Query-by-playing application (app)

Final Report

J.H. Mervitz
12014223

Submitted as partial fulfilment of the requirements of Project EPR402
in the Department of Electrical, Electronic and Computer Engineering
University of Pretoria

November 2017

Study leader: Prof J.P. Jacobs

Part 1. Preamble

This report describes the work I did in designing and developing a query-by-playing mobile application, using MATLAB and Android platforms.

Project proposal and technical documentation
This main report contains a copy of the approved Project Proposal (Part 3 of the report) and technical documentation (Part 5 of the report). The latter provides details of the datasets, references, and software code. I have included this appendix on a CD or DVD that accompanies this report.

Project history
The design, simulation and final implementation of this project was performed in MATLAB and in Android. The four main algorithms featured in the project have been designed and tested in MATLAB, and are based on existing algorithms. Any other accompanying algorithms are entirely my own.

Language editing
This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was ________________________________.

________________________________ ________________

Language editor signature *Date*

Declaration

I, ________________________________ understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the Internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else.

Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.

J.H. Mervitz	Date

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ACF	Auto Correlation Function
CSV	Comma Separated Values
dB	Decibels
DTW	Dynamic Time Warping
FFC	Fundamental Frequency Change
FFT	Fast Fourier Transform
GP	Gaussian Pulse
GPDE	Gross Pitch Detection Error
GUI	Graphical User Interface
HMM	Hidden Markov Models
HPS	Harmonic Product Spectrum
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MRR	Mean Reciprocal Rate
NPP	Non-pitched Percussive
OSS	Onset Signal Strength
P	Precision
PDA	Pitch Detection Algorithm
PNP	Pitched Non-percussive
PP	Pitched Percussive
PS	Phase Slope
QBH	Query-by-Humming
QBP	Query-by-Playing
R	Recall
SDK	Software Development Kit
SF	Spectral Flux
STFT	Short Time Fourier Transform
SW	Smith-Waterman
UAT	User Acceptance Test
UI	User Interface
UIDS	User Interface Development System
XML	Extensible Markup Language

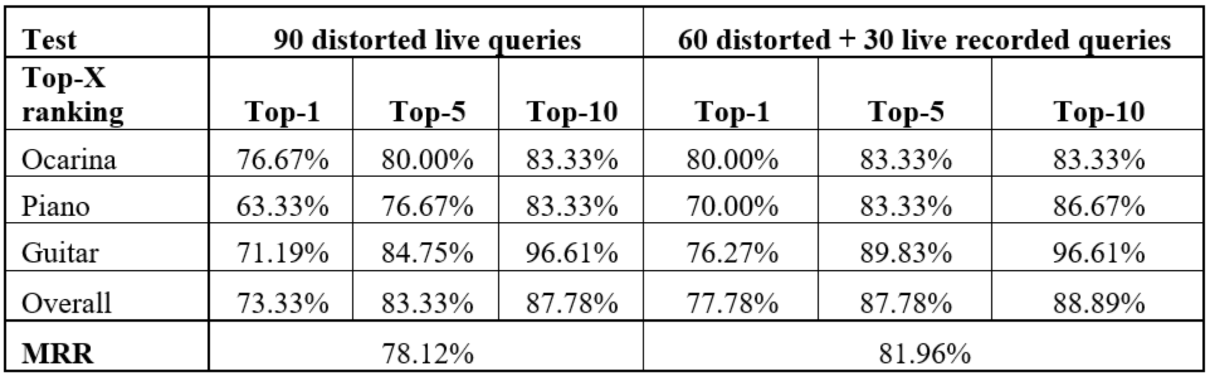
Part 2. Summary

nte

This report details the work done on the design and implementation of a Query-by-Playing (QBP) system, where the objective is to identify song titles when imperfect queries are played using a guitar, piano or woodwind (in this case an ocarina) instrument.

What has been done

A literature survey was compiled on Query-by-Humming (QBH) systems and all the relevant subsystems that follow. The subsystems included onset and pitch detection, feature representations and similarity matching methods. The onset and pitch detection algorithms were firstly designed, implemented and simulated in MATLAB. The procedure adopted for testing and correcting design issues comprised of using tools such as Sonic Visualizer and MidiIllustrator to produce queries and the relevant annotations. The feature representation obtained from the literature namely, onset ratios and pitch intervals was utilised. The matching algorithm was also designed, implemented and simulated in MATLAB. Then the onset and pitch detection algorithms were integrated into one functional system. The database of 500 MIDI songs were adapted from MIDI files downloaded from the Internet. The database was tailored to the specific needs of the project i.e. the vocal track was extracted and stored for each MIDI file. Once the system was fully functional in MATLAB it was ported to Android whereby a Java implementation was done with identical coding logic as in the MATLAB implementation. The testing strategy used for Android and MATLAB comprised of simulating two groups, the first group simulated was 90 perfect queries and the second group simulated was 90 imperfect queries (16.67 % distortions). The overall User Acceptance Test (UAT) was done with 90 live imperfect queries of which 30 were performed by musicians and pre-recorded. The UAT results are shown below in **Table 1**.



Test	**90 distorted live queries**			**60 distorted + 30 live recorded queries**		
Top-X ranking	**Top-1**	**Top-5**	**Top-10**	**Top-1**	**Top-5**	**Top-10**
Ocarina	76.67%	80.00%	83.33%	80.00%	83.33%	83.33%
Piano	63.33%	76.67%	83.33%	70.00%	83.33%	86.67%
Guitar	71.19%	84.75%	96.61%	76.27%	89.83%	96.61%
Overall	73.33%	83.33%	87.78%	77.78%	87.78%	88.89%
MRR	78.12%			81.96%		

Table 1.
Acceptance test

What has been achieved

Successful identification of song titles, within mission-critical specifications, for imperfect queries distorted with 16.67 % distortions. As seen in **Table 1**, the core mission-critical

specifications have been met.

Findings

Although the project was designed and implemented for QBP applications it was discovered that the functionality can be extended and also used for QBH applications.

The Smith-Waterman (SW) matching algorithm used in genome sequence matching was discovered to be a very effective matching algorithm in QBP systems. It was also found that any imperfections or deterioration in one of the feature representations had a direct impact on the Top-10 hit rate for matching. With regards to the SW algorithm, a combination of using the algorithm once for every feature representation proved to be very effective.

It was established that the testing strategy using two control query groups one for perfect queries and the other for imperfect queries proved to be effective in validating the functionality of the SW matching algorithm, thereby eliminating the effect of a poor performing algorithm in the interpretation of the results associated with imperfect queries.

The strategy of implementing, testing and optimizing each relevant subsystem (onset detection, pitch detection and matching) contributed to the superior performance of the overall QBP system.

Part 3. Project Proposal

This section contains the problem identification in the form of the complete approved Project Proposal, unchanged from the final approved version.

Project Proposal	**April 2017**	**Note:**

Student to complete this section					
Mervitz	**JH**	**Mr.**	**12014223**		
Query-by-playing application (app)			**Study leader: Prof. JP. Jacobs**		
Class group: English group	**Project number:**	JPJ3	Revision number:	0	
Type of project: Design	Degree programme enrolled for: Computer Engineering				
Student declaration: I understand what plagiarism is and that I have to complete my project on my own.	Student signature		Date		

Declaration by language editor (proofreader)	
I have been allowed adequate time to read this document carefully and to make corrections where necessary (date received indicated below). To the best of my knowledge, correct formatting, spelling and grammar are used throughout the document.	
D. Swart (language editor)	Date

Declaration and recommendation by study leader		
1. Have you (the study leader) been allowed adequate time to read and comment on the Project Proposal?	Yes	No
2. Is the Project Proposal a <u>correct</u> and <u>complete</u> description of what is required?	Yes	No
3. Is the Project Proposal <u>clear</u> and <u>unambiguous</u>?	Yes	No
4. Recommendation: Do you recommend that the Project Proposal be approved?	Yes	No
Prof. JP. Jacobs (Study leader)	Date	

This section to be used by the Project lecturer					
Content /20		Attended lectures:	Yes	No	
Subtract for editing errors /10		Language editing adequate:	Yes	No	
Final mark /20		Approved? (If "No", a revision must be submitted):	Yes	No	Prof. J.J. Hanekom

1. Problem statement

1.1 Motivation

The motivation for this project stems from people's need to identify tunes or melodies played by them, but for which they are unable to recall the title of the song. This project's aim is to retrieve music and/or song title(s) from raw audio of music played on a keyboard.

1.2 Context

Due to the proliferation of multi-media devices there has been a great deal of interest in Music Information Retrieval (MIR) systems. Much of the research based on Query by Humming (QBH) systems have focused on music processing, feature extraction, signal representation, similarity matching, database queries, and search algorithms.

QBH systems can largely be segmented into note-based and frame-based systems [6]. In note-based systems the feature extraction produces a feature set such as musical interval, duration, tempo and beat, and pitch detection which serves as an estimator or proxy for pitch from each note sequence. Frame-based systems make use of pitch vectors or contours and method statements which enable the distance between the pitch vectors and those contained in stored music files to be calculated. This design project will make use of the existing body of research to design an MIR application for implementation on a smartphone and/or tablet device.

1.3 Technical challenge

This project will concentrate on MIR of monophonic audio signals generated by a single instrument in the presence of background and device (microphone) noise.

The technical engineering challenges associated with this project are the design of a computer algorithm that (i) uses digital signal processing methods, (ii) converts a monophonic audio signal into a format/representation suitable for matching, (iii) matching the resultant representation to a large database of ground truth melodies to select the nearest match, (iv) software development and porting of the algorithm to a smartphone and/or tablet device, and (v) live testing of the application and/or algorithm with live music queries in an acoustically untreated room.

Specific technical challenges that the algorithms will need to address are player induced issues. The player may play in a different key and/or tempo compared to the ground truth. The player may also make errors associated with pitch, rhythm and extra notes may be introduced or notes may be omitted.

The challenges that the matching algorithm will need to address are (i) time and speed imperfection in tunes played by a user, (ii) time variations between the query and the ground truth melody references and (iii) determining an optimal cost function to select the most likely candidates.

1.4 Limitations

Limitations to contend with, are the following:

- Live testing in an acoustically untreated room, the presence of background noise and microphone generated noise;
- the unpredictable distortions of tunes by users;
- an unknown start position and portion of the song; and
- a selection of the Musical Instrument Digital Interface (MIDI) track for comparison of the query.

2. Project requirements

ELO 3: Design part of the project

2.1 Mission requirements of the product

The requirements of the product and/or application are detailed in the following list:

- Software application for a smartphone or tablet;
- Simple graphical user interface;
- Musical Information Retrieval (MIR) of song titles from a database following the playing of a song or a tune into the microphone of the device; and
- Displaying the title of the song(s) matched in the database.

2.2 Field conditions

The product and/or application should be able to operate in an acoustically untreated room with background and device (microphone) noise.

2.3 Student tasks

The following student tasks are envisaged to meet the objectives of this project:

- To perform a literature study of systems and MIR techniques used;
- to document literature study;
- to design a computer algorithm capable of meeting the project requirements and specifications;
- implementing and testing the computer algorithm using MATLAB/ Python;
- to develop and test an Android based client application for smartphone and/or tablet with database which has been compiled;
- to perform relevant simulations and measurements;
- the comparing of measurement outcomes to the requirements and specifications;
- to document the design and results throughout design and development process; and
- to do a live testing of the application/algorithm with instrument generated music queries in an acoustically untreated room.

ELO 4: Investigative part of the project

2.4 Research questions

The following research questions are associated with the project:

- Which method(s) for onset detection should be used for this project?
- Which methods of pitch detection should be used for this project?
- Which retrieval or matching algorithms should be applied in this project?
- What search algorithm is suitable for the database of ground truth melodies?
- What form of digital signal processing should be applied to a raw audio signal in the presence of background and other noise?

2.5 Student tasks and experiments

The tasks to be completed by the student are listed below:

- Conduct simulations to verify and/or implement the onset detection algorithm with MATLAB.
- Conduct simulations to verify and/or implement the pitch detection algorithm with MATLAB.
- Conduct simulations to verify the ground truth melody search and matching algorithm using MATLAB.
- Configure a database on the target device and test MIR using live music received through the device's microphone.

3. Functional analysis

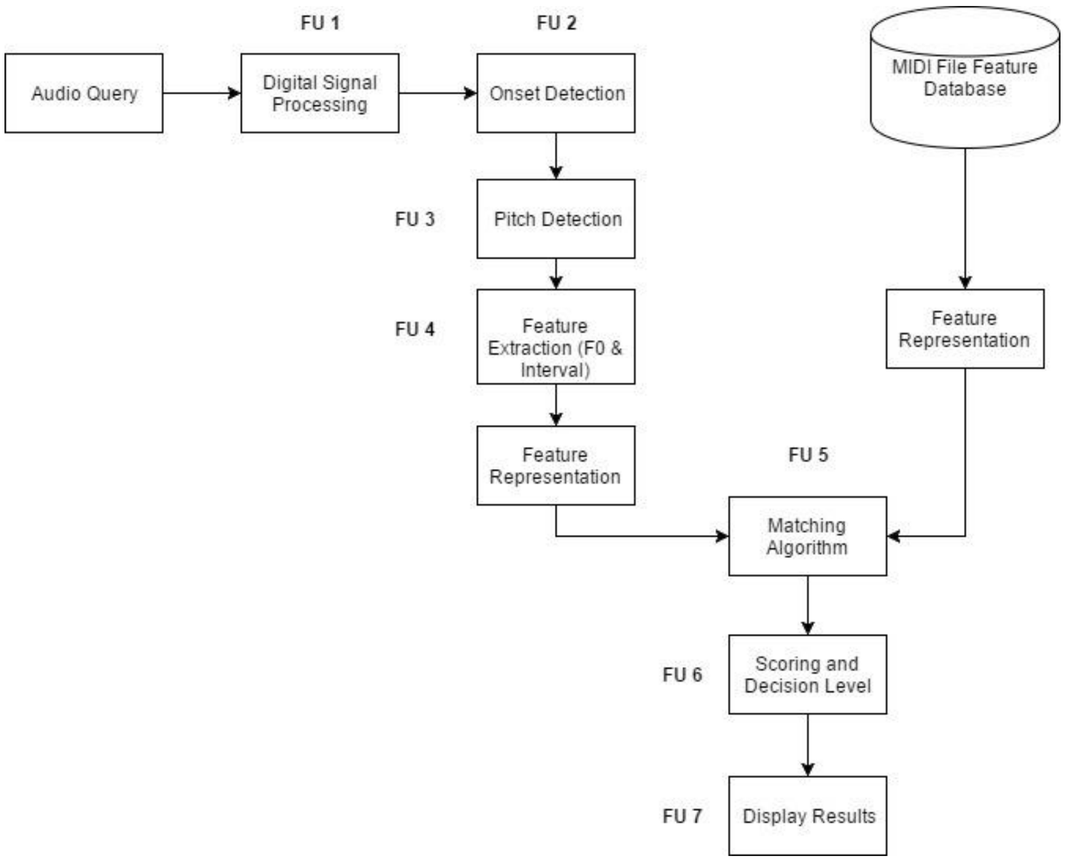


Figure 1: Functional Block Diagram

Audio query: The audio query is generated by playing a melody or tune on a musical instrument, and recorded by the device.

Digital signal processing: Digital signal processing will be performed on the raw audio signal such as filtering and sampling.

Onset detection: Onset detection looks for transients and/or transient regions in the signal which may be defined as a sudden burst of spectral energy or a change in the short time of the signal. Changes in phase slope (group delay) are also an indicator for onset detection.

Pitch detection: The chosen algorithm is designed to extract fundamental frequency of a signal which is an estimator of the pitch.

Feature extraction: The fundamental frequency and the pitch interval are extracted as representative features of the signal.

Feature representation: The melody features are represented by pitch and pitch interval sequence.

MIDI file feature database: Large database of stored ground truth melodies in MIDI format.

Matching algorithm: The algorithm performs a database search and determines the degree of similarity between the feature representation of the audio query and the feature representation of the MIDI files stored in the database. Possible candidates are: (i) Dynamic Time Warping (DTW) as described in [2] and (ii) Hidden Markov Models.

Scoring and decision level: The ranking of possible candidates is determined based on chosen criteria and a decision level will be implemented to select the most likely candidate.

Display results: Display of the results in a user-friendly format.

4. Specifications

4.1 Mission-critical system specifications

Table 1 below details the mission-critical specifications for this project.

SPECIFICATION (IN MEASURABLE TERMS)	**ORIGIN OR MOTIVATION OF THIS SPECIFICATION**	**HOW WILL YOU CONFIRM/ MEASURE THAT YOUR SYSTEM COMPLIES WITH THIS SPECIFICATION?**
The length of each query is required to be at least 15 seconds. Top-1 Hit Rate for monophonic music ≥ 55 %. Top-5 Hit Rate for monophonic music ≥ 75%. Top-10 Hit Rate for monophonic music ≥ 85 %. N is the number of queries, which is at least equal to 80. The database size will consist of at least 500 MIDI files. The genres applied will be traditional pop, dance pop, lounge music, country pop and indie pop music with the main track selected as the vocal track in the MIDI file.	This specification was chosen as it was observed from [3] that both the top-1 and the top-10 hit rate and the MRR rise as the length of query increases. The top-X hit rate criterion is a widespread measure associated with the matching success of music information retrieval systems as shown in [3]. The top-X hit rate reports the proportion of queries for which $r_i \leq X$ where r_i is the ranking of ith query and X is integer [3].	The raw analogue signal will be processed and the representative signal will be matched to all representative signals of the database of ground truth melodies. The number of queries and the ranking of each query is recorded. Count the number of queries which are ranked top X and divide by the total number of queries.
Estimated MRR (Mean Reciprocal Rate) ≥ 60 %.	This specification was chosen because it appears to be the average MRR achievable as shown in [1]. The mean reciprocal rank (MRR) is calculated using the following formula: MRR = $\left(\frac{1}{N}\right)\sum_{i=1}^{N}\frac{1}{r_i}$, where N is the number of queries, let r_i denote the rank of correct answer in the retrieved melodies [3].	The raw analogue signal will be processed and the representative signal will be matched to all representative signals of the database of ground truth melodies. The number of queries and the ranking of each query is recorded.
Gross Pitch Detection Error (GPDE) associated with the query ≤ 15 %.	This specification was chosen as it appears to be achievable using an algorithm such as YIN or spectro-temporal autocorrelation (STA) for pitch detection as an	The accuracy of the pitch detection algorithm will be measured by comparing the pitches of detected notes to those of known ground truth melodies.

SPECIFICATION (IN MEASURABLE TERMS)	**ORIGIN OR MOTIVATION OF THIS SPECIFICATION**	**HOW WILL YOU CONFIRM/ MEASURE THAT YOUR SYSTEM COMPLIES WITH THIS SPECIFICATION?**
	objective estimation of pitch and corresponding MIDI number [5].	The relationship between MIDI number and frequency is given by the following formula: $MIDI\ No = 69 + 12\log_2(\frac{f}{440Hz})$ The algorithm will be tested by using a query with a known sequence of pitches, and the difference between the detected pitches and the known pitches will be used to calculate the gross pitch detection error.
F-measure for onset detection ≥ 75 %. Onsets are as taken as being correctly detected within a tolerance of ± 75 ms around the onset.	This specification was chosen for two possible methods of onset detection such as phase slope and spectral difference as shown in [4] and in [7].	$F = \frac{2PR}{P+R}$, count number of correctly detected onsets (N_{tp}), number of missed onsets (N_{fn}) and number of false alarms (N_{fp}) to calculate P (precision) and R (recall). Where, $P = \frac{N_{tp}}{N_{tp} + N_{fp}}$ and $R = \frac{N_{tp}}{N_{tp} + N_{fn}}$

Table 1. Mission-critical system specifications

4.2 Field conditions

Table 2 below indicates the field specifications for the project.

SPECIFICATION	**ORIGIN OR MOTIVATION**
Live testing of the application in an acoustically untreated room.	Project concept document for project JPJ3 of ERP 400 & EPR 402.

Table 2. Field conditions

4.3 Functional unit specifications

The functional unit specifications for the project are described below in Table 3.

SPECIFICATION	**ORIGIN OR MOTIVATION**
FU1. Digital Signal Processing.	Sample at the Nyquist frequency assuming a maximum frequency that corresponds to the highest MIDI number.
FU2. Onset Detection.	This is used to determine the onset of musical notes.
FU3. Pitch Detection.	Detection of the fundamental frequency (F0) which is used as an estimator for pitch. Generate several F0 candidates and peak selection.

FU4. Feature Extraction.	Features such as F0 and pitch interval are extracted for the representative signal.
FU5. Matching Algorithm.	This algorithm is used to match the representative signal of the audio query with all the representative signal from the MIDI file database. The algorithm should have the capability to deal with variations in pitch interval and fundamental frequency.
FU 6. Scoring and Decision Level.	The difference of each query/database representative signal pair will be evaluated and scored. The decision level will be used to select the most likely candidate pair.
FU 7. Display Results	Display the results of the musical query to the user.

Table 3. Functional unit specifications

5. Deliverables

5.1 Technical deliverables

DELIVERABLE	DESIGNED AND IMPLEMENTED BY STUDENT	OFF-THE-SHELF
Android tablet and/or smartphone.		X
Android application developed for Music Information Retrieval (MIR).	X	
Keyboard with the choice of three instruments such as piano, guitar and woodwind to produce the query.		X
User manual for the application.	X	
Large database of MIDI files.		X
Design of onset and pitch detection algorithms.	X	
Design of a matching algorithm.	X	

Table 4. Deliverables

5.2 Demonstration at the examination

The following visualized steps are to be performed at the demonstration:

1. The keyboard and the Android tablet and/or smartphone will be setup in an acoustically untreated room.
2. The keyboard will be placed in such a way that the microphone of the Android device will clearly receive any sound played.
3. Before the commencement of a query the song that will be played on the query producing instrument (keyboard) will be conveyed to the examiners.
4. The application is then setup correctly on the Android device to allow for recording of the played query.
5. The returned ranked list of most likely song candidates as displayed in the screen of the device are then shown to the examiners.
6. The above query process is repeated to exhibit the full extent of system functionality to examiners.
7. Point out to examiners where specifications have been validated with prior results or measurements contained within the final report.

