TUGAS AKHIR - KI141502

RANCANG BANGUN APLIKASI MUSICMOO DENGAN METODE MIR (MUSIC INFORMATION RETRIEVAL) PADA MODUL FINGERPRINT, COVER SONG RECOGNITION, DAN SONG RECOMMENDATION

M Faris Ponighzwa Rizkanda
NRP 5113100034

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Dosen Pembimbing II
Dwi Sunaryono, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Januari 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

RANCANG BANGUN APLIKASI MUSICMOO DENGAN METODE MIR (MUSIC INFORMATION RETRIEVAL) PADA MODUL FINGERPRINT, COVER SONG RECOGNITION, DAN SONG RECOMMENDATION

Mochammad Faris Ponighzwa Rizkanda
NRP 5113100034

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Dosen Pembimbing II
Dwi Sunaryono, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Januari 2017

[Halaman ini sengaja dikosongkan]

FINAL PROJECT– KI141502

DESIGN AND IMPLEMENTATION OF MUSICMOO APPLICATION USING MIR (MUSIC INFORMATION RETRIEVAL) OF FINGERPRINT, COVER SONG RECOGNITION, AND SONG RECOMMENDATION

Mochammad Faris Ponighzwa Rizkanda
NRP 5113100034

Supervisor I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

Supervisor II
Dwi Sunaryono, S.Kom., M.Kom.

Department of Informatics
Faculty of Information and Technology
Institut Teknologi Sepuluh Nopember
January 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Rancang Bangun Aplikasi MusicMoo dengan Metode MIR (Music Information Retrieval) pada Modul Fingerprint, Cover Song Recognition, dan Song Recommendation

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
Mochammad Faris Ponighzwa Rizkanda
NRP : 5113 100 034

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D. ..... (pembimbing 1)
NIP: 19590803 198601 1001

Dwi Sunaryono, S.Kom., M.Kom. ..... (pembimbing 2)
NIP: 19720528 199702 1001

SURABAYA
JANUARI 2017

[Halaman ini sengaja dikosongkan]

Rancang Bangun Aplikasi MusicMoo dengan Metode MIR (Music Information Retrieval) pada Modul Fingerprint, Cover Song Recognition, dan Song Recommendation

Nama Mahasiswa : Mochammad Faris Ponighzwa R.
NRP : 5113 100 034
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing 1 : Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.
Dosen Pembimbing 2 : Dwi Sunaryono, S.Kom., M.Kom.

ABSTRAK

Industri musik sudah mulai merambah ke bidang komputer. Salah satunya adalah aplikasi SoundHound, Shazam, dan masih banyak lagi. Namun semua aplikasi tersebut hanya melakukan deteksi dari potongan suara yang direkam. Semua aplikasi tersebut bekerja dengan cara melakukan ekstrak fingerprint hanya dari beberapa segmen sinyal audio yang direkam.

Pertama, penulis membangun database yang berisi fitur lagu. Di dalam fitur tersebut terdapat kumpulan nilai yang mengidentifikasikan suatu lagu. Dari deskripsi ini digunakan untuk melakukan pencarian pada suatu lagu. Kedua, melakukan proses pada fitur audio yang terkait. Ketiga, melakukan klasifikasi dan hasilnya adalah detail informasi fingerprint, cover song, dan rekomendasi pada suatu lagu.

***Kata kunci: Analisa Audio, MIR,* MPEG-7**

[Halaman ini sengaja dikosongkan]

Design and Implementation of MusicMoo Application using MIR (Music Information Retrieval) Method of Fingerprint Modul, Cover Song Recognition, and Song Recommendation

Student Name : Mochammad Faris Ponighzwa Rizkanda
Student ID : 5113 100 034
Major : Informatics Department FTIf-ITS
Advisor 1 : Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.
Advisor 2 : Dwi Sunaryono, S.Kom., M.Kom.

.

ABSTRACT

Music industry has started using computer engineering factor to produce the best quality of an audio. The example of application using music as their main theme was SoundHound, Shazam, and etc. From application example above, they just try to identify the original song from its pieces of recorded song. All application example above works by extracting fingerprint on a few segments of a recorded audio signal.

First, writes construct a database contain of some feature of an audio. An audio feature contain of set value that used to describe an audio. This set of value was used to search a song. Second, to process related audio feature. Third, to classify and the result was information of Fingerprint, Cover Song, and Recommendation of an audio.

***Keyword: Audio Analysis, MIR,* MPEG-7**

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

Rancang Bangun Aplikasi MusicMoo dengan Metode MIR (Music Information Retrieval) pada Modul Fingerprint, Cover Song Recognition, dan Song Recommendation

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Bapak, Ibu, kakak dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D. selaku dosen pembimbing I.
3. Bapak Dwi Sunaryono, S.Kom., M.Kom. selaku dosen pembimbing II.
4. Bapak Dedy Rahman Wijaya, selaku mahasiswa S-3 sekaligus anak bimbing Prof. Riyanarto yang telah banyak membimbing dan memberikan arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.
5. Umy Chasanah Noor Rizqi, selaku rekan dan pedamping hidup yang selalu mendukung dan menyemangati penulis.
6. Johanes Andre, selaku tim yang membantu dalam pengerjaan Tugas Akhir ini.
7. Grup "ululululululululu", yang selalu memberikan penyemangat sendiri bagi penulis.
8. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS lainnya yang telah banyak menyampaikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
9. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu persatu.

Penulis menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan tugas akhir maupun penyusunan buku laporan ini, namun penulis berharap buku tugas akhir ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan penulisan buku tugas akhir ini

Surabaya, January 2017

Mochammad Faris Ponighzwa R.

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

DAFTAR PERSAMAAN

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Akhir-akhir ini, industri musik berkembang sangat pesat. Tidak terkecuali, industri musik di Indonesia juga mulai terkena dampaknya. Aplikasi seperti Shazam dan SoundHound dapat mendeteksi potongan suara yang direkam dan menebak judul lagu dari potongan suara tersebut. Sehingga, masyarakat tidak perlu merasa kesusahan ketika menginginkan judul lagu dari potongan suara yang terdengar di tempat umum.

Namun aplikasi tersebut hanya sebatas mengidentifikasi dan melakukan pencocokan dari suatu audio. Padahal, telah ditemukan suatu istilah MIR dengan kepanjangan *Music Information Retrieval*, yang berarti pencarian keterangan informasi dari suatu musik. MIR merupakan istilah yang digunakan ketika ingin mendapatkan informasi dari suatu musik. Informasi yang didapat dapat berupa judul lagu, genre, tempo, dan lainnya.

Pada penelitian ini, penulis akan melakukan implementasi MIR. Penulis akan melakukan penelitian metode dan langkah-langkah yang tepat, agar istilah MIR dapat diimplementasikan. Fokus dalam penelitian kali ini adalah, pada modul *fingerprint*, *cover song*, dan *song recommendation*. *Fingerprint* yang berarti adalah sidik jari, dengan maksud audio akan diidentifikasi dengan hanya potongan audio yang dicari. *Cover song* berarti penyanyian kembali dari lagu asli (istilah bahasa inggris disebut dengan *cover*), sehingga potongan suara yang

direkam akan dicari lagu asli dari potongan suara tersebut. Dan yang terakhir adalah *song recommendation*, yang berarti sistem akan merekomendasikan lagu dari lagu yang sudah dicari saat ini.

Standar yang digunakan penulis adalah MPEG-7 yang sudah menjadi standar dalam konten multimedia berdasarkan ISO/IEC 15938 yang berekstensi XML. XML ini akan diperoleh dari *library MPEG7AudioEnc* yang bersifat *open source*. XML tersebut akan dilakukan *query* untuk diambil fitur-fitunya dalam bentuk matriks. Fitur-fitur inilah yang digunakan untuk melakukan pemanggilan informasi musik/MIR. Kumpulan fitur yang ada dalam *file* XML ini akan disimpan dalam *database* untuk pemanggilan informasi *fingerprint, cover song,* dan *song recommendation* yang dimaksud.

1.2. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Mengetahui fitur lagu yang efektif untuk *fingerprint*, berbasis MPEG-7.
2. Mengetahui fitur lagu yang mempengaruhi dalam *cover song recognition*.
3. Mengetahui fitur lagu yang mempengaruhi *song recommendation*.

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini antara lain:

1. Bagaimana cara mengidentifikasi suatu musik dari *fingerprint*?
2. Bagaimana cara mengidentifikasi *cover song* dari musik?
3. Bagaimana cara mengetahui kemiripan dari suatu musik, sehingga dapat dilakukan rekomendasi?

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, antara lain:

1. Hasil dari tugas akhir ini adalah sebuah aplikasi "MusicMoo" dengan modul *fingerprint*, *cover song recognition*, dan *song recommendation.*
2. Audio yang digunakan berekstensi .wav.
3. Identifikasi lagu dengan *fingerprint* hanya sebatas menampilkan judul.
4. *Cover song recognition* didapat dengan menggunakan metode *supervised learning*, yang artinya membutuhkan data *training*.
5. *Song recommendation* menghasilkan daftar dari suara yang direkam. Daftar ini didapatkan, dari pencocokan modul *fingerprint* sebelumnya.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

a. Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi latar belakang pembuatan tugas akhir, rumusan masalah, batasan masalah, tujuan pembuatan, manfaat, metodologi hingga jadwal kegiatan pembuatan tugas akhir. Selain itu proposal tugas akhir ini memberikan ringkasan dari tugas akhir. Proposal tugas akhir juga berisi tinjauan pustaka yang digunakan sebagai referensi pembuatan tugas akhir ini.

b. Studi literatur

Pada bab ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan program yaitu mengenai MPEG-7, MPEG-7 *Low Descriptors*, Xquery, dan *Wavelet*.

c. Analisis dan desain perangkat lunak

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

d. Implementasi perangkat lunak

Implementasi perangkat lunak ini dibangun dengan sistem perangkat bergerak yang berbasis bahasa pemrograman Java (Android Studio) dan *database* menggunakan MySQL. Sedangkan untuk *pre-processing* dan *processing* sinyal menggunakan bahasa Python.

e. Pengujian dan evaluasi

Pada tahapan ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya aplikasi, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

f. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah

   c. Batasan Tugas Akhir
   d. Tujuan
   e. Metodologi
   f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.6. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka aplikasi.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi fitur-fitur penunjang aplikasi.

Bab V Pengujian dan Evaluasi

Membahas tentang lingkungan pengujian, skenario pengujian, dan evaluasi pengujian setelah aplikasi selesai dikembangkan.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir.

2.1. Penelitian Terkait

Pada penelitian sebelumnya yang terkait dengan tugas akhir ini, identifikasi musik dengan memanfaatkan *fingerprint* berdasarkan standar MPEG-7 sudah dilakukan dengan menggunakan *Sliding Algorithm* [1]. Penelitian tersebut yang dijadikan acuan dalam implementasi pada modul *fingerprint*. Modul *fingerprint* inilah yang dijadikan dasar dalam implementasi modul *cover song recognition*. Penelitian tentang modul *cover song recognition* sudah pernah dilakukan sebelumnya. Pada penelitian tersebut identifikasi dilakukan dengan cara membandingkan *raw signal* dan dicari *chroma* yang terkandung dalam sinyal tersebut [2]. Pada penelitian yang digunakan pada tugas akhir ini melakukan deteksi *cover song* namun dengan menggunakan fitur yang berbasis dengan standar ISO MPEG-7. Penelitian ini menggunakan acuan MPEG-7 dikarenakan hasil ekstraksi fitur dari MPEG-7 berupa metadata. Metadata ini akan digunakan sebagai bahan dasar untuk melakukan klasifikasi suara, sehingga dalam penelitian ini tidak menggunakan klasifikasi yang berbasis konten. Pada penelitian *cover song* yang telah dijelaskan sebelumnya, klasifikasi dan *query* masih berbasis konten dengan cara menerima *input* berupa *raw-signal*. Pada penelitian ini digunakan ekstraksi fitur dari *file* berekstensi .wav dan dilakukan klasifikasi. Sehingga dapat dikatakan bahwa penelitian ini menggunakan metadata dalam pengolahannya, bukan melakukan klasifikasi berbasis konten. Contoh aplikasi dari deteksi suatu musik adalah SoundHound dan Shazam, namun kedua aplikasi musik tersebut melakukan deteksi lagu harus berupa lagu asli dari penyanyi. Sehingga ketika

aplikasi merekam lagu berupa lagu yang dinyanyikan kembali, maka aplikasi tersebut tidak dapat melakukan deteksi.

