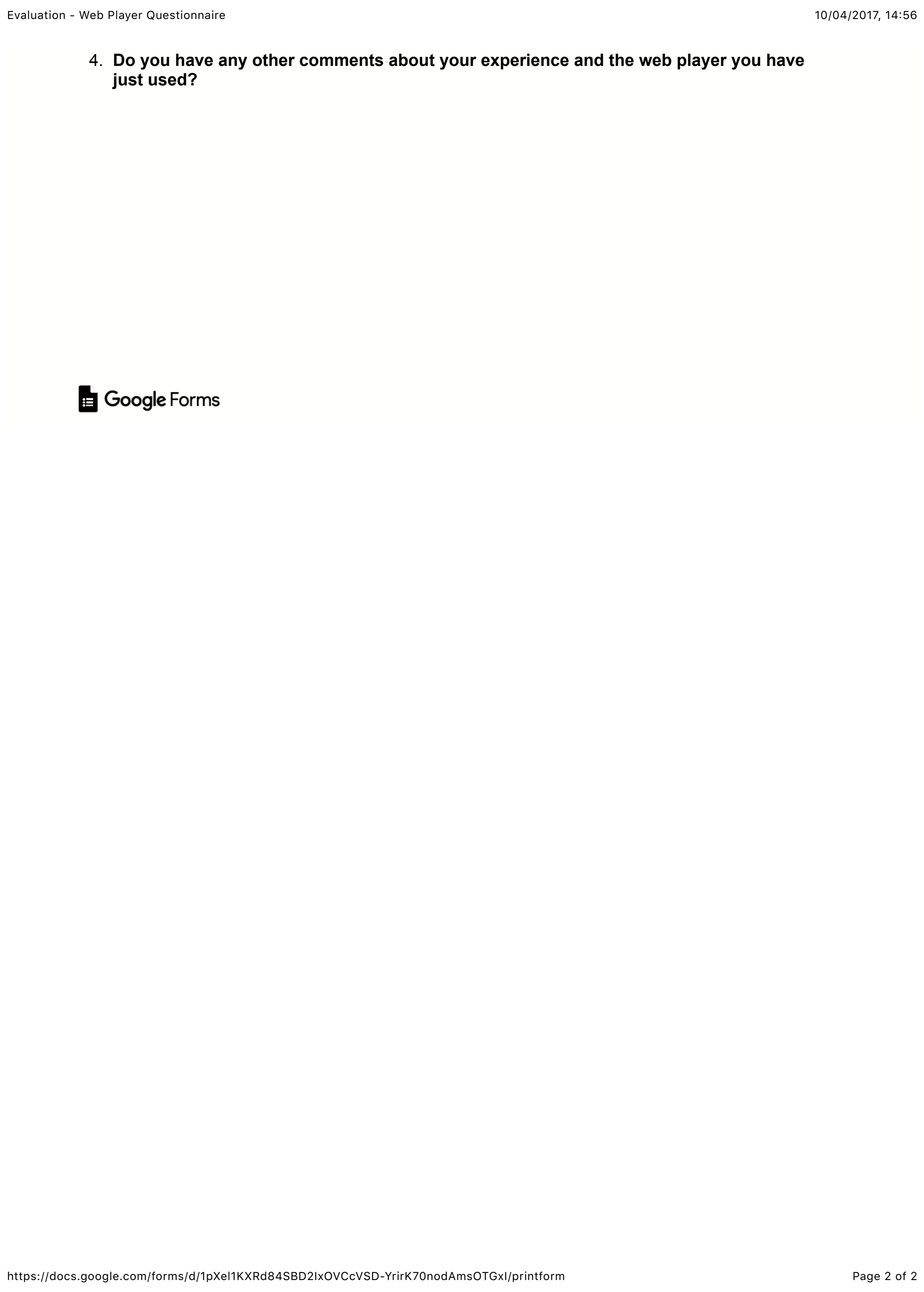
*Required

1. **Participant ID ***

2. **Player Evaluated ***

3. **Usability Study ***
   Please choose how you feel against each statement.
   Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this web player frequently.					
I found the web player unnecessarily complex.					
I thought the web player was easy to use.					
I think that I would need the support of a technical person to be able to use this web player.					
I would imagine that most people would learn to use this web player very quickly.					
I found the web player very cumbersome to use.					
I felt very confident using the web player.					
I needed to learn a lot of things before I could get going with this web player.					
I found it quick to navigate and play specific content within a programme.					
I thought it was difficult to understand what content was within a programme.					

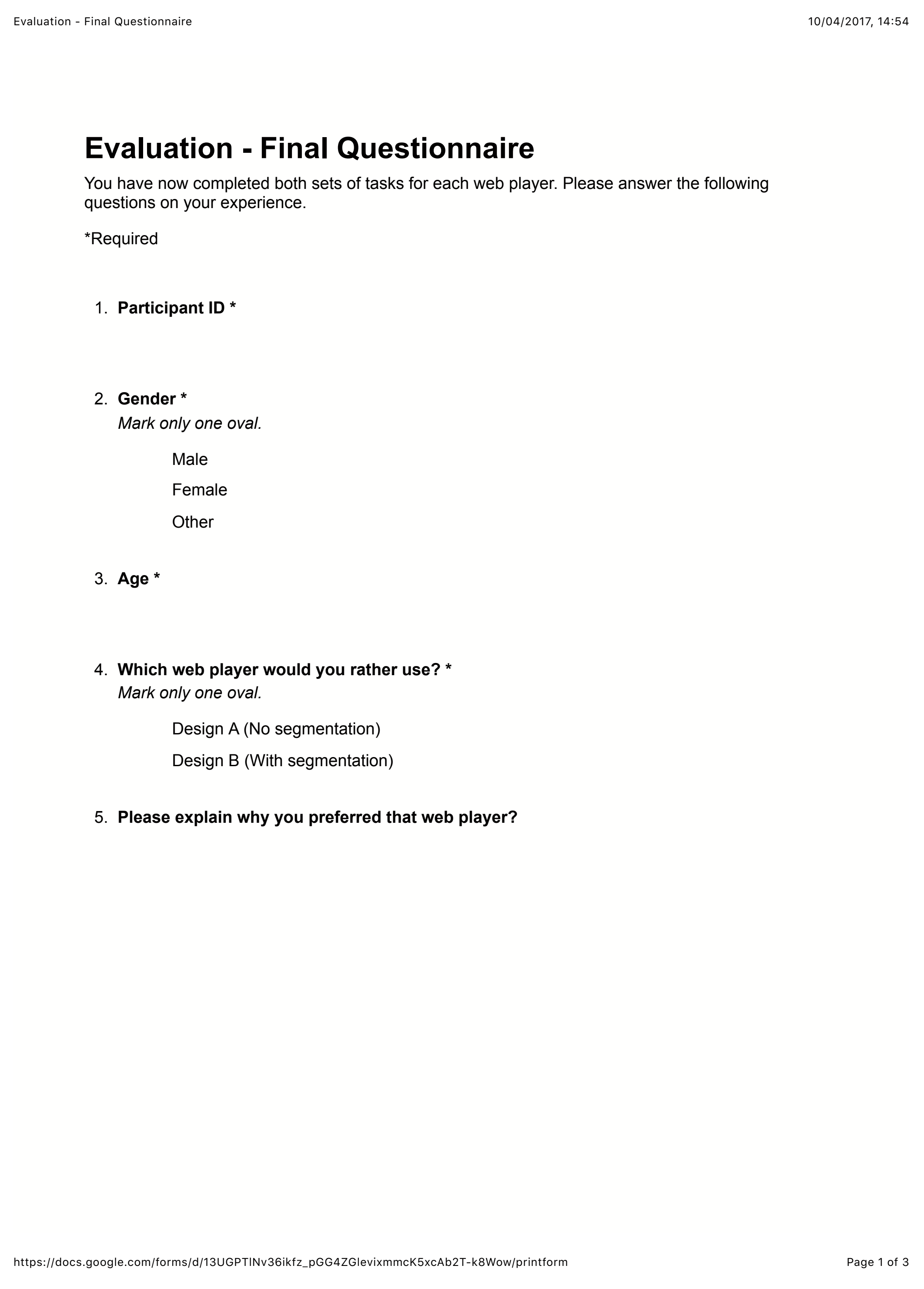


Evaluation - Web Player Questionnaire 10/04/2017, 14:56

4. **Do you have any other comments about your experience and the web player you have just used?**