6. References

[1] Zhuoran Chen, Xingce Wang, Guoxing Zhao and Mingquan Zhou, "A new mechanism of similarity evaluation for content-based music information retrieval," *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, Beijing, 2010, pp. 238-242.

[2] H. M. Yu, W. H. Tsai and H. M. Wang, "A Query-by-Singing System for Retrieving Karaoke Music," in *IEEE Transactions on Multimedia*, vol. 10, no. 8, pp. 1626-1637, Dec. 2008.

[3] C. W. Lin, J. J. Ding and C. M. Hu, "Advanced query by humming system using diffused hidden Markov model and tempo based dynamic programming," *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Jeju, 2016, pp. 1-7.

[4] A. Holzapfel, Y. Stylianou, A. C. Gedik and B. Bozkurt, "Three Dimensions of Pitched Instrument Onset Detection," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1517-1527, Aug. 2010.

[5] N. Yang, H. Ba, W. Cai, I. Demirkol and W. Heinzelman, "BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1833-1848, Dec. 2014.

[6] Gi Pyo Nam, Kang Ryoung Park, Sung-Joo Park, et al. “A new query-by-humming system based on the score level fusion of two classifiers”. In: *International Journal of Communication* Systems 25.6 (2012), pp. 717–733.

[7] Sebastian B¨ock, Florian Krebs, and Markus Schedl. “Evaluating the Online Capabilities of Onset Detection Methods.” In: *ISMIR*. 2012, pp. 49–54.

Part 4. Main Report

1. Literature Study

1.1 BACKGROUND AND CONTEXT OF THE PROBLEM

Due to the proliferation of multi-media devices there has been a great deal of interest in Music Information Retrieval (MIR) systems. Much of the research based on Query by Humming (QBH) systems has focused on music processing, feature extraction, signal representation, similarity matching, database queries, and search algorithms. Query by Playing (QBP) closely follows the elements introduced in QBH, apart from the query producing instrument that differs. Below, all relevant and considered articles on the algorithms are discussed as well as a brief summary on how they are used in this project.

QBH systems can be broadly segmented into note-based or frame-based systems. In note-based systems the feature extraction from the query produces a feature set such as musical interval, duration, tempo and pitch. Frame-based systems make use of pitch vectors or pitch contours that enables the distance between the pitch vectors and those contained in stored music files to be calculated [1]. There are several challenges associated with an imperfect query due to humans not being able to hum or sing perfectly. People may hum in a different pitch, different key or different tempo. The QBP application is subject to similar challenges associated with query imperfections.

One example of a QBH system was an implementation based on pitch contour tracking, and melody conversion into a string contour and thereafter the songs stored in a MIDI database were matched using a string matching method and a frame-based matching algorithm called Dynamic Time Warping (DTW). The method discussed in [2] was not very effective for their large scale database (around 250 songs) and proved to be very time consuming. They concluded that the DTW algorithm performed well for query imperfections with different tempo, but poorly for query imperfections such as different pitch or different key.

An alternative QBH system was implemented using music segmentation based on the extraction of note intervals, note durations, relative pitch and duration matched to a MIDI melody feature database. The melody features were extracted from recorded humming queries in WAV format and the DTW algorithm was used to calculate the similarity score and ranking. The results showed that a segmentation method based on note intervals, note durations and relative pitch achieved high retrieval accuracy [3].

1.1.1 Onset detection

Music is to a large extent an event-based phenomenon, and it is the detection of these events in note-based music which gives rise to a range of music applications such as content delivery, indexing, compression and retrieval [4].

Note onset detection, also referred to as note segmentation, forms the basis of many MIR systems. This crucial piece of information provides the starting time of a note that was played and gives insight in useful analysis and indexing techniques. **Figure 1** depicts a graphical representation of a typical onset [4].

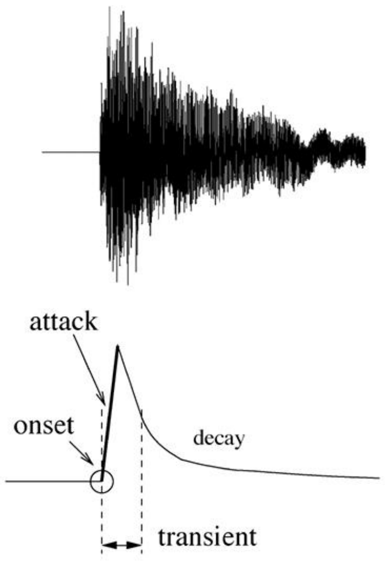


Figure 1.
Graphical depiction of an onset.

Onsets usually occur at transient regions in the signal. The concept of a note onset is defined as a single instant in time chosen to mark the temporally extended onset, which coincides with the start of the transient or the earliest time at which the transient can reliably be detected. A sudden burst in energy can create onsets and can be analysed using time-based or frequency-based methods. Frame-based methods attempt to accurately detect onsets by extracting short transient frames which can be very useful if onsets occur close to one another. The utilisation of each method largely depends on the waveform characteristics of the query producing instrument.

Musical instruments can be classified into three categories, pitched non-percussive (PNP), pitched percussive (PP), non-pitched percussive (NPP), and their performance was evaluated separately. Following the testing of several onset detection methods with PNP, PP and NPP instruments it was concluded that onset detection has a strong relationship with the specific type of musical instrument, and that one should use a detection method with minimal complexity that satisfies the required application. For signals that are percussive, time-domain methods will usually suffice. For signals with strongly pitched transients, spectral methods such as spectral difference (spectral flux) performed well. These recommendations are relevant to instruments selected for this QBP project. [4].

For the purposes of this project only pitched instruments are used in order to purely demonstrate other detection mechanisms such as pitch detection in addition to onset detection. The instruments used are guitar, piano and woodwind. The piano is classified as a pitched percussive instrument and is more suited to the time-domain onset detection methods, whilst the woodwind instrument is classified as a pitched non-percussive instrument which may be more suited to a frequency-based onset detection method.

Onsets can also be classified as "hard" or "soft". Hard onsets are onsets with sudden increases in energy typically generated by a piano or guitar, and their detection is suited to time domain methods. Soft onsets show a more gradual change in energy, such as onsets typically generated by a woodwind instrument, and their detection is more suited to frequency-based methods [5].

For every onset detection algorithm there are three basic steps that are followed. These are listed below [6]:

- Pre-processing the audio signal
- Computing the Onset Signal Strength (OSS)
- Informative peak picking to detect transients in the OSS

Pre-processing is performed on raw audio signal data which is usually done in the presence of noise. For the proposes of this project no filtering is required for noise as the assumption is made that noise levels will be kept at a minimum when recording and the query will be played in close proximity to the device microphone.

OSS is usually determined directly after pre-processing or sometimes on the raw audio signal. A popular frame-based method is called the Spectral Flux (SF) method and is useful when sudden energy changes take place in the signal that indicate attack parts in new notes. This method makes use of Short Time Fourier Transforms (STFT) to determine the spectral difference between successive frames [6]. It is important to find the optimal frame size, also called the window length, as the onset detection accuracy largely depends on it. This method is highly recommended with the use of both hard and soft onset detection.

It is possible to describe a signal in the frequency domain by applying a Fourier transform in a frequency-based method called Phase Slope (PS). This method is favourable when observing the properties of the group delay which is defined by **Equation 1.1** below [6].

$$\tau(\omega) = -\frac{d\phi(\omega)}{d\omega} \tag{1.1}$$

Computing the derivative of the phase component with regards to frequency, results in a value that is proportional to the distance between the centre of the current window being analysed, and the position of the impulse. However, PS is not the only frequency-based method that exists, Fundamental Frequency Change (FFC) works in contrast to standard onset detection since pitch detection is required. In some cases, onsets tend to be difficult to observe. This is usually common with woodwind instruments, for example a flute where energy changes are difficult to detect. To effectively detect onsets for this type of instrument it is necessary to introduce an additional OSS method. The solution lies in calculating the F0 estimations [6].

Time based methods are often just as effective as frequency or frame based methods. Edge detection has proved to be useful in the detection of onsets that have a sudden rise and a rapid exponential decay, as in the case of a guitar or a piano that only produce hard onsets. The Gaussian filter is one such edge detector that excels in hard onset detection. The absolute value of the raw signal provides an envelope of the signal which is then convolved with a filter based on the first derivative of the Gaussian Pulse (GP). The output produces positive and negative spikes depending on the polarity of the edges [7].

All three mentioned categories of onset detection remain vulnerable to unwanted onsets because further processing and filtering has not been done yet. Peak picking is a technique used to detect the salient maxima in OSS and to eliminate any points below certain thresholds. In the case of the SF algorithm a moving median filter is used for adaptive thresholding [6]. Absolute thresholding is also an option as with the case of the Gaussian edge detector [7].

Onset detection performance should in all cases be evaluated to determine the goodness of the detection method. MIREX evaluates onset detection by computing the F-measure as shown in **Equation 1.2** below [6].

$$F = \frac{2PR}{P+R} \tag{1.2}$$

The equation considers both Precision (P) and Recall (R). These two factors depend upon correctly detected onsets, false positives and false negatives. False positives, also referred to as false alarms are onsets that are mistakenly detected by the onset detection algorithm. False negatives or otherwise known as missed onsets are onsets that exists, but which the detector failed to detect [6].

1.1.2 Pitch detection

Pitch detection is the next step in MIR systems and with the location of relevant onsets, the procedure of finding the pitches is simplified. The algorithms used in this form of detection is called Pitch Detection Algorithms (PDA) and can be achieved in the time or frequency domain. PDA attempts to estimate the period of the quasi-periodic signal and then performs an inversion of the obtained value to give the frequency. Three different PDAs are considered and discussed below.

The YIN estimator is a well-known PDA that makes use of the autocorrelation method and is effective for both music and speech signals. The algorithm has no restriction of the upper frequency search range and is best suited for high pitched music and speech. The simplicity of implementation and execution speed makes this algorithm attractive for low-end processing and real-time applications. Below **Figure 2** shows a sample waveform and the result of an Auto Correlation Function (ACF) performed on the waveform. Other methods exists that follows the ACF to improve the error rates of the algorithm [8].

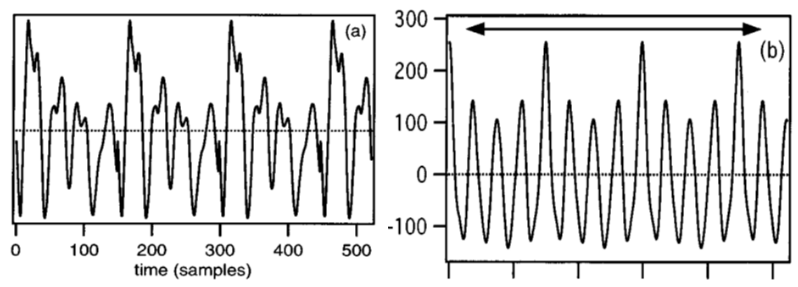


Figure 2.
Autocorrelation performed on a waveform.

The Harmonic Product Spectrum (HPS) is also known for its simplicity and computational friendliness. The HPS algorithm uses **Equation 1.3** below to measure the maximum coincidence for the harmonics of each spectral frame [9].

$$Y(\omega) = \Pi_{r=1}^{R} |X(\omega r)| \qquad (1.3)$$

The resulting correlation array $Y(\omega)$ is then searched for a maximum value. **Figure 3** below depicts a graphic representation of the HPS method [9].

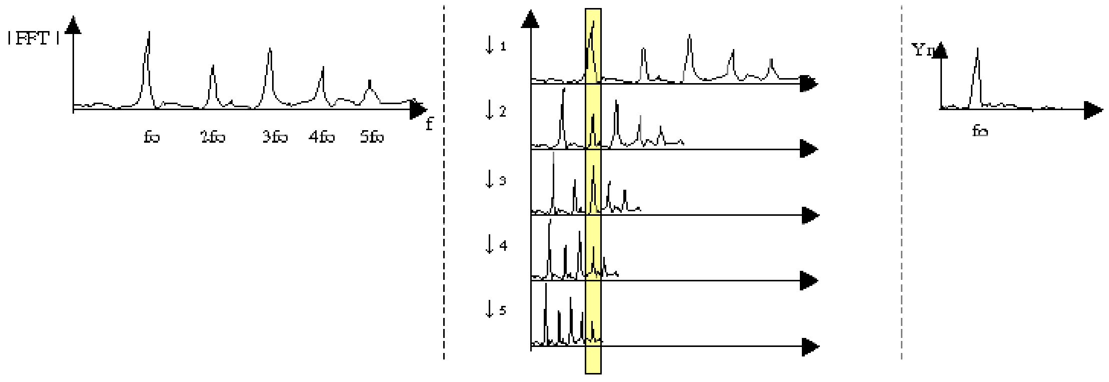


Figure 3.
Graphic representation of the HPS algorithm.

The algorithm methodology primarily consists of the following steps:

- The appropriate window function is selected depending on the algorithm requirements.
- The sampled input signal is multiplied with the window function element by element.
- An N point FFT is then performed on the result of the previous step.
- Downsampling or harmonic compression shrinks the spectrum.

- All the downsampled components are summed together into one vector.
- The maximum value of the vector provides the desired frequency in Hz.

The algorithms mentioned thus far performed well without the presence of noise and execute relatively fast. The BaNa Algorithm was developed as a noise resilient pitch estimator. The Gross Pitch Detection Error (GPDE) of this algorithm has proved to be the lowest. The main disadvantage with the BaNa algorithm is resource-hungry approach in determining pitches.

Listed below are some key features of the BaNa Algorithm [10].

- Pre-processing: This includes the implementation of a bandpass filter prior to pitch extraction.
- Pitch Determination: This step involves the discovery of harmonic peaks in the signal and then calculating the most likely pitch candidates.
- Pitch Selection: Pitches are then selected from the candidates using a cost function of pitch differences in adjacent frames.

1.1.3 Feature representation

The onset and pitch detection algorithms have been discussed up until this point. It is important to note that for any further processing, the raw onset and pitch information can't be used. One has to consider a feature representation that is useful to any matching algorithm. A feature representation uses the raw onset and pitch information and mathematically manipulates these inputs to produce a simplified and recognisable format for any matching algorithm. It is suggested in [11] that onset ratios be used to represent onset information while using pitch interval to represent pitch information.

1.1.4 Matching

For a user of any MIR system it is desirable to have rapid access to large music databases. Content-based and descriptive-based methods are the main methods used for matching. Any matching algorithm requires a representation that is common to the query and the content stored in the database. The matching algorithm will produce a rank list as an output of most likely candidates and then be displayed to the user [12]. The following algorithms discussed use feature representations and similarity matching to match the query feature set to that of the database feature by finding the most likely matching sequence.

The Hidden Markov Model (HMM) involves observing a sequence of events. The observation x_t can be observed at time t is generated by an unobservable, underlying discrete state s_t that follows a first-order Markov process. Given the model parameter θ, the joint probability of the observation sequence and the underlying state sequence is expressed below in **Equation 1.4** [13].

$$p(x,s|\theta) = \pi_{s_t}\Pi_{t=1}^{T-1}a_{s_t s_{t+1}}\Pi_{t=1}^{T}b_{s_t x_t} \tag{1.4}$$

Dynamic Time Warping (DTW) is classified as a frame-based method with two (2) time dynamic or speed dynamic time series. The series comparison that is performed using this method is non-linear and is between the input and the template. Provided with two (2) time series *x* and *y*, that represents the testing data and the training set respectively, DTW will try to find the time dimension which effectively minimizes the difference between the two series. Given a warping path the distance can be calculated using **Equation 1.5** [14].

$$D(i,j) = Dist(i,j) + min[D(i-1,j), D(i,j-1), D(i-1,j-1)] \tag{1.5}$$

Figure 4 below is a graphic representation of the workings of a classic DTW algorithm [2].

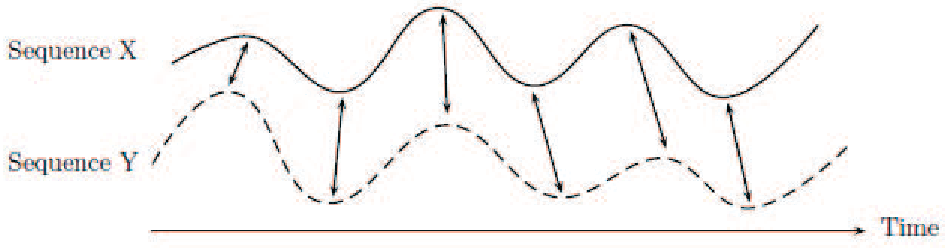


Figure 4.
Classic DTW algorithm.

A typical QBH system can be implemented with the query transcribed into a sequence of notes and makes use of the DTW algorithm to identify the best match to a MIDI tunes database. This implementation considered the identification of the melody despite being performed at a different pitch or different tempo. In this implementation each note is described by its pitch, onset time and duration. The pitch sequence is encoded as sequence of pitch intervals as shown in **Equation 1.6** below, and the duration sequence was converted to a tempo invariant representation using the relevant duration sequence as seen in **Equation 1.7** below. Using this form of encoding and conversion it was found that query matching was less sensitive to query imperfections [11].

$$A = (a_2 - a_1, a_3 - a_2, a_n - a_{n-1}) \tag{1.6}$$

Where $a_1, a_2, a_3 ... a_n$ are a series of detected pitches.

$$B = (\frac{b_2}{b_1}, \frac{b_3}{b_2}, \frac{b_n}{b_{n-1}}) \tag{1.7}$$

Where $b_1, b_2, b_3 ... b_n$ are a series of detected onsets.

The Smith-Waterman (SW) algorithm runs more efficiently than any of its competitors, matching short sequences to longer sequences. Originally developed for molecular sequence analysis, the SW algorithm employs a similarity function that will output a positive value for similar values and a negative value for dissimilar values. A gap penalty is used when a gap exists between the two sequences being matched. The algorithm invokes a two-dimensional scoring matrix for path trace-backs and accumulative scoring [15].

The SW algorithm identifies the optimal local alignment between two sequences. The optimal local alignment is identified by comparing the query sequence and the sequences stored within the database on a character-to-character level. The algorithm is based on a dynamic programming technique which is a methodology used to divide problems into sub-problems and to solve these sub-problems before combining the solutions to the sub-problem in order to address the overall problem [16].

The algorithm consists of the following basic steps:

- Initialisation: The first row and column of the scoring matrix is initialised with zeros.
- Iteration: For each node on the matrix an accumulated similarity cost is calculated.
- Backtracking: The node with the maximum score is selected and the chain of predecessors is followed until location (0,0) is reached in the grid.

The SW algorithm has been used with some success in automatic cover song detection. In this implementation the SW algorithm was used to find local matches in the binary similarity matches [17]. Other audio applications where the SW algorithm has been used are music sequence alignment and audio thumb nailing capable of detecting repeating patterns (thumbnails) in audio streams that stem from shared Internet videos [18].

1.2 APPLICATION OF THE LITERATURE

This project is largely concerned with the identification of queries from a large database and is closely related to QBH systems, hence many of the above-mentioned algorithms are applicable in this scenario. The fact that three unique instruments are used to generate the query, resulted in interesting choices in algorithms. Not one algorithm works for all three instruments but rather various algorithms perform well for certain instruments. It is assumed that the user will know beforehand what instrument is played and can enter that into the application. The algorithms are carefully considered and only the most suitable algorithms are chosen in each case.

Onset detection remains the most integral part of this project since the pitches depend on correctly detected onsets. The SF method was decided upon for a woodwind instrument because of the energy changes, although not significantly visible, in the waveform of the instrument. The Gaussian edge detector will fail in the case of a woodwind instrument because for every note played, the waveform resonates after the initial onset creating spikes and peaks that the edge detector can detect as onsets. For the remainder of the instruments (guitar and piano) both the SF method and the Gaussian edge detector is valid. However, the Gaussian edge detector was chosen since the literature revealed that pitch percussive instruments perform the best with this method of detection. Experiments performed during design and implementation prove that the Gaussian edge detector outperforms the SF method for guitar and piano.

The pitches need to be detected accurately to match with the correct database song. The previously mentioned BaNa algorithm is renowned for its resilience in noisy environments however, for the purposes of this project noise does not need to be considered and therefore this algorithm was ruled out. It is also very computationally expensive. The YIN and the HPS algorithms both execute fast and accurately. Experiments conducted showed that the HPS algorithm performed faster and proved to be more accurate than the YIN algorithm. Therefore, the base HPS was considered with modification and adaptation for each instrument.

Matching is a crucial part of the project and uses the onset and pitch detector results to find the best matching songs in the fastest possible time. The onsets are used to determine the normalised ratio between successive times. This forms the first feature representation. The second feature representation is that of successive pitch intervals. This allows someone to play out of key and slightly out of tempo. The DTW algorithm is renowned for its accuracy but also for its computational inefficiency while the SW algorithm performs the fastest of any matching algorithm and still manages to be accurate. For user convenience and acceptable response times, especially on a mobile platform, the SW algorithm is used for matching.

The database of songs are stored in MIDI format and truncated for matching speed after being processed. The truncation is assumed valid because of the repetitive nature of the songs in the specific genre. Only the melody/vocal track is selected and stored in the MIDI file for processing convenience. Trimmed portions of the MIDI files can be used to generate large test data sets making use of an accurate MIDI to WAV converter.

2. Approach

The solutions to the engineering problems provided by this project were carefully studied and approached in such a manner that sufficient consideration was given to design alternatives and each design's trade-off. All available tools were utilised fully to aid in the design and implementation as well as engineering problem solving. Below each entity in the functional analysis as proposed in the project proposal (**Part 3**), **Section 3** is described systematically.

2.1 AUDIO RECORDING AND QUERY PRODUCTION

The audio recording of a query is of paramount importance as this crucial step has an effect that propagates throughout the remainder of the system. The two factors that contribute the most to overall performance of the system is the audio sampling rate, and the distance between the source and the microphone. The three (3) select instruments have the following frequency ranges:

- Guitar: 80 - 1 200 Hz.
- Piano: 27.5 Hz - 4 186 Hz.
- Woodwind: 75 Hz - 5 000 Hz.