2.2. MPEG-7

MPEG-7 adalah deskripsi standar konten multimedia dalam ISO/IEC 15938 [3]. Konten multimedia yang dimaksud adalah gambar, musik (suara), dan video. Namun pada penelitian kali ini fokus hanya pada kontem multimedia berupa musik. Dalam melakukan ekstraksi fitur, MPEG-7 menghasilkan beberapa fitur yang disebut dengan *Low Level Descriptors*. Hasil ekstraksi fitur musik (suara) berdasarkan MPEG-7 merupakan sebuah metadata yang disimpan dalam bentuk matriks berukuran $n \times m$. Nilai *m* menyatakan sebagai *subband* metadata dari musik. *Subband* metadata bergantung pada fitur yang akan digunakan. Sebagai contoh apabila menggunakan fitur *Audio Signature Type* maka jumlah *m* adalah 16. Nilai *n* bergantung pada durasi dan ukuran dari suatu sumber suara. Sehingga semakin lama suatu sumber suara, maka akan semakin besar pula nilai *n* yang didapatkan berdasarkan ekstraksi fitur MPEG-7.

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & \cdots & A_{1,M-1} & A_{1,M} \\ A_{2,1} & A_{2,2} & A_{2,3} & \cdots & A_{2,M-1} & A_{2,M} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{N-1,1} & A_{N-1,2} & A_{N-1,3} & \cdots & A_{N-1,M-1} & A_{N-1,M} \\ A_{N,1} & A_{N,2} & A_{N,3} & \cdots & A_{N,M-1} & A_{N,M} \end{bmatrix}$$

Gambar 2.1 Matriks fitur MPEG-7

Gambar 2.1 merupakan format matriks yang dihasilkan oleh MPEG-7. Matriks inilah yang dianggap sebagai suatu fitur yang diproses oleh sistem. Kumpulan dari semua fitur akan disimpan dalam suatu dokumen yang berformat XML. Detail dari masing-masing fitur dijelaskan pada subbab selanjutnya.

Dalam penelitian ini proses ekstraksi fitur *file* .wav ditangani oleh suatu *library MPEG7AudioEncApp* yang

dirancang oleh Holger Crysandt [4]. Cara kerja dari *library* ini menerima *input* berupa nilai *string* yang menyimpan informasi berupa alamat *directory* dari *file* musik yang berformat .wav. *Output* dari *Library* ini adalah dokumen XML yang meyimpan detail dari fitur MPEG-7 dalam skema yang telah ditentukan. Detail dari implementasi *library* ini dibahas pada Bab IV subbab Implementasi Ekstraksi Fitur.

2.3. MPEG-7 Low Level Descriptors

Suatu lapisan paling dasar yang didapat setelah melakukan ekstraksi audio .wav ke bentuk XML [5]. *Low Level Descriptor* memiliki beberapa fitur seperti *Audio Spectrum Projection, Audio Spectrum Flatness, Audio Spectrum Envelope,* dan sebagainya. Setiap fitur menggambarkan properti dari sebuah sumber suara. Sebagai contoh *Audio Spectrum Flatness* menggambarkan properti kerataan sinyal dari sebuah sumber suara. Berdasarkan dari beberapa fitur tersebut, eksperimen ini menggunakan 3 fitur dalam pengolahan metadata. Fitur tersebut adalah *Audio Spectrum Flatness, Audio Spectrum Projection,* dan *Audio Signature Type*. *Audio Spectrum Flatness* dan *Audio Spectrum Projection* akan digunakan untuk modul *cover song recognition*. *Audio Signature Type* akan digunakan pada modul *fingerprint*.

```
1. <?xml version="1.0" encoding="UTF-
   8" standalone="no"?>
2. <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001" xmlns:mpe
   g7="urn:mpeg:mpeg7:schema:2001" xmlns:xsi="http://w
   ww.w3.org/2001/XMLSchema-instance">
3. <Description xsi:type="ContentEntityType">
4. <MultimediaContent xsi:type="AudioType">
5. <Audio xsi:type="AudioSegmentType">
6. <MediaInformation xsi:type="MediaInformationType">

7. <MediaProfile xsi:type="MediaProfileType">
8. <MediaFormat xsi:type="MediaFormatType">
9. <Content href="MPEG7ContentCS" xsi:type="Controlled
   TermUseType">
```

```
10. <Name>audio</Name>
11. </Content>
12. <FileSize>35642924</FileSize>
13. <AudioCoding>
14. <AudioChannels>2</AudioChannels>
15. <Sample bitsPer="16" rate="44100.0"/>
16. </AudioCoding>
17. </MediaFormat>
18. <MediaInstance xsi:type="MediaInstanceType">
19. <InstanceIdentifier xsi:type="UniqueIDType">UniqueI
    D0000000002</InstanceIdentifier>
20. <MediaLocator xsi:type="MediaLocatorType">
21. <MediaUri>file:/D:/TA/MPEG-
    7/Lagu/Isyana%20Sarasvati%20-
    %20Keep%20Being%20You.wav</MediaUri>
22. </MediaLocator>
23. </MediaInstance>
24. </MediaProfile>
25. </MediaInformation>
26. <CreationInformation xsi:type="CreationInformationT
    ype">
27. <Creation xsi:type="CreationType"/>
28. </CreationInformation>
29. <AudioDescriptor hiLimit="12000.0" loLimit="50.0" x
    si:type="AudioFundamentalFrequencyType">
30. <SeriesOfScalar hopSize="PT10N1000F" totalNumOfSamp
    les="40404">
31. <Raw>1297.0588 1297.0588 352.8 352.8 54.51174 54.51
    174 1297.0588 1297.0588
32. 3150.0 3150.0 3150.0 3150.0 832.0755 832.0755 2756.
    25 2756.25
33. ..................................................
    ..........
34. 1191.8918 689.0625 689.0625 1297.0588 1297.0588 315
    0.0 3150.0 1075.6097</Raw>
35. </SeriesOfScalar>
36. </AudioDescriptor>
37. <AudioDescriptor xsi:type="AudioHarmonicityType">
38. <HarmonicRatio>
```

Kode Sumber 2.1 Format Skema dalam XML

Kode Sumber 2.1 merupakan hasil *file* yang berformat XML hasil dari ekstraksi fitur berbasis MPEG-7. Namun format pada Kode Sumber 2.1 merupakan potongan dari *file* XML, sebab apabila ditampilan seutuhnya akan memakan tempat sebab terlalu panjang. Masing-masing fitur dalam *file* XML akan dijelaskan secara detail pada masing-masing subbab berikut [6]:

2.3.1. *Audio Power*

Audio Power (*AP*) dalam MPEG-7 merepresentasikan kekuatan temporal sinyal audio sesaat. Koefisien pada *Audio Power* merupakan kuadrat rata-rata dalam suatu sinyal yang tidak tumpang tindih.

$$AP(l) = \frac{1}{N_{hop}} \sum_{n=0}^{N_{hop}-1} \left| s(n + lN_{hop}) \right|^2 \ (0 \ \leq l \leq L-1) \tag{2.1}$$

dimana:

L = total jumlah frame waktu
s(n) = rata-rata *square waveform*
l = indeks frame
N_{hop} = sampel antara *successive non-overlapping*

2.3.2. *Temporal Centroid*

Temporal Centroid (*TC*) didefinisikan sebagai waktu rata-rata energi sinyal *envelope*.

$$TC = \frac{N_{hop}}{F_s} \ \frac{\sum_{l=0}^{L-1}(lEnv(l))}{\sum_{l=0}^{L-1} Env(l)} \tag{2.2}$$

$$Env(l) = \sqrt{\frac{1}{N_w}\sum_{n=0}^{N_w-1} s^2\left(lN_{hop}+n\right)\,(0 \le l \le L-1)} \tag{2.3}$$

dimana:

N_{hop} = Sampel antara *successive non-overlapping*
Env(l) = Sinyal envelope
N_w = Panjang dari *frame* dalam beberapa waktu
n = Indeks waktu
F_s = Frekuensi *sampling*
s = Spektrum yang diekstak dari *frame* ke-*l*
L = Jumlah total dari *frame*

2.3.3. *Log Attack Time*

Log Attack Time (*LAT*) didefinisikan sebagai waktu yang dibutuhkan untuk mencapai maksimum amplitudo sinyal dari waktu batas minimum. Tujuannya adalah sebagai deskripsi *onsets* sampel suara tunggal dari alat musik yang berbeda. Dalam MPEG-7 standar, *LAT* didefinisikan sebagai logaritma (basis desimal) dari durasi dari waktu T_{Start} ketika sinyal mulai dan T_{Stop} ketika mencapi nilai maksimum.

$$LAT = \log_{10}(T_{stop} - T_{start}) \tag{2.4}$$

dimana:

T_{start} = Waktu ketika sinyal dimulai
T_{Stop} = Waktu ketika sinyal maksimum

2.3.4. *Audio Spectrum Centroid*

Audio Spectrum Centroid (*ASC*) didefiniskan sebagai pusat gravitasi dari log-frekuensi spektrum daya. Semua

koefisien daya di bawah 62,5 Hz dijumlahkan dan diwakili oleh koefisien tunggal, untuk mencegah nol non-komponen DC dan sangat komponen frekuensi rendah dari yang memiliki berat proporsional.

$$ASC = \frac{\sum_{k'=0}^{(N_{FT}/2)} \log_2\left(\frac{f'(k')}{1000}\right) P'(k')}{\sum_{k'=0}^{(N_{FT}/2)} P'(k')} \tag{2.5}$$

$$K_{low} = floor(6.25/\Delta F) \tag{2.6}$$

$$P'(k') = \begin{cases} \sum_{k=0}^{K_{low}} P(k), & for\ k' = 0 \\ P(k' + K_{low}), & for\ 1 \le k' \le \frac{N_{FT}}{2} - K_{low} \end{cases} \tag{2.7}$$

$$f(x) = \begin{cases} 31.35, & for\ k' = 0 \\ f(k' + K_{low}), & for\ 1 \le k' \le \frac{N_{ft}}{2} - K_{low} \end{cases} \tag{2.8}$$

dimana:

N_{FT} = Ukuran dari *Fast Fourrier Transform*
$f'(k')$ = Sinyal envelope pada indeks *k*'
$P'(k')$ = Power spektrum yang diekstrak pada *frame* ke-*l*
ΔF = Interval frekuensi dari dua sinyal *FFT*
k = Frekuensi indeks

2.3.5. *Harmonic Spectral Centroid*

Harmonic Spectral Centroid (*HSC*) didefinisikan rata-rata (yang melebihi durasi dari sinyal) *amplitude-weighted* (pada skala linear) dari harmonic puncak spektrum. Nilai dapat diperoleh dari rata-rata centroid lokal atas dari total jumlah frame.

$$HSC = \frac{1}{L}\sum_{l=0}^{L-1} LHSC,$$

(2.9)

Dimana nilai *LHSC* diperoleh dari:

$$LHSC_l = \frac{\sum_{h=1}^{N_h}(f_{h,l}\, A_{h,l})}{\sum_{h=1}^{N_h} A_{h,l}}$$

(2.10)

$f_{h,l}$ = Frekuensi.
$A_{h,l}$ = Amplitudo puncak harmonik
N_h = Jumlah harmonik yang diperhitungkan
L = Jumlah frame dalam segmen suara
LHSC = Lokal Harmonic Spectral Centroid
h = Indeks dari komponen harmonis
l = Indeks Frame

2.3.6. *Harmonic Spectral Deviation*

Harmonic Spectral Deviation mengukur penyimpangan harmonis puncak dari amplop dari spektrum local.

$$HSD = \frac{1}{L}\sum_{l=0}^{L-1} LHSD_l$$

(2.11)

$$LHSD_l = \frac{\sum_{h=1}^{N_h} \left|\log_{10}(A_{h,l}) - \log_{10}(SE_{h_l})\right|}{\sum_{h=1}^{N_h} \log_{10}(A_{h,l})} \tag{2.12}$$

$$SE_{h_l} = \begin{cases} \frac{1}{2}\left(A_{h,l} + A_{h+1,l}\right) \; if \; h = 1 \\ \frac{1}{3}\left(A_{h_{-1},l} + A_{h,l} + A_{h+1,l}\right) \; if \; 2 \le h \le N_h - 1. \\ \frac{1}{2}\left(A_{h-1,l} + A_{h,l}\right) \; if \; h = N_h \end{cases} \tag{2.13}$$

dimana:

$A_{h,l}$ = Amplitudo puncak harmonik
N_h = Jumlah harmonik yang diperhitungkan
L = Jumlah frame dalam segmen suara
h = Indeks dari komponen harmonis