https://docs.google.com/forms/d/1pXel1KXRd84SBD2IxOVCcVSD-YrirK70nodAmsOTGxI/printform Page 2 of 2

D.5 Final Questionnaire



Evaluation - Final Questionnaire 10/04/2017, 14:54

Evaluation - Final Questionnaire

You have now completed both sets of tasks for each web player. Please answer the following questions on your experience.

*Required

1. **Participant ID ***

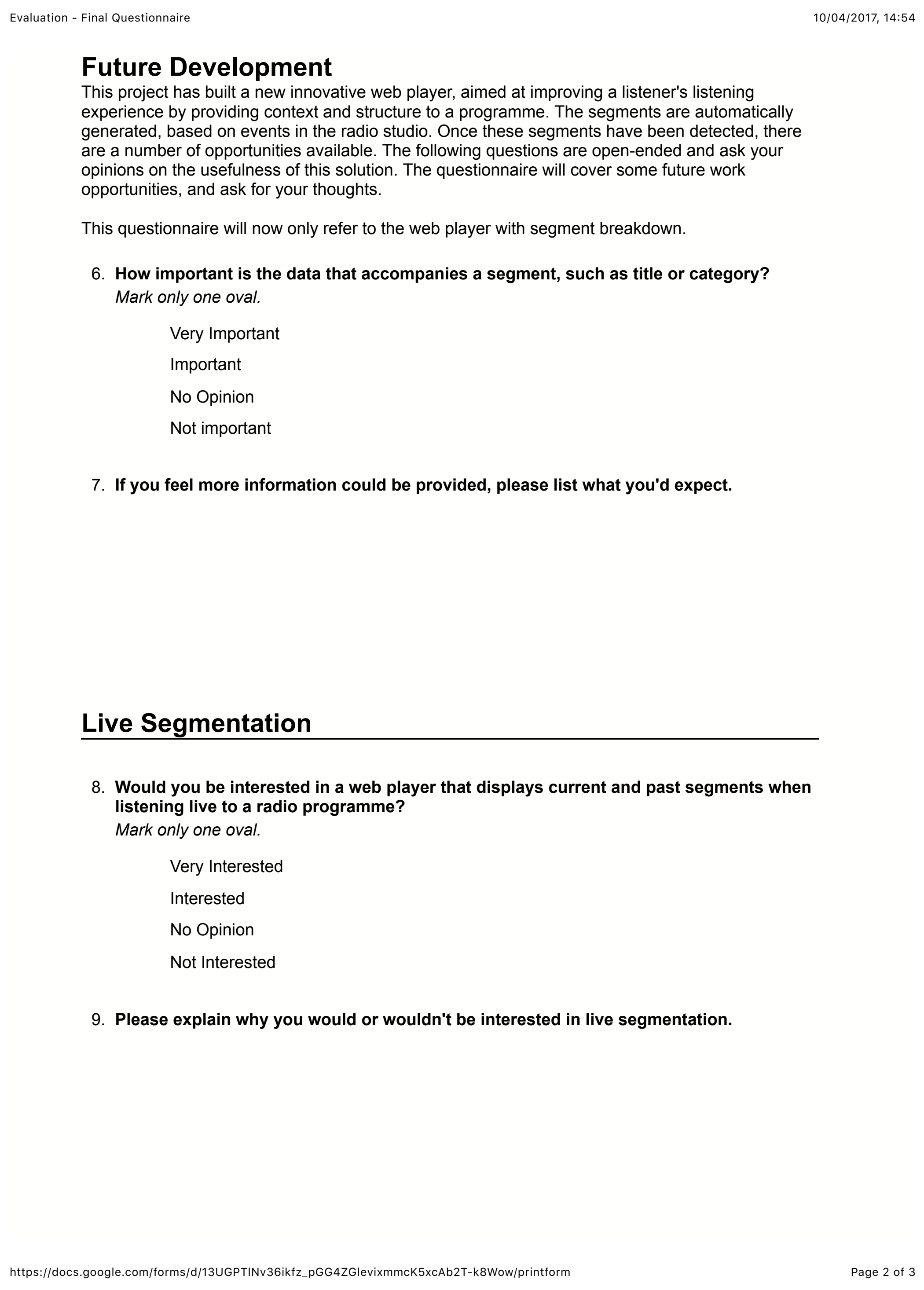
2. **Gender ***
   Mark only one oval.
   - Male
   - Female
   - Other

3. **Age ***

4. **Which web player would you rather use? ***
   Mark only one oval.
   - Design A (No segmentation)
   - Design B (With segmentation)

5. **Please explain why you preferred that web player?**

https://docs.google.com/forms/d/13UGPTlNv36ikfz_pGG4ZGlevixmmcK5xcAb2T-k8Wow/printform Page 1 of 3



Future Development

This project has built a new innovative web player, aimed at improving a listener's listening experience by providing context and structure to a programme. The segments are automatically generated, based on events in the radio studio. Once these segments have been detected, there are a number of opportunities available. The following questions are open-ended and ask your opinions on the usefulness of this solution. The questionnaire will cover some future work opportunities, and ask for your thoughts.

This questionnaire will now only refer to the web player with segment breakdown.

6. **How important is the data that accompanies a segment, such as title or category?**
   Mark only one oval.

   Very Important

   Important

   No Opinion

   Not important

7. **If you feel more information could be provided, please list what you'd expect.**

Live Segmentation

8. **Would you be interested in a web player that displays current and past segments when listening live to a radio programme?**
   Mark only one oval.