Initially the sampling rate of the audio was set to 44.1 kHz which is the standard CD quality for audio so that the algorithms have enough samples to work with. Later it was found that the algorithms became computationally expensive and would take more time to execute. The audio sampling rate was then halved to 22.05 kHz with favourable results and still meets the Nyquist criteria due to the frequency ranges of the chosen instruments. The algorithms executed much faster and performed within the same degree of accuracy as the previous sampling rate. Regarding the proximity of the microphone to the audio source, it was found that the closer the microphone is to the source the better the algorithms performed since the signal to noise ratio improved.

Queries are produced from the preprocessed database songs. MATLAB is used to extract a query of any length from the full MIDI file using Java commands. The query is then stored as a MIDI file and is further processed in MidiIllustrator so that the instrument can be changed to what is required. The next step is to convert the MIDI query to a WAV file using a standard MIDI to WAV converter that utilises a realistic sound font file.

2.2 FEATURE EXTRACTION

2.2.1 Onset detection

Following from the literature, [4] mentions that the SF method for onset detection is viable in case of hard and soft onsets. The SF method was then tested by using the onset performance measurement of F measure and accuracy to determine which onset algorithm works best for which type of onset. To aid in the endeavour, Sonic Visualizer was used to analyse the WAV query and its corresponding MIDI file. The program superimposes the annotations over the audio waveform. Once it was determined that the annotation lines up with the onsets in the waveform, the average F measure and accuracy was calculated for 90 queries. In addition to the SF method, the Gaussian edge detector and FFC was considered. This procedure was carried out for each of the three instruments and results showed that the Gaussian edge detector performs the best for hard onsets while the SF method performs the best for soft onsets. The onset detection algorithms were designed, implemented and tested in MATLAB before being ported to Android.

2.2.2 Pitch detection

For pitch detection, HPS, BaNa and YIN were considered. To effectively evaluate the PDAs, average Gross Pitch Detection Error (GPDE) is calculated for 90 queries. Ground-truth pitches are obtained from the MIDI queries. Initially BaNa and HPS outperformed YIN as the GPDE was significantly lower. However, due to the computationally expensive nature of the BaNa algorithm, the HPS algorithm was used. The HPS algorithm was then modified to cater for each selected instrument's down-sampling ratio. The down-sampling ratio was determined through tests and simulations. The frequencies obtained from the pitch algorithm are converted to MIDI numbers.

2.3 DATABASE RETRIEVAL

The database consists of 500 pop songs. Each song in the database is prepared by only extracting the melody or vocal stave and storing it. The instrument that is stored in the MIDI file in the database has no effect on matching. The songs are stored by assigning a number to each song as the file name and the song titles are stored in a separate CSV file with the song number indexing the appropriate song name. When retrieving a song from the database the Java MIDI parser is able to extract all the relevant information and that is stored. Only half of this information is used to increase execution time of the matching algorithm.

2.4 FEATURE REPRESENTATION

Many feature representations were tried and tested. The feature representations are passed to the matching algorithm and then the Top-X rankings are calculated for 90 queries. The feature representation that yielded the best Top-X ranking results was chosen. The first attempt involved using the onset and pitch information as is. The second attempt used absolute pitch intervals and standard onset differences however, the third attempt performed the best with the onset ratios and the standard pitch intervals being used.

2.5 MATCHING AND FEEDBACK

Three (3) matching algorithms were considered namely, (i) SW, (ii) DTW and (iii) HMM. The performance measurement for matching is the Top-X rankings. The SW algorithms proved to be the most accurate when testing it over 90 queries as the algorithm gave the best Top-X rankings. The problem with the DTW algorithm was the fact that it executed slow due to its intensive resource usage and struggled performance-wise with matching. It was decided that the SW algorithm be used for matching because it performed accurate enough and the execution time was satisfactory.

The user is able to get feedback after the feature extraction in the form of a piano roll. This is a plot of onset times against MIDI numbers. In terms of matching results, the user will see the Top 5 most likely songs from the database when querying a song.

3. Design and implementation

The design approach adopted is a top-down design making use of the functional decomposed blocks of the functional block diagram detailed in **Figure 1** of the project proposal (**Part 3**). Each major functional block will be discussed in terms of design, simulation, performance evaluation and optimisation.

3.1 AUDIO RECORDING

Audio recording is implemented using the built-in microphone of the target mobile device. The acquisition process for all audio queries uses the Android app through the built-in microphone in the target mobile device. The audio is recorded by way of the AudioRecorder library from the Android SDK with a sample frequency of 22.05 kHz.

3.1.1 Audio recording process

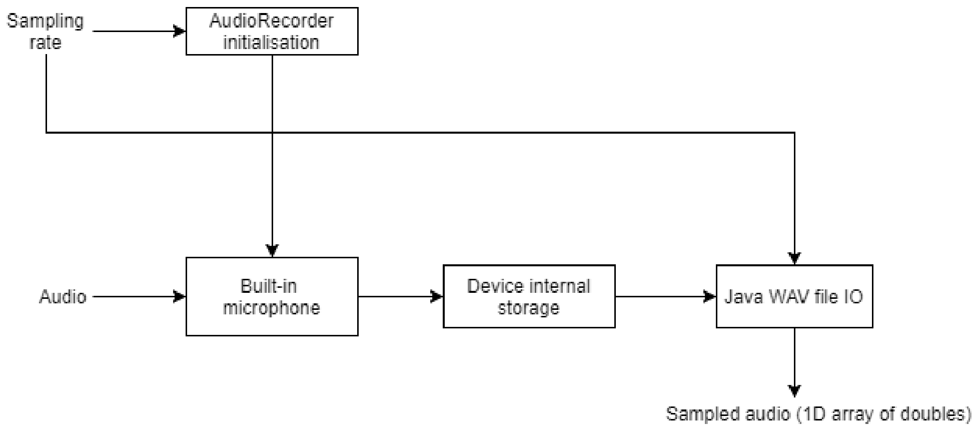


Figure 5.
Audio recording and retrieval process.

The above figure, **Figure 5**, describes the audio recording acquisition and retrieval process. The desired sampling rate is specified to the AudioRecorder object which ensures proper initialisation of the built-in microphone. Upon the completion of the recording, the WAV file produced is stored in the device's internal storage. The Java WAV file IO object can then fetch the file, process it and produce the sampled audio array of doubles.

3.2 ONSET DETECTION

In this project two (2) onset detection algorithms were designed and implemented, namely:

- The SF method

- The Gaussian edge detector

Onsets form the first feature representation used in the matching process.

3.2.1 SF method

3.2.1.a. Design

The SF method uses a function that calculates the spectral difference vector for a WAV file, and then employs additional methods such as peak picking to determine only the appropriate onsets.

Below in **Figure 6** the block diagram describing the SF method is depicted.

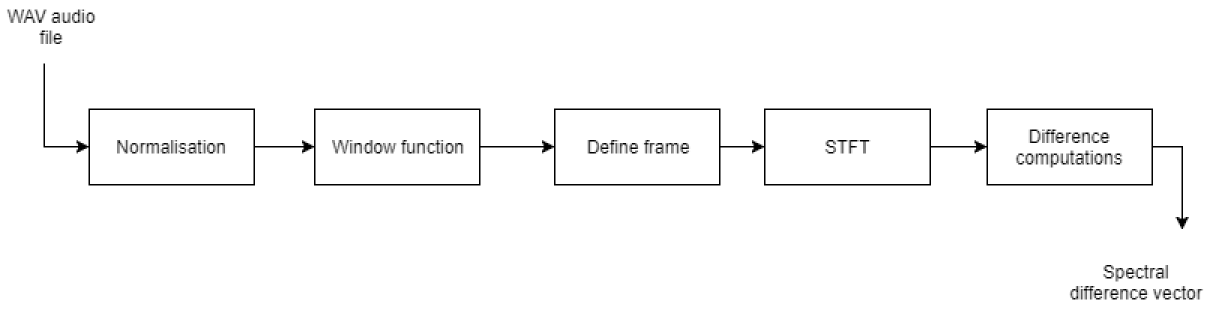


Figure 6.
SF method block diagram.

The algorithm consists of the following steps:

- Input parameters: The sampled audio data, the sampling rate, a fixed window size and a fixed step size.
- Normalisation: The sampled audio data is first normalised to ensure that it is range bound.
- Kaiser window: A window function is created that will be used in the STFT process to develop piece-wise approximations of the magnitude profile.
- Frame definition: A frame is then defined using the window length and the step size.
- STFT: The spectral difference computations that calculate the SF difference vector uses many STFTs in succession.

Window function

The Kaiser window was chosen because the first side lobe is significantly smaller than the main lobe. Additionally, the maximum stop-band attenuation is the lowest of all the window functions (-74 dB). This makes the Kaiser window function ideal for the SF method.

Step size and window length

The optimal step size and window length was found to be 40 ms.

Corresponding time vector

It is important to know the time values that correspond to the centre of each frame. These time values accompany the SF difference vector. The time vector is calculated in **Equation 3.1** and **Equation 3.2** below.

$$c_n = \frac{step_{size}}{2} \tag{3.1}$$

$$c_{time} = f_{start} + \frac{step_{size}}{2} \tag{3.2}$$

Where c_n is the absolute centre time of each frame, c_{time} is the relative centre time taking the frame's start time (f_{start}) into account.

Algorithm flow diagram

The following figure, **Figure 7**, describes the flow diagram of the SF method. Following the SF block diagram, depicted in **Figure 6**, each current and previous window is analysed by taking the STFT and then passing the information to a difference function that ultimately produces the SF difference vector. This vector is passed to the peak picking algorithm that performs filtering and only produces the appropriate onsets.

Difference computations

The function that computes the SF differences is described in **Figure 8** below.

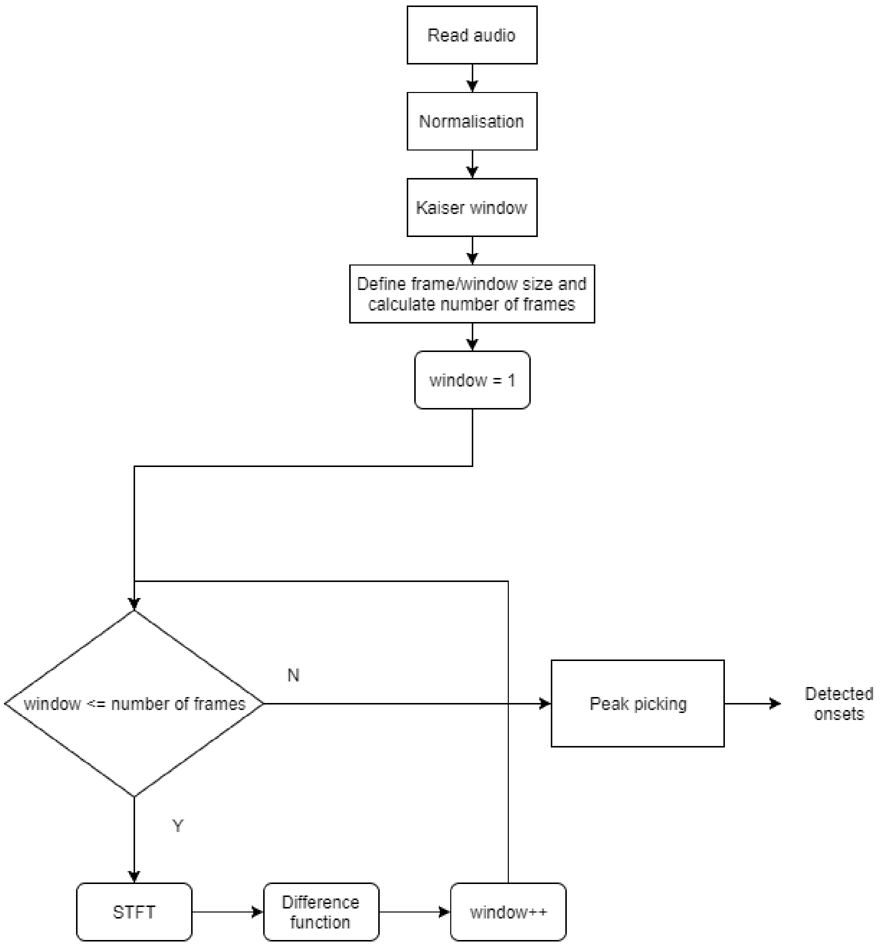


Figure 7.
SF method flow diagram.

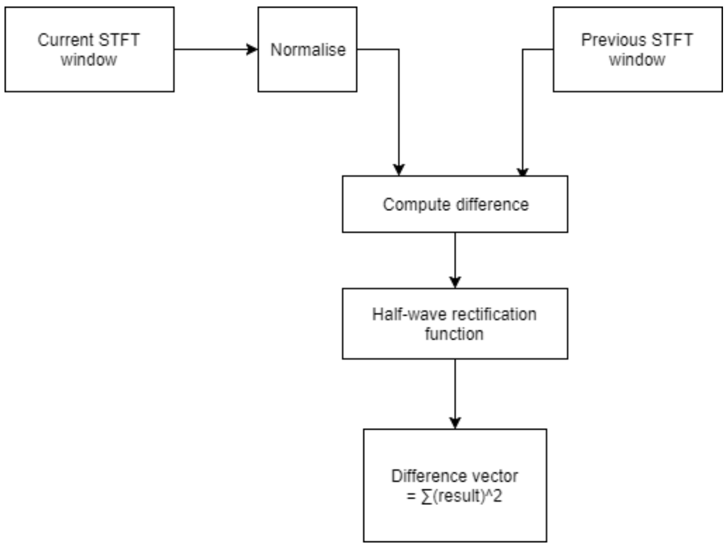


Figure 8.
Difference function flow diagram.

The spectral difference function computes the SF difference vector by analysing the current and the previous windows. The windows originate from the STFT function and are first normalised before passing through a difference function. The difference between the two (2) successive windows are then half-wave rectified where after the result is summed and squared to produce the SF difference vector. **Equation 3.3** is describes the half-wave rectifier function used and **Equation 3.4** describes the SF difference function used.

$$H_f(x) = \frac{x+|x|}{2} \tag{3.3}$$

$$Diff_{vector}(x) = \Sigma(result)^2 \tag{3.4}$$

Peak picking

Peak picking uses the SF difference vector together with filtering, thresholding and redundancy elimination techniques. The block diagram of the peak picking algorithm is shown below in **Figure 9**.

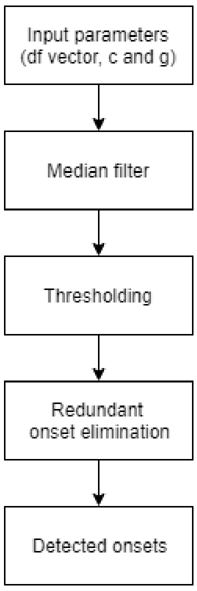

Input parameters (df vector, c and g)
Median filter
Thresholding
Redundant onset elimination
Detected onsets

Figure 9.
Peak picking block diagram.

The algorithm can be described as follows:

- Input parameters:
  - df vector: SF difference function vector.
  - c: Absolute minimum threshold.
  - g: Thresholding scaling factor (θ).
- Median Filtering: A moving filter used for dynamic thresholding that is discussed in more detail in the **Median filter** section below.

- Thresholding: Employs a function, described in **Equation 3.5** below, that distinguishes between all points above it. These points are used for further processing.

$$T_f(x) = \theta \times filter_{out}(x) + c \tag{3.5}$$

- Redundancy elimination: Redundant onsets are eliminated according to a refractory period and is discussed below in more detail below in the **Redundant onset elimination** section.

Median Filter

The flow diagram that describes the median filter is shown below in **Figure 10**.

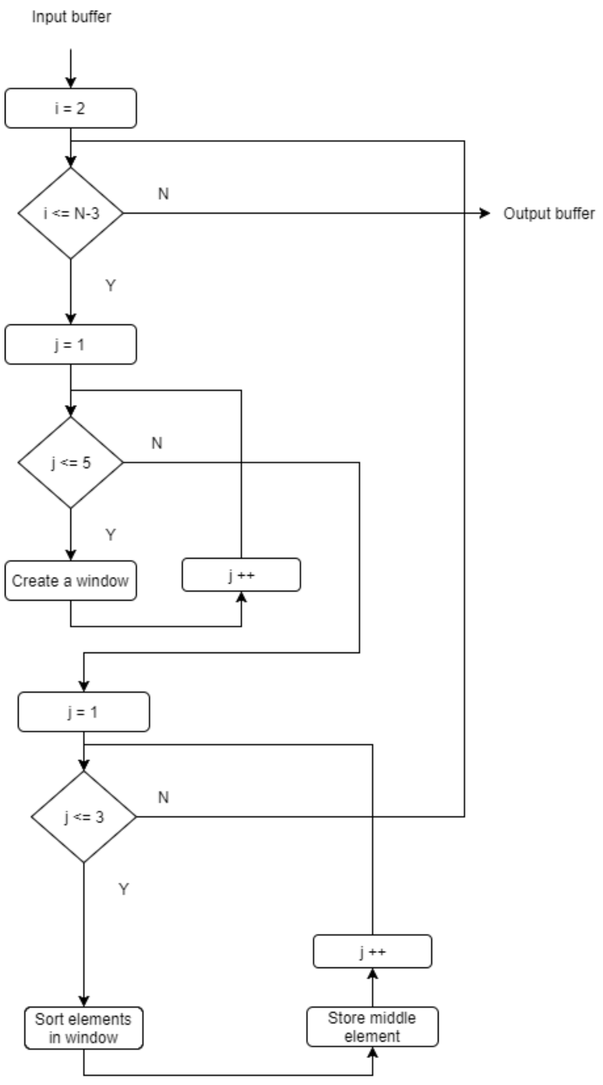


Figure 10.
Median filter flow diagram.

The input buffer is broken down into windows of five (5) elements in length at a time. The windows then get sorted in ascending order whereby the middle element (median) of each window is extracted to form part of the output buffer.

Redundant onset elimination

Below, in **Figure 11** the redundant onset elimination flow diagram is shown.

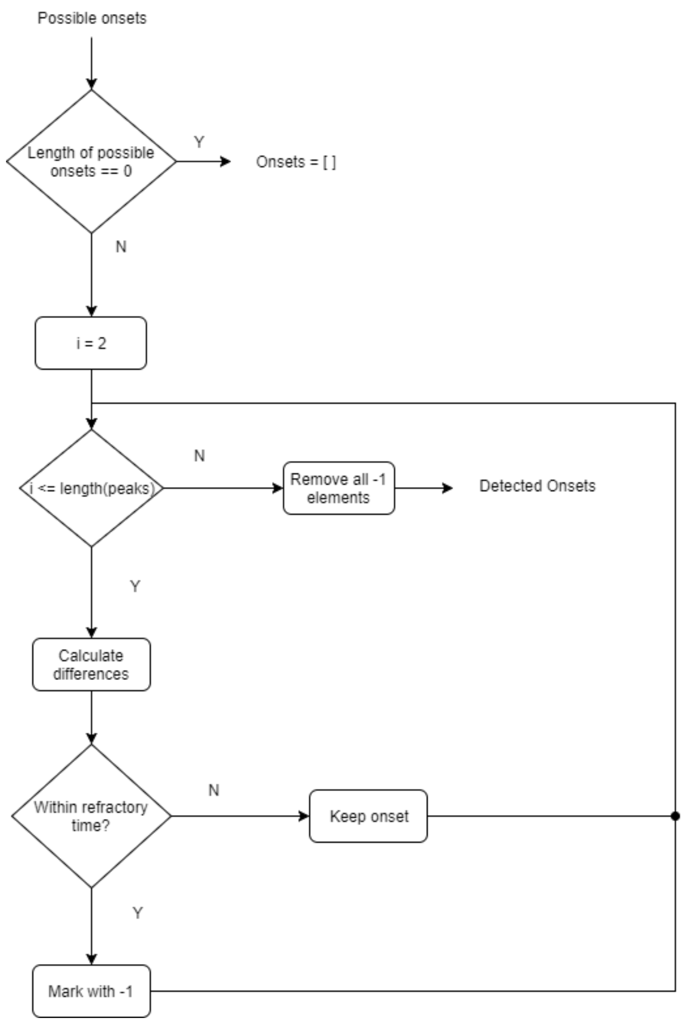


Figure 11.
Redundant onset elimination flow diagram.

The algorithm is designed to accept the array of possible onsets that need to be processed. This vector can contain multiple onsets that exists at a peak and also lie very close together. Onsets that lie within a certain refractory time (50 ms) are eliminated while the rest are kept as detected onsets.

3.2.1.b. Simulations

Instrument simulation

All three (3) types of instruments were simulated using the SF method and the F measure and accuracy were recorded in **Table 2** below. The following experimental set-up was used:

- Aim: The aim of this experiment is to demonstrate the performance of the SF method across all three (3) categories of instruments.
- Queries: 30 unique queries per instrument group were used.
- Approximate length of a query: 30 seconds.
- Types of instruments: Guitar, piano and ocarina (woodwind).
- SF method parameters:
  - Window length: 40 ms.
  - Step size: 40 ms.
  - Minimum threshold c: 0.2.
  - Threshold scaling factor θ: 0.8.
- Performance measurement: F measure and accuracy.
- Platform: MATLAB.

Instrument	F measure (%)	Accuracy (%)
Piano	94.67	79.10
Guitar	92.28	84.82
Ocarina	90.08	85.21

Table 2.
Tabulated simulation for F measure and accuracy for SF method.

3.2.1.c. Optimisations

The following optimisations were applied on the algorithm:

- Window function: Originally the Hamming window was used in the spectral difference computations but it was found that a Kaiser window yielded better results.
- Window length: Since the window function was altered the optimal window length was found to be 40 ms.
- Step size: Due to the same reasons as the window length above, the optimal step size was found to be 40 ms.

- Minimum threshold c: The minimum threshold for the peak picking function was determined to be 0.2.
- Threshold scaling factor θ: The threshold scaling factor gave optimal results when set to 0.8.

3.2.2 Gaussian edge detection

3.2.2.a. Design

The Gaussian edge detector block diagram that uses time based convolution is illustrated below in **Figure 12**.