2.3.7. *Harmonic Spectral Spread*

Harmonic Spectral Spread (*HSS*) mengukur rata-rata *spread spectrum* dalam kaitannya dengan *Harmonic Spectral Centroid* (*HSC*). Pada tingkat *frame*, didefinisikan sebagai *power-weighted RMS* (*Root Mean Square*) dari penyimpangan *HSC* lokal.

$$HSS = \frac{1}{L}\sum_{l=0}^{L-1} LHSS_l \tag{2.14}$$

dimana nilai *LHSS*:

$$LHSS_l = \frac{1}{LHSS_l}\sqrt{\frac{\sum_{h=1}^{N_h}[(f_{h,l} - LHSC_l)^2 \, A_{h,l}^2 \,]}{\sum_{h=1}^{N_h} A_{h,l}^2}} \tag{2.15}$$

dimana:

L = jumlah frame dari segment suara
LSHH = cerminan modulasi vibrato yang kurang jelas dari LHSD
$f_{h,l}$ = frekuensi.
$A_{h,l}$ = amplitudo puncak harmonik.
N_h = jumlah harmonik yang diperhitungkan
L = jumlah frame dalam segmen suara.
LHSC = lokal Harmonic Spectral Centroid

2.3.8. *Harmonic Spectral Variation*

Harmonic Spectral Variation (*HSV*) mencerminkan variasi spektral antara frame yang berdekatan. Pada level *frame*, didefinisikan sebagai pelengkap untuk 1 dari korelasi dinormalisasi antara amplitudo puncak harmonik yang diambil dari dua frame yang berdekatan.

$$HSV = \frac{1}{L}\sum_{l=0}^{L-1} LHSV_l \tag{2.16}$$

$$LHSV_l = 1 - \frac{\sum_{h=1}^{N_h}(A_{h,l-1}\ A_{h,l})}{\sqrt{\sum_{h=1}^{N_h} A_{h,l-1}^2}\ \sqrt{\sum_{h=1}^{N_h} A_{h,l}^2}} \tag{2.17}$$

dimana:

L = Jumlah frame dari segment suara
$A_{h,l}$ = Amplitudo puncak harmonik
N_h = Jumlah harmonik yang diperhitungkan
h = Indeks dari komponen harmonis

2.3.9. *Audio Spectrum Spread Type*

Audio Spectrum Spread (*ASS*) merupakan contoh perhitungan lain dari bentuk *spectral*. Dalam MPEG-7, didefinisikan sebagai *central moment* kedua dari spektrum log-frekuensi. Untuk *frame rate* pada suatu sinyal. Fitur *Audio Spectrum Spread* diekstraksi dengan mengambil *RMS* dari standar deviasi spektrum fitur *Audio Spectrum Centroid*.

$$ASS = \sqrt{\frac{\sum_{k'=0}^{\left(\frac{N_{FT}}{2}\right)-K_{low}} \left[\log_2\left(\frac{f'(k')}{1000}\right) - ASC\right]^2 P'(k')}{\sum_{k'=0}^{\left(\frac{N_{FT}}{2}\right)-K_{low}} P'(k')}} \quad (2.18)$$

dimana:

N_{FT} = Ukuran dari *Fast Fourrier Transform*
$f'(k')$ = Sinyal envelope pada indeks *k'*
$P'(k')$ = Power spektrum yang diekstrak pada *frame* ke-*l*
k = Frekuensi indeks
ASC = Dijelaskan pada subbab 2.3.4 *Audio Spectrum Centroid*
K_{low} = Dijelaskan pada subbab 2.3.4 *Audio Spectrum Centroid*

2.3.10. *Audio Spectrum Projection*

Audio Spectrum Projection (ASP) digunakan untuk mewakili fitur rendah dimensi spektrum setelah proyeksi terhadap dasar peringkat berkurang. *Audio Spectrum Projection* merupakan spektogram yang digunakan klasifikasi sebagai suara dari berbagai sumber. Perbedaan antara *Audio Spectrum Projection* dan MFCC dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbedaan Antara MFCC dan *Audio Spectrum Projection*

Steps	MFCC	*Audio Spectrum Projection*
1	*Convert* ke *frame*	*Convert* ke *frame*
2	Setiap frame, didapatkan amplitudo dari spektrum	Setiap frame, didapatkan amplitudo dari spektrum
3	Mel-scaling	Skala logaritma oktaf *band*
4	Logaritma	Nosmalisasi
5	Aplikasi DCT	Reduksi Fitur pada seperti PCA, ICA, NMF, dan sebagainya.

Dari Tabel 2.1, perbedaan antara MFCC dan *Audio Spectrum Projection* dijelaskan [7]. Kedua MFCC dan *Audio Spectrum Projection* memiliki metode yang sama untuk memperoleh amplitudo sumber spektrum ini. Perbedaan yang MFCC menggunakan metode sendiri (*Mel-scaling*) untuk melakukan skala amplitudo dari sinyal, sementara *Audio Spectrum Projection* menggunakan metode logaritmik skala untuk melakukan skala amplitudo. *Audio Spectrum Projection* menggunakan metode normalisasi untuk mengurangi kompleksitas nilai spektrum sementara MFCC menggunakan metode logaritma untuk mengurangi kompleksitas. Terakhir, MFCC menggunakan DCT (*Discrete Cosine Transforms*) untuk reduksi panjang sinyal, sementara *Audio Spectrum Projection* digunakan PCA, ICA dari NMF untuk metode reduksi dimensi.

2.3.11. *Audio Spectrum Envelope Type*

Audio Spectrum Envelope (*ASE*) didefinisikan sebagai log-frekuensi daya spektrum yang dapat digunakan untuk menghasilkan spektogram mengurangi sinyal audio asli. Fitur ini diperoleh dengan menjumlahkan energi dari pangkat spektrum asli dalam seri frekuensi *band*.

$$ASE(b) = \sum_{k=loK_b}^{hiK_b} P(k) \quad (1 \le b \le B_{in}),$$

(2.19)

$$r = 2^j \; octaves \; (-4 \; \le j \le +3)$$

(2.20)

$$loF_b = loEdge \times 2^{(b-1)r} \; (1 \le b \le B_{in})$$

(2.21)

$$hiF_b = loEdge \times 2^{br} \; (1 \le b \le B_{in})$$

(2.22)

dimana:

loEdge	= 62.5 Hz
B_{in}	= 8/r
b	= Frekuensi dari indeks *band*
k	= Frekuensi indeks
hiK_b	= Frekuensi *integer* batas dari hiF_b
loK_b	= Frekuensi *integer* batas dari loF_b

2.3.12. *Audio Spectrum Flatness Type*

Audio Spectrum Flatness (*ASF*) didefinisikan sebagai kerataan sinyal pada suatu spektrum. Dapat juga digunakan sebagai fitur yang mengukur similaritas antara satu sinyal dengan sinyal yang lain.

$$ASF = \frac{\sqrt[hiK'_b - loK'_b + 1]{\prod_{k'=loK'_b}^{hiK'_b} P_g(k')}}{\frac{1}{hiK'_b - loK'_b + 1} \sum_{k'=loK'_b}^{hiK'_b} P_g(k')} \qquad (1 \le b \le B).$$

(2.23)

$$loEdge = 2^{\frac{1}{4}n} \times 1\ kHz \tag{2.24}$$

$$hiEdge = 2^{\frac{1}{4}B} \times loEdge \tag{2.25}$$

$$loF_b = 0.95 \times loEdge \times 2^{\frac{1}{4}(b-1)} \tag{2.26}$$

$$hiF_b = 1.05 \times loEdge \times 2^{\frac{1}{4}b} \tag{2.27}$$

dimana:

n = Nilai penentu batas *band* terbawah. Nilai rekomendasi minimum adalah -8.
B = Nilai penentu batas atas *band*
b = Frekuensi dari indeks *band*
$P_g(k')$ = Power spektrum yang diekstrak pada *frame* ke-*l*
hiK_b = Frekuensi *integer* batas dari hiF_b
loK_b = Frekuensi *integer* batas dari loF_b

2.3.13. *Audio Fundamental Frequency*

Audio Fundamental Frequency (*AFF*) merupakan *descriptor* memberikan perkiraan frekuensi dasar f_0, dimana sinyal diasumsikan periodik.

Terdapat 2 parameter yang ditambahkan dalam standarisasi MPEG-7 ini, yaitu:

- *loLimits* yang berarti batas bawah dari rentang frekuensi dimana f_0 telah dicari.
- *hiLimits* yang berarti batas atas dari rentang frekuensi dimana f_0 telah dicari.

Ukuran kepercayaan pada kehadiran periodisitas di bagian dianalisis dari sinyal, nilainya antara 0 dan 1.

2.4. *Discrete Wavelet Transform*

Discrete Wavelet Transform adalah metode yang dilakukan untuk kompresi terhadap sinyal namun masih mempertahankan sinyal asli. Sinyal yang telah dilakukan *wavelet* akan memiliki panjang lebih pendek dari aslinya. Ada beberapa nilai yang dapat diambil jika melakukan *wavelet* terhadap suatu audio, nilai tersebut adalah *low-pass* dan *high-pass*. Nilai *low-pass* merupakan suara dari penyanyi dari lagu (dalam hal ini perempuan dan laki-laki akan memiliki nilai yang hampir mirip), dan nilai *high-pass* yang merupakan suara dari instrumen dari lagu yang dilakukan *Discrete Wavelet Transform*. Dalam penelitian ini nilai yang diambil dari *Discrete Wavelet Transform* hanya pada nilai *low-pass filter* saja. Pengambilan nilai *low-pass filter* dapat dilakukan dengan cara mengambil nilai *approximation* dari *Discrete Wavelet Transform*. *Discrete Wavelet Transform* sangat bergantung pada level dekomposisisi yang akan digunakan. Semakin tinggi level dekomposisi yang digunakan, maka akan semakin tinggi resiko sinyal dapat rusak. Sebaliknya, semakin rendah level dekomposisi yang digunakan, maka sinyal akan semakin rentan terhadap *noise* [8]. Untuk menentukan level dekomposisi yang tepat, maka perlu diaplikasikan *pseudocode* pada Kode Sumber 2.2 dan hasil dari *pseudocode* tersebut akan dibandingkan dengan Tabel 2.2 [9].

```
1.  frekuensi(arr):
2.      npArr = array(arr)
3.      mean = mean(npArr)
4.      fin = npArr - mean
5.
6.      hasil = max(abs(np.fft.fft(fin.transpose(), axi
    s = 0) ) )
7.      fft = np.array(abs(np.fft.fft(fin.transpose(),
    axis=0 )))
8.      maxIndex = np.where(fft == hasil)[0][0]
9.
10.     fre = float(maxIndex) * 1024 / len(arr)
11.
```

```
12.     return nilaiDekomposisi(fre)
```

Kode Sumber 2.2 Pseudocode level dekomposisi

Tabel 2.2 Tabel Level Dekomposisi pada *Wavelet*

Decomposition level (L)	Frequency range (Fr) (Hz)
1	256-512
2	128-256
3	64-128
4	32-64
5	16-32
6	8-16
7	4-8
8	2-4
9	1-2
10	0.5-1
11	0.25-0.5
12	0.125-0.25
13	0.0625-0.125

2.5. Xquery

Xquery adalah bahasa untuk melakukan *query* / pemanggilan data di dalam suatu dokumen XML [10]. Dalam penelitian kali ini Xquery dieksekusi pada bahasa pemrograman Java dengan menggunakan *library BaseX*.

```
1.  private static String AudioSpectrumFlatnessType(Str
    ing string) throws BaseXException{
2.
3.     String query =
4.       "declare default element namespace \"urn:mpeg:
    mpeg7:schema:2001\";" +
```

```
5.                          "declare namespace mpeg7 =
    \"urn:mpeg:mpeg7:schema:2001\";" +
6.                          "declare namespace xsi = \"
    http://www.w3.org/2001/XMLSchema-instance\";" +
7.                          "for $x in doc(\"C:/Users/p
    onighzwa/Desktop/CoverSong2/Test/XML/"+ string+".xm
    l\")/Mpeg7/Description/MultimediaContent/" +
8.                          "Audio/AudioDescriptor\n re
    turn if($x/@xsi:type=\"AudioSpectrumFlatnessType\")
    then data($x/SeriesOfVector/Raw) else \"\"";
9.
10.         //System.out.println(new XQuery(query).exec
    ute(context));
11.         String hasil = new XQuery(query).execute(co
    ntext);
12.         return hasil;
13.     }
```

Kode Sumber 2.3 Xquery pada fitur MPEG-7

Kode Sumber 2.3 merupakan salah satu contoh implementasi Xquery dalam pengambilan suatu fitur pada dokumen XML. *Library BaseX* menerima *input* berupa varibel *string* untuk eksekusi Xquery pada dokumen XML. Hasil dari proses tersebut adalah variabel *string* yang merupakan isi dari *tag* dalam suatu dokumen XML.