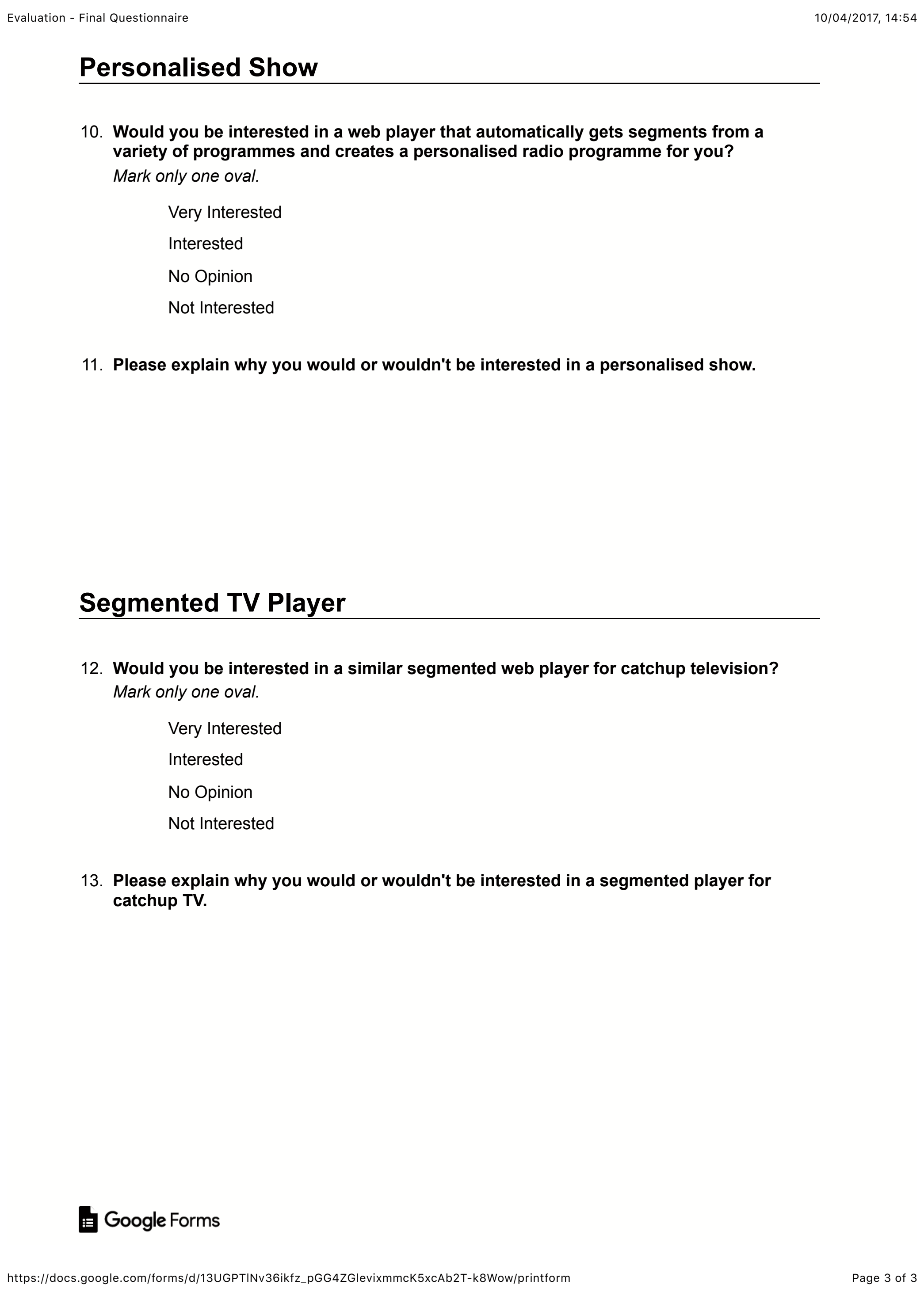
   Very Interested

   Interested

   No Opinion

   Not Interested

9. **Please explain why you would or wouldn't be interested in live segmentation.**



Evaluation - Final Questionnaire 10/04/2017, 14:54

Personalised Show

10. **Would you be interested in a web player that automatically gets segments from a variety of programmes and creates a personalised radio programme for you?**
Mark only one oval.

    Very Interested
    Interested
    No Opinion
    Not Interested

11. **Please explain why you would or wouldn't be interested in a personalised show.**

Segmented TV Player

12. **Would you be interested in a similar segmented web player for catchup television?**
Mark only one oval.

    Very Interested
    Interested
    No Opinion
    Not Interested

13. **Please explain why you would or wouldn't be interested in a segmented player for catchup TV.**

Google Forms

https://docs.google.com/forms/d/13UGPTlNv36ikfz_pGG4ZGlevixmmcK5xcAb2T-k8Wow/printform Page 3 of 3

D.6 Questionnaire Responses

D.6.1 Web Player Questionnaire - Web Player Comments

Design A (non-segmented player)

- **Participant 1 -** Found myself listening to large 'talk' segments within the shows to see if they were what I was looking for when ultimately not. Lots of skipping through tracks trying to find the end points
- **Participant 4 -** First two tasks were fine as you were only searching for specific sections of a programme in a reduced amount of time. Last task was difficult for 2 reasons: increased length of the programme, and increased amount of radio hosts talking on the programme (compared to the previous two tasks where it at least seemed like there was less talking).
- **Participant 6 -** It was difficult to know if you were close to what you were looking for which made it quite frustrating in the longer segments
- **Participant 7 -** Difficult to skip small units of time eg. to find the end of a song / transitions between speaking and music

Design B (segmented player)

- **Participant 1 -** Recognition on segments made it very easy to read through and click to skip straight to the segment being searched for
- **Participant 4 -** Having key words of the talking segments was really useful to quickly identify the specific parts of the programme. Might have been useful to make the scrolling pane of all the different parts of the programme a bit bigger so you have a bigger overview of all the parts on the screen at once rather than having to scroll through it.
- **Participant 6 -** The first time using the web player it took bit of time to work out how it works but from the second task it was very natural to use
- **Participant 7 -** Found it very easy to navigate to different songs and speech content, and have an idea of what is going to be said in the speech sections of a show. Saves a lot of time trying to find the start and end of songs as it navigates quickly to the transition point.
- **Participant 8 -** It was awesome

D.6.2 Web Player Questionnaire - System Usability Scale (SUS)

Table D.1 System Usability Scale (SUS) - Design A

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
I think that I would like to use this web player frequently.	0	6	2	2	0
I found the web player unnecessarily complex.	3	6	0	1	0
I thought the web player was easy to use.	0	1	3	5	1
I think that I would need the support of a technical person to be able to use this web player.	5	5	0	0	0
I would imagine that most people would learn to use this web player very quickly.	0	0	2	6	2
I found the web player very cumbersome to use.	2	1	0	3	4
I felt very confident using the web player.	0	2	2	5	1
I needed to learn a lot of things before I could get going with this web player.	4	3	3	0	0
I found it quick to navigate and play specific content within a programme.	8	1	1	0	0
I thought it was difficult to understand what content was within a programme.	0	0	1	3	6

Table D.2 System Usability Scale (SUS) - Design B

I think that I would like to use this web player frequently.	0	0	1	5	4
I found the web player unnecessarily complex.	5	5	0	0	0
I thought the web player was easy to use.	0	0	0	4	6
I think that I would need the support of a technical person to be able to use this web player.	8	2	0	0	0
I would imagine that most people would learn to use this web player very quickly.	0	0	0	7	3
I found the web player very cumbersome to use.	5	5	0	0	0
I felt very confident using the web player.	0	0	0	6	4
I needed to learn a lot of things before I could get going with this web player.	7	2	1	0	0
I found it quick to navigate and play specific content within a programme.	0	0	0	2	8
I thought it was difficult to understand what content was within a programme.	6	4	0	0	0

Table D.3 User Experiment - Final Questionnaire: Participant Age

Age	Count
20	1
21	3
22	4
23	1

Table D.4 User Experiment - Final Questionnaire: Participant Gender

Gender	Count
Male	7
Female	2
Other	0

Table D.5 User Experiment - Final Questionnaire: Which web player would you rather use?

Web Player	Count
Design A (No segmentation)	0
Design B (With segmentation)	10

D.6.3 Final Questionnaire - General

Please explain why you preferred that web player?

Design A No participants preferred Design A.