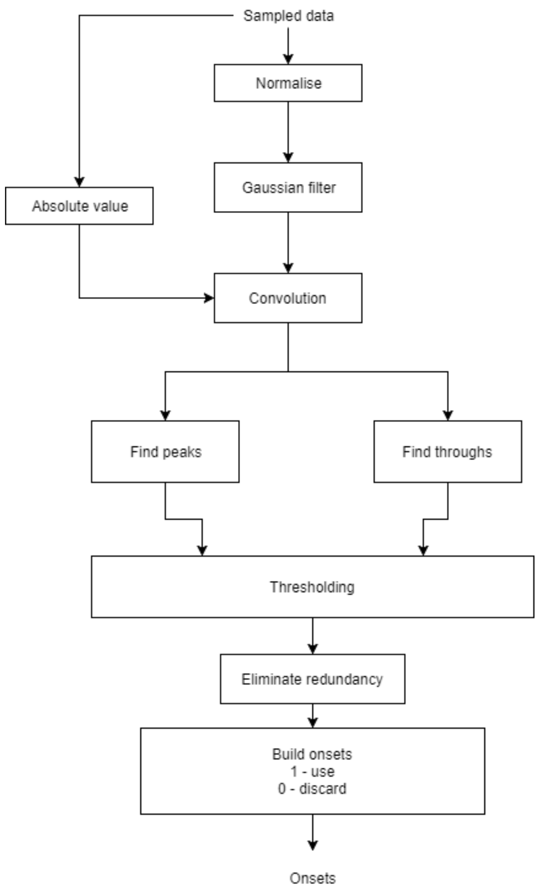


Figure 12.
Gaussian edge detector with convolution block diagram.

The algorithm has the following input parameters:

- Sampled audio query: The sampled audio query data.

- Upper threshold: This parameter provides control over the threshold procedure for peaks.
- Lower threshold: This parameter provides control over the threshold procedure for troughs.

This method makes use of the first order derivative of the Gaussian function to create a filter. The derivation of the first order Gaussian derivative is provided in **Equation 3.6** below. The signal is initially processed to derive its absolute value which provides the signal with a detectable envelope. The signal is also normalised to ensure that it is range bound between +1 and -1. The sampled audio signal is convolved with the Gaussian filter using time based convolution to produce the edges which are processed later on for higher and lower thresholds depending on peaks and troughs.

Derivation of first order Gaussian derivative

$$\begin{aligned} f(x) &= ae^{-\frac{(x-b)^2}{2c^2}} \\ \frac{df(x)}{dx} &= -\frac{(x-b)}{c^2} ae^{-\frac{(x-b)^2}{2c^2}} \end{aligned} \tag{3.6}$$

Where the constants are defined as follows:

- a: Assumed to be 1 for simplicity
- b: *(Filter length)*/2
- $2c^2$: alpha (α)
- c^2 : $\frac{\alpha}{2}$

The parameter *(*b) is is chosen as (Filter length)/2 for the following reasons:

- Values between 0 and (Filter length)/2 is positive.
- The centre value of (Filter length)/2 is equal to 0.
- Values between 0 and (Filter length)/2 is negative.

This enables the Gaussian filter to have a complete swing from the maximum value (positive) to the minimum value (negative), also including zero. Selection of other parameters such as α and *filter length* are discussed below.

Gaussian filter

Figure 13 below depicts the Gaussian filter.

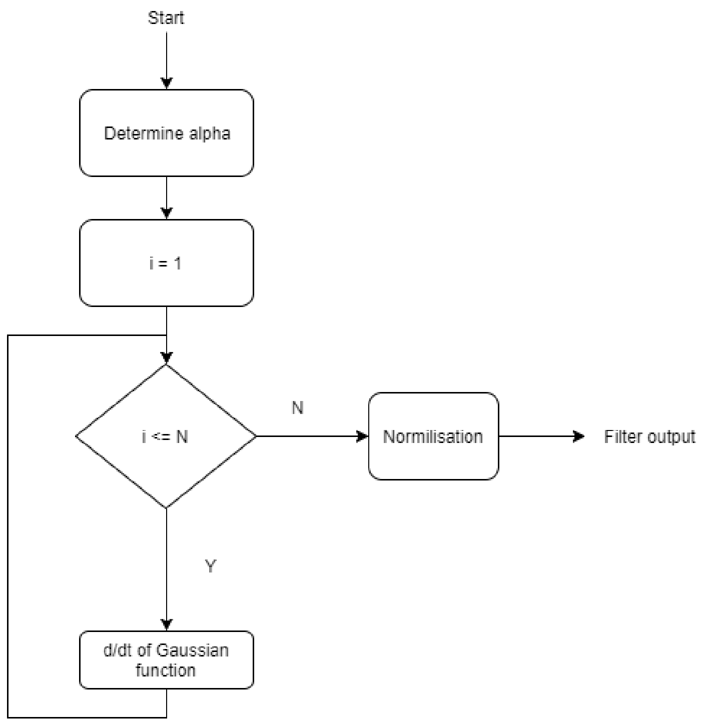


Figure 13.
Gaussian filter flow diagram.

The filter is constructed using the first order Gaussian derivative function. The alpha value is determined before processing and is described in the following section, **Determination of alpha**. The filter length is determined and selected through simulation. The filter output is normalised before further processing takes place.

Determination of alpha

Below in **Figure 14**, the flow diagram describes the selection of the alpha value. The starting position is assumed to be 1 000, using the alpha at the specified value the maximum value of the first order Gaussian derivative is calculated and checked for polarity. if polarity is positive the value of alpha is used otherwise alpha is incremented by 1 000 and the process repeats until a maximum alpha value is reached. The optimal alpha was calculated and found to be 65 000.

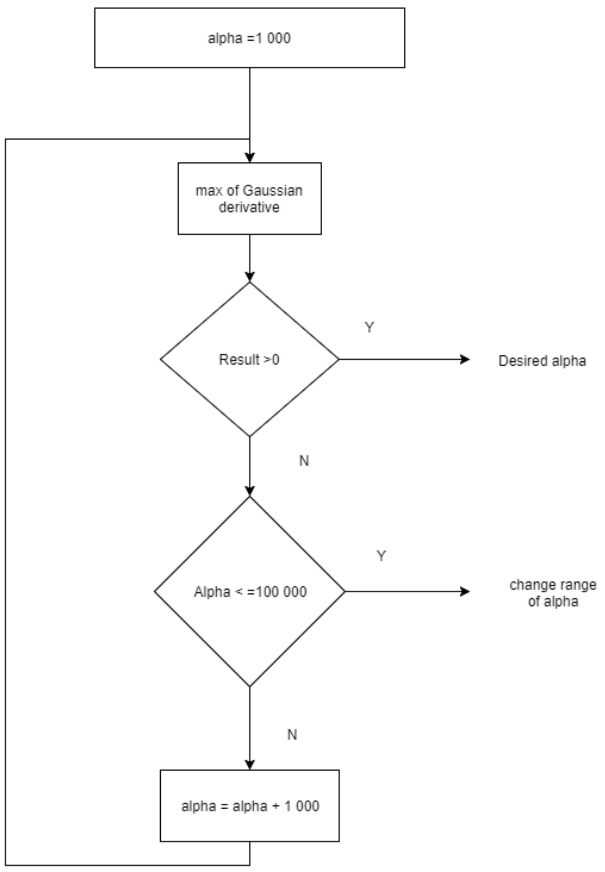


Figure 14.
Flow diagram of how the alpha value is determined.

3.2.2.b. Simulations

Filter length simulation

The filter length versus the F measure for the Gaussian edge detector was simulated and is shown below in **Figure 15**. The following experimental set-up was used:

- Aim: The aim of this experiment is to demonstrate the performance of the Gaussian edge detector.
- Queries: 30 unique queries were used.
- Approximate length of a query: 30 seconds.
- Type(s) of instrument(s): Guitar.
- Gaussian edge detector parameters:

  - Upper threshold: 0.1.
  - Lower threshold: -0.3.

- Performance measurement used: F measure.

- Platform: MATLAB.

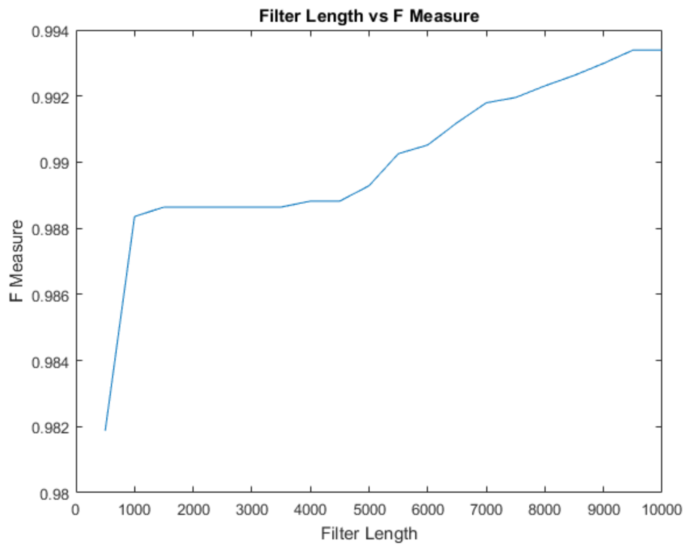


Figure 15.
Filter length vs F measure.

The filter length that is decided upon was 3 000. Simulations showed that the larger the filter length the more computationally expensive the algorithm, but also the better the F measure. A trade-off exists between computational complexity and F measure performance.

Instrument simulation

All three (3) types of instruments were simulated using the Gaussian edge detector with the F measure and accuracy compared as seen below in **Table 3**.

- Aim: The aim of this experiment is to demonstrate the performance of the Gaussian edge detector across all three (3) categories of instruments.

- Queries: 30 unique queries per instrument group were used.

- Approximate length of a query: 30 seconds.

- Types of instruments: Guitar, piano and ocarina (woodwind).

- Gaussian edge detector parameters:

  - Upper threshold: 0.1.

    - Lower threshold: -0.3.

- Performance measurement: F measure and accuracy.
- Platform: MATLAB.

Instrument	**F measure (%)**	**Accuracy (%)**
Piano	99.07	97.16
Guitar	98.65	98.19
Ocarina	72.85	81.27

Table 3.
Tabulated results for F measure and accuracy for Gaussian edge detector.

3.2.2.c. Optimisations

The optimisations performed on the Gaussian edge detector includes the following:

- Filter length: The filter length was optimised through a simulation as discussed above in the **Filter length simulation** section.
- Alpha value: The optimal alpha value was found to be 65 000. This was computed using the method discussed above in the **Determination of alpha** section.
- Redundant onset elimination: This algorithm is taken from the SF method and optimises the overall performance.

3.2.3 Performance evaluation

The performance of onset detectors are evaluated using the onset performance evaluator. This algorithm takes the detected onsets, the actual ground-truth onsets and the detection window length as inputs. The outputs are accuracy and F measure. The equation is described below in **Equation 3.7**:

$$F = \frac{2PR}{P+R} \tag{3.7}$$

Where:

$$P = \frac{N_{tp}}{N_{tp}+N_{fp}} \tag{3.8}$$

$$R = \frac{N_{tp}}{N_{tp}+N_{fn}} \tag{3.9}$$

The variables used in the above equations are described below:

- N_{tp}: The number of correctly detected onsets.

- N_{fn}: The number of missed onsets.
- N_{fp}: The number of false alarms.

3.3 PITCH DETECTION

The pitch is the second feature required to be extracted from the sampled audio query. The design and implementation of the chosen pitch detection algorithm namely Harmonic Product Spectrum (HPS) is described in this section.

3.3.1 HPS algorithm

3.3.1.a. Design

The block diagram describing the HPS algorithm is shown below in **Figure 16**.

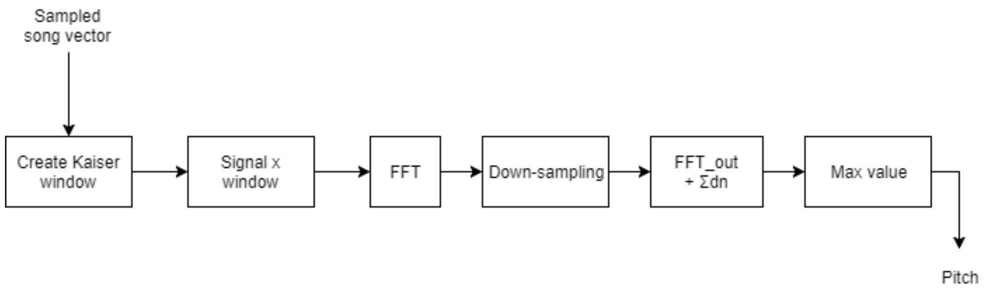


Figure 16.
HPS block diagram.

The following describes the HPS algorithm in detail:

- Step 1: The inter-onset sampled song data is passed to the HPS algorithm.
- Step 2: The window is created using a Kaiser window function.
- Step 3: The coefficients of the window function is multiplied with an inter-onset portion of the sampled signal.
- Step 4: The result of the multiplication is passed through an FFT function.
- Step 5: The absolute value of the result of the previous step is computed and then the base 10 logarithm is taken.
- Step 6: The resultant output of Step 5 is down-sampled. The rate of down-sampling is dependent on the type of instrument selected.
- Step 7: The output of the FFT function is then added to the down-sampled components.

- Step 8: The maximum value of the result in step 7 is then found. This value is then the pitch in Hz.

Equation 3.10 shown below indicates what generates the logarithm of the absolute value of the spectrum.

$$f = log(abs(FFT(x_{sampled} \times Kaiser(N)))) \tag{3.10}$$

The following parameters are used:

- $x_{sampled}$: inter-onset sampled song data.
- N: Length of the inter-onset sampled song data.

Down-sampling

The down-sampling process is described in **Figure 17** below.

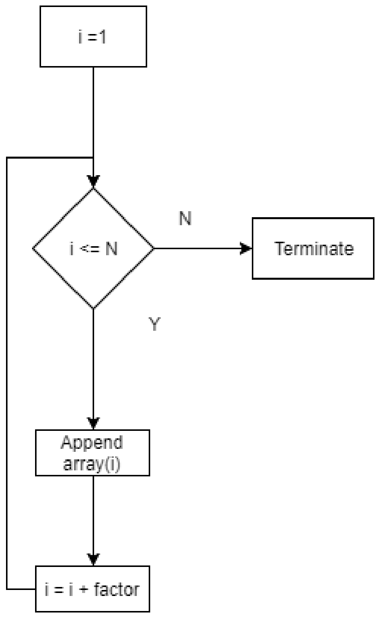


Figure 17.
Down-sampling flow diagram.

For an array length of N and down-sampling rate (factor), the down-sampling algorithm involves keeping the first sample and discarding the next factor -1 samples, and so on. The effect is to reduce the sampling rate by a chosen factor which has the effect of increasing the frequency resolution.

3.3.1.b. Simulations

FFT size simulation

The FFT size *n* is determined by varying *n* with regards to GPDE (Gross Pitch Detection Error) and can be seen in **Table 4**. The following experimental set-up was used:

- Aim: The aim of this experiment is to demonstrate the FFT size that yields the best performance.
- Queries: 30 unique queries were used.
- Approximate length of a query: 30 seconds.
- Types of instruments: Guitar.
- HPS parameters:
  - Down-sampling factor: 4.
- Performance measurement: GPDE.

n	**GPDE (%)**
2	100
4	100
6	93.27
8	75.79
10	46.46
12	11.36
14	5.62
16	2.06

Table 4.
GPDE for each *n*.

As seen from the table above, *n* should be equal to 16 to obtain the best pitch detection accuracy. Any value for *n* that is higher than 16 is not practical to use, as the algorithm becomes too computationally expensive.

Down-sampling factor simulation

The down-sampling factor is done in steps from one to four in order to find the lowest GPDE for each instrument. The process can be seen in **Table 4** below. The following experimental set-up was used:

- Aim: The aim of this experiment is to demonstrate the down-sampling factor that yields the best GPDE.

- Queries: 30 unique queries were used per instrument group.
- Approximate length of a query: 30 seconds.
- Types of instruments: Guitar, piano and ocarina (woodwind).
- HPS parameters:
  - *n*: 16
- Performance measurement: GPDE.

Down-sample factor	Piano (%)	Guitar (%)	Woodwind (%)
1	3.47	29.83	2.81
2	3.22	8.7	2.81
3	1.26	5.67	2.74
4	4.02	2.17	2.74

Table 5.
GPDE for each instrument based on the down-sampling factor.

It is clear form the table above that the following down-sample factors be used in each case:

- Piano: 3
- Guitar: 4
- Ocarina (woodwind): 4

3.3.1.c. Optimisations

the following optimisations are made for the HPS algorithm:

- Window function: Initially the window function used was the Hamming window but it was later changes to a Kaiser window to increase the performance of the HPS algorithm.
- FFT size *n*: The FFT size was optimised for performance and was found to be 16 as discussed in the section **FFT size simulation**.
- Down-sampling factor: The optimal down-sampling factor for each instrument was found in the **Down-sampling factor simulation** section.

3.3.2 Performance evaluation

The performance of the HPS algorithm is evaluated by providing perfect queries and ground-truth onsets for the function. The HPS algorithm will determine the pitches in the sampled audio query by using the onsets provided. The pitches are then converted to MIDI numbers and compared to the ground-truth pitches to obtain the GPDE.

3.4 MATCHING

The matching algorithm accepts two (2) feature representations as input and finds the variable length sections in both that match the best in terms of similarity.

3.4.1 SW algorithm

3.4.1.a. Design

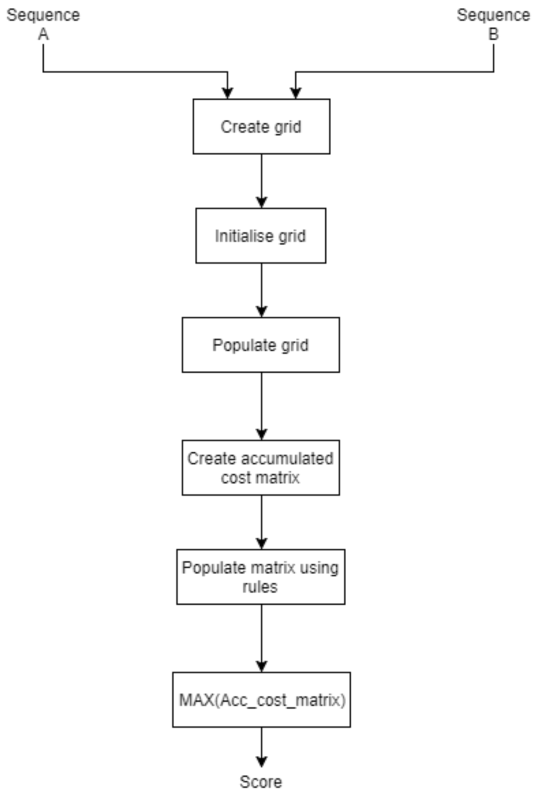


Figure 18.
SW block diagram.

Figure 18 above is a block diagram describing the SW algorithm. Assume two sequences A and B of lengths *M* and *N* respectively.

- Create grid: Construct *M* by *N* similarity grid.
- Initialise grid: The whole grid is initialised with zeros.

- Populate grid: Populate the grid with one (1) for a similarity and $-penalty$ for a dissimilarity.
- Create accumulated cost matrix (T): Construct $M+1$ by $N+1$ cost grid (T).
- Populate matrix using the following rules: Three (3) rules, described in more detail in **Figure 20**, are used to populate the accumulated cost matrix namely:
  - T1
  - T2
  - T3
- Calculate the maximum of accumulated cost matrix: $MAX(T1, T2, T3)$

Populate grid

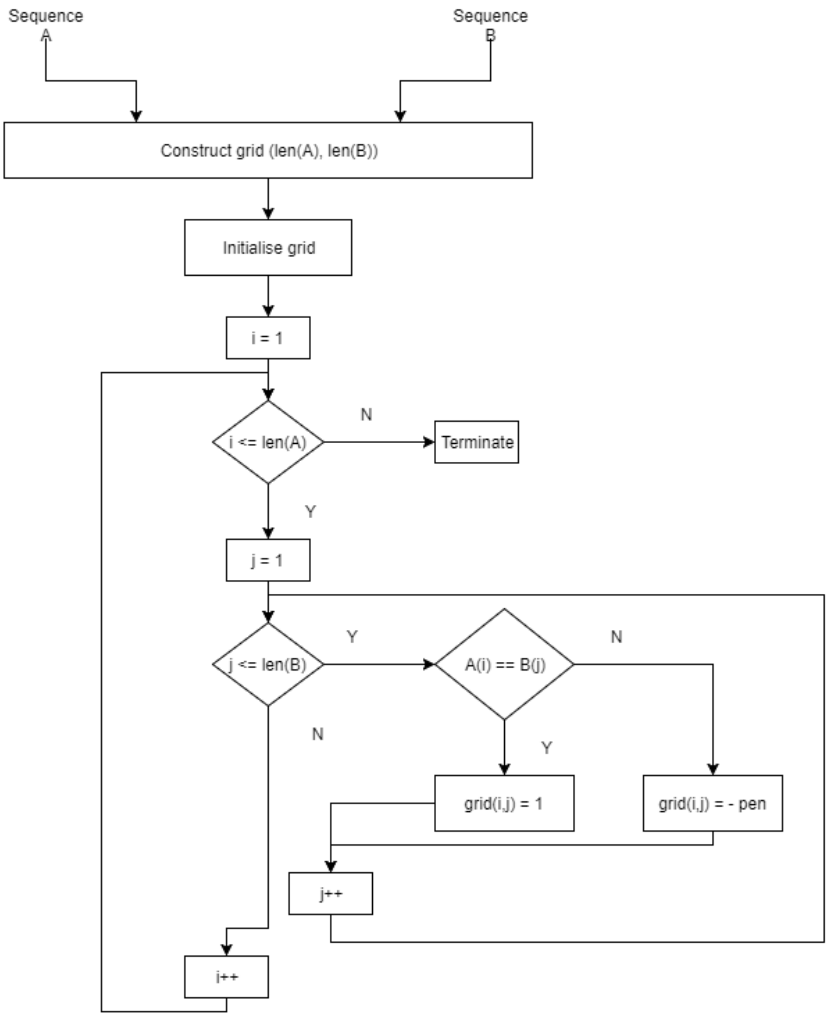


Figure 19.
SW populate grid flow diagram.

Figure 19 above, shows the steps associated with populating the SW grid in detail. In the event that $A(i) == B(j)$, where i iterates through sequence A and j iterates through sequence B, the grid coordinate (i, j) is populated with 1, otherwise $-penalty$ if $A(i) \neq B(j)$.