2.6. MIR (Music Information Retrieval)

MIR merupakan suatu istilah yang melakukan pengolahan sinyal dari suatu music untuk tingkat yang lebih lanjut. Dari MIR dapat dilakukan identifikasi *fingerprint*, *genre identification*, *cover identification*, dan sebagainya [11].

2.7. Fingerprint

Fingerprint merupakan suatu istilah yang memiliki definisi sebagai ciri khas dari sebuah lagu. *Fingerprint* digunakan untuk mengidentifikasi pada sebuah lagu. Bentuk dari *fingerprint* yang dimiliki pada sebuah lagu adalah sebuah sinyal. Sinyal ini

bersifat unik, sehingga antara satu lagu dengan lagu lain memiliki *fingerprint* yang berbeda. Sehingga dari sinyal inilah yang digunakan untuk mengidentifikasi suatu musik (*audio*). *Fingerprint* suatu musik didapatkan dengan cara melakukan pengambilan fitur *Audio Signature Type*[12].

$$\underset{m}{\begin{bmatrix} 0.78 & 0.71 & 0.69 & \dots & \dots & 0.72 \\ 0.72 & 0.58 & 0.53 & \dots & \dots & 0.57 \\ 0.71 & 0.71 & 0.60 & \dots & \dots & 0.53 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0.66 & 0.70 & 0.47 & \dots & \dots & 0.46 \end{bmatrix}}_{n}$$

Gambar 2.2 Sinyal dalam Matriks

Gambar 2.2 merupakan contoh dari sinyal yang tersimpan dalam bentuk matriks. Matriks tersebut memiliki ukuran $n \times m$ dengan nilai *m* adalah 16 untuk *Audio Signature Type* dan nilai *n* yang tergantung dari ukuran atau durasi dari *file* musik (audio) yang terkait. Matriks inilah yang merupakan ciri khas dari suatu musik yang diolah sistem untuk melakukan pendeteksian sebuah lagu.

2.8. *Cover Song*

Istilah yang digunakan untuk menggambarkan kondisi dimana sebuah lagu dinyanyikan kembali oleh artis lain. Pengemasan dalam hal menyanyi dapat menggunakan nada yang sama dan alat musik yang sama, atau nada yang beda dan alat musik yang berbeda. Namun ketika menyanyikan lagu tersebut harus menggunakan runtutan nada yang sama agar tidak mengubah lagu asli. Untuk melakukan pendeteksian *cover song*, penelitian ini menggunakan 2 fitur yaitu fitur *Audio Spectrum Projection* dan *Audio Spectrum Flatness*. Penelitian ini menggunakan *Audio Spectrum Projection* karena fitur ini

memang ada untuk melakukan klasifikasi sebuah musik (audio) yang berasal dari berbagai macam sumber suara [7].

Hasil ekstraksi fitur *Audio Spectrum Projection* dan *Audio Spectrum Flatness* berupa sebuah sinyal. Sinyal tersebut akan tersimpan dalam bentuk matriks yang sama dengan Gambar 2.2. Perbedaannya hanya terletak pada nilai *m*, sebab *Audio Spectrum Projection* memiliki jumlah *m* sebanyak 9 dan *Audio Spectrum Flatness* memiliki jumlah *m* sebanyak 21.

2.9. Play Framework

Suatu *framework* dalam pemrograman bahasa Java yang digunakan sebagai aplikasi berbasis web [13]. *Play Framework* digunakan sebagai server yang mengolah data audio / musik untuk dilakukan ekstrak data dan memperolah sinyal-sinyal yang terkait. Dalam penelitian ini *Play Framework* akan dipanggil melalui *file routes*. Sehingga ketika alamat yang tersimpan dalam *file routes* dipanggil, *Play Framework* akan melakukan kompilasi ekstraksi fitur sesuai dengan alamat yang dipanggil.

```
1.  GET     /AudioPower                 controllers.HomeCont
    roller.AudioPower
2.  GET     /AudioHarmonicity           controllers.HomeCont
    roller.AudioHarmonicity
3.  GET     /AudioCentroid              controllers.HomeCont
    roller.AudioSpectrumCentroid
4.  GET     /AudioSpread                controllers.HomeCont
    roller.AudioSpectrumSpread
5.  GET     /Descriptor/:nama           controllers.Descript
    or.index(nama: String)
```

Kode Sumber 2.4 *File routes Play Framework*

```
1.  public class Descriptor extends Controller {
2.
3.      public Result index(String nama){
4.          try {
5.
```

```
6.              String configSource = "C:/Users/ponighzwa/
    IdeaProjects/CobaRP/config/config.xml";
7.              InputStream inputStream = new FileInputStr
    eam(configSource);
8.              Reader reader = new InputStreamReader(inpu
    tStream);
9.              Config config = ConfigXML.parse(reader);
10.
11.             String input = "C:/xampp/htdocs/musicmoo/u
    ploads/"+nama;
12.             File file = new File(input);
13.             AudioInputStream audioInputStream = AudioS
    ystem.getAudioInputStream(file);
14.             Document mpeg7 = MP7DocumentBuilder.encode
    (audioInputStream,config);
15.             . . . . . .. . . .
```

Kode Sumber 2.5 *Class* pada *Play Framework*

Kode Sumber 2.4 merupakan contoh *file routes* yang terdapat pada *Play Framework*. Pada baris ke 5 Kode Sumber 2.4 akan mengambil kelas *Descriptor* untuk dilakukan kompilasi. Sehingga, ketika *routes* "/*Descriptor/:nama*" dipanggil akan melakukan kompilasi pada Kode Sumber 2.5.

2.10. Flask

Suatu *web application framework* yang terdapat pada bahasa pemrograman Python [14]. *Flask* digunakan sebagai server yang melakukan perhitungan dan klasifikasi data *testing* terhadap dataset. Eksperimen ini memilih *Flask*, sebab *Flask* memiliki tingkat fleksibel yang sangat tinggi untuk digunakan bersamaan dengan *library* Python pada umumnya. Sama halnya dengan *Play Framework*, *Flask* memiliki *routes* yang dapat dipanggil. Ketika *routes* dalam *Flask* dipanggil, maka *Flask* akan melakukan kompilasi sesuai dengan *routes* yang bersangkutan.

```
1.  @app.route("/slidingSignature/")
2.  def slidingSignature():
```

```
3.      conn = MySQL.connect()
4.      cursor = conn.cursor()
5.
6.      cursor.execute('Select * from Lagu')
7.      conn.commit()
8.      data1 = cursor.fetchall()
9.
10.     cursor.execute('Select * from Testing')
11.     conn.commit()
12.     testing = cursor.fetchall()
```

Kode Sumber 2.6 Potongan dari *Flask*

Kode Sumber 2.6 merupakan penggalan kode dari aplikasi *Flask*. Pada baris 1 dapat diamati bahwa penggalan kode tersebut memiliki *routes* "*/slidingSignature/*". Sehingga ketika "*/slidingSignature/*" dijalankan akan menjalankan fungsi "*slidingSignature()*" untuk melakukan kompilasi.

2.11. KNN

Dalam penelitian ini implementasi klasifikasi KNN tidak dapat langsung diaplikasikan ke dalam studi kasus. Hal ini disebabkan karena perbedaan dimensi matriks pada fitur antara data latih dan data uji. Sebagai suatu contoh pada kasus bunga *iris*, data latih dan data uji merupakan suatu data yang *single*. Namun pada implementasi dalam penelitian ini, data latih dan data uji berbentuk suatu matriks yang berbeda ukurannya. Perbedaan ukuran sebabkan oleh perbedaan ukuran dan panjang *file* dari suatu musik.

Pada Gambar 2.3 kasus klasifikasi bunga *iris* dapat langsung diaplikasikan klasifikasi KNN dengan implementasi perhitungan rumus jarak yang ada. Namun pada Gambar 2.4 kasus klasifikasi tidak dapat langsung diimplementasikan karena terdapat perbedaan dimensi antara data uji dan data latih. Untuk menangani kasus ini, maka klasifikasi KNN perlu dikombinasikan dengan algoritma *Sliding Algorithm* yang akan dibahas pada subbab selanjutnya.

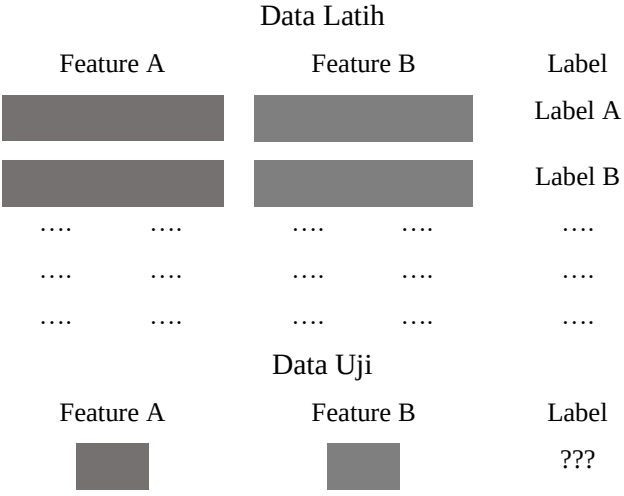
Training Iris Data

Sepal Width	Sepal Length	Petal Width	Petal Height	Iris Class
5.1	3.5	1.4	0.2	I.Setosa
4.9	3.0	1.4	0.2	I.Setosa
….	….	….	….	….
….	….	….	….	….

Testing Iris Data

Sepal Width	Sepal Length	Petal Width	Petal Height	Iris Class
4.0	3.3	2.0	0.5	??

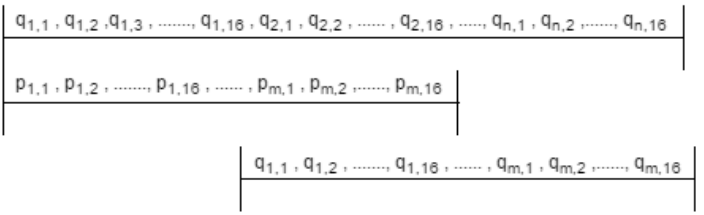
Gambar 2.3 Klasifikasi bunga *iris*



Gambar 2.4 Klasifikasi musik standar MPEG-7

2.12. *Sliding Algorithm*

Suatu algoritma yang digunakan dalam penelitian ini untuk mencari *distance* dari *subband* matriks. Dalam eksperimen ini, perhitungan *distance* yang digunakan adalah metode *Euclidian Distance*. Hasil *distance* yang didapat akan dilakukan klasifikasi menggunakan KNN. *Sliding Algorithm* diusulkan, karena ditemukan masalah dalam melakukan klasifikasi. Masalah tersebut adalah fitur dalam MPEG-7 tidak berupa satu data, melainkan berbentuk matriks.



Gambar 2.5 *Sliding Algorithm*

$$d(p,q) = \sqrt{\sum_i^n (p_i - q_i)^2} \tag{2.28}$$

Gambar 2.5 menjelaskan bagaimana *Sliding Algorithm* bekerja [1]. *Sliding Algorithm* akan membandingkan setiap bagian dari matriks, hasil *distance* terkecil dari metode ini dianggap mewakili nilai *similarity* dari matriks tersebut. Pada Gambar 2.5 matriks q memiliki dimensi $n \times 16$ namun dijadikan 1 *array*, sedangkan pada matriks q memiliki dimensi $m \times 16$ dan juga dijadikan 1 *array* dengan keterangan tambahan, nilai n > m. Karena mememiliki perbedaan dimensi antara matriks q dan p maka *Sliding Algorithm* perlu diimplementasikan. Perhitungan distance pada *Sliding Algorithm* yang dipakai menggunakan Persamaan 2.28, yang merupakan rumus dari *Euclidian Distance* [15]. *Euclidian Distance* merupakan suatu algoritma yang menghitung kemiripan dengan cara menghitung selisih tiap fitur.

Pada penelitian ini fitur yang dilakukan perhitungan *Euclidian Distance* adalah *subband* pada masing-masing sinyal.

BAB III
ANALISIS PERANCANGAN SISTEM

Bab ini membahas tahap analisis dan perancangan sistem yang akan dibangun. Analisis membahas semua persiapan yang akan menjadi pokok pikiran pembuatan aplikasi ini. Mulai dari masalah yang melatar belakangi, hingga analisis gambaran awal sistem yang akan dibuat. Perancangan sistem membahas hal-hal yang berkaitan dengan pondasi atau dasar pembuatan aplikasi, yang meliputi perancangan basis data, tampilan antar muka halaman aplikasi, hingga perancangan alur proses yang akan diimplementasikan di dalam aplikasi.