Design B

- **Participant 1 -** So much easier to know where talk segments started/ended without skipping through the songs, and the titles of each meant I wasn't listening to lots of the segment to figure out if it was the bit I was searching for.
- **Participant 2 -** It gave you a breakdown of what to expect in the show, so if you were looking for a particular song or type of content it removed the guesswork.
- **Participant 3 -** Easier to navigate to content
- **Participant 4 -** Quicker to identify the parts, I really liked that you could see the breakdown of what is in the show as you can see the tracks played, what the hosts talked about, and perhaps skip over irrelevant parts (or skip to relevant parts) depending on what you are looking for. It made searching very easy.
- **Participant 5 -** It was, approximately, a thousand times easier to find things - also much easier to be confident that you'd found the correct segment.
- **Participant 6 -** I preferred design B because it meant I could see what I was looking for and where it came in the programme relative to other segments. I was more confident using design B as with design a I didn't trust myself to not have skipped it so went really slowly and backtracked sometimes whereas with design B I had confidence that the text had brought me to the right segment

- **Participant 7** - It gives you far superior control over navigating through the content. It opens up the option of listening to specific parts of a show (e.g. speech links or particular songs), which on a non-segmented player is so cumbersome to achieve that you probably wouldn't bother. It also gives you an indication of what is about to be said in speech links, and the option to skip to different tracks if you'd prefer, which could be a convenient form of music discovery (eg. you like Radio 1s playlist & speech links, but want to hear songs you haven't heard before, you can skip to different parts of the show for that purpose).
- **Participant 8** - It was a lot easier to find specific segments within a show that I would want to listen to (highlights, special moments etc.), plus, I could flick through a show to see if it is worth listening to the whole thing or not.
- **Participant 9** - Saves time when trying to locate audio quickly
- **Participant 10** - Very easy to find specific section of show. Easy to learn and understand functionality.

D.6.4 Final Questionnaire - Future Development

Table D.6 User Experiment - Final Questionnaire: How important is the data that accompanies a segment, such as title or category?

Importance	Count
Very Important	5
Important	5
Not important	0
No Opinion	0

If you feel more information could be provided, please list what you'd expect.

- **Participant 1** - Really helped in this test environment - in real life use I could mostly only imagine myself using it to skip to the end of speech segments to get to the next song without really caring what each segment was
- **Participant 4** - (It was just the right amount of information, key words rather than long sentences)
- **Participant 5** - Genre colours?
- **Participant 7** - Titles of features (I can imagine this form of segmentation being used for finding popular speech features in a listeners favourite show for instance).

Live Segmentation

Please explain why you would or wouldn't be interested in live segmentation.

- **Participant 1** (*Very Interested*) - Very quick and simple to see if recent segments of particular interest and skip directly to them
- **Participant 2** (*Very Interested*) - I don't always listen live, and I like being able to skip songs if it's specific show content I want to listen to.
- **Participant 3** (*Interested*) - *No response.*

Table D.7 User Experiment - Final Questionnaire: Would you be interested in a web player that displays current and past segments when listening live to a radio programme?

Importance	Count
Very Interested	4
Interested	6
Not Interested	0
No Opinion	0

- **Participant 4** (*Interested*) - Searching for specific parts of the entire programme would become a great deal easier if you didn't want to painstakingly go through the whole thing to identify what you wanted.
- **Participant 5** (*Interested*) - Generally listen to spotify, but also enjoy radio content, proper segmentation would make it much easier to work the content around own music preferences.
- **Participant 6** (*Interested*) - Important because some segments make more sense in the grand scheme of a programme / it would be useful if you had just missed the beginning of a programme and wanted to catch up there and then
- **Participant 7** (*Very Interested*) - If you are mainly listening for a speech element to a show, then it can be frustrating to wait for long periods of music if it isn't what you are looking for. It could also provide a snapshot to the listener of whether the music and format of the show suits their listening tastes at that time.
- **Participant 8** (*Very Interested*) - Especially for long shows, it helps the presenters when deciding what to say next on air, to space plugs / adverts out throughout the duration of a show, for example.
- **Participant 9** (*Interested*) - *No response.*
- **Participant 10** (*Interested*) - Would be able to quickly jump to a section of the show that I would want to replay.

Personalised Show

Table D.8 User Experiment - Final Questionnaire: Would you be interested in a web player that automatically gets segments from a variety of programmes and creates a personalised radio programme for you?

Importance	Count
Very Interested	1
Interested	7
Not Interested	0
No Opinion	2

Please explain why you would or wouldn't be interested in a personalised show.

- **Participant 1** (*Interested*) - Very quick and simple to see if recent segments of particular interest and skip directly to them

- **Participant 2** (*Interested*) - I don't always listen live, and I like being able to skip songs if it's specific show content I want to listen to.
- **Participant 3** (*No Opinion*) - *No response.*
- **Participant 4** (*Interested*) - Searching for specific parts of the entire programme would become a great deal easier if you didn't want to painstakingly go through the whole thing to identify what you wanted.
- **Participant 5** (*Interested*) - Generally listen to spotify, but also enjoy radio content, proper segmentation would make it much easier to work the content around own music preferences.
- **Participant 6** (*Interested*) - Important because some segments make more sense in the grand scheme of a programme / it would be useful if you had just missed the beginning of a programme and wanted to catch up there and then
- **Participant 7** (*Interested*) - If you are mainly listening for a speech element to a show, then it can be frustrating to wait for long periods of music if it isn't what you are looking for. It could also provide a snapshot to the listener of whether the music and format of the show suits their listening tastes at that time.
- **Participant 8** (*Interested*) - Especially for long shows, it helps the presenters when deciding what to say next on air, to space plugs / adverts out throughout the duration of a show, for example.
- **Participant 9** (*Very Interested*) - *No response.*
- **Participant 10** (*No Opinion*) - Would be able to quickly jump to a section of the show that I would want to replay.

Segmented TV Player

Table D.9 User Experiment - Final Questionnaire: Would you be interested in a similar segmented web player for catchup television?

Importance	Count
Very Interested	2
Interested	4
Not Interested	3
No Opinion	1

Please explain why you would or wouldn't be interested in a segmented player for catchup TV.

- **Participant 1** (*No Opinion*) - Very quick and simple to see if recent segments of particular interest and skip directly to them
- **Participant 2** (*Not Interested*) - I don't always listen live, and I like being able to skip songs if it's specific show content I want to listen to.
- **Participant 3** (*Very Interested*) - *No response.*
- **Participant 4** (*Not Interested*) - Searching for specific parts of the entire programme would become a great deal easier if you didn't want to painstakingly go through the whole thing to identify what you wanted.

- **Participant 5** (*Interested*) - Generally listen to spotify, but also enjoy radio content, proper segmentation would make it much easier to work the content around own music preferences.
- **Participant 6** (*Interested*) - Important because some segments make more sense in the grand scheme of a programme / it would be useful if you had just missed the beginning of a programme and wanted to catch up there and then
- **Participant 7** (*Interested*) - If you are mainly listening for a speech element to a show, then it can be frustrating to wait for long periods of music if it isn't what you are looking for. It could also provide a snapshot to the listener of whether the music and format of the show suits their listening tastes at that time.
- **Participant 8** (*Interested*) - Especially for long shows, it helps the presenters when deciding what to say next on air, to space plugs / adverts out throughout the duration of a show, for example.
- **Participant 9** (*Very Interested*) - *No response.*
- **Participant 10** (*Not Interested*) - Would be able to quickly jump to a section of the show that I would want to replay.

D.7 Experiment Results

D.7.1 User Tasks - Time Taken

Table D.10 User Experiment: Time taken to complete task.