Calculate accumulated cost matrix and final score

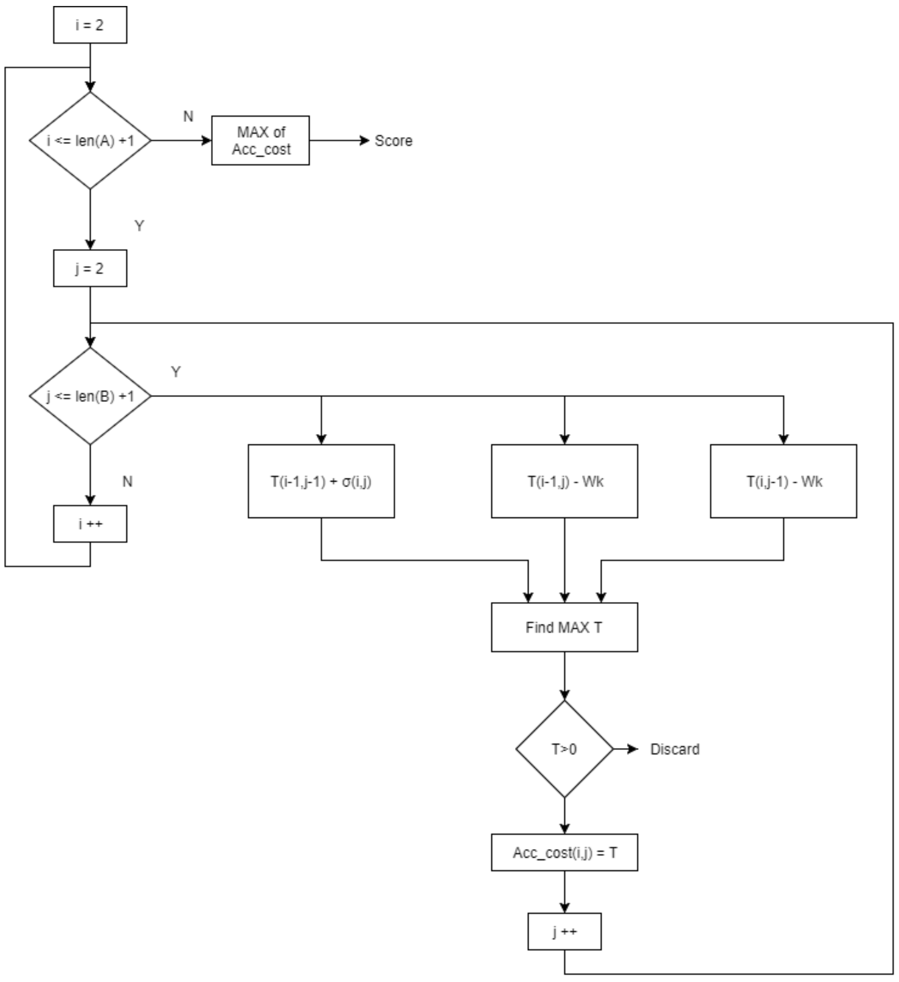


Figure 20.
Calculation of accumulated cost matrix and final score flow diagram.

Figure 20 above describes the method statement for calculating the accumulated cost matrix and final score. The following rules are used:

- Rule 1: $T1(i,j) = T(i-1, j-1) + grid(i,j)$
- Rule 2: $T2(i,j) = T(i-k, j) - (1 + penalty \times k)$
- Rule 3: $T2(i,j) = T(i, j-m) - (1 + penalty \times m)$

3.4.1.b. Optimisations

The SW algorithm is called twice for each database item. Once for the onset ratios and again for the pitch intervals to get a combined score. The optimal penalty for a gap was selected for each instrument as follows:

- Piano: The onset detection for piano performs very well and therefore a penalty value of 0.9 is used for onsets and 0.9 for pitch.
- Guitar: The onset detection for guitar performs very well and therefore a penalty value of 0.9 is used for onsets and 0.9 for pitch.
- Flute: The onset detection for flute performs within specification but not as well as the other instruments therefore a penalty value of 0.9 is used for onsets and 0.5 for pitch.

3.4.2 Performance evaluation

The SW algorithm's performance was evaluated by creating a list of scores for each database item when a song is queried. This list is then sorted in descending order in terms of score. The ranked list is then checked for Top-1, Top-5 and Top-10 respectively. The Top-X rankings accumulate as songs are queried to give and indication of matching performance.

3.5 DATA DESIGN

3.5.1 Internal software data structure

One-dimensional arrays of type double are passed internally between components i.e. the SF method calculates and outputs a vector of spectral differences as a one-dimensional array of doubles, which is then used by the peak picking algorithm.

3.5.2 Global data structure

Global data structures comprise of one-dimensional double arrays i.e. the sampled audio signal, array of onsets and pitches are one-dimensional double arrays.

3.5.3 Temporary data structure

One-dimensional double and integer arrays are used as temporary data structures to store onset and pitch information during the matching process.

A single two-dimensional double array is utilised during the matching process to store that database ranking and score.

Java array lists are used to store information that is later plotted in the piano roll.

3.5.4 Database description

A database is used to store all the complete MIDI songs locally. The items are stored with a numerical file name, enabling iteration with a for-loop. The song titles and artist names are stored in a CSV file whereby each entry correspond to the correct MIDI file i.e. entry number 1 in the CSV file corresponds to MIDI file number 1. Below in **Figure 21** a graphical representation of the database is provided.

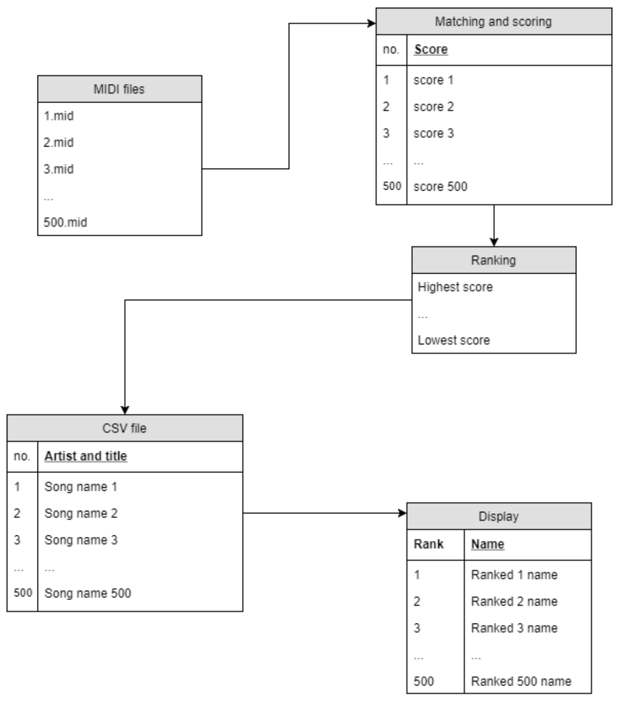


Figure 21.
Database description block diagram.

The database of 500 MIDI songs were obtained from a collection of open repositories [19].

3.6 ARCHITECTURAL AND COMPONENT-LEVEL DESIGN

3.6.1 Program structure

The program structure is modular and consists of input and output, indexed data structures, pipelines, procedural structures and algorithmic structures with a combination of sequential, iterative and conditional structures.

The QBP project's architecture consists of five (5) processing pipelines, where in some cases the pipelines are connected. However, all the pipelines ultimately connect to a Graphical User Interface (GUI). The pipelines are listed below.

- Query pipeline: Consists of taking the query and storing the sampled signal. The signal is stored in a format that can be processed and analysed.
- Onset extraction pipeline: The input is to take a sampled signal and to process it then for onsets. The output is a feature representation of the onsets.
- Pitch extraction pipeline: The input is to take a sampled signal and then to process it for pitches. The output is a feature representation of the pitches.
- Retrieval pipeline: MIDI files are loaded from the database, the contents are extracted and analysed so that a feature representation can be made.
- Matching and scoring pipeline: The feature representations from the onsets, pitches and database are fed into a matching algorithm that produces a score for each database item. The list is then ranked according to score and then made visible to the user.

3.6.2 Architecture diagram

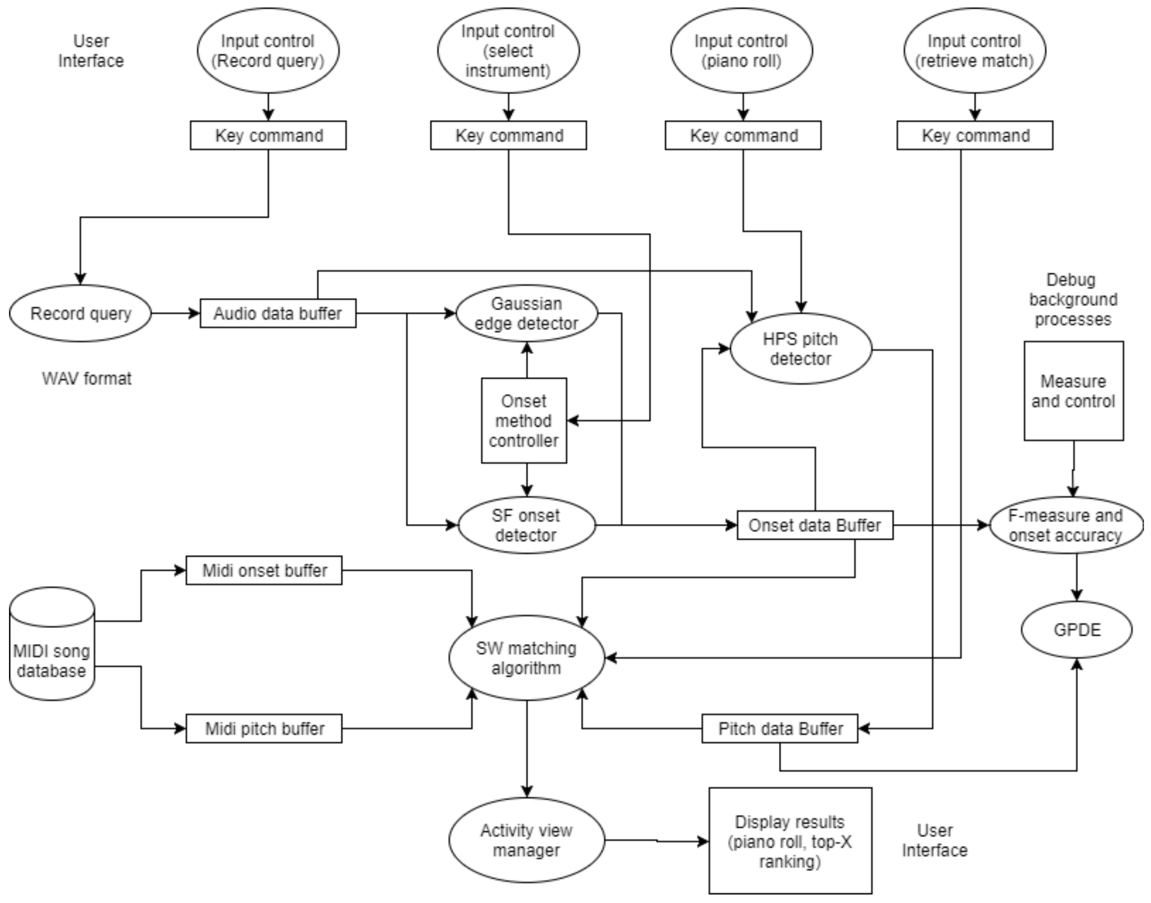


Figure 22.
Architecture block diagram.

The software architecture block diagram depicted in **Figure 22** is structured into input, output, control and process blocks. Several of the functional blocks are interconnected using data buffers.

3.6.2.a. User interface - Control inputs

The user interface comprises of several control inputs to initiate selected event-driven processes. The input control for record query initiates an audio recording in WAV format. The samples are stored in an audio data buffer for further downstream processing. The input control for select instrument is used by the onset method control function to select between the Gaussian edge detector process (for guitar and piano) and the SF onset detector for woodwind instruments. The input control for piano roll is used to initiate the onset detection process followed by the pitch detection process. The input control, Retrieve match is used to initiate the SW similarity matching process.

3.6.2.b. Control and process functions

The Record query process is used to record the audio query in WAV format. The sampled audio data is stored using an object orientated process in an audio data buffer. The audio data buffer links the Gaussian edge detector process, SF process and the HPS pitch detection process to the Record query process.

The HPS pitch detection process is linked to the output of the Gaussian edge detector process and the output of the SF process through an onset data buffer.

The SW similarity matching process is linked to the output of the HPS pitch detection process using a pitch data buffer. The SW similarity matching process is also linked to the output of the Gaussian edge detector process and the output of the SF process through an onset data buffer.

Onset and pitch data originating from the MIDI song database are linked to the SW similarity matching process using two data buffers namely a MIDI onset buffer and a MIDI pitch buffer.

3.6.2.c. User interface - View manager and display

The Activity view manager process is linked to the SW similarity matching process, and the piano roll results and matching results are processed by the activity view manager to displayed on the user interface.

3.6.2.d. Measurement and control debugger

Measure and control is a background process used for debugging and measurements. This process is used to calculate the F-measure and onset accuracy as well as the GPDE.

3.6.3 Alternatives

Pipe and filter architectures are suitable for the design of signal processors, and the filter blocks are oblivious to their neighbours allowing them to be constructed in parallel. The architecture requires encoding and decoding of the inputs and outputs. Event-based architectures are suitable for implementation of GUIs and supports re-use since new agents can easily be added. The data-centric architecture supports the implementation of databases which promotes ease of changing and/or adding information, and reduces the need to replicate complex data. Object-orientated architectures support abstract data types and object broker systems, and problems can be expressed as a set of interacting agents.

The software architecture selected represents a hybrid software architecture and makes use of several of the architecture types mentioned earlier. The reason for selecting this hybrid style of software architecture is that no single architecture type is suitable for describing the architecture when used on its own. The hybrid style is also most suited to the design and implementation of a project of this nature which comprises of a User Interface (UI) (input/output), database, signal processing, data types and data structures.

3.7 DESCRIPTION OF COMPONENTS

3.7.1 Class relationship diagram

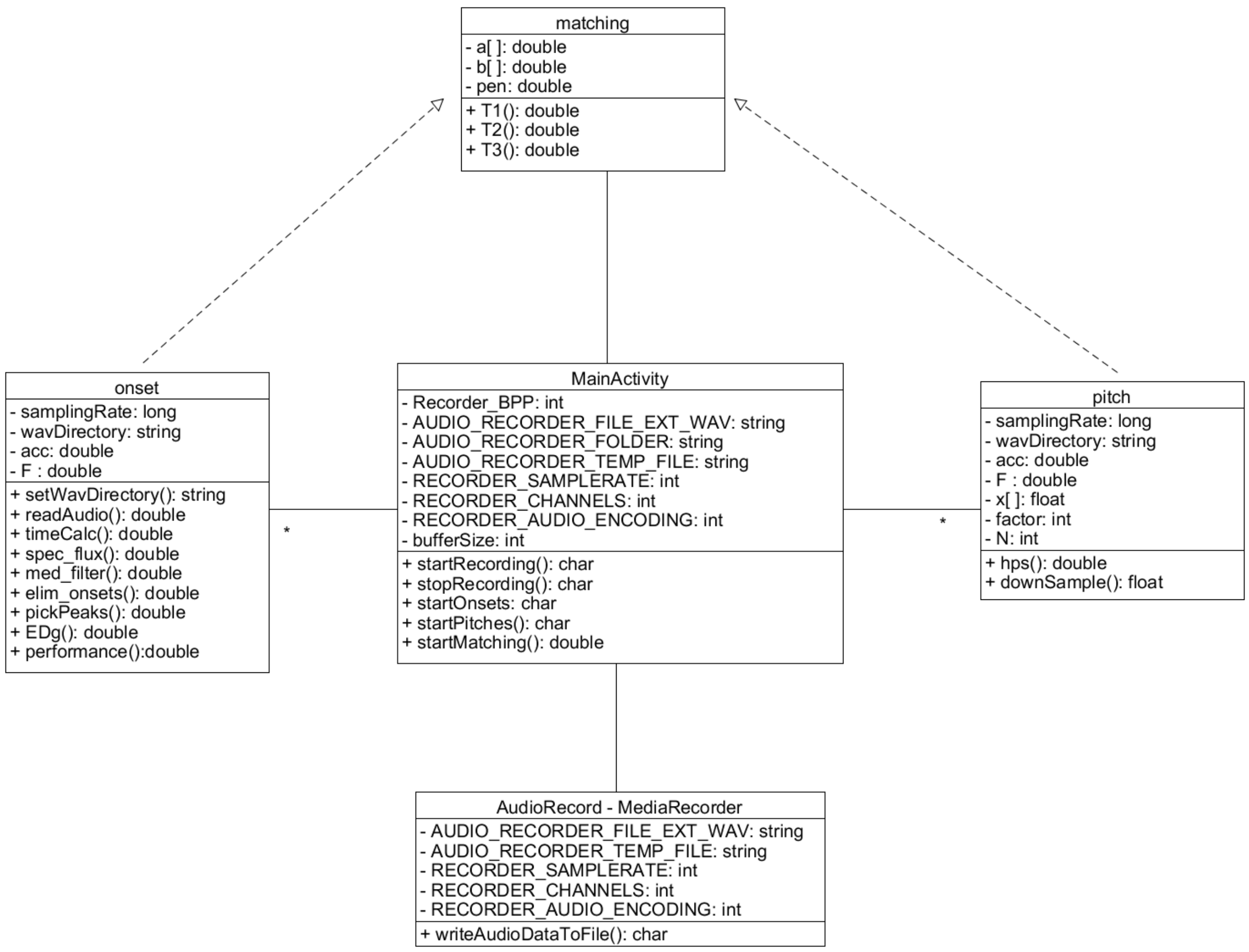


Figure 23.
Class relationship diagram.

3.7.2 Description for class Main Activity

Class name: MainActivity	**Method Description**
	A method to initialize, create and make available for use objects such as the WAV recorder, onset detector, pitch detector, matching algorithm, and plotting libraries for the piano roll.
	Processing Narrative (PSPEC)
	When a user opens the mobile application a quick-start instruction guide is loaded to explain: • Recording of an audio query; • Selecting an instrument; • Creating a piano roll; • Retrieving the ranking list of songs (the layout for the above elements is also created).
	Program Description Language

	Public class MainActivity extends AppCompatActivity {
	// initialization of WAV recorder, sampling rate and storage directories
	// onCreate creates layout for user interface
	// creation of an onset object and calling the object's primary function
	// creation of a pitch object and calling the object's primary function
	// creating feature representation and plotting the piano roll
	// database retrieval
	// creation of a matching object and calling the object's primary function
	//display ranked and sorted song names
	}

	Restrictions and/or limitations
	Since MainActivity is self-contained there are no practical restrictions.

Table 6.
Description for MainActivity part one.

Class name: MainActivity	**Local data structures**
	The following local data structures are used: • One-dimensional array of doubles • Two-dimensional array of doubles • Java array lists
	Performance issues
	Performance is limited because of the limited application of parallel processing due to the sequential nature of the program flow.
	Design constraints
	The class is constrained by the limitation of only one instance per handset user. The design is constrained by a single query that needs to be processed before the next query can be processed.
	Interface Description
	MainActivity interfaces with the following components: • onset() • pitch() • matching()

Table 7.
Description for MainActivity part two.

3.7.3 Description for class onset

Class name: onset	Method Description
	A method to read recorded WAV files and analyze and process sampled audio data using the SF method and the Gaussian edge detector method to obtain onsets.
	Processing Narrative (PSPEC)
	The object is responsible for calling the following functions: • readAudio • spec_flux method • med_filter method • elim_onsets method • pickPeaks method • EDg method • performance method
	Program Description Language

	Public class onset {
	// readAudio reads in the stored WAV file and provides a one−dimensional array sampled audio data
	// timeCalc calculates the center time of each element in the SF difference array
	// spec_flux method calculates a SF difference vector based on the SF algorithm
	// med_filter method filters the SF algorithm output using a median filter for peak picking
	// elim_onsets method is applied to eliminate redundant onsets
	// pickPeaks method uses spec_flux, med_filter and elim_onsets to detect only relevant onsets
	// EDg method calculates the onsets using the Gaussian edge detector
	// performance calculates the F−measure and accuracy using detected onsets and ground truth onsets
	}

Table 8.
Description for onset part one.

Class name: onset	**Restrictions and/or limitations**
	Onsets that occur within a period of less than 50 ms of each other cannot be detected, due to the refractory period of 50 ms to eliminate redundant onsets. Onsets can only be obtained for the specified instruments (guitar, piano and woodwind).
	Local data structures
	One-dimensional array of doubles are used.
	Performance issues
	The sampling rate had to be reduced from 44.1 kHz to 22.05 kHz to enhance the performance due to large array sizes of the sampled data. The speed of execution improved, and the overall performance of the algorithm was not impacted negatively.
	Design constraints
	The class is constrained by the limitation of only one instance per handset user. The design is constrained by a single query which needs to be processed before the next query can be processed.
	Interface Description
	MainActivity()

Table 9.
Description for onset part two.

3.7.4 Description for class pitch

Class name: pitch	**Method Description**
	A method that uses sampled audio data and detected onsets to obtain pitches using the HPS algorithm.
	Processing Narrative (PSPEC)
	The object is responsible for calling the following functions: • hps method • downSample method
	Program Description Language

	Public class pitch {
	// hps processes the sampled audio data and the detected onsets and returns the pitch(es) using the HPS algorithm
	// downSample method down−samples an data array that is passed to it
	}

	Restrictions and/or limitations
	Pitches that are associated with onsets that occur within a period of less than 50 ms of each other cannot be detected, due to the refractory period of 50 ms to eliminate redundant onsets. Pitches can only be obtained for the specified instruments (guitar, piano and woodwind).
	Local data structures
	One-dimensional array of doubles are used.
	Performance issues
	The sampling rate had to be reduced from 44.1 kHz to 22.05 kHz to enhance the performance due to large array sizes of the sampled data. The speed of execution improved, and the overall performance of the algorithm was not impacted negatively.
	Design constraints
	The class is constrained by the limitation of only one instance per handset user. The class is constrained by the limitation that detected onsets are required to be obtained before the pitches can be obtained.
	Interface Description
	MainActivity()

Table 10.
Description for pitch.