3.1. Analisis

Tahap analisis meliputi analisis masalah, analisis kebutuhan, deskripsi umum sistem, dan kasus penggunaan sistem yang dibuat.

3.1.1. Analisis Permasalahan

Pada era globalisasi ini, pengolahan data sangatlah dibutuhkan. Pengolahan data tidak hanya sebatas pemberian *id* pada *object*, namun konten atau isi dari *object* tersebut yang paling penting. Sehingga, pemberian label *id* belum tentu mencerminkan isi dari *object*. Penelitian yang merujuk pada tema musik diajukan, sebab pada tema musik dataset dan pengaplikasiannya dapat diperoleh dengan mudah melalui internet.

Oleh karena itu, penelitian ini diajukan dengan menggunakan tema musik. Namun tujuan utama pada penelitian ini adalah tentang cara mengolah dan mendapatkan *object*, bukan dari label *id* yang diberikan pada objek melainkan mengolah konten atau isi dari *object* tersebut. Fokus utama pada penelitian bertema musik kali ini hanya pada *fingerprint*, *cover song recognition*, dan *song recommendation*. Ketiga modul tersebut

akan diimplementasikan dalam aplikasi bernama *MusicMoo* dalam *platform* perangkat bergerak.

3.1.2. Analisis Kebutuhan

Kebutuhan utama dalam penelitian ini adalah pada *fingerprint*, *cover song recognition* dan *song recommendation*. *Fingerprint*, berarti berusaha mengidentifikasi lagu penuh dari potongan lagu yang diberikan. *Cover song recognition* berarti melakukan deteksi terhadap lagu yang dinyanyikan lagi oleh orang lain. *Song recommendation*, berarti memberikan daftar lagu yang direkomendasikan berdasarkan lagu yang dicari oleh pengguna sekarang.

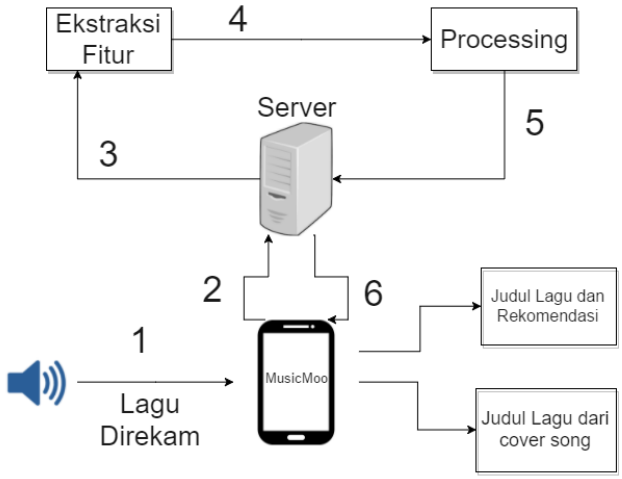
Penelitian kali ini, akan menggunakan standar MPEG-7, yang merupakan standar ISO dari konten dari sebuah audio. Dari konten standar ISO MPEG-7 akan dibandingkan kontennya dengan audio yang lain. Daftar kebutuhan fungsional perangkat lunak yang dibangun dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Kebutuhan Fungsional Perangkat Lunak

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-1	Melihat judul potongan lagu	Sistem dapat mengenali lagu dari *fingerprint* yang dimiliki dari potongan lagu
F-2	Melihat judul *cover* lagu	Sistem dapat mengenali lagu asli dari potongan lagu yang dinyayikan oleh artis lain.
F-3	Menampilkan daftar lagu	Sistem memberikan daftar dari lagu yang saat ini sedang direkam oleh pengguna.

3.1.3. Deskripsi Umum Sistem

Aplikasi yang akan dibuat pada Tugas Akhir ini adalah program aplikasi perangkat bergerak. Alur kerja pada sistem akan digambarkan pada Gambar 3.1.



Keterangan Proses :
Proses 1 : Perekaman musik
Proses 2 : Musik diupload pada server
Proses 3 : Ekstraksi Fitur pada potongan Musik
Proses 4 : Hasil ekstraksi dikirimkan pada server *Python*
Proses 5 : Kalkulasi pada fitur dari potongan Musik
Proses 6 : Hasil dikirimkan kembali pada aplikasi android untuk ditampilkan

Gambar 3.1 Arsitektur Sistem

Pada Gambar 3.1 di atas, akan dijelaskan dengan rinci mulai dari *input* (Proses 1), *pre-processing*, *processing*, dan *output* (Proses 6). Detail runtutan aplikasi dari *input* hingga *output* akan dijelaskan pada subbab berikut:

3.1.3.1. *Input*

Lagu akan didengarkan dari aplikasi perangkat bergerak berbasis Android. Lagu yang didengarkan hanya berupa potongan lagu, bukan lagu seutuhnya. Potongan lagu yang didengarkan akan disimpan dalam penyimpanan *internal* dalam ekstensi audio .wav (Proses 1). Ketika sudah dilakukan

penyimpanan, lagu akan di-*upload* ke server dan diterima oleh XAMPP (Proses 2).

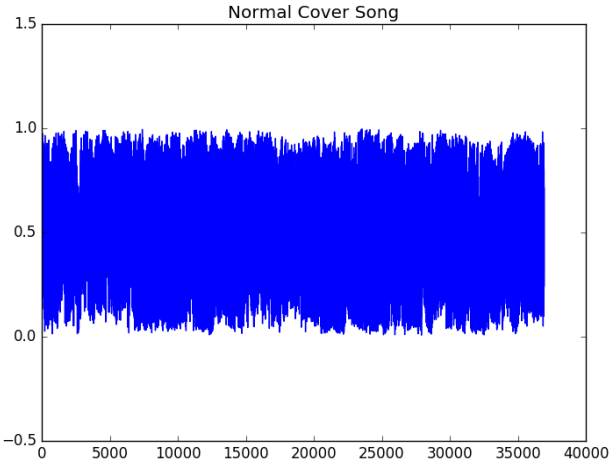
3.1.3.2. Proses

Proses yang terjadi pada sistem dilakukan oleh server Java dan server Python. Server Java memiliki fungsi utama untuk melakukan ektraksi fitur dari *Input*. Server Python memiliki fungsi utama untuk melakukan perhitungan hasil dari ekstraksi fitur. Perhitungan yang dimaksud adalah *pre-processing* sinyal dan *processing* sinyal. Detail dari perancangan ekstraksi fitur, *pre-processing*, dan *processing* akan dibahas pada subbab berikut.

3.1.3.2.1 Ekstraksi Fitur

Setelah lagu sudah diunggah, langkah selanjutnya adalah melakukan ekstrak fitur. Fitur yang digunakan sesuai dengan modul masing-masing. Pada modul *fingerprint* dan *song recommendation* akan menggunakan fitur *Audio Signature*, sedangkan pada modul *cover song recognition* akan menggunakan 2 fitur yaitu, *Audio Spectrum Projection* dan *Audio Spectrum Flatness*. Ekstraksi fitur audio dengan format .wav akan menggunakan *Library MPEG7AudioEncApp* dengan menggunakan bahasa pemograman Java (Proses 3).

Setelah dilakukan ekstraksi fitur, kemudian dilakukan pengambilan fitur pada dokumen hasil menggunakan bahasa Xquery. Xquery perlu dilakukan, sebab hasil ekstraksi dari *Library MPEG7AudioEncApp* merupakan dokumen dengan format dokumen XML. Hasil dari Xquery dalam dokumen XML adalah dalam bentuk *string*. Data *string* inilah yang diambil untuk dilakukan *processing* data.

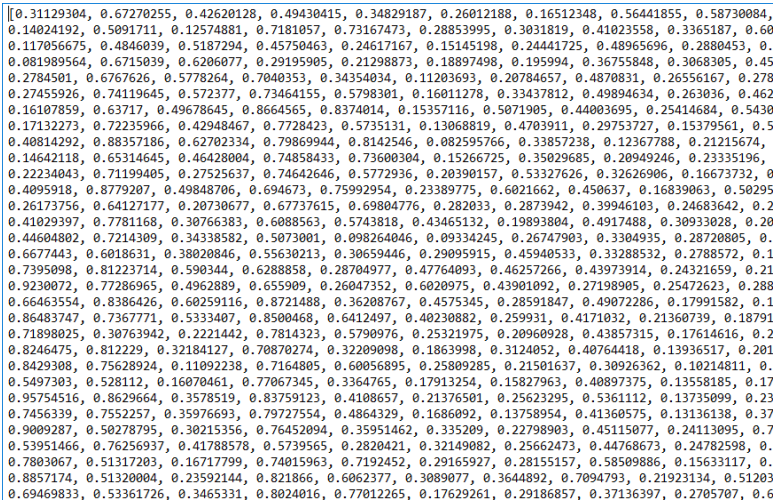


Gambar 3.2 Sinyal dari suatu lagu

Gambar 3.2 merupakan contoh dari fitur *Audio Spectrum Flatness* yang merupakan hasil dari ekstraksi. Fitur-fitur audio standar MPEG-7 tidak berupa *single* data, namun fitur audio merupakan data yang berupa matriks. Sehingga, fitur yang digunakan untuk masing-masing modul perlu dilakukan transformasi 1 dimensi agar dapat dilakukan *pre-processing* dan *processing* data. Keluaran dari tahapan ekstraksi fitur adalah *string* dari fitur-fitur yang digunakan.

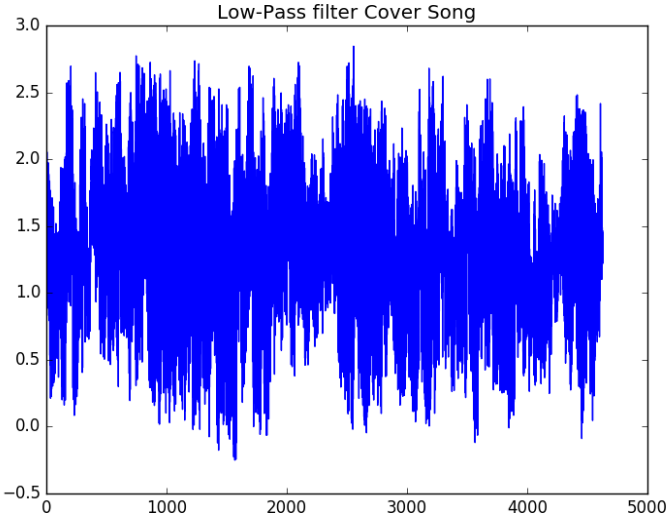
3.1.3.2.2 *Pre-processing*

Tahapan *pre-processing* mempunyai *input* berupa nilai *string* ekstraksi fitur. Nilai *string* tersebut telah memiliki format yang sesuai terhadap tipe data *list* pada Python. Sehingga, tahapan *pre-processing* dimulai dengan melakukan perubahan tipe data pada masing-masing *string* menjadi tipe data *float* pada Python (Proses 4).



Gambar 3.3 Fitur dalam bentuk text

Gambar 3.3 merupakan contoh dari keluaran ekstraksi fitur yang perlu dilakukan perubahan tipe data. Setelah dilakukan perubahan tipe data, data berupa matriks tersebut perlu diaplikasikan *Discrete Wavelet Transform* untuk mengambil nilai tertentu. Namun untuk *fingerprint* tidak perlu diaplikasikan *Discrete Wavelet Transform*, tahapan *pre-processing* hanya sampai perubahan tipe data.



Gambar 3.4 Sinyal dari lagu setelah di-*wavelet*

Gambar 3.4 merupakan contoh hasil fitur *Audio Spectrum Flatness* yang hanya diambil *low-pass filter* saja. Hasil dari implementasi *Discrete Wavelet Transform* akan digunakan sebagai *input* pada tahapan *processing*. Hasil dari *Discrete Wavelet Transform* merupakan sekumpulan nilai dalam bentuk *array* 1 dimensi.

3.1.3.2.3 *Processing*

Dataset untuk tahapan processing adalah 11 untuk modul *fingerprint* dan 50 untuk modul *cover song recognition* dalam format audio berekstensi .wav. Lagu yang dijadikan dataset, tidak seutuhnya disimpan dalam *database*, melainkan dipotong menjadi 1 menit. Setiap potongan akan disimpan pada *database* sesuai dengan modul masing-masing. Untuk modul *cover song recognition* digunakan 5 judul lagu dengan

masing-masing judul memiliki data *training* sebanyak 10 lagu dengan komposisi 5 penyanyi laki-laki dan 5 perempuan. Untuk modul *fingerprint* judul lagu yang dipotong 1 menit dan disimpan hanya 1 lagu untuk setiap judul.