			Player A			**Player B**	
Participant	**Group**	**Task - 10 Min**	**Task - 30 Min**	**Task - 60 Min**	**Task - 10 Min**	**Task - 30 Min**	**Task - 60 Min**
1	2	76	153	102	49	32	24
2	1	59	54	300	32	28	82
3	2	64	73	132	29	20	8
4	1	80	80	300	20	44	40
5	2	31	64	300	39	25	17
6	2	87	143	300	80	29	22
7	1	86	80	191	51	57	48
8	2	81	187	300	65	46	32
9	1	107	81	156	17	69	51
10	1	60	80	300	50	51	53
Average	*1*	*78*	*75*	*249*	*34*	*50*	*55*
Average	*2*	*68*	*124*	*227*	*52*	*30*	*21*
Average		**73**	**100**	**238**	**43**	**40**	**38**

D.7.2 Two-way ANOVA Statistical Analysis

Table D.11 Empirical Evaluation: Two-way ANOVA (*web player design* × *programme duration* conducted on *task completion time*)

Source	***design***	***audio_ duration***	**df**	**Mean Square**	**F**	**Sig.**
design	Linear		1	139876.817	125.047	0.000
Error(design)	Linear		9	1118.594		
audio_duration		Linear	1	63600.625	26.605	0.001
		Quadratic	1	10622.008	8.321	0.018
Error (audio_ duration)		Linear	9	2390.569		
		Quadratic	9	1276.545		
*design * audio_ duration*	Linear	Linear	1	72675.625	30.428	0.000
		Quadratic	1	10360.208	8.999	0.015
Error (design audio_duration)*	Linear	Linear	9	2388.458		
		Quadratic	9	1151.264		

D.7.3 Wilcoxon Statistical Analysis

Table D.12 Empirical Evaluation: Wilcoxon Signed Ranks Test (SUS_DESIGN_B – SUS_DESIGN_A)

	N	**Mean Rank**	**Sum of Ranks**
Negative Ranks	0[a]	.00	.00
Positive Ranks	10[b]	5.50	55.00
Ties	0[c]		
Total	10		

a SUS_DESIGN_B < SUS_DESIGN_A

b SUS_DESIGN_B > SUS_DESIGN_A

c SUS_DESIGN_B = SUS_DESIGN_A

E Code

Due to space limitations, not all the code could be included in the appendix, however the full code is available in the electronic submission. The following code has been included:

1. *server/eventCollection/mic_event_collector.py* - The script which is run on a Raspberry Pi to detect and record when the microphone is live. The script also records the microphone, while it is live, and sends the output to a server for speech to text and text to keyword analysis.
2. *server/segmentProcessor/main.py* - This is the main script which segments a radio programme, using the events stored in various databases. The script analyses the events, determines segments, receives metadata about the segments and finally exports the segments into a JSON file for the webplayer.
3. *player/player.html* - This is the main web player HTML file which displays the segmented listen again player to the audience. Various other files, including CSS and JS, are used to ensure the page displays and functions correctly.
4. *player/src/js/add-on/popcorn.segment.js* - This is one of the main JavaScript files for the web player. It is a custom plugin which interfaces with the *Popcorn.js* library and controls the web player.

E.1 Event Collectors - Mic Live Collector

Listing E.1 *server/eventCollection/mic_event_collector.py*

```python
import sys
import time
import logging
import socket

import RPi.GPIO as GPIO
import MySQLdb

from addons.mic_recorder import MicRecorder

device_name = "URB_S1_MicLive_Collector"
network_name = "undefined"
event_pin = 4
event_wait = 1000

# Database Configuration
log_enabled = True
db = None
cursor = None
host_address = "138.38.*.*"
port_number = 3306
username = "radio"
password = "***"
database_name = "RadioCatchup"

event_table = "CATCHUP_EVENT"
message_table = "CATCHUP_LOG"

# Event Collection - Add ons
recorder = None
recorder_device = 2
recorder_rate = 44100
recorder_channel = 1
recorder_chunk = 4096
recorder_min_seconds = 5
recorder_max_seconds = 180
recording_location = "/home/pi/RadioCatchup/server/
    eventCollection/recorded"
recording_end_script = "/home/pi/RadioCatchup/server/
    eventCollection/addons/upload_to_server.sh"

logging.basicConfig(level=logging.DEBUG, filename='/home
    /pi/RadioCatchup/server/eventCollection/
    eventCollection.log')


def log_event(type, timestamp, active, text):
    global device_name, network_name

    if not log_enabled: return False

    add_event = ("INSERT INTO " + event_table +
                 "(device, network, type, timestamp,
    active, text) "
                 "VALUES (%(device)s, %(network)s, %(
    type)s, %(timestamp)s, %(active)s, %(text)s)")

    data_event = {
        'device': device_name,
        'network': network_name,
            'type': type,
            'timestamp': timestamp,
            'active': active,
            'text': text,
    }

    cursor.execute(add_event, data_event)
    db.commit()
    row_id = cursor.lastrowid
    print("EventCollection: Event added (id-" + str(
    row_id) + "): ", data_event)

    return row_id


def log_message(type, timestamp, msg):
    global device_name

    if not log_enabled: return False

    add_message = ("INSERT INTO " + message_table +
                   "(device, network, type, timestamp,
    msg) "
                   "VALUES (%(device)s, %(network)s, %(
    type)s, %(timestamp)s, %(msg)s)")

    data_message = {
            'device': device_name,
            'network': network_name,
            'type': type,
            'timestamp': timestamp,
            'msg': msg,
    }

    cursor.execute(add_message, data_message)
    db.commit()
    row_id = cursor.lastrowid
    print("EventCollection: Message logged (id-" + str(
    row_id) + "): ", data_message)

    return row_id


def mic_live_active(timestamp):
    mic_start_record(timestamp)
    log_event(eventTypes["mic_live_active"], timestamp,
    eventActiveStatus["mic_live_active"], "")


def mic_live_not_active(timestamp):
    mic_end_recording(timestamp)
    log_event(eventTypes["mic_live_not_active"],
    timestamp, eventActiveStatus["mic_live_not_active"],
    "")


def mic_start_record(timestamp):
    global recorder, recording_location

    recording_id = str(timestamp)
    log_message(eventTypes["mic_live_active"], timestamp
    , "Starting recording")

    if recorder is not None:
        recorder.end_recording()
        recording_id += "-DUP"
        #    LOG RECORDER STARTED WHILE RECORDING
        log_message(timestamp)

    recorder = MicRecorder(recording_id, recording_location, recording_end_script, device=recorder_device,
                                rate=recorder_rate, channels=recorder_channel, chunk=recorder_chunk,
                                max_seconds=recorder_max_seconds, min_seconds=recorder_min_seconds)
    recorder.start()


def mic_end_recording(timestamp):
    global recorder

    if recorder is not None:
        log_message(eventTypes["mic_live_not_active"], timestamp, "Ending recording")
        recorder.end_recording()
        recorder = None


def event_detected(channel):
    state = GPIO.input(channel)
    timestamp = int(time.time())

    if state and channel is event_pin:
        mic_live_active(timestamp)
    elif not state and channel is event_pin:
        mic_live_not_active(timestamp)
    else:
        pass

eventTypes = {
    "status": 0,
    "mic_live_active": 1,
    "mic_live_not_active": 1
}

eventActiveStatus = {
    "mic_live_active": 1,
    "mic_live_not_active": 0
}

command = {
    0: mic_live_active,
    1: mic_live_not_active
}


def main(argv):
    global db, cursor, device_name, network_name

    network_name = socket.gethostname()

    try:
        print("EventCollection: Attempting to connect to DB: " + host_address + "/" + database_name)
        db = MySQLdb.connect(host=host_address, port=port_number, user=username, passwd=password,
                                db=database_name, connect_timeout=5)
        cursor = db.cursor()

    except MySQLdb.Error as err:
        print("EventCollection: Error connecting to DB: " + host_address + " " + database_name)
        print(err)

    else:
        log_message(eventTypes["status"], int(time.time()), "Event collector starting")

        GPIO.setmode(GPIO.BCM)
        GPIO.setup(event_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        GPIO.add_event_detect(event_pin, GPIO.BOTH, callback=event_detected, bouncetime=event_wait)

        while True:
            pass

        log_message(eventTypes["status"], int(time.time()), "Event collector closing")

        cursor.close()
        db.close()


if __name__ == "__main__":
    try:
        main(sys.argv)
    except:
        print("EventCollection: Error (see log file for more details)")
        logging.exception(str(int(time.time())) + "-EventCollection")
```