3.7.5 Description for class matching

Class name: matching	**Method Description**
	A method that uses the SW algorithm to produce a similarity score between a feature representation created from the detected onsets and detected pitches and a feature representation derived from the MIDI database of songs.
	Processing Narrative (PSPEC)
	The object is responsible for calling the following functions: • SW method • T1 method • T2 method • T3 method
	Program Description Language

	Public class matching {
	// SW method calls T1, T2 and T3 and produces a similarity score between the query feature representation and the database ground truth feature representation
	// T1 processes condition 1 of the SW algorithm
	// T2 processes condition 2 of the SW algorithm
	// T3 processes condition 3 of the SW algorithm
	}

Table 11.
Description for matching part one.

Class name: matching	Restrictions and/or limitations
	Two instances of the SW algorithm had to be called. One for onset feature similarity scoring and one for pitch similarity scoring since the algorithm can only process two one-dimensional data arrays at any one time.
	Local data structures
	One-dimensional array of doubles are used.
	Performance issues
	A complete database representation of the songs could be not be used due to the extended execution time of the SW algorithm. The songs in the database were truncated with affecting the overall similarity matching performance as songs are largely repetitive.
	Design constraints
	The class is constrained by the limitation that detected onsets and detected pitches are required to be obtained, before the similarity score can be obtained.
	Interface Description
	MainActivity()

Table 12.
Description for matching part two.

3.8 SOFTWARE INTERFACE DESCRIPTION

3.8.1 Human interface

The user interface has been designed around the display of an Android mobile device consisting of a Super AMOLED capacitive touch screen. Several distinctive icons were placed on the bottom of the screen to serve as control inputs and the mobile device screen was used to display the piano roll and ranked songs. The built-in microphone is used as an interface for recording musical songs generated by a person.

3.9 USER INTERFACE DESIGN

3.9.1 Description of user interface

The menu will consist of various GUI components such as icons, buttons, text fields, and list objects. The quick-start menu will be arranged in such a way that the user will intuitively and efficiently be able to perform the operations. A detailed description of the menu and the user's interactions are described in **Section 8.1**. The screen images of the user interface are presented in **Section 3.9.2** below.

3.9.2 Objects and actions

3.9.2.a. Welcome screen

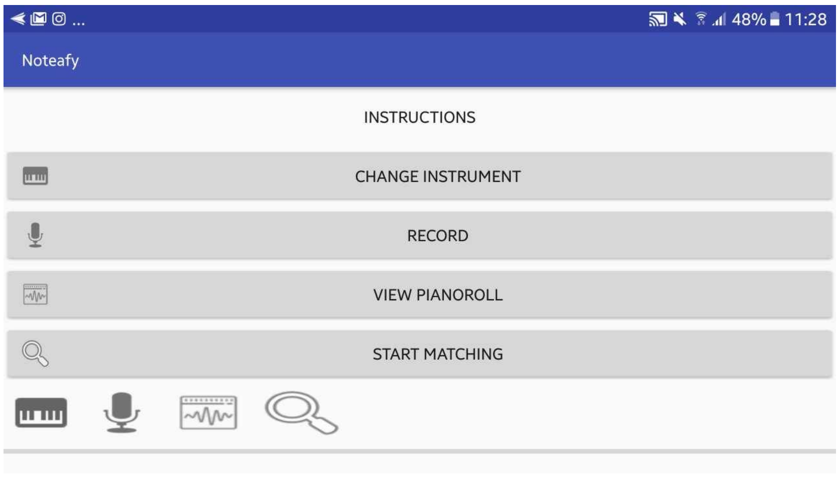


Figure 24.
Start-up screen image.

The user is greeted with a quick-start instruction guide of how to operate the application. All the icons and the buttons are briefly explained so that the user can start operating the application shortly after reading the instructions.

3.9.2.b. Recording screen

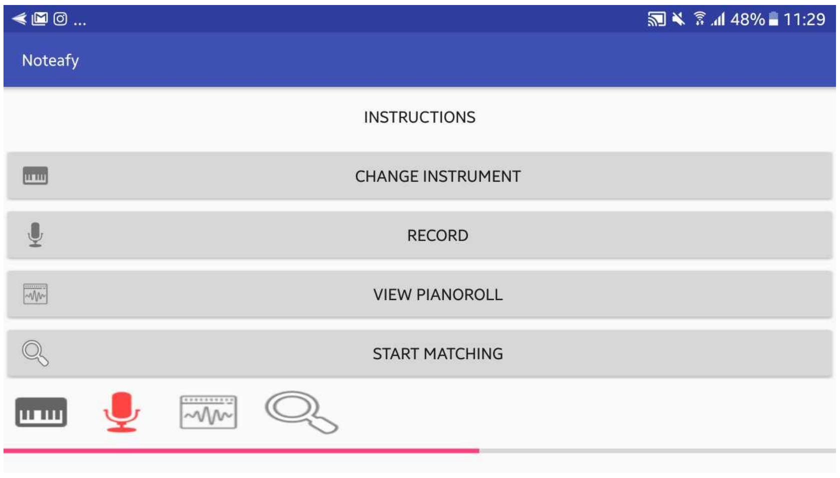


Figure 25.
Recording screen image.

In the event that the user starts recording audio, the microphone icon changes colour and a progress bar is displayed. The user can clearly see the period of time that the microphone is active and how long a sufficient audio query needs to be.

3.9.2.c. Piano roll screen

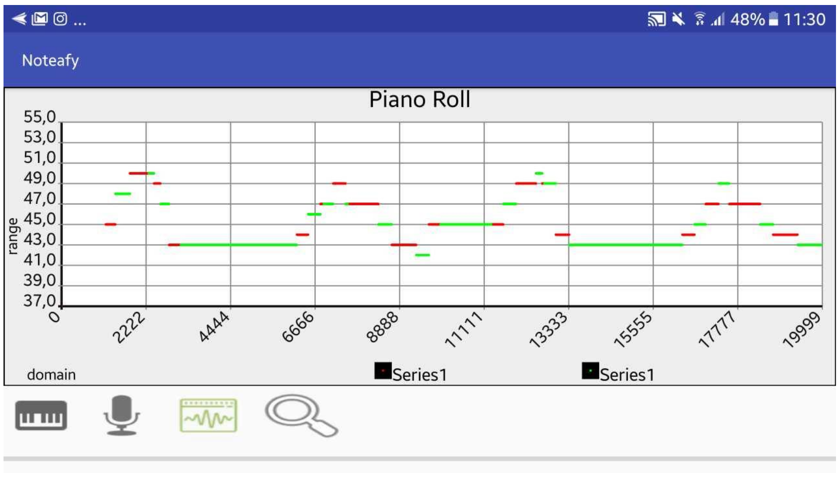
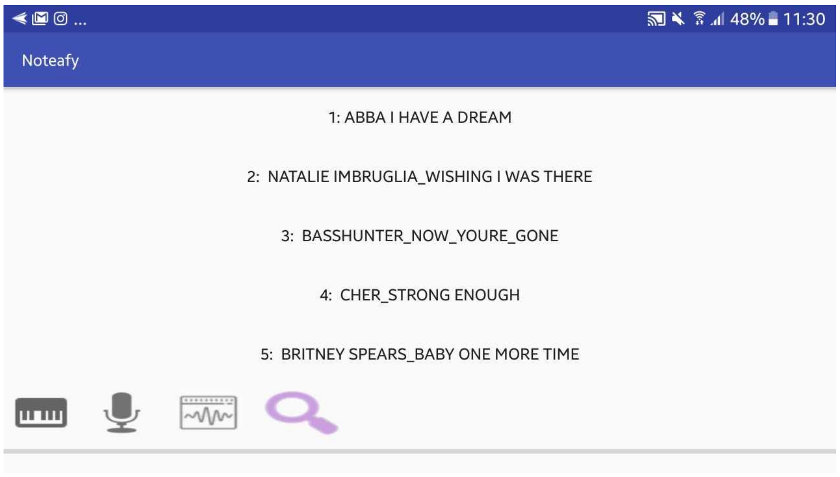


Figure 26.
Piano roll screen image.

The piano roll screen makes the pitch contour as well as onset times and note durations clearly visible to the user, especially when the mobile device's display is in landscape mode.

3.9.2.d. Matching screen



1: ABBA I HAVE A DREAM

2: NATALIE IMBRUGLIA_WISHING I WAS THERE

3: BASSHUNTER_NOW_YOURE_GONE

4: CHER_STRONG ENOUGH

5: BRITNEY SPEARS_BABY ONE MORE TIME

Figure 27.
Matching screen image.

Once the user is ready to search the database the matching icon can be pressed and the top five results are shown after a short period of time.

3.9.3 Interface design rules

The following five elements as described in [20] for the UI design have been taken into account:

- Cognition
- Ergonomics
- Utility
- Image
- Ownership

UI Design Element	Application	Relevance
Cognition	The layout of the icons and the use thereof are initiative and consistent. A quick-start guide is available at start-up of the application.	Addresses the expectation of the user and promotes ease of use.
Ergonomics	Colours selected have meaning and their brightness has been toned down. The size of the icons was made large enough and they were spaced out. Pictorial symbols were used for the icons.	Allows for ease of use, and is softer on the eye. The size and spacing of the icons reduces the incidence an erroneous operations. The pictorial nature of the icons promotes ease of understanding and use of the application.
Utility	The piano roll is a visual representation of the onsets versus the pitches. The audio recording includes a visual progress bar.	Visual representation of the piano roll has more meaning than a list of onset and pitch values. Provides the users with feedback on audio recording length progress to avoid disappointment due to truncated query lengths.
Image	The icons have the same proportion to one another. The positions of the icons and main display remain constant. Due to the use of the mobile device the main screen is large enough to display in landscape or portrait mode.	Promotes an attractive look and feel.
Ownership	The use of Android smart phones is pervasive. The application including the database does not occupy large amounts of storage space.	Smart devices are already owned by users. The QBP application can be bolted-on to the existing mobile devices. The storage cost is insignificant and songs are local without the need to access the Internet with associated data charges.

Table 13.
Description of user interface.

3.9.4 Components available

Android provides a wide variety of GUI components available for use. The following components are used:

- Activity: This refers to a single focused environment housing all UI components, created so that the user is able to interact with all these components.
- Linear Layout: A layout that arranges other views either horizontally in a single column or vertically in a single row. The linear layout arranges UI components or views into a single column, horizontally or a single row, vertically.
- Button: A UI component providing the user with something to tap and select.
- Icon: A UI component that gives other UI components meaning.
- Progress Bar: A UI component that acts as an indicator of progress i.e. seconds elapsed, so that the user can stay informed.
- Android XY Plot: A modified view that can be used to create plots and communicate meaningful information to the user.

3.9.5 User Interface Development System (UIDS) description

The UIDS is provided by Android Studio which allows the use of an XML file to specify the layout of GUI components. The UIDS is able to generate XML code from a visual layout editor which is then later edited in code. Java was employed to handle the functionality of XML elements such as event handlers and autonomous updates of GUI components.

3.10 USE CASES

The use cases for the system can be found in **Appendix 1**, **Section 8.1**.

3.11 RESTRICTIONS, LIMITATIONS AND CONSTRAINTS

This section describes the issues that have an impact on the design and implementation of the software.

Due to the large data arrays and long execution times resulting from using a 44.1 kHz sampling rate, the sampling rate was lowered to 22.05 kHz without affecting the overall performance of the system.

The user is required to consistently use either a portrait or landscape orientation on the display of the mobile phone device. Switching between orientations causes a loss of data, causing the the user to re-record the audio query and the application to recreate the piano roll.

Audio queries are required to be at least 15 seconds in length to provide a long enough feature representation for the matching algorithm. Shorter query lengths negatively affect the performance.

The MIDI database of 500 songs is locally stored on the device and can only be downloaded to and updated locally within the device storage.

The software implementation is largely sequential in nature, for example the audio query must be recorded before the onset and pitch detection can be initiated. The pitch detection can only be initiated once the onset detection has been completed, and matching can only commence once pitch and onset detection are complete. This limits the ability to implement parallel processing and pipe-lining techniques.

3.12 DESIGN SUMMARY

Task	Implementation	Task completion
Investigate the use of standard audio recording libraries	The AudioRecorder library from the Android SDK was selected.	Completed.
Design of the SF method for onset detection.	Implemented in MATLAB and Android	Completed.
Design of Gaussian edge detector for onset detection.	Implemented in MATLAB and Android.	Completed.
Design of redundant onset eliminator for onset detection.	Implemented in MATLAB and Android.	Completed.
Simulation of SF method and Gaussian edge detector.	Performed in MATLAB with 30 unique queries per instrument group.	Completed.
Optimisation of SF method.	Optimal values were obtained for the window function, window length, step size and other parameters.	Completed.
Optimisation of Gaussian edge detector.	Optimal values were obtained for the filter length and the value of alpha.	Completed.
Performance evaluation for onset detection.	The performance of the onset detectors was measured in terms of accuracy and F measure.	Completed.

Table 14.
Design summary part one.

Task	Implementation	Task completion
Design of HPS pitch algorithm.	Implemented in MATLAB and Android.	Completed
Simulation of HPS pitch algorithm.	Simulation with 30 unique queries per instrument and selection of relevant down-sampling factors.	Completed.
Optimisation of HPS pitch algorithm.	Optimised the window function, FFT size and down-sampling factors.	Completed.
Performance evaluation for pitch detection.	The performance of the HPS algorithm was measured in terms of GPDE.	Completed.
Design of SW matching algorithm.	Implemented in MATLAB and Android.	Completed.
Optimisation of SW matching algorithm.	The optimal penalty for each instrument group was determined.	Completed.
Performance evaluation for matching.	Obtained list of Top-X rankings for 90 queries.	Completed.
Database design.	Implemented in MATLAB and Android for storage 500 MIDI songs.	Completed.
Software design.	Implemented in Android.	Completed.
Design of user interface.	Implemented in Android.	Completed.

Table 15.
Design summary part two.

4. Results

4.1 SUMMARY OF RESULTS ACHIEVED

Description or requirement of specification (intended outcome)	**Actual outcome**	**Location in report**
Mission requirement of the product		
Software application for smartphone.	The software application was implemented in Android for a smartphone.	Section 3.7
Simple GUI.	The student designed and implemented a simple GUI.	Section 3.9
MIR of song titles from a database following the playing of a song into the microphone of the device.	The student designed and implemented an algorithm for matching.	Section 3.7.5
Displaying the title of the song(s) matched in the database.	Top-5 song titles retrieved and displayed on GUI.	Section 3.9.2.d
Field conditions		
Live testing of the application in an acoustically untreated room with background and device (microphone) noise.	User acceptance testing was conducted in a live acoustically untreated environment.	Section 4.5.2
Specifications		
Top-1 Hit Rate for monophonic music ≥ 55 %. Top-5 Hit Rate for monophonic music ≥ 75 %. Top-10 Hit Rate for monophonic music ≥ 85%.	Top-1 Hit Rate for monophonic music = 77.78 %. Top-5 Hit Rate for monophonic music = 87.78 %. Top-10 Hit Rate for monophonic music = 88.89%.	Section 4.5.2.a
Mean Reciprocal Rate (MRR) ≥ 60 %.	Mean Reciprocal Rate (MRR) = 81.96 %.	Section 4.5.2.a
Gross Pitch Detection Error (GPDE) ≤ 15 %.	Gross Pitch Detection Error (GPDE) = 1.48	Section 4.3.2.d
F-measure for onset detection ≥ 75 %.	F-measure for onset detection = 95.64 %.	Section 4.3.2.b

Table 16.
Summary of results achieved part one.

Description or requirement of specification (intended outcome)	**Actual outcome**	**Location in report**
Deliverables		
Android application developed for MIR.	The student implemented the software application in Android for a smartphone.	Section 3.7
User manual for the application.	The student implemented a quick-start guide on the welcome screen.	Section 3.9.2.a
Large database of MIDI files.	The student completed the design and implementation of a database of 500 MIDI songs.	Section 3.5.4
The student completed the design and implementation of onset and pitch detection algorithms.	The student designed and implemented onset and pitch detection algorithms.	Section 3.7.3 and Section 3.7.4
Design of matching algorithm.	The student completed the design and implementation of the matching algorithm.	Section 3.7

Table 17.
Summary of results achieved part two.

4.2 COMPONENT-LEVEL QUALIFICATION TESTS

4.2.1 Onset Detection qualification test

4.2.1.a. F-Measure and onset accuracy test

Objectives of the F-Measure test

The objective of the F-Measure test is to verify that the F-measure specifications associated with the two chosen methods of OSS namely, SF and Gaussian edge detection have been achieved.

F-Measure and onset accuracy

The F-measure is a measure of the onset detection accuracy. It records the degree to which the chosen OSS algorithms correctly detect onsets, fail to detect onsets and/or incorrectly detect onsets. The formula for F-measure as recorded in the mission-critical system specifications is in **Section 4**, **Part 3**. The average F-measure is computed as the average for 90 queries. The onset accuracy per query is computed as the number of correctly detected onsets expressed as a ratio of the total number of ground truth onsets. The average onset accuracy is computed as the average for 90 queries.

Specification

The specification for F-Measure set out in the mission-critical system specifications in **Section 4**, **Part 3** was recorded as: F-measure for onset detection ≥ 75 %.

Experimental Software tools

The recording of F-Measure is performed within the MATLAB simulation environment.

Experimental parameters and set-up

The query list was populated with 90 perfect queries in WAV format each with a query length of between 15 and 20 seconds. There are 30 unique queries associated with each of the three musical instruments namely piano, guitar and ocarina (woodwind). The ground-truth songs associated with the 90 perfect queries were previously annotated with ground truth onsets, several of which were hand annotated using Sonic Visualizer and other applicable tools.

Experimental protocol

The queries are read in one after the other and each query is processed independently. The first step is to read the associated WAV file and the query is thereafter processed by the respective OSS algorithm. The detected onsets are compared to the ground truth onsets and the F-measure is calculated and recorded for each of the 90 perfect queries. Thereafter the average F-measure for the 90 perfect queries is calculated and recorded.

4.2.1.b. Results and observations

Measurements

Instrument	Average F-measure (%)	Average accuracy (%)
Piano	98.93	96.91
Guitar	98.54	97.68
Woodwind	90.55	85.61
Overall	96.01	93.40

Table 18.
MATLAB average F-measure and accuracy per instrument.

Description of results

The average F-measure and average accuracy expressed in percentages were recorded for each of the musical instruments namely piano, guitar and ocarina (woodwind). The overall average F-measure is 96.01 % and overall average accuracy is 93.40 % for the 90 perfect queries, respectively recorded in **Table 18** above.

4.2.2 Pitch detection qualification test

4.2.2.a. GPDE test

Objectives of the GPDE test

The objective of the GPDE test is to verify that the GPDE specification associated with the HPS pitch detection algorithm has been achieved.

GPDE

The GPDE is a measure of the pitch detection accuracy. It records the degree to which the chosen HPS algorithm correctly detects pitches. The average GPDE is computed as the average for 90 queries.

Specification

The specification for GPDE set out in the mission-critical system specifications in **Section 4**, **Part 3** was recorded as: Gross Pitch Detection Error (GPDE) $\leq$ 15 %.

Experimental Software tools

The recording of GPDE is performed within the MATLAB simulation environment.

Experimental parameters and set-up

The query list was populated with 90 perfect queries in WAV format each with a query length of between 15 and 20 seconds. There are 30 unique queries associated with each of the three musical instruments namely piano, guitar and ocarina (woodwind). The actual pitches of the ground-truth songs associated with the 90 perfect queries are contained within the MIDI database.

Experimental protocol

The queries are read in one after the other and each query is processed independently. The first step is to read the associated WAV file and the query is thereafter processed by the HPS algorithm. The detected pitches are compared to the ground truth pitches and the GPDE is calculated and recorded for each of the 90 perfect queries. The average GPDE is computed as the average of the GPDEs for 90 perfect queries.

4.2.2.b. Results and observations

Measurements

Instrument	**Average GPDE (%)**
Piano	1.26
Guitar	1.30
Woodwind	2.74
Overall	2.65

Table 19.
MATLAB average GPDE per instrument.

Description of results

The average GPDE expressed as a percentage was recorded for each of the musical instruments namely piano, guitar and ocarina (woodwind). The overall average for GPDE is recorded in **Table 19** above as 2.65 %.

4.2.3 Matching qualification test

4.2.3.a. SW matching test

Objectives of the SW matching test

The objective of the SW matching test is to verify that similarity matching specifications associated with the SW algorithm have been achieved.

Specification

The specification for Top-X and MRR set out in the mission-critical system specifications in **Section 4**, **Part 3** was recorded as: Top-1 Hit Rate for monophonic music ≥ 55 %. Top-5 Hit Rate for monophonic music ≥ 75 %. Top-10 Hit Rate for monophonic music ≥ 85%. Mean Reciprocal Rate (MRR) ≥ 60 %.

Experimental Software tools

The recording of Top-X and MRR is performed within the MATLAB simulation environment.

Experimental parameters and set-up

The query list was first populated with 90 perfect queries in WAV format each with a query length of between 25 and 30 seconds. There are 30 unique queries associated with each of the three musical instruments namely piano, guitar and ocarina (woodwind).

Thereafter the query list was populated with 90 distorted queries. The distorted queries were formed by altering every sixth note in the sequence using a circular application of four (4) possible distortions namely: delete note, increase semitone, decrease semitone or insert note. The level of distortion was introduced to simulate the query imperfections that may be introduced by a person playing a live query.

The 90 distorted queries were replicated and two separate sets were formed, one with an average query length of 15 to 20 seconds and the second set with an average query length of 25 to 30 second.

Experimental protocol

The queries are read in one after the other and each query is processed independently. The first step is to read the associated WAV file and the query is thereafter processed by the SW algorithm. The Top-X and MRR are calculated for the 90 perfect queries, the 90 distorted queries (length 15-20 seconds) and the 90 distorted queries (length 25-30 seconds).

4.2.3.b. Results and observations

Measurements

Measurement	Result (%)
Top-1	100
Top-5	100
Top-10	100
MRR	100

Table 20.
MATLAB Matching results for 25 to 30 second perfect queries.

Measurement	Result (%)
Top-1	82.22
Top-5	88.89
Top-10	91.11
MRR	84.38

Table 21.
MATLAB Matching results for 15 to 20 second imperfect queries.