Data *testing* data hasil tahapan *pre-processing* akan dilakukan perhitungan dan klasifikasi terhadap setiap *dataset* yang ada dalam *database*. Proses klasifikasi menggunakan *Sliding Algorithm* untuk mencari nilai *similarity* terkecil dari masing-masing lagu. Hasil terkecil dianggap sebagai lagu yang dicari dalam *dataset* (Proses 5).

3.1.3.3. *Output*

Output dari sistem hampir sama untuk masing – masing modul. *Output* berupa tipe data *string* yang akan dikirim kembali pada aplikasi Android (Proses 6). Tipe data *string* inilah yang mengandung informasi judul lagu sesuai dengan permintaan modul. Namun, untuk modul *song recommendation* akan mengembalikan *string* dengan beberapa nilai yang mengandung judul lagu yang direkomendasikan berdasarkan kemiripan lagu yang dicari saat ini.

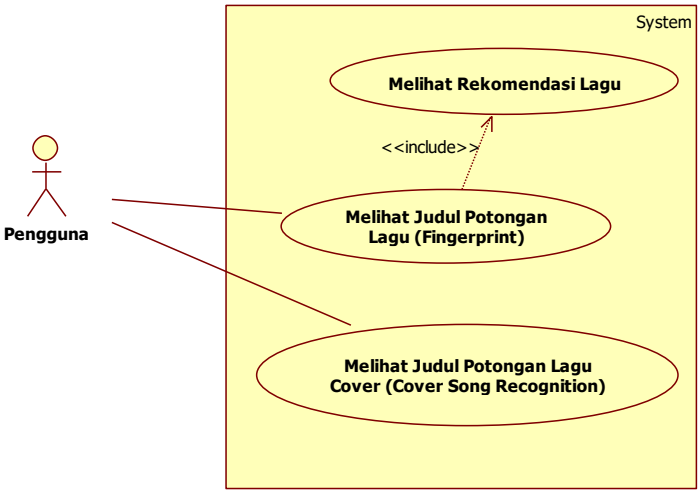
3.1.4. Kasus Penggunaan

Sesuai pada spesifikasi kebutuhan fungsional yang telah dipaparkan, maka akan disusun tabel kasus penggunaan, yang diharapkan dapat memenuhi kebutuhan fungsional. Kasus penggunaan dijelaskan lebih lanjut pada Tabel 3.2 dan diagram kasus penggunaan ditunjukkan pada Gambar 3.5.

Tabel 3.2 Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	Nama	Aktor
UC-1	Melihat judul potongan lagu *(Fingerprint)*	Pengguna
UC-2	Melihat judul potongan lagu *cover (Cover Song Recognition)*.	Pengguna

UC-3	Melihat *song recommendation* lagu.	Pengguna



Gambar 3.5 Diagram Kasus Penggunaan

3.1.4.1. Melihat Judul Potongan Lagu (*Fingerprint*) (UC-1)

Kasus penggunaan kode UC-1 diakses saat pengguna memilih menu deteksi lagu. Pengguna akan diminta untuk merekam sebuah potongan lagu pada aplikasi. Akan terdapat 3 tombol, yaitu tombol rekam, berhenti, dan kirim. Tombol rekam berguna ketika pengguna akan mulai merekam potongan lagu pada aplikasi. Tombol berhenti berguna ketika pengguna selesai merekam potongan lagu, dan aplikasi akan melakukan penyimpanan potongan. Tombol kirim berguna ketika pengguna ingin mengetahui judul dari potongan lagu yang telah direkam.

3.1.4.2. Melihat Judul Potongan Lagu Cover (*Cover Song Recognition*) (UC-2)

Kasus penggunaan kode UC-2 diakses saat pengguna memilih menu deteksi lagu *cover*. Pengguna akan diminta untuk merekamkan sebuah potongan lagu *cover* pada aplikasi. Akan terdapat 3 tombol, yaitu tombol rekam, berhenti, dan kirim. Tombol rekam berguna ketika pengguna akan mulai merekam potongan lagu *cover* pada aplikasi. Tombol berhenti berguna ketika pengguna selesai merekam potongan lagu *cover*, dan aplikasi akan melakukan penyimpanan potongan. Tombol kirim berguna ketika pengguna ingin mengetahui judul asli dari potongan lagu *cover* yang telah direkam.

3.1.4.3. Melihat Song Recommendation Lagu (UC-3)

Kasus penggunaan kode UC-3 diakses pengguna saat memilih kasus penggunaan UC-1. Pengguna akan mendapatkan *list* berupa rekomendasi yang mungkin akan disukai pengguna berdasarkan kemiripan dari judul lagu yang dicari pada saat ini.

3.2. Perancangan Sistem

Tahap ini meliputi perancangan basis data, tampilan antarmuka, dan perancangan alur proses penggunaan sistem yang diharapkan dapat memenuhi tujuan dari pengembangan aplikasi ini. Perlu diketahui bahwa aplikasi ini dibangun dalam kondisi lingkungan tertentu, dan dapat dioperasikan dalam lingkungan tertentu pula. Lingkungan implementasi adalah lingkungan di mana sistem tugas akhir ini dibangun. Lingkungan implementasi dibagi dua yaitu perangkat keras dan perangkat lunak.

Perangkat keras

Lingkungan implementasi perangkat keras dari tugas akhir ini adalah sebagai berikut:

- Komputer : Prosesor Intel® Core™ i3-CPU (3.30GHz),RAM 4 GB, Graphic Intel ® Sandybridge Desktop
- Android : OS Android v5.1 Lollipop Chipset Snapdragon 616 CPU, Quad-core 1.7 GHz & quad-core 1.0 GHz GPU, Adreno 405 RAM 3 GB

Perangkat lunak

Lingkungan implementasi perangkat lunak dari tugas akhir ini adalah sebagai berikut:

- Sistem operasi : Ubuntu 16.04 LTS , Android Lolipop
- IDE : Gedit text Editor, Android Studio 2.0, IntelliJ 2016.1.1
- Basis Data : MySQL 5.6
- SDK : SDK Android Lolipop level 23.

3.2.1. Perancangan Basis Data

Pada subbab ini dijelaskan mengenai perancangan basis data yang dalam hal ini digunakan untuk menyimpan *dataset* lagu baik dari modul *cover song* maupun *fingerprint*. *Database* menggunakan SQL *database*, namun tidak ada relasi antar tabel. Untuk menangani penyimpanan sinyal, *noSQL database* seharusnya yang digunakan. Namun, karena kekurangan sumber daya manusia maka eksperimen ini menggunakan SQL *database*. Berikut adalah daftar tabel beserta atributnya.

3.2.1.1. Tabel *Fingerprint* dan *Song Recommendation*

Tabel untuk modul *fingerprint* dan *song recommendation*, memiliki beberapa atribut. Atribut tersebut adalah *id*, judul, dan, *signature*. *id* merupakan atribut yang wajib dimiliki oleh sebuah *record* pada suatu tabel, *judul* digunakan sebagai label dari suatu lagu dalam satu *record*. Atribut *signature* memiliki peran paling penting, sebab identitas dari suatu lagu (metadata lagu) disimpan pada atribut tabel *signature*. Detail dari penjelasan disajikan pada Tabel 3.3 dan tabel pada *database* dapat dilihat pada Gambar 3.6.

Tabel 3.3 Tabel atribut modul *fingerprint*

Fingerprint	
id	Int (pk,auto increment)
judul	Varchar (250)
signature	Text

id	judul	signature	
39	CarelessWhisper1	[0.7818754, 0.70698506, 0.6896374, 0.6454572, 0.60142446,...	11K
40	CarelessWhisper2	[0.5995887, 0.6492162, 0.56979877, 0.48182374, 0.45647013...	11K
41	CarelessWhisper3	[0.62717164, 0.5354338, 0.5451959, 0.66517514, 0.6076055,...	11K
42	CarelessWhisper4	[0.69670504, 0.70051026, 0.6593561, 0.60889053, 0.4961917...	11K
43	CarelessWhisper5	[0.8116155, 0.76436883, 0.5743023, 0.45064625, 0.660636, ...	11K
44	ItsMyLife1	[0.6943918, 0.65440446, 0.65582544, 0.6674884, 0.602984, ...	11K
45	EverybodyKnew1	[0.7503462, 0.70866627, 0.7950047, 0.7595779, 0.7448186, ...	11K
46	EverybodyKnew2	[0.617767, 0.35687435, 0.39234802, 0.28745475, 0.4540707,...	11K
47	EverybodyKnew3	[0.45403126, 0.5288235, 0.56136113, 0.437311, 0.451636, 0...	11K
48	EverybodyKnew4	[0.63154924, 0.2564481, 0.65704703, 0.5901824, 0.25508752...	11K
49	ItsMyLife2	[0.61557156, 0.58826095, 0.61071193, 0.5752341, 0.4718219...	11K
50	ItsMyLife3	[0.7475947, 0.71742725, 0.58259374, 0.543677, 0.4693467, ...	11K
51	Faded1	[0.7442439, 0.73090243, 0.6874964, 0.76301455, 0.7970525,...	11K
52	Faded2	[0.7377208, 0.67339605, 0.49892992, 0.3897184, 0.35633084...	11K
53	ItsMyLife4	[0.62386614, 0.6604635, 0.5289524, 0.5450339, 0.53907824,...	11K
54	Faded3	[0.60820377, 0.62444615, 0.6155331, 0.57929134, 0.3179771...	11K

Gambar 3.6 Tabel Modul *Fingerprint* pada *database*

3.2.1.2. Tabel *Cover Song Recognition*

Tabel untuk modul *cover song recognition*, memiliki beberapa atribut. Atribut tersebut adalah *id*, judul, *flatness* dan, *projection*. *id* merupakan atribut yang wajib dimiliki oleh sebuah *record* pada suatu tabel, *judul* digunakan sebagai label dari suatu lagu dalam satu *record*. Atribut *flatness* dan *projection* memiliki

peran paling penting, sebab identitas dari suatu lagu (metadata lagu) disimpan pada atribut tabel *flatness* dan *projection*. Detail dari penjelasan disajikan pada Tabel 3.4 dan tabel pada *database* dapat dilihat pada Gambar 3.7.

Tabel 3.4 Tabel atribut modul *cover song*

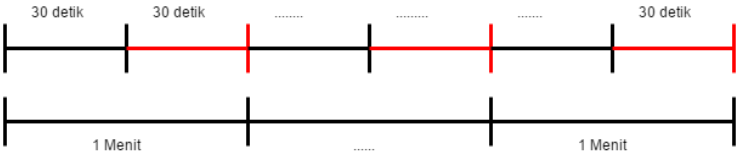
Cover Song	
id	Int (pk,auto increment)
judul	Varchar (250)
projection	Medium BLOB
flatness	Medium BLOB

id	judul	flatness		projection	
1	Heathens	[0.6768899, 0.67863613, 0.70004195, 0.6...	1M	[256.28793, -0...	696K
2	Heathens-1	[0.1305556, 0.34193388, 0.5180551, 0.47...	1M	[221.2429, -0....	698K
3	Heathens-2	[0.334596, 0.68305075, 0.8974304, 0.719...	1M	[154.14124, -0...	700K
4	Heathens-3	[0.27892512, 0.59555453, 0.62290275, 0....	1M	[349.92798, -0...	699K
5	Heathens1	[0.8965845, 0.638196, 0.7666013, 0.8707...	1M	[249.39134, -0...	705K
6	Heathens1-1	[0.5827877, 0.30394018, 0.28832832, 0.1...	1M	[340.08875, -0...	706K
7	Heathens1-2	[0.8218572, 0.8747742, 0.5461863, 0.418...	1M	[292.50247, -0...	706K
8	Heathens2	[0.54795235, 0.5875927, 0.8199391, 0.24...	1M	[257.09122, -0...	698K
9	Heathens2-1	[0.28810108, 0.6340984, 0.5559035, 0.41...	1M	[249.17459, -0...	736K
10	Heathens2-2	[0.96300435, 0.635596, 0.53484416, 0.50...	1M	[219.41267, -0...	701K
11	Heathens3	[0.74653965, 0.037001375, 0.060232628, ...	1M	[331.9478, -0....	695K
12	Heathens3-1	[0.9188308, 0.784991, 0.1339925, 0.0947...	1M	[262.65457, -0...	697K
13	Heathens3-2	[0.4263677, 0.58840746, 0.61648256, 0.9...	1M	[200.14955, -0...	700K
14	Heathens3-3	[0.055413175, 0.32622638, 0.711635, 0.6...	1M	[268.74567, -0...	700K
15	Heathens4	[0.38927892, 0.363901, 0.1504497, 0.196...	1M	[243.32878, -0...	701K
16	Heathens4-1	[0.6208855, 0.21314873, 0.27608928, 0.1...	1M	[242.60815, -0...	700K

Gambar 3.7 Tabel Modul *Cover Song* pada *database*

3.2.2. Perancangan Data Latih

Perancangan data latih dalam penelitian pada tugas akhir ini berasal dari lagu yang terdapat pada YouTube. Video lagu pada situs tersebut akan dilakukan proses *convert* menjadi *file* musik dengan format .wav. *File* musik .wav inilah yang akan digunakan sebagai data latih. Namun *file* .wav tidak disimpan pada *database*, melainkan hanya fitur yang sesuai dengan masing-masing modul.