E.2 Segment Processor

Listing E.2 *server/segmentProcessor/main.py*

```python
import datetime
import json
import logging
import sys
import time
import requests

from server.segmentProcessor.informationSupplier import
    mic_live_retriever, play_log_retriever,
    programme_data_retriever

logging.basicConfig(level=logging.DEBUG, filename='
    segmentProcessor.log')


def generate_programme_data(series, episode, start_date,
     end_date, presenters, description):
    programme_data = dict()
    programme_data['seriesName'] = series
    programme_data['episodeName'] = episode
    programme_data['broadcastStart'] = time.mktime(
    start_date.timetuple())
    programme_data['broadcastEnd'] = time.mktime(
    end_date.timetuple())
    programme_data['presenters'] = presenters
    programme_data['description'] = description

    return programme_data


def generate_media_data(artist, title, poster, mp3):
    media_data = dict()
    media_data['artist'] = artist
    media_data['title'] = title
    media_data['poster'] = poster
    media_data['mp3'] = mp3

    return media_data


def generate_segment_data(start_date, end_date):
    segments = []

    speech_segments = mic_live_retriever.
    retrieve_segments(start_date, end_date)
    segments.extend(speech_segments)
    # playout_audio_segments includes music and advert
    segments
    playout_audio_segments = play_log_retriever.
    retrieve_segments(start_date, end_date)
    segments.extend(playout_audio_segments)
    # news_segments = NOT IMPLEMENTED

    # Sorts the segments based on their start time
    segments = sorted(segments, key=lambda x: x['options
    ']['start'], reverse=False)

    return segments


```

```
def download_show_logging(destination, start_date, end_date):
    start = start_date.strftime("%Y%m%d%H%M%S")
    end = end_date.strftime("%Y%m%d%H%M%S")

    # Download logging URL
    url = "http://138.38.*.*/cgi/get_media.cgi?format=mpeg128_joint_stereo;channel_id=142" \
          ";start_timestamp=" + start + ";" \
          "end_timestamp=" + end + ";"

    d = start_date.strftime("%d")
    m = start_date.strftime("%m")
    Y = start_date.strftime("%Y")
    H = start_date.strftime("%H")
    i = start_date.strftime("%M")
    s = start_date.strftime("%S")
    dur = str(((end_date - start_date).seconds / 60))

    # channel = "142" # URB Output
    channel = "22728" # S1 Output

    print("SegmentProcessor: Requesting logging from nonstop.")
    nonstop_url = "http://nonstop.urb.bath.ac.uk:8000/ns/logging/request.php?d="+d+"&m="+m+"&Y="+Y+"&H="+H+"&i="+i+"&s="+s+"&dur="+dur+"&channel="+channel
    r = requests.get(nonstop_url)
    nonstop_file = r.text[19:-3].replace("\\", "").replace(" ", "%20")
    print("SegmentProcessor: Logging successfully downloaded to nonstop - " + nonstop_file)

    print("SegmentProcessor: Downloading logging from nonstop.")
    r = requests.get(nonstop_file, stream=True)
    with open(destination, 'wb') as f:
        for chunk in r.iter_content(chunk_size=1024):
            if chunk:
                f.write(chunk)
    print("SegmentProcessor: Logging successfully downloaded locally - " + destination)

    return destination


def main(args):
    if len(args) == 4:
        # EXAMPLE: python main.py "21/11/06 15:00" "21/11/06 16:00" "breakfast_show"
        start_date = datetime.datetime.strptime(args[1], "%d/%m/%y %H:%M")
        end_date = datetime.datetime.strptime(args[2], "%d/%m/%y %H:%M")
        filename = args[3]
    else:
        # TEST (script called without parameters)
        start_date = datetime.datetime(2017, 3, 30, 20, 59, 00)
        end_date = datetime.datetime(2017, 3, 30, 21, 9, 00)
        filename = "test"

    print "SegmentProcessor: Processing date range " + start_date.strftime("%d/%m/%y %H:%M") + " - " + end_date.strftime("%d/%m/%y %H:%M")

    show_id = programme_data_retriever.retrieve_programme_scheduled(start_date, end_date)
    prog_info = programme_data_retriever.retrieve_programme_information(show_id)
    print ("SegmentProcessor: Retrieved show " + str(show_id) + " programme information - " + str(prog_info))

    series = prog_info["series"]
    episode = prog_info["episode"]
    description = prog_info["description"]
    presenters = programme_data_retriever.retrieve_programme_presenters(show_id)

    poster = "img/default-poster.png"
    mp3 = "data/shows/" + filename + ".mp3"

    length = (start_date - end_date).seconds
    mp3_destination = "/Users/Chris/Programmes/Workspace/RadioCatchup/player/" + mp3
    json_destination = "/Users/Chris/Programmes/Workspace/RadioCatchup/player/data/shows/" + filename + ".json"

    download_show_logging(mp3_destination, start_date, end_date)

    data = dict()
    data['programme'] = generate_programme_data(series, episode, start_date, end_date, presenters, description)
    data['media'] = generate_media_data(series, episode, poster, mp3)
    data['plugins'] = generate_segment_data(start_date, end_date)

    with open(json_destination, 'w') as outfile:
        json.dump(data, outfile)

    print ("SegmentProcessor: Show successfully processed.")
    print ("SegmentProcessor: AUDIO - " + str(mp3_destination))
    print ("SegmentProcessor: JSON - " + str(json_destination))


if __name__ == "__main__":
    try:
        main(sys.argv)
    except:
        print("SegmentProcessor: Error (see log file for more details)")
        logging.exception(str(time.time()) + "-SegmentProcessor")
```

E.3 Web Player - HTML

Listing E.3 *player/player.html*

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<title>URB Radio Catchup - The Hangover Cure</title>

<!-- CSS Imports -->
<link href="dist/css/skin.urb.min.css">
<link href="lib/css/bootstrap.min.css">
<link href="lib/css/font-awesome.min.css">
<link href="lib/css/font-awesome-animation.min.css">

<!-- JS Imports -->
<script src="lib/js/bootstrap.min.js"></script>
<script src="lib/js/jquery.jplayer.min.js"></script>
<script src="lib/js/jquery.jplayer.min.js"></script>
<script src="lib/js/popcorn.ie8.js"></script>
<script src="lib/js/popcorn.js"></script>
<script src="lib/js/popcorn.player.js"></script>
<script src="lib/js/popcorn.jplayer.js"></script>
<script src="dist/radioCatchup.min.js"></script>
<script type="text/javascript">
var player;