Measurement	Result (%)
Top-1	93.33
Top-5	94.44
Top-10	95.56
MRR	94.12

Table 22.
MATLAB Matching results for 25 to 30 second imperfect queries.

Description of results

The Top-1, Top-5, Top-10 and MRR results are recorded for 90 perfect queries, 90 imperfect queries (length 15-20 seconds) and 90 imperfect queries (length 25-30 seconds) in **Table 20**, **Table 21** and **Table 22** respectively.

4.3 COMPONENT TEST PLANS AND PROCEDURES

4.3.1 Component test strategy overview

The next section provides an overview of the testing strategy used for each identified component.

4.3.2 Component test procedure

This section describes the software component level test procedure.

4.3.2.a. Component: Audio recording

Test case number	CL_001
Features to be tested	Audio Recording
Testing approach	The audio is recorded by way of the standard off-the-shelf AudioRecorder library from the Android SDK with a sample frequency of 22.05 kHz. The testing is to confirm that the resultant saved WAV file is correctly retrieved and processed by the Java WAV file IO object to produce a the sampled audio array of doubles.
Pass/fail criteria	Pass if the WAV file and sampled audio array is created.
Testing environment	Android environment.

Table 23.
Audio recording test component.

4.3.2.b. Component: Onset detection

Test case number	CL_002
Features to be tested	F-Measure and onset accuracy
Testing approach	The query length is between 15 and 20 seconds. There are three (3) sets of 30 unique queries each associated with each of the three (3) musical instruments respectively namely piano, guitar and ocarina (woodwind). Each query is processed independently. The first step is to read the associated WAV file and the query is thereafter processed by the respective OSS algorithm. The detected onsets are compared to the ground truth onsets and the F-measure is calculated and recorded for each of the 90 perfect queries. The average F-measure for the 90 perfect queries is calculated and recorded.
Pass/fail criteria	Pass if the F-measure for onset detection ≥ 75 %.
Testing environment	Android environment.

Table 24.
Onset detection test component.

Onset detection test results

Instrument	Average F-measure (%)	Average accuracy (%)
Piano	98.90	96.98
Guitar	98.67	97.98
Woodwind	89.39	85.20
Overall	95.64	93.41

Table 25.
Android average F-measure and accuracy per instrument.

4.3.2.c. Description of results

The average F-measure and average accuracy expressed in percentages were recorded for each of the musical instruments namely piano, guitar and ocarina (woodwind). The overall average F-measure is 95.64 % and overall average accuracy is 93.41 % for the 90 perfect queries respectively as recorded in **Table 25** above.

4.3.2.d. Component: Pitch detection

Test case number	CL_003
Features to be tested	Pitch detection and GPDE
Testing approach	The query list was populated with 90 perfect queries in WAV format each with a query length of between 15 and 20 seconds. There are 30 unique queries associated with each of the three musical instruments namely piano, guitar and ocarina (woodwind). The actual pitches of the ground-truth songs associated with the 90 perfect queries are contained within the MIDI database. The detected pitches are compared to the ground truth pitches and the GPDE is calculated and recorded for each of the 90 perfect queries. The average GPDE is computed as the average of the GPDE's for 90 perfect queries.
Pass/fail criteria	Pass if the Gross Pitch Detection Error (GPDE) ≤ 15 %.
Testing environment	Android environment.

Table 26.
Pitch detection test component.

Pitch detection test results

Instrument	**Average GPDE (%)**
Piano	1.65
Guitar	1.35
Woodwind	1.44
Overall	1.48

Table 27.
Android average GPDE per instrument.

Description of results

The average GPDE expressed as a percentage was recorded for each of the musical instruments namely piano, guitar and ocarina (woodwind). The overall average for GPDE is recorded in **Table 27** above, as 1.48 %.

4.4 SYSTEM TEST PLANS AND PROCEDURES

4.4.1 System test strategy overview

The next section provides an overview of the system testing strategy.

4.4.2 System level: Matching

Test case number	SL_001
Features to be tested	Similarity Matching
Testing approach	The query list was first populated with 90 perfect queries in WAV format each with a query length of between 25 seconds and 30 seconds. There are 30 unique queries associated with each of the three musical instruments namely piano, guitar and ocarina (woodwind). Thereafter the query list was populated with 90 distorted queries. The distorted queries were formed by altering every sixth note in the sequence using a circular application of four (4) possible distortions namely: delete note, increase semitone, decrease semitone or insert note. The level of distortion was introduced to simulate the query imperfections that may be introduced by a person playing a live query. The 90 distorted queries were replicated and two separate sets were formed one with an average query length of 15 to 20 seconds and the second set with an average query length of 25 to 30 seconds.

Table 28.
Matching system level test part one.

Test case number	SL_001
Pass/fail criteria	Pass if all of the below conditions are met: • Top-1 Hit Rate for monophonic music ≥ 55 %. • Top-5 Hit Rate for monophonic music ≥ 75 %. • Top-10 Hit Rate for monophonic music ≥ 85 %. • MRR (Mean Reciprocal Rate) ≥ 60 %.
Testing environment	Android environment.

Table 29.
Matching system level test part two.

4.4.2.a. Matching test results

Measurement	**Result (%)**
Top-1	100
Top-5	100
Top-10	100
MRR	100

Table 30.
Android Matching results for 25 to 30 second perfect queries.

Measurement	**Result (%)**
Top-1	73.33
Top-5	82.22
Top-10	84.44
MRR	77.33

Table 31.
Android Matching results for 15 to 20 second imperfect queries.

Measurement	Result (%)
Top-1	80
Top-5	84.44
Top-10	89.89
MRR	83.17

Table 32.
Android Matching results for 25 to 30 second imperfect queries.

4.4.2.b. Description of results

The Top-1, Top-5, Top-10 and MRR results are recorded for 90 perfect queries, 90 imperfect queries (length 15-20 seconds) and 90 imperfect queries (length 25-30 seconds) in **Table 30**, **Table 31** and **Table 32** respectively.

4.5 USER ACCEPTANCE TEST (UAT) PLANS AND PROCEDURES

4.5.1 UAT test strategy overview

The next section provides an overview of the user acceptance testing strategy.

4.5.2 UAT

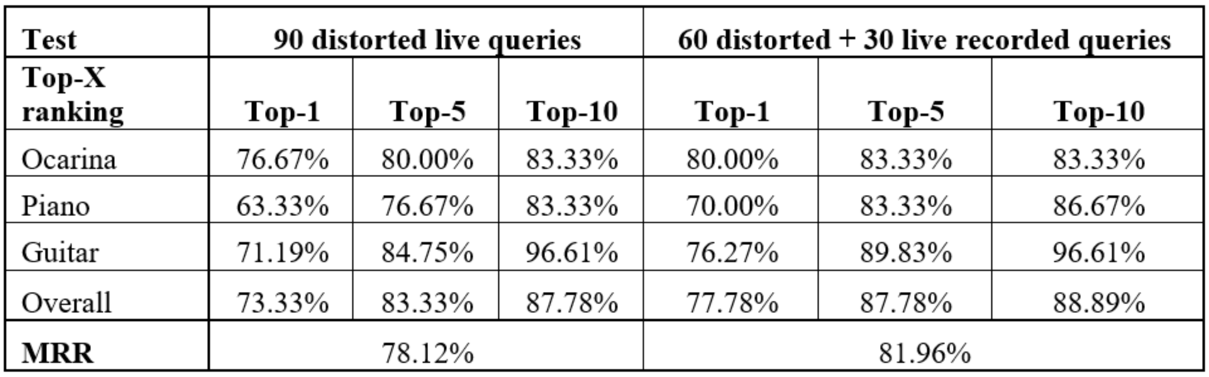
Test case number	UAT_001
Features to be tested	Audio recording, piano roll and matching
Testing approach	There are 90 live queries played with 30 unique queries associated with each of the three musical instruments namely piano, guitar and ocarina (woodwind). The average query length is 15 to 20 seconds. The first query list was formed with 90 imperfect queries. The imperfect queries were formed by altering every sixth note in the sequence using a circular application of four (4) possible distortions name: delete note, increase semitone, decrease semitone or insert note. The level of distortion was introduced to simulate the query imperfections that may be introduced by a person playing a live query. The second query list was formed with 30 live recordings and 60 imperfect queries formed using the same method statement described in the paragraph above. The 30 live recordings comprised of 10 live recordings per instrument and were played by the following persons: Guitar - Dr. Gerrit Roos, Ocarina - Marlette van Dalen, and Piano - Carina Heymans.

Table 33.
UAT part one.

Test case number	UAT_001
Pass/fail criteria	Pass if all of the conditions below are met: • The record /stop record button works. • The piano roll is displayed when the piano roll button is pressed after the audio recording. • The Top-5 hit list is displayed after the search button has been pressed. • Top-1 Hit Rate for monophonic music ≥ 55 %. • Top-5 Hit Rate for monophonic music ≥ 75 %. • Top-10 Hit Rate for monophonic music ≥ 85 %. • MRR (Mean Reciprocal Rate) ≥ 60 %.
Testing environment	Live testing in acoustically untreated environment.

Table 34.
UAT part two.

4.5.2.a. UAT test results



Test	**90 distorted live queries**			**60 distorted + 30 live recorded queries**		
Top-X ranking	**Top-1**	**Top-5**	**Top-10**	**Top-1**	**Top-5**	**Top-10**
Ocarina	76.67%	80.00%	83.33%	80.00%	83.33%	83.33%
Piano	63.33%	76.67%	83.33%	70.00%	83.33%	86.67%
Guitar	71.19%	84.75%	96.61%	76.27%	89.83%	96.61%
Overall	73.33%	83.33%	87.78%	77.78%	87.78%	88.89%
MRR	78.12%			81.96%		

Table 35.
Acceptance test

4.5.2.b. Description of results

The Top-1, Top-5, Top-10 and MRR results are recorded for the first set of 90 distorted unique live queries and the second set of 90 unique queries (comprising 60 distorted live queries and 30 pre-recorded live queries) respectively in **Table 35** above.

5. Discussion

This section contains a discussion and interpretation of the results in the previous section.

5.1 INTERPRETATION OF THE RESULTS

The success of any MIR system is dependent on successful design and implementation of a number of interconnected and interrelated subsystems. This was also evident in the results obtained for the overall QBP system and its associated subsystems. During the literature study and design, the design considerations associated with onset detection algorithms were taken into account. The design was informed by the inferences drawn from the literature study namely that the chosen onset detection algorithms should be closely matched to the type of musical instrument. The specifically chosen musical instruments informed the choice of onset detection algorithms. For PP instruments with hard onsets such as the piano it was found that temporal algorithms such as the Gaussian edge detector performed well, whilst frequency based methods such as the SF method was more suited to soft onsets associated with the ocarina (woodwind) instrument.

When the results for onset detection accuracy are compared in both the MATLAB simulated results in **Table 18** and the Android component results in **Table 25**, it was found that PP instruments outperformed the PNP instrument. The MATLAB simulated average F-measures for the woodwind instrument and piano was 90.01 % and 98.93 % respectively. For the Android results the average F-measures for the woodwind instrument and piano was 89.39 % and 98.90 % respectively, which amounts to a difference of 9.51 %. This is consistent with the inferences drawn from the literature study where it was discovered that hard onsets which exhibit a marked change in energy are generally easier to detect. The choice of the HPS pitch detection algorithm proved to be well suited for all three pitched type instruments (guitar, piano and ocarina) as evidenced by the average GPDE result of 1.48 % in **Table 27**.

It was found that the SW matching algorithm performed well when supported by a testing approach using separate query groups. It was decided to create two control groups of queries, the first group comprised of 90 perfect queries and the second group was made up of 90 imperfect queries. The 90 imperfect queries were further subdivided into 60 distorted queries and 30 live recorded queries. For the perfect queries the matching algorithm produced a result of 100 % for Top-1, Top-5, Top-10 and the MRR as shown in **Table 30**. The approach of base-lining perfect queries worked well since it confirmed that the SW algorithm performed as expected. The next logical step was to perform live testing with imperfect queries and the integrity of the SW algorithm had already been proven for the perfect query baseline.

Thereafter the query list was populated with 90 distorted queries. The distorted queries were formed by altering every sixth note in the sequence using a circular application of four (4) possible distortions namely: delete note, increase semitone, decrease semitone or insert note. The level of distortion of 16.67 % was introduced to simulate the query imperfections that may be introduced by the user playing a live query. This approach worked well as it was not practical to find players to perform all 90 queries. Instead 30 queries were performed live and pre-recorded by three musicians playing different instruments (10 for guitar, 10 for piano and 10 for ocarina), and the rest of the 90 queries was formed by distorting 60 perfect queries (20 per instrument) by 16.67 %.

The overall UAT test results are presented in **Table 35**. For the imperfect queries the matching algorithm produced results for Top-1 =77.78 %, Top-5 = 87.78 %, Top-10 = 88.89 % and the MRR = 81.96 % as shown in table 33. When results for the instruments are compared it was found that the ocarina achieved the lowest Top-10 result of 83.33 % and the piano achieved the lowest Top-1 result of 70 %. Due to the lower F-measure associated with the ocarina the onset feature representation for ocarina would be more negatively impacted when compared to the piano or the guitar yielding a lower Top-10 hit rate for the ocarina when compared with guitar and piano. In general, the piano queries contained less notes than the queries of the other instruments and therefore a 16.67 % distortion had a significant impact on the piano resulting in a lower Top-1 hit rate when compared to the other instruments.

The approach to segment the detected onsets and detected pitches into two distinct feature representations supported the one-dimensional vector of the SW algorithm. A SW algorithm was performed independently on each of the feature representations and their database counterparts. The use of two feature representations and the combination of their results enhanced the matching performance, contrary to the results that would have been achieved if only one parameter had been used.

The literature informed the construct of the feature representations. In respect of onsets the design choice was made to use the onset ratios as opposed to the onset interval, and in respect of the pitch representation it was decided to use the pitch interval. These choices supported the matching of imperfect queries such as when playing in a different key, because then the pitch interval would remain relatively unaffected. Likewise, the onset ratio feature representation supported a greater tolerance for queries with varying onset intervals.

Increasing the query length had a positive effect on the matching results. The system testing produced better matching results for an average query length of 25-30 seconds as shown in **Table 32** when compared to the matching results for an average query length of 15-20 seconds as shown in **Table 31**. The improved results can be explained in terms of the SW algorithm. The SW algorithm identifies the optimal local alignment between two sequences, by comparing the query sequence with the sequence stored in the database of ground truths on an array item by array item basis. Therefore, the SW algorithm performs better for longer query lengths as there are more array items in the query sequence available for matching.

It was found that placing the device microphone closer to the music source worked well as this improves the signal to noise ratio in a normal test environment with the presence of background noise.

In conclusion, all the mission-critical specifications were met. In hindsight the choice of mission-critical specification for Top-1 and F-measure could have been more optimistic, but the original (less optimistic) specification was based on results obtained in the literature for

QBH systems.

5.2 ASPECTS TO BE IMPROVED

Following the current design of the SF onset detection method, the F-measure for an ocarina (woodwind) does not perform as well as the piano and guitar. This could have been improved if multiple onset detection methods (fusion) were utilised for the ocarina. This would also have affected the Top-10 ranking score for ocarina.

The Gaussian edge detector execution speed can be increased by further optimising time-based convolution used in the algorithm, further reducing the waiting time of a user when prompting the piano roll.

With regards to the SW matching algorithm, matching speed can be improved if the algorithm was better optimised especially for longer query lengths.

5.3 STRONG POINTS

The reduction in the sampling rate from 44.1 kHz to 22.05 kHz served to reduce processing times of the onset and pitch algorithms, ultimately enhancing the overall user experience. The Gaussian edge detector performed very well due to the Gaussian filter used and the appropriately calculated alpha value. This produced exceptional results for the piano and guitar onsets. The HPS algorithm could successfully identify most pitches and the fact that a Kaiser window was used further improved the results.

The feature representations decided upon namely, the onset ratios and pitch intervals worked very well. The feature representations provided a sturdy foundation for matching purposes that could withstand imperfect queries with a 16.67 % distortion. The good performance of the matching process is also largely attributed to the selection of the SW matching algorithm.

5.4 FAILURE MODES OF THE DESIGN

The product has two failure modes that are associated with volatility of data. The first failure mode depends on the orientation of the mobile device's display as the user must consistently use a chosen orientation. Changing the orientation from landscape to portrait or vice versa causes a loss of data. The second failure mode occurs when the mobile device's screen locks and shuts down. In this case the user's data is also lost.

5.5 DESIGN ERGONOMICS

The ergonomic design is detailed in **Table 13**, **Section 3.9.3**. Colours selected have meaning and their brightness has been toned down. This allows for ease of use, and is softer on the eye. The size of the icons was also enlarged and they were adequately spaced. The size and spacing of the icons reduces the incidence an erroneous operation. Pictorial symbols were used for the icons. The pictorial nature of the icons promotes ease of understanding and use of the application.

5.6 HEALTH AND SAFETY ASPECTS OF THE DESIGN

Sounds above 85 dB are harmful the human ear. To ensure that the user does not undergo permanent hearing damage, instruments such as piano, guitar and ocarina should not be played above 85 dB. The user should rather play at a lower volume and position the phone's microphone closer to the instrument.

5.7 SOCIAL AND LEGAL IMPACT OF THE DESIGN

The product is designed for English users but can be extended to support other languages. The genre specified for the product can be extended for other genres so that other cultures can be included. This will also extend the user base of the product.

The MIDI database consists of MIDI files that were freely available on the Internet. For a commercial application the rights to the MIDI files would need to be legally secured.

5.8 ENVIRONMENTAL IMPACT AND BENEFITS OF THE DESIGN

The application is a software product that could be updated and ported to new versions and therefore the lifetime of the product can be extended.

6. Conclusion

6.1 SUMMARY OF THE WORK

A QBP software application that met the mission-critical requirements and the mission-critical specifications was designed and implemented.

6.2 SUMMARY OF THE OBSERVATIONS AND FINDINGS

Although the project was designed and implemented as a QBP application it was found that the application can be extended to also be used for QBH applications.

The SW algorithm that is widely used in DNA genomic sequence matching was found to be an effective matching algorithm for QBP applications. It was also discovered that that the interrelated nature of the subsystems within the overall system played an important role in the overall success of the QBP project. Furthermore, it was observed that any deterioration in any one of the feature representation such as onsets had a direct correlation with a deterioration in the Top-10 hit rate for matching. Confirmation was also received that the implementation of two SW algorithms, one for each of the feature representations i.e. onset ratio and pitch interval respectively, delivered superior results.

Then it was established that the testing strategy using two control query groups one for perfect queries and the other for imperfect queries proved to be effective in validating the functionality of the SW matching algorithm, thereby eliminating the effect of a poorly performing algorithm in the interpretation of the results associated with imperfect queries.

6.3 SUGGESTIONS FOR FUTURE WORK

Referring to the above findings one can suggest that the initial positive results obtained for limited QBH testing can inform further investigation into the use of the chosen SW matching algorithm for QBH.

It can also be suggested that further investigation and work can be done to improve the onset detection for the ocarina (woodwind) instrument that is characterised by soft onsets, pitch changes and moderate changes in spectral energy.

Further research can be done into extending the application of the QBP algorithm to more pitched instruments beyond the initial three chosen for this project.

7. References

[1] G. P. Nam, K. R. Park, S.-J. Park, S.-P. Lee, and M.-Y. Kim, “A new query-by-humming system based on the score level fusion of two classifiers,” *International Journal of Communication Systems*, vol. 25, no. 6, pp. 717–733, 2012.

[2] R. A. Putri and D. P. Lestari, “Music information retrieval using query-by-humming based on the dynamic time warping,” in *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*. IEEE, 2015, pp. 65–70.

[3] M. Li, Z. Zhao, and P. Shi, “Query by humming based on music phrase segmentation and matching,” in *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*. IEEE, 2015, pp. 1966–1970.

[4] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, “A tutorial on onset detection in music signals,” *IEEE Transactions on speech and audio processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[5] R. Zhou and J. D. Reiss, “Music onset detection,” *Machine Audition: Principles, Algorithms and Systems: Principles, Algorithms and Systems*, p. 297, 2010.

[6] A. Holzapfel, Y. Stylianou, A. C. Gedik, and B. Bozkurt, “Three dimensions of pitched instrument onset detection,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1517–1527, 2010.

[7] M. Fabiani and A. Friberg, “Expressive modifications of musical audio recordings: Preliminary results.” in *ICMC*, 2007.

[8] A. De Cheveigné and H. Kawahara, “Yin, a fundamental frequency estimator for speech and music,” *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[9] P. De La Cuadra, A. S. Master, and C. Sapp, “Efficient pitch detection techniques for interactive music.” in *ICMC*, 2001.

[10] M. Vasilik, L. Stillings, and C. Cortazar, “Pitch detection for music in noisy environments.”

[11] M. Rocamora, P. Cancela, and A. Pardo, “Query by humming: Automatically building the database from music recordings,” *Pattern Recognition Letters*, vol. 36, pp. 272–280, 2014.

[12] L. Lu, H. You, H. Zhang *et al.*, “A new approach to query by humming in music retrieval.” in *ICME*, 2001, pp. 22–25.

[13] Y. Qi, J. W. Paisley, and L. Carin, “Music analysis using hidden markov mixture models,” *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5209–5224, 2007.

[14] Z. Ren, C. Fan, and Y. Ming, “Music retrieval based on rhythm content and dynamic time warping method,” in *Signal Processing (ICSP), 2016 IEEE 13th International Conference on*. IEEE, 2016, pp. 989–992.

[15] T. Giannakopoulos and A. Pikrakis, *Introduction to Audio Analysis: A MATLAB® Approach*. Academic Press, 2014.

[16] G. Wieds, "Bioinformatics explained: Smith-waterman," *Bioinformatics Explained*, 2007.

[17] S. Ravuri *et al.*, "Automatic cover song detection: Moving from high scores to general classification," *Proceedings of Music Information Retrieval Evaluation eXchange, MIREX*, 2009.

[18] A. Pikrakis, "Audio thumbnailing in video sharing sites," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012, pp. 1284–1288.

[19] T. M. Man. (2015) 130,000 midi file collection 3.65gb uncompressed / 1.02 gb compressed zip. [Online; accessed 19-July-2017]. [Online]. Available: https://www.reddit.com/r/WeAreTheMusicMakers/comments/3ajwe4/the_largest_midi_collection_on_the_internet/

[20] K. R. Fowler and K. Fowler, *Electronic instrument design: architecting for the life cycle*. Oxford University Press on Demand, 1996.