Gambar 3.8 Perancangan dalam data latih

Gambar 3.8 merupakan perancangan data latih dan data uji. Data latih akan dipotong 1 menit secara merata sesuai dengan durasi total dari musik tersebut. Data uji akan diambil berupa 30 detik terakhir dari masing-masing data latih. Apabila terdapat *overlapping* dalam durasi data latih, maka sisa dari waktu *overlap* akan dilakukan pemotongan lagu yang berdurasi 1 menit terakhir dari waktu *overlap*. Misal terdapat lagu berdurasi 4 menit 10 detik, maka akan terdapat 4 potongan lagu berdurasi 1 menit. Namun masih terdapat 10 detik waktu *overlap*, maka akan dipotong lagi satu bagian lagu dengan durasi 1 menit terhitung dari 10 detik terakhir berjalan mundur ke belakang.

3.2.3. Perancangan Tampilan Antarmuka

Subbab ini menjelaskan bagaimana rancangan antarmuka yang akan berinteraksi secara langsung dengan pengguna pada saat tahap implementasi.

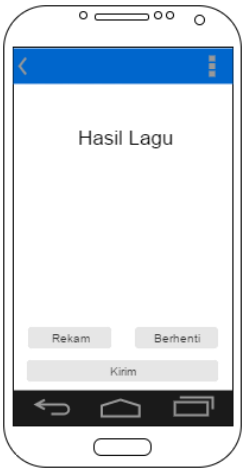
3.2.3.1. Perancangan Halaman Menu Utama

Pada halaman ini terdapat 3 tombol utama, yaitu tombol "*Fingerprint*", "*Cover Song*", dan "Detail Lagu". Tombol "*Fingerprint*" akan menampilkan halaman untuk merekam audio dengan hasil berupa judul lagu. Tombol "*Cover Song*" akan menampilkan halaman untuk merekam audio dengan hasil berupa judul lagu asli dari potongan lagu *cover* yang direkam. Pada saat tombol "Detail Lagu" ditekan akan menampilkan halaman untuk merekam dengan pada modul lain. Detail dari rancangan penjelasan tersebut dapat dilihat pada Gambar 3.9.

Gambar 3.9 Rancangan pada Menu Utama

3.2.3.2. Perancangan Halaman Merekam *Audio*

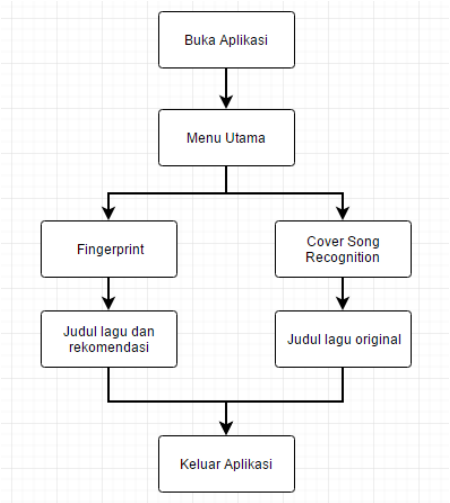
Pada halaman ini terdapat 3 tombol utama, yaitu tombol "Rekam", "Berhenti", dan "Kirim". Tombol "Rekam" berfungsi untuk melakukan pemberitahuan kepada sistem bahwa pengguna memerintahkan untuk memulai proses perekaman. Tombol "Berhenti" berfungsi untuk melakukan pemberitahuan kepada sistem bahwa pengguna memerintahkan untuk menghentikan proses perekaman. Pada saat tombol "Berhenti" ditekan, sistem akan melakukan penulisan data rekaman dengan format .wav. Tombol "Kirim" berfungsi untuk melakukan pengunggahan dari audio yang telah direkam kepada server. Detail dari rancangan penjelasan tersebut dapat dilihat pada Gambar 3.10.



Gambar 3.10 Rancangan Halaman Rekam

3.2.4. Perancangan Alur Proses Penggunaan Aplikasi

Pada tahap ini akan dijelaskan mengenai rancangan alur proses penggunaan aplikasi yang digunakan sebagai acuan dalam pembangunan aplikasi. Sehingga dapat terlihat jelas antara *input*, *proses*, dan *output*. Alur penggunaan aplikasi akan dijelaskan pada Gambar 3.11.



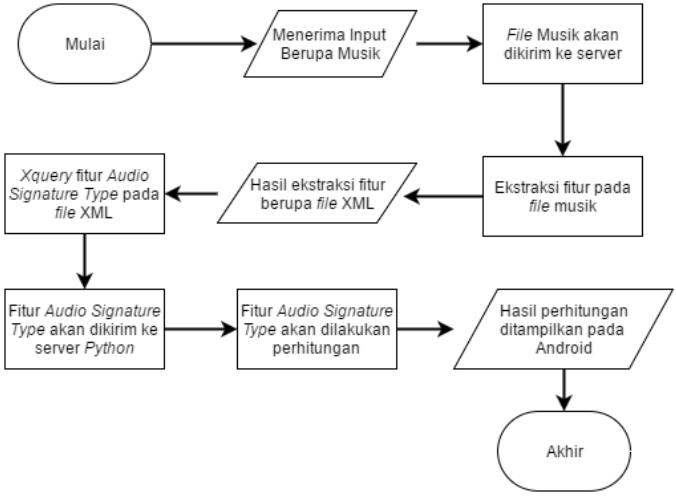
Gambar 3.11 Diagram Alur Penggunaan Aplikasi

3.2.4.1. Modul *Fingerprint*

Proses ini dilakukan dengan melakukan rekaman dari potongan suara berupa lagu asli. Potongan lagu yang direkam akan disimpan dalam penyimpanan lokal pada perangkat bergerak yang digunakan. Rekaman akan disimpan dalam ekstensi audio .wav. Ekstensi audio .wav akan dilakukan *upload* kepada server dan dilakukan perhitungan. Server akan mengembalikan nilai berupa tipe data *string* yang berisi tentang judul lagu dari potongan lagu yang dicari.

Pada Gambar 3.12 merupakan diagram alur dari modul *fingerprint*. Pertama aplikasi Android akan merekam musik dari lingkungan sekitar. Hasil dari proses rekam aplikasi Android akan menghasilkan *file* berupa format .wav. *File* tersebut akan dikirim pada server untuk dilakukan ekstraksi fitur. Proses ekstraksi fitur terjadi pada server Java dan menghasilkan dokumen XML sesuai dengan standar ISO MPEG-7. Xquery akan diaplikasikan pada dokumen XML untuk diambil fitur yang

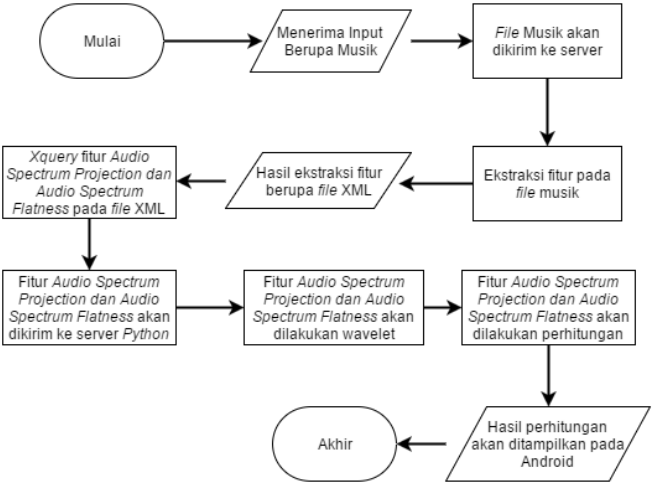
sesuai dengan modul *fingerprint*. Fitur yang sesuai dengan modul *fingerprint* adalah fitur *Audio Signature Type*. Fitur ini digunakan dalam perhitungan pada server Python untuk mencari kemiripan sebuah musik. Hasil dari perhitungan akan ditampilkan pada aplikasi Android berupa judul dari lagu yang direkam.



Gambar 3.12 Diagram Alur Modul *Fingerprint*

3.2.4.2. Modul *Cover Song Recognition*

Proses pendeteksi *cover song* memiliki kemiripan dengan modul *fingerprint*. Potongan audio dari lagu akan didengarkan oleh aplikasi Android dan di-*upload* ke server. Perbedaan hanya pada tahapan *pre-processing* pada server. Modul *cover song recognition* mengaplikasikan *Discrete Wavelet Transform* pada tahapan *pre-processing*. Setelah tahapan perhitungan dan klasifikasi selesai, server akan mengembalikan *string* yang berisi informasi tentang judul asli dari potongan *cover song* yang diperdengarkan.

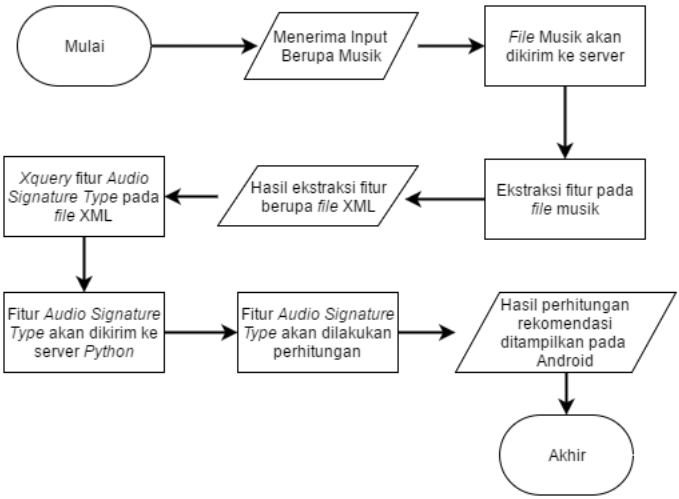


Gambar 3.13 Diagram Alur Modul *Cover Song Recognition*

Pertama aplikasi Android akan merekam musik dari lingkungan sekitar. Hasil dari proses rekam aplikasi Android akan menghasilkan *file* berupa format .wav. *File* tersebut akan dikirim pada server untuk dilakukan ekstraksi fitur. Proses ekstraksi fitur terjadi pada server Java dan menghasilkan dokumen XML sesuai dengan standar ISO MPEG-7. Xquery akan diaplikasikan pada dokumen XML untuk diambil fitur yang sesuai dengan modul *cover song recognition*. Fitur yang sesuai dengan modul *cover song recognition* adalah fitur *Audio Spectrum Projection* dan *Audio Spectrum Flatness*. Kedua fitur ini digunakan dalam perhitungan pada server Python untuk mencari kemiripan sebuah musik dengan data latih yang terdapat dalam *database*. Hasil dari perhitungan akan ditampilkan pada aplikasi Android berupa judul lagu asli dari lagu yang direkam. Diagram alur dari proses ini dapat dilihat pada Gambar 3.13.

3.2.4.3. Modul *Song Recommendation*

Proses pemberian rekomendasi pada lagu akan mengikuti pada modul *fingerprint*. Sebab modul *song recommendation* merupakan modul yang *include* pada modul induk, yaitu *fingerprint*. Salah satu *output* dari modul *fingerprint* selain lagu asli adalah, modul *song recommendation*. Modul *song recommendation* akan memberikan lagu rekomendasi yang sesuai berdasarkan nilai kemiripan potongan lagu yang dicari pada saat ini.



Gambar 3.14 Diagram Alur Modul *Song Recommendation*

Pertama aplikasi Android akan merekam musik dari lingkungan sekitar. Hasil dari proses rekam aplikasi Android akan menghasilkan *file* berupa format .wav. *File* tersebut akan dikirim pada server untuk dilakukan ekstraksi fitur. Proses ekstraksi fitur terjadi pada server Java dan menghasilkan dokumen XML sesuai dengan standar ISO MPEG-7. Xquery akan diaplikasikan pada dokumen XML untuk diambil fitur yang sesuai dengan modul *fingerprint*. Fitur yang sesuai dengan modul

fingerprint adalah fitur *Audio Signature Type*. Fitur ini digunakan dalam perhitungan pada server Python untuk mencari kemiripan sebuah musik. Hasil dari perhitungan akan ditampilkan pada aplikasi Android berupa *list* judul dari lagu yang direkam. Diagram alur dari penjelasan ini dapat dilihat pada Gambar 3.14.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari analisis dan perancangan sistem yang telah dibahas pada Bab III. Namun dalam penerapannya, rancangan tersebut dapat mengalami perubahan minor sewaktu-waktu apabila dibutuhkan.