$(document).ready(function(){
  var player = setupCatchupPlayer("#player_1", "timeline-container", "seekbar-container",
          "data/shows/hangover-cure-2017-02-24.json", {});
});
</script>

</head>
<body>
  <div class="jp-container">
    <div class="programme-header col-xs-12">
      <div class="brand-logo hidden-xs">
        <img src="img/URBIcon.png">
      </div>
      <div  class="programme-information" style="display: inline-block">
        <div class="programme-information-title" id="programme-title">The Hangover Cure</div>
        <div class="programme-information-subtitle" id="programme-subtitle">with Ollie Brookes</div>
        <div class="programme-information-date-time">
          <i class="fa fa-clock-o" aria-hidden="true"></i>
          <span class="time" id="programme-time">16:00 - 18:00</span>
          <i class="fa fa-calendar-o" aria-hidden="true"></i>
          <span class="date" id="programme-date">Tuesday 24th February 2017</span>
        </div>
      </div>
    </div>
    <div id="jp_container_1" class="jp-player-container">
      <div class="jp-player col-xs-12 col-sm-8 col-equal-width">
        <div class="jp-player-video">
          <div id="player_1" class="jp-jplayer"></div>
          <div class="jp-gui">
            <div class="jp-interface">
              <div class="jp-progress">
                <div class="jp-seek-bar">
                  <div id="seekbar-container" class="jp-seek-elements"></div>
                  <div class="jp-play-bar" style="width: 27.53%;">
                    <div class="jp-play-bar-cursor"></div>
                  </div>
                </div>
              </div>
              <div class="jp-controls">
                <button class="fa fa-play fa-fw fa-2x align-left jp-play" aria-hidden="true" role="button"
                    tabindex="0"></button>
                <button class="fa fa-pause fa-fw fa-2x align-left jp-pause" aria-hidden="true" role="button"
                    tabindex="0"></button>
                <button class="fa fa-step-forward fa-fw fa-2x align-left jp-step-forward" aria-hidden="true" role="button"
                  tabindex="2"></button>

              <span style="color: white; float: right; padding: 0 3%;">
                <span class="jp-current-time"></span>
                /
                <span class="jp-duration"></span>
              </span>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="jp-sidebar col-xs-12 col-sm-4 col-equal-width">
      <div class="sidebar-container" id="sidebar-container">
        <div id="img-sidebar" class="sidebar-background"></div>
        <div class="sidebar-header">Welcome to Radio Recap!</div>
        <div class="sidebar-body">
          <div class="sidebar-image-container col-xs-4 col-sm-12">
            <img class="sidebar-image img-responsive" src="img/URBIcon.png">
          </div>
          <div class="sidebar-info col-xs-6 col-sm-12">
            <div class="sidebar-info-title">Press play to start.</div>
            <div class="sidebar-info-subtitle"></div>
            <div class="sidebar-info-text"></div>
            <div class="sidebar-info-links col-xs-6 col-sm-12"></div>
          </div>
          </div>
        </div>
      </div>
    </div>
    <div class="col-xs-12 jp-timeline">
      <div class="timeline-header">
        <div style="display:inline-block">Programme Timeline</div>
      </div>
      <div id="timeline-container" class="timeline-container" style="max-height: 1000px;"></div>
    </div>
  </div>
</body>

</html>
```

E.4 Web Player - Plugin JS

Listing E.4 *player/src/js/add-on/popcorn.segment.js*

```
// Popcorn.js plugin: Segment
(function ( Popcorn ) {
  var
      defaultImageURL = "img/URBIcon.png",
      defaultSeekbarContainerID = "seekbar-container",
      defaultTimelineContainerID = "timeline-container",
      defaultSidebarContainerID = "sidebar-container",
      seekbarClassesToAdd = "jp-seek-element",
      timelineClassesToAdd = "list-group-item timeline-segment",
      timelineTitleClassesToAdd = "timeline-title",
      timelineSubtitleClassesToAdd = "timeline-subtitle",
      timelineDisplayTimeClassesToAdd = "timeline-display-time",
      timelinePlayIconClassesToAdd = "timeline-play-icon fa fa-fw",
      sidebarBackgroundClass = ".sidebar-background",
      sidebarHeaderClass = ".sidebar-header",
      sidebarImageClass = ".sidebar-image",
      sidebarTitleClass = ".sidebar-info-title",
      sidebarSubtitleClass = ".sidebar-info-subtitle",
      sidebarTextClass = ".sidebar-info-text",
      sidebarLinksClass = ".sidebar-info-links";

  var refreshSegmentSeekbar = function(element, segmentOptions, mediaDuration) {
    var elementClassName = seekbarClassesToAdd,

      segmentDuration = segmentOptions.end - segmentOptions.start,
      segmentSize= (segmentDuration) / mediaDuration * 100,
      segmentPosition = (segmentOptions.start / mediaDuration) * 100;

    // Set classes for certain content
    if (segmentOptions.skipItem == true) {
      elementClassName += " skipped";
    }
    if (segmentOptions.disabled == true) {
      elementClassName += " disabled";
    }
    element.className = elementClassName + " segment-" + segmentOptions.type;

    // Update segment GUI - Recalculate and set width and left position
    element.style.width = segmentSize + "%";
    element.style.left = segmentPosition + "%";

    //Set tooltip
    if (segmentOptions.title != "") {
      var tooltip = segmentOptions.title;
      if (segmentOptions.subtitle != "") {
        tooltip += " - " + segmentOptions.subtitle;
      }
      element.setAttribute('title', tooltip);
      element.setAttribute('data-toggle', 'tooltop');
      $(element).tooltip();
    }

  };

  var refreshSegmentTimeline = function(element, elementTitle, elementSubtitle,
                                        elementDisplayTime, elementPlayIcon,
                                        segmentOptions, context) {

    var elementClassName = timelineClassesToAdd;
    var elementTitleClassName = timelineTitleClassesToAdd;
    var elementSubitleClassName = timelineSubtitleClassesToAdd;
    var elementDisplayTimeClassName = timelineDisplayTimeClassesToAdd;
    var elementPlayIconClassName = timelinePlayIconClassesToAdd;

    // Set classes for certain content
    if (segmentOptions.skipItem == true) {
      elementClassName += " skipped";
    }
    if (segmentOptions.disabled == true) {
      elementClassName += " disabled";
    }

    element.className = elementClassName + " timeline-" + segmentOptions.type;
    elementPlayIcon.className = elementPlayIconClassName;