8. Appendixes

8.1 APPENDIX 1: USE CASES

8.1.1 Use case diagram

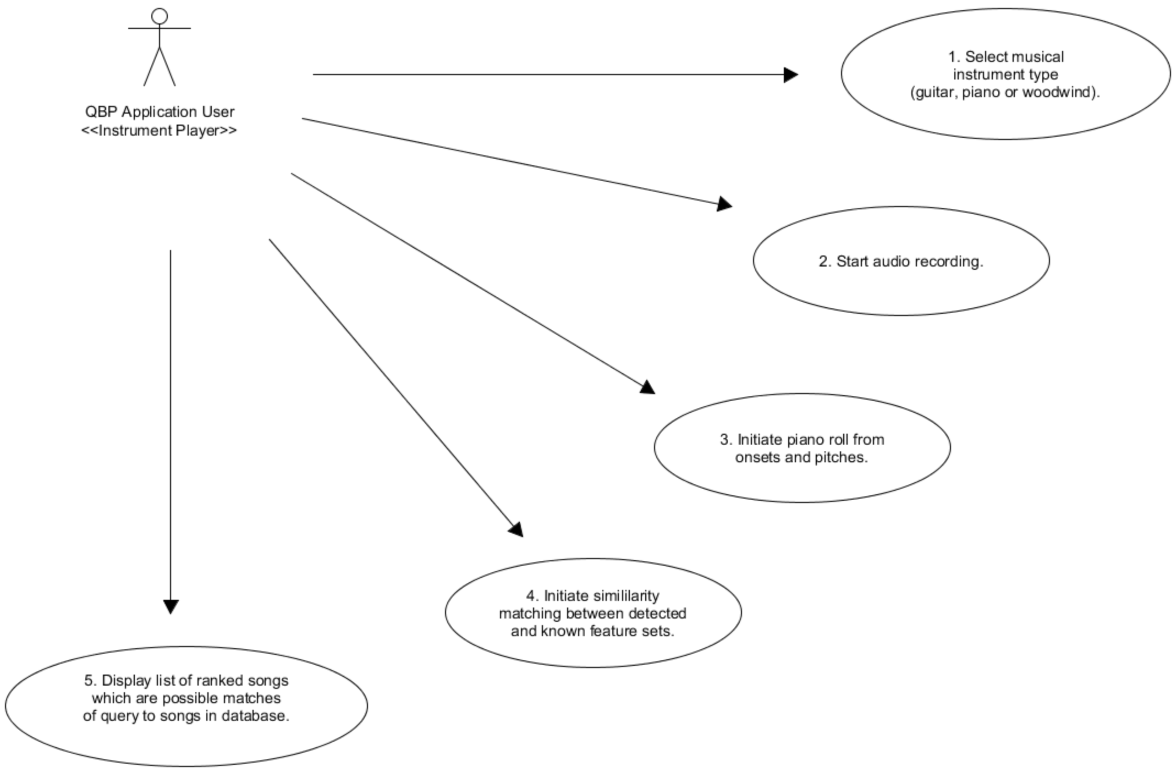
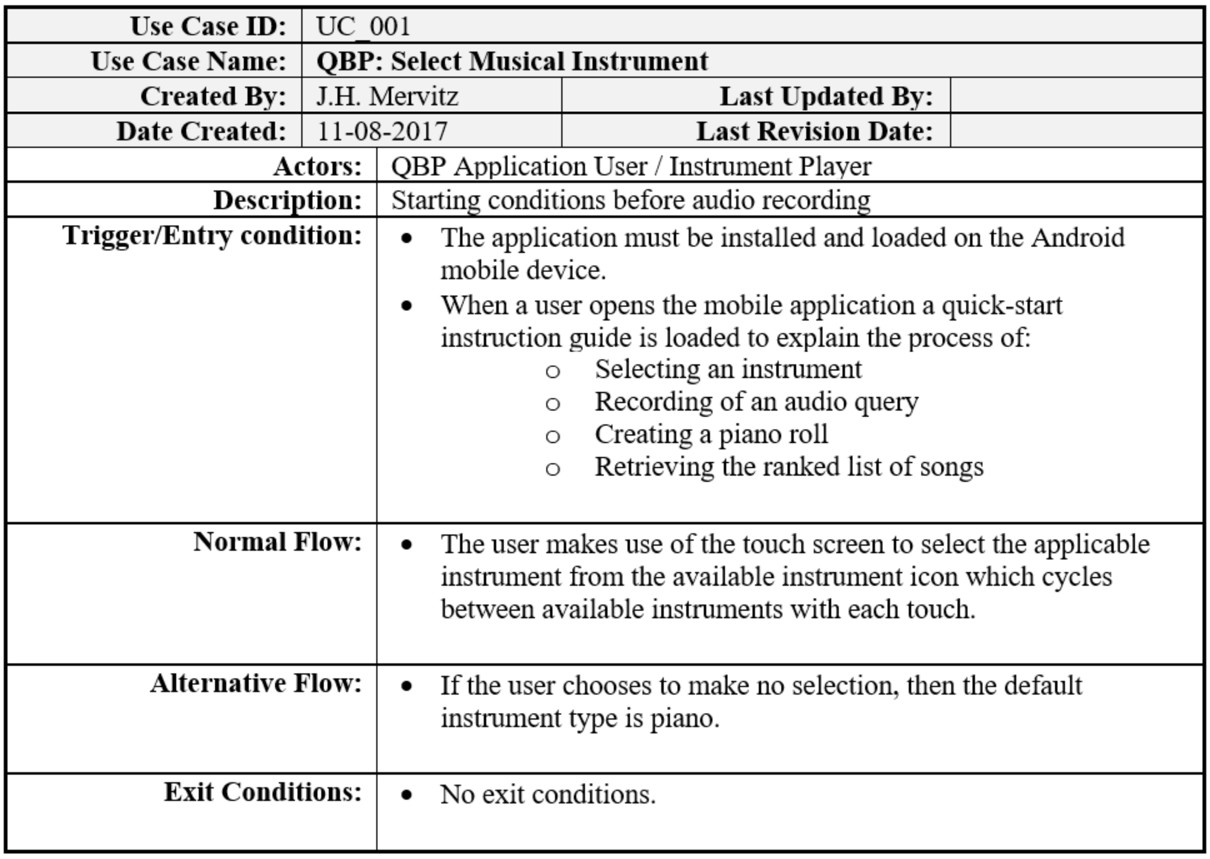

QBP Application User
<<Instrument Player>>
1. Select musical instrument type (guitar, piano or woodwind).
2. Start audio recording.
3. Initiate piano roll from onsets and pitches.
4. Initiate simililarity matching between detected and known feature sets.
5. Display list of ranked songs which are possible matches of query to songs in database.

Figure 28.
Use case diagram.

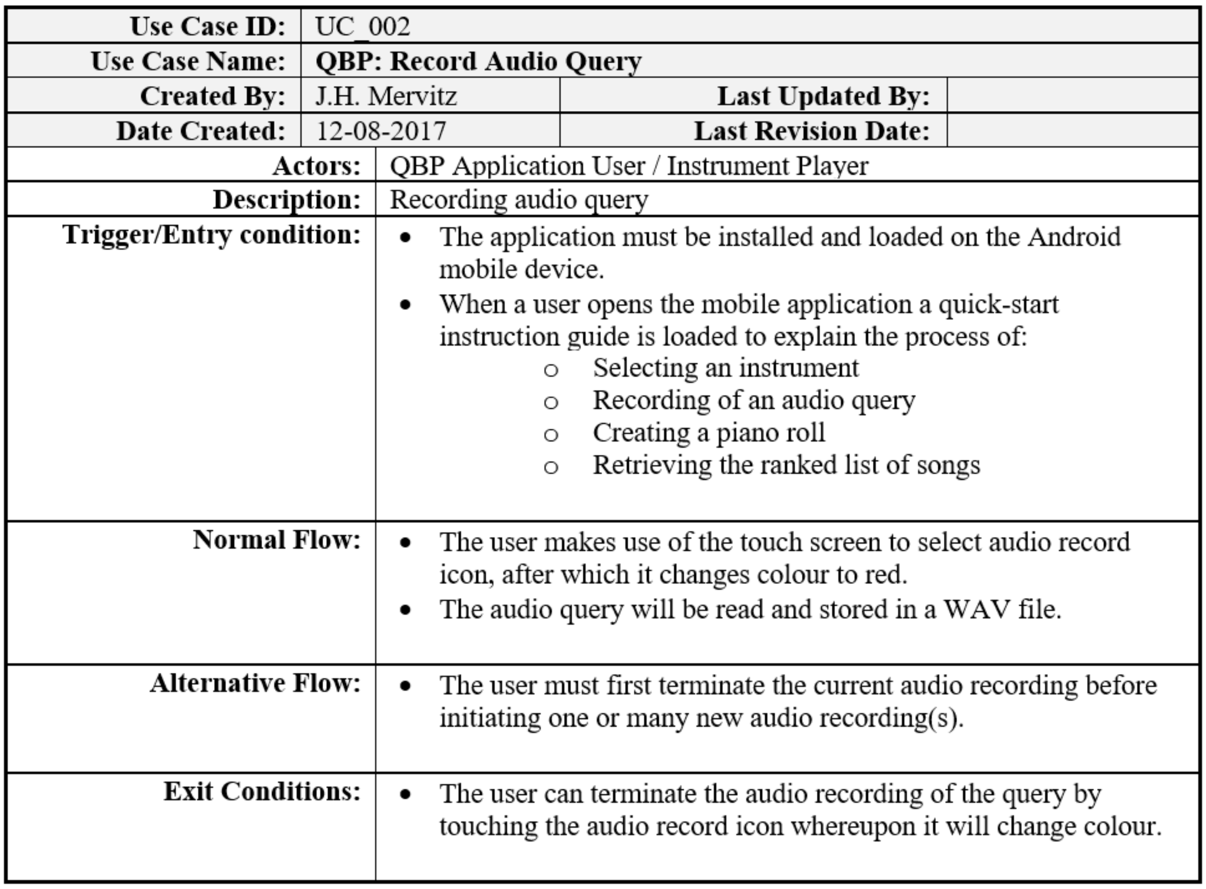
8.1.2 Use case one



Use Case ID:	UC_001		
Use Case Name:	**QBP: Select Musical Instrument**		
Created By:	J.H. Mervitz	**Last Updated By:**	
Date Created:	11-08-2017	**Last Revision Date:**	
Actors:	QBP Application User / Instrument Player		
Description:	Starting conditions before audio recording		
Trigger/Entry condition:	• The application must be installed and loaded on the Android mobile device. • When a user opens the mobile application a quick-start instruction guide is loaded to explain the process of: o Selecting an instrument o Recording of an audio query o Creating a piano roll o Retrieving the ranked list of songs		
Normal Flow:	• The user makes use of the touch screen to select the applicable instrument from the available instrument icon which cycles between available instruments with each touch.		
Alternative Flow:	• If the user chooses to make no selection, then the default instrument type is piano.		
Exit Conditions:	• No exit conditions.		

Table 36.
Use case one.

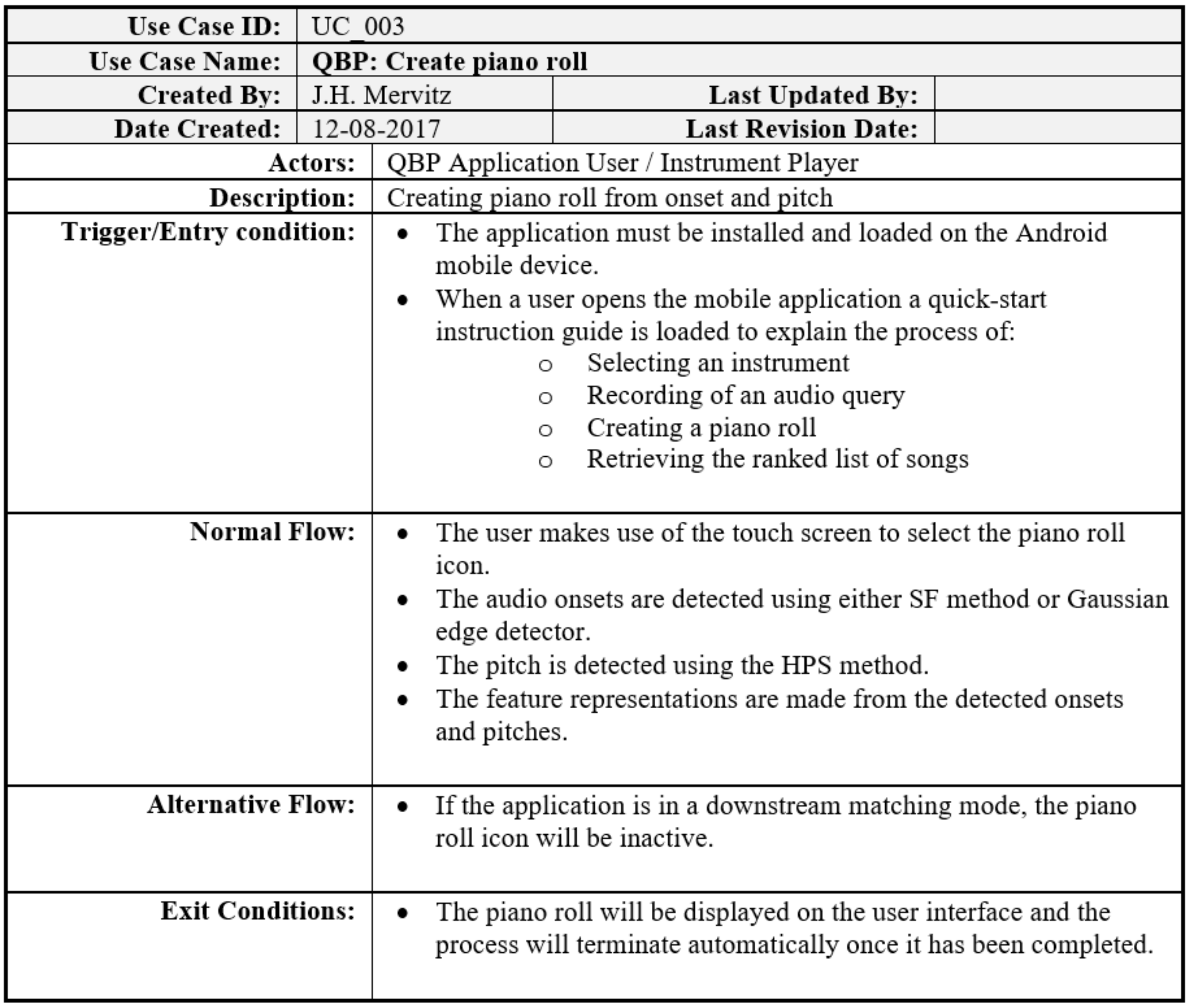
8.1.3 Use case two



Use Case ID:	UC_002		
Use Case Name:	**QBP: Record Audio Query**		
Created By:	J.H. Mervitz	**Last Updated By:**	
Date Created:	12-08-2017	**Last Revision Date:**	
Actors:	QBP Application User / Instrument Player		
Description:	Recording audio query		
Trigger/Entry condition:	• The application must be installed and loaded on the Android mobile device. • When a user opens the mobile application a quick-start instruction guide is loaded to explain the process of: o Selecting an instrument o Recording of an audio query o Creating a piano roll o Retrieving the ranked list of songs		
Normal Flow:	• The user makes use of the touch screen to select audio record icon, after which it changes colour to red. • The audio query will be read and stored in a WAV file.		
Alternative Flow:	• The user must first terminate the current audio recording before initiating one or many new audio recording(s).		
Exit Conditions:	• The user can terminate the audio recording of the query by touching the audio record icon whereupon it will change colour.		

Table 37.
Use case two.

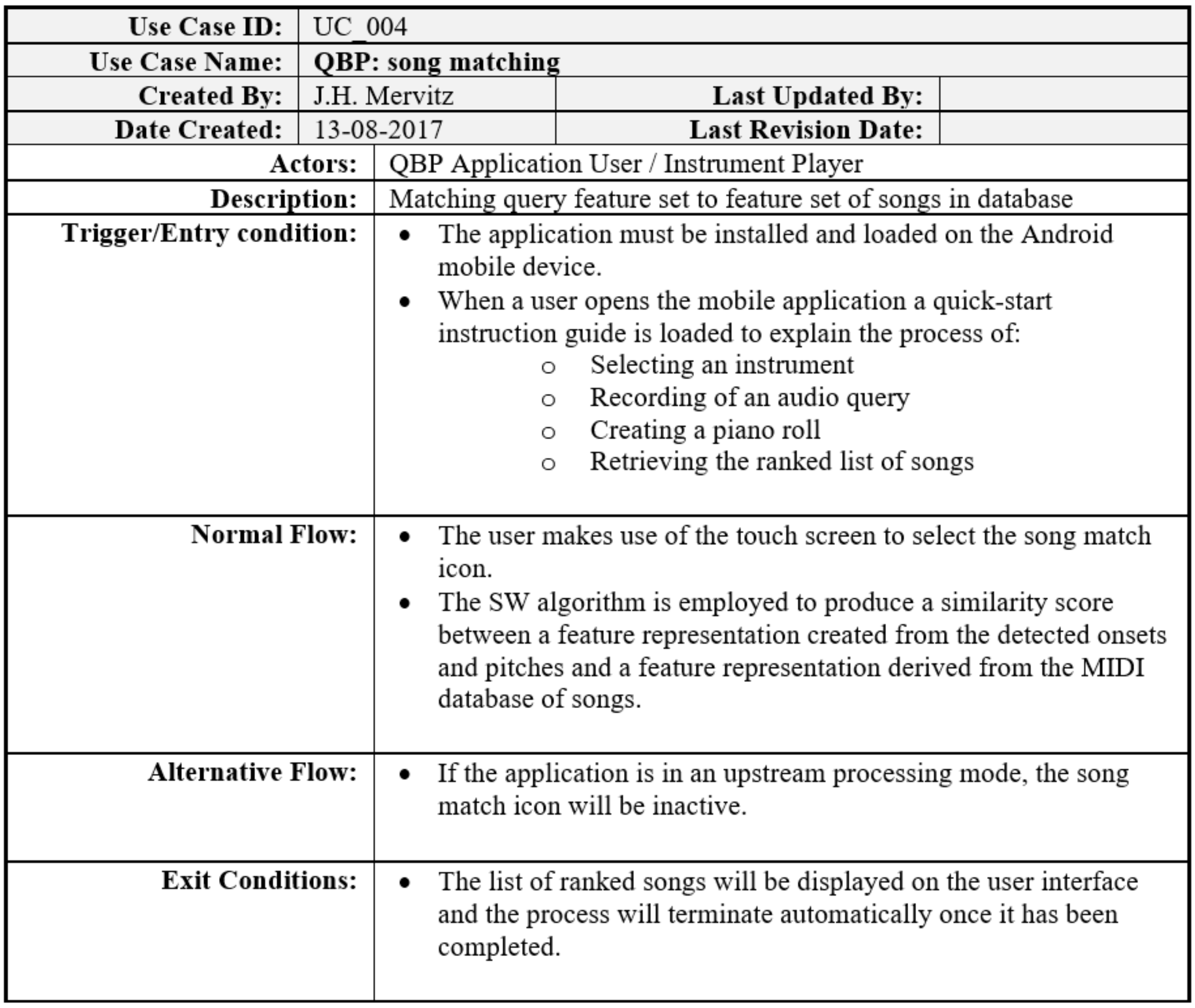
8.1.4 Use case three



Use Case ID:	UC_003		
Use Case Name:	**QBP: Create piano roll**		
Created By:	J.H. Mervitz	**Last Updated By:**	
Date Created:	12-08-2017	**Last Revision Date:**	
Actors:	QBP Application User / Instrument Player		
Description:	Creating piano roll from onset and pitch		
Trigger/Entry condition:	• The application must be installed and loaded on the Android mobile device. • When a user opens the mobile application a quick-start instruction guide is loaded to explain the process of: o Selecting an instrument o Recording of an audio query o Creating a piano roll o Retrieving the ranked list of songs		
Normal Flow:	• The user makes use of the touch screen to select the piano roll icon. • The audio onsets are detected using either SF method or Gaussian edge detector. • The pitch is detected using the HPS method. • The feature representations are made from the detected onsets and pitches.		
Alternative Flow:	• If the application is in a downstream matching mode, the piano roll icon will be inactive.		
Exit Conditions:	• The piano roll will be displayed on the user interface and the process will terminate automatically once it has been completed.		

Table 38.
Use case three.

8.1.5 *Use case four*



Use Case ID:	UC_004		
Use Case Name:	**QBP: song matching**		
Created By:	J.H. Mervitz	**Last Updated By:**	
Date Created:	13-08-2017	**Last Revision Date:**	
Actors:	QBP Application User / Instrument Player		
Description:	Matching query feature set to feature set of songs in database		
Trigger/Entry condition:	• The application must be installed and loaded on the Android mobile device. • When a user opens the mobile application a quick-start instruction guide is loaded to explain the process of: o Selecting an instrument o Recording of an audio query o Creating a piano roll o Retrieving the ranked list of songs		
Normal Flow:	• The user makes use of the touch screen to select the song match icon. • The SW algorithm is employed to produce a similarity score between a feature representation created from the detected onsets and pitches and a feature representation derived from the MIDI database of songs.		
Alternative Flow:	• If the application is in an upstream processing mode, the song match icon will be inactive.		
Exit Conditions:	• The list of ranked songs will be displayed on the user interface and the process will terminate automatically once it has been completed.		

Table 39.
Use case four.

Part 5. Technical Documentation

This main report is supplemented with technical documentation. This provides more detail on the software that was used in the experiments, including program listings, a user guide and circuit designs. This section appears on the CD/DVD that accompanies this report.

The CD/DVD is organised into the directories listed below: (give the directories as they appear on the CD).

Author
Datasets/30 live recorded queries
Datasets/90 imperfect 15 to 20 second queries
Datasets/90 imperfect 15 to 20 second queries
Datasets/90 perfect queries
Datasets/Database
Project Photos
References
Software
Technical Documentation