4.1. Lingkungan Implementasi

Dalam implementasinya, lingkungan yang digunakan sama seperti yang dituliskan pada rancangan, yakni menggunakan beberapa perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam implementasi pengembangan aplikasi ini adalah Komputer dan sebuah perangkat bergerak berbasis Android. Spesifikasi komputer yang digunakan adalah Prosesor Intel® Core™ i3-CPU (3.30GHz), RAM 4 GB, Graphic Intel ® Sandybridge Desktop. Sedangkan spesifikasi dari perangkat bergerak yang digunakan adalah Oppo F1, OS Android v5.1 Lollipop Chipset Snapdragon 616 CPU, Quad-core 1.7 GHz & quad-core 1.0 GHz GPU, Adreno 405 RAM: 3 GB

4.1.2. Lingkungan Implementasi Perangkat Lunak

Penjelasan perangkat lunak yang digunakan dalam implementasi aplikasi ini adalah sebagai berikut:

- Linux Ubuntu 16.04 LTS sebagai *Operating System* computer server.
- MySQL untuk mengimplementasikan rancangan basis data.
- Android Studio sebagai IDE untuk mengembangkan aplikasi Android.

4.2. Implementasi Basis Data

Subbab ini membahas tentang implementasi basis data yang telah dirancang dan dibahas pada Bab III. Berikut merupakan contoh basis data yang telah dibuat.

Tabel 4.1 Fitur yang disimpan dalam *database*

Fitur	Modul	
	Fingerprint	*Cover Song Recognition*
Audio Power	X	X
Audio Waveform	X	X
Temporal Centroid	X	X
Log Attack Time	X	X
Audio Spectrum Centroid	X	X
Audio Harmonicity	V	X
Harmonic Spectral Centroid	X	X
Harmonic Spectral Deviation	X	X
Harmonic Spectral Spread	X	X
Harmonic Spectral Variaton	X	X
Audio Spectrum Spread	X	X
Audio Spectrum Projection	X	V
Audio Spectrum Basis	X	X
Audio Spectrum Flatness	X	V
Audio Spectrum Envelope	X	X
Audio Fundamental Frequency	X	X
Audio Signature	V	X

Keterangan:

X: Fitur yang tidak dipakai dalam penelitian.

V: Fitur yang dipakai dalam penelitian.

Tabel 4.1 mendeskripsikan fitur lagu yang akan disimpan kedalam *database*. Pemilihan fitur berdasarkan standar deskripsi fitur MPEG-7 yang telah dibahas pada subbab sebelumnya. Untuk modul *fingerprint*, fitur yang disimpan dalam *database* hanya *Audio Signature* yang merupakan ciri khas dari suatu audio. Untuk modul *cover song recognition*, fitur yang disimpan dalam *database* adalah *Audio Spectrum Projection* dan *Audio Spectrum Flatness*. *Audio Spectrum Projection* digunakan sebab menggambarkan karakteristik sinyal agar dapat dilakukan klasifikasi. *Audio Spectrum Flatness* digunakan sebab menggambarkan sifat kedataran pada suatu signal, sehingga dapat digunakan sebagai parameter seberapa mirip satu signal dengan yang lain.

4.3. Implementasi Tampilan Antarmuka

Pada halaman ini pengguna diminta untuk melakukan perekaman terhadap potongan audio. Potongan audio akan disimpan dalam bentuk .wav, namun pengekstrakan fitur tidak dilakukan pada perangkat bergerak, melainkan pada server. Hasil dari implementasi ini adalah *string* yang berisi informasi dari masing-masing modul. Tampilan antarmuka akan diimplementasikan pada aplikasi berbasis Android, sehingga dalam melakukan implementasi tampilan antarmuka akan dilakukan pada dokumen XML. Kode sumber yang digunakan dalam implementasi dari tampilan antarmuka dapat dilihat pada Kode Sumber 4.1. Hasil dari Kode Sumber 4.1 adalah tampilan aplikasi berbasis Android pada Gambar 4.1. Untuk implementasi halaman menu utama dapat dilihat pada Kode Sumber 4.2 dan hasil dari kode sumber tersebut dapat dilihat pada Gambar 4.2.

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:Android="http://schemas.Android
    .com/apk/res/Android"
```

```
3.      xmlns:tools="http://schemas.Android.com/tools"

4.      Android:layout_width="match_parent"
5.      Android:layout_height="match_parent"
6.      Android:paddingBottom="@dimen/activity_vertical
    _margin"
7.      Android:paddingLeft="@dimen/activity_horizontal
    _margin"
8.      Android:paddingRight="@dimen/activity_horizonta
    l_margin"
9.      Android:paddingTop="@dimen/activity_vertical_ma
    rgin"
10.     Android:orientation="vertical"
11.     tools:context="com.example.ponighzwa.musicmoov4
    .Rekam">
12.
13.     <TextView
14.         Android:layout_width="match_parent"
15.         Android:layout_height="wrap_content"
16.         Android:text="@string/app_info"
17.         Android:textSize="20dip"
18.         Android:textAlignment="center"
19.         Android:layout_weight="0.5"
20.         Android:id="@+id/hasil"/>
21.
22.     <ImageView
23.         Android:layout_width="match_parent"
24.         Android:layout_height="wrap_content"
25.         Android:src="@drawable/rekam"
26.         Android:layout_weight="2"
27.         Android:scaleType="fitCenter"/>
28.
29.     <LinearLayout
30.         Android:orientation="horizontal"
31.         Android:layout_width="fill_parent"
32.         Android:layout_height="wrap_content"
33.         >
34.
35.         <Button
36.             Android:layout_width="wrap_content"
37.             Android:layout_height="wrap_content"
38.             Android:id="@+id/btnStart"
```

```
39.             Android:text="@string/start_recording"

40.             Android:layout_weight="1.0"/>
41.
42.         <Button
43.             Android:layout_width="wrap_content"
44.             Android:layout_height="wrap_content"
45.             Android:id="@+id/btnStop"
46.             Android:text="@string/stop_recording"
47.             Android:layout_weight="1.0"/>
48.     </LinearLayout>
49.     <Button
50.         Android:layout_width="match_parent"
51.         Android:layout_height="wrap_content"
52.         Android:id="@+id/btnUpload"
53.         Android:text="Kirim"
54.         />
55. </LinearLayout>
```

Kode Sumber 4.1 Implementasi Halaman Rekam Audio

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout xmlns:android="http://schemas.android
    .com/apk/res/android"
3.      xmlns:tools="http://schemas.android.com/tools"
4.      android:layout_width="match_parent"
5.      android:layout_height="match_parent"
6.      android:paddingBottom="@dimen/activity_vertical
    _margin"
7.      android:paddingLeft="@dimen/activity_horizontal
    _margin"
8.      android:paddingRight="@dimen/activity_horizonta
    l_margin"
9.      android:paddingTop="@dimen/activity_vertical_ma
    rgin"
10.     android:orientation="vertical"
11.     tools:context="com.example.ponighzwa.musicmoov4
    .MainActivity">
12.     <Button
13.         android:layout_width="wrap_content"
14.         android:layout_height="wrap_content"
15.         android:id="@+id/signature"
```

```
16.         android:text="signature"
17.         android:layout_gravity="center"
18.         android:layout_margin="5dp"/>
19.     <Button
20.         android:layout_width="wrap_content"
21.         android:layout_height="wrap_content"
22.         android:id="@+id/cover"
23.         android:text="Cover Song"
24.         android:layout_gravity="center"
25.         android:layout_margin="5dp"/>
26.     <Button
27.         android:layout_width="wrap_content"
28.         android:layout_height="wrap_content"
29.         android:id="@+id/detail"
30.         android:text="Detail Lagu"
31.         android:layout_gravity="center"
32.         android:layout_margin="5dp"/>
33. </LinearLayout>
```

Kode Sumber 4.2 Implementasi Halaman Menu Utama

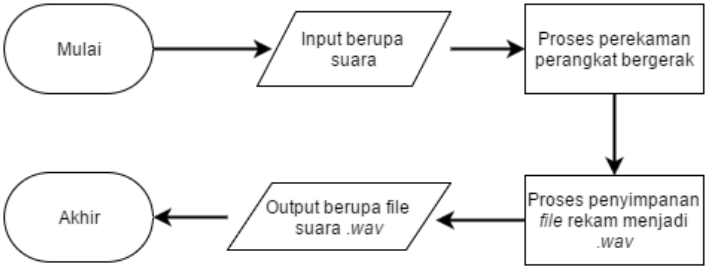
Gambar 4.1 Implementasi Tampilan Rekam Lagu pada Android

Gambar 4.2 Hasil Implemenasi Halaman Menu Utama

4.4. Implementasi Alur Proses Aplikasi

Pada subbab ini akan dibahas mengenai implementasi dari masing-masing proses alur aplikasi. Proses yang dimaksud adalah detail dari proses yang menerima *input* kepada sistem yaitu implementasi perekaman audio hingga hasil dari masing-masing modul yaitu implementasi modul *fingerprint*, *cover song recognition*, dan *song recommendation*. Detail dari masing-masing proses akan dijelaskan pada subbab berikut.

4.4.1. Implementasi Perekaman Audio



Gambar 4.3 Diagram Alur Proses Rekam

Gambar 4.3 merupakan diagram alur yang akan diimplementasikan pada proses rekam audio. Implementasi rekam audio akan menerima *input* suara dari lingkungan dan memberikan *output* berupa *file* rekam yang memiliki format .wav. Proses perekaman audio terjadi dalam aplikasi perangkat bergerak Android. Proses perekaman audio, menggunakan *samplerate* 44100 Hz dan *Encoding* PCM 16 Bit untuk mendapatkan hasil rekaman dengan kualitas terbaik. Hal ini sangat penting, sebab terdapat kemungkinan apabila sumber suara kurang bagus ketika diperdengarkan oleh aplikasi. Apabila sumber suara kurang baik dan dipertemukan dengan hasil rekaman yang normal, maka informasi suara yang diperdengarkan akan terdapat informasi yang hilang. Kode Sumber 4.3 menunjukkan variabel yang akan digunakan untuk proses perekaman audio, mulai dari *sample rate, audio encoding,* dan *audio channels*.

Alur proses pada proses perekaman audio adalah sebagai berikut. Ketika aplikasi pertama dibuka, akan menjalankan Kode Sumber 4.4. Kemudian, ketika pengguna ingin melakukan perekaman, fungsi *custom listener* (Kode Sumber 4.5) akan mengarahkan ke

fungsi *start recording* untuk memulai perekaman (Kode Sumber 4.6). Ketika fungsi *start recording* dijalankan, aplikasi akan menulis *raw* audio menjadi *file* yang dapat dibaca (Kode Sumber 4.7). Setelah audio direkam, pengguna akan memilih tombol "Berhenti" yang akan dibawa ke fungsi *stop recording* (Kode Sumber 4.8). Fungsi *stop recording* akan menggandakan data *temporary* menjadi data yang dapat dibaca oleh aplikasi (Kode Sumber 4.9). Data yang dibaca aplikasi tersebut, akan ditulis dengan ekstensi .wav (Kode Sumber 4.10).

```
1.  private static final int RECORDER_BPP = 16;
2.  private static final String AUDIO_RECORDER_FILE_EXT
    _WAV = ".wav";
3.  private static final String AUDIO_RECORDER_FOLDER =
    "AudioRecorder";
4.  private static final String AUDIO_RECORDER_TEMP_FIL
    E = "record_temp.raw";
5.  private static final int RECORDER_SAMPLERATE = 4410
    0;
6.  private static final int RECORDER_CHANNELS = AudioF
    ormat.CHANNEL_IN_STEREO;
7.  private static final int RECORDER_AUDIO_ENCODING =
    AudioFormat.ENCODING_PCM_16BIT;
8.
9.  private AudioRecord recorder = null;
10. private int bufferSize = 0;
11. private Thread recordingThread = null;
12. private boolean isRecording = false;
13.
14. public String selectedFilePath = "/storage/emulated
    /0/AudioRecorder/clip.wav";
15. private String SERVER_SIGNATURE = "http://10.151.64
    .169/MusicMoo/signature.php";
16. private String SERVER_COVER = "http://10.151.64.169
    /MusicMoo/cover.php";
17. private String SERVER_TEMPO = "http://10.151.64.169
    /MusicMoo/tempo.php";
18. private String SERVER