    // Set event listener
    element.onclick = function(){
      if (!segmentOptions.disabled) {
        context.currentTime(segmentOptions.start);
      }
    };

    if (segmentOptions.displayStart != "") {
      elementDisplayTime.className = elementDisplayTimeClassName;
      elementDisplayTime.innerHTML = segmentOptions.displayStart;
    }

    if (segmentOptions.title != "") {
      elementTitle.innerHTML = segmentOptions.title;
      elementTitle.className = elementTitleClassName;
    }

    if (segmentOptions.subtitle != "") {
      elementSubtitle.className = elementSubitleClassName;
      elementSubtitle.innerHTML = segmentOptions.subtitle;
    } else if (segmentOptions.displayKeywords != "") {
      elementSubtitle.className = elementSubitleClassName;
      keywords = segmentOptions.displayKeywords;
      console.log(keywords.length);
      if (keywords.length > 10) {
        keywords = keywords.substring(0, 15) + "...";
      }
      elementSubtitle.innerHTML = segmentOptions.displayKeywords;
    }
  };

  var refreshSegmentSidebar = function (element, segmentOptions) {
    var sidebarBackgroundElement = $(element).find(sidebarBackgroundClass),
        sidebarHeaderElement = $(element).find(sidebarHeaderClass),
        sidebarImageElement = $(element).find(sidebarImageClass),
        sidebarTitleElement = $(element).find(sidebarTitleClass),
        sidebarSubtitleElement = $(element).find(sidebarSubtitleClass),
        sidebarTextElement = $(element).find(sidebarTextClass),
        sidebarLinksElement = $(element).find(sidebarLinksClass);

    if (sidebarBackgroundElement.length){
      var imageURL = "";
      if ('displayPicture' in segmentOptions && segmentOptions.displayPicture != "") {
        imageURL = segmentOptions.displayPicture;
      }
      $(sidebarBackgroundElement).css('background-image', 'linear-gradient(rgba(255, 255, 255, 0.4), ' +
            'rgba(255, 255, 255, 0.5) ), url("' + imageURL + '")')
    }

    if (sidebarHeaderElement.length){
      var headerText = ""
      switch (segmentOptions.type) {
          case "music":
            headerText = "Now playing...";
            break;
          case "news":
            headerText = "URB News";
            break;
          case "speech":
            if (typeof radioCatchupShowTitle !== 'undefined') {
              headerText = radioCatchupShowTitle
            }
            break;
          case "advert":
            headerText = "Advert";
            break;
          default:
            headerText = "URB Catchup";
            break;
      }
      $(sidebarHeaderElement).html(headerText);
    }

    if (sidebarImageElement.length) {
      var imageURL = "";
      if ('displayPicture' in segmentOptions && segmentOptions.displayPicture != "") {
        imageURL = segmentOptions.displayPicture;
      } else {
        imageURL = defaultImageURL;
      }
      $(sidebarImageElement).attr('src', imageURL);
    }

    if (sidebarTitleElement.length) {
      var title = "";
      if ('title' in segmentOptions && segmentOptions.title != "") {
        title = segmentOptions.title;
    } else if (typeof radioCatchupShowTitle !== 'undefined') {
      title = radioCatchupShowTitle
    }
    $(sidebarTitleElement).html(title);
  }

  if (sidebarSubtitleElement.length) {
    var subtitle = "";
    if ('subtitle' in segmentOptions && segmentOptions.subtitle != "") {
      subtitle = segmentOptions.subtitle;
    } else if (typeof radioCatchupShowSubtitle !== 'undefined') {
      subtitle = radioCatchupShowSubtitle
    }
    $(sidebarSubtitleElement).html(subtitle);
  }

  if (sidebarTextElement.length) {
    var displayKeywords = "";
    if ('displayKeywords' in segmentOptions && segmentOptions.displayKeywords != "") {
      displayKeywords = segmentOptions.displayKeywords;
    }
    $(sidebarTextElement).html(displayKeywords);
  }

  if (sidebarLinksElement.length) {
    var urlLink = "";
    if ('urlLink' in segmentOptions && segmentOptions.urlLink != "") {
      urlLink = segmentOptions.urlLink;
      }
    }
};

var centreCurrentSegmentTimeline = function (element, container, segmentOptions) {
  container.scrollTop = element.offsetTop - 50;
};

var nextSegmentTime = function(context) {
  var segments = context.data.trackRefs;
  var currentTime = context.currentTime();
  var newTime = context.duration();

  for (var segment in segments) {
    var segmentStartTime = segments[segment].start;

    if (segmentStartTime >= currentTime && segmentStartTime < newTime) {
      newTime = segmentStartTime;
    }
  }

  return newTime;
}

Popcorn.plugin( "segment" , function( options ) {
  // DECLARE VARIABLES
  var seekbarContainerID = (options.seekbarContainer) ?
        (options.seekbarContainer) : defaultSeekbarContainerID,
      seekbarContainer = document.getElementById(seekbarContainerID),
      seekbarElement,

      timelineContainerID = (options.seekbarContainer) ?
          (options.timelineContainer) : defaultTimelineContainerID,
      timelineContainer = document.getElementById(timelineContainerID),
      timelineElement,
      timelineElementTitle,
      timelineElementSubtitle,
      timelineElementDisplayTime,
      timelineElementPlayIcon,

      segmentPlayingContainerID = (options.segmentPlayingContainer) ?
          (options.segmentPlayingContainer) : defaultSidebarContainerID,
      segmentPlayingContainer = document.getElementById(segmentPlayingContainerID),
      segmentPlayingElement;

  return {

    _setup: function( track ) {
      // Convert relevant parameters to floats
      options.start = parseFloat(options.start);
      options.end = parseFloat(options.end);

      // Ensure containers exist, otherwise can't add any elements.
      if (seekbarContainer) {
        // SEEKBAR ELEMENT
        seekbarElement = document.createElement( "div" );
        seekbarElement.id = "SB-" + options._id;
        seekbarContainer.appendChild(seekbarElement);
      }
      if (timelineContainer) {
        // TIMELINE ELEMENT
        timelineElement = document.createElement("a");
        timelineElement.id = "TL-" + options._id;
        timelineContainer.appendChild(timelineElement);

        // ELEMENT PARTS
        timelineElementTitle = document.createElement("span");
        timelineElementSubtitle = document.createElement("span");
        timelineElementDisplayTime = document.createElement("span");
        timelineElementPlayIcon = document.createElement("i");
        timelineElement.appendChild(timelineElementDisplayTime);
        timelineElement.appendChild(timelineElementTitle);
        timelineElement.appendChild(timelineElementSubtitle);
        timelineElement.appendChild(timelineElementPlayIcon);
      }
      if (segmentPlayingContainer) {
        // SIDEBAR ELEMENT
        segmentPlayingElement = document.createElement( "div" );
        segmentPlayingElement.id = "SP-" + options._id;
        segmentPlayingContainer.appendChild(segmentPlayingElement);
      }
    },

    _update: function( track, updates ) {
    },

    start: function( event, options ) {
      // 1. Check item can be played
      if (options.disabled == true || options.skipItem == true){
        this.currentTime(nextSegmentTime(this));
      } else {
        $(timelineElement).addClass('segment-playing');
      }
      // 2. Update segment sidebar UI
      refreshSegmentSidebar(segmentPlayingContainer, options);
      // 3. Update segment timeline UI
      centreCurrentSegmentTimeline(timelineElement, timelineContainer, options);
    },

    durationchange: function ( event ) {
      refreshSegmentSeekbar(seekbarElement, options, this.duration());
      refreshSegmentTimeline(timelineElement, timelineElementTitle, timelineElementSubtitle,
          timelineElementDisplayTime, timelineElementPlayIcon,
          options, this);
    },

    end: function( event, options ) {
        $(timelineElement).removeClass('segment-playing');
        refreshSegmentSidebar(segmentPlayingContainer, {});
      },

      _teardown: function( options ) {
      }
    };
  },
  {
    about: {
    name: "Popcorn Segment Plugin",
    version: "0.1",
    author: "Chris Couch",
    website: "chris.io"
  },
    options: {
      timelineContainer: "timeline-container",
      seekbarContainer: "seekbar-container",
      segmentPlayingContainer: "segment-playing-container",

      title: {
        elem: "input",
        type: "text",
        label: "Title"
      },
      // ...
    }
  });

})( this.Popcorn );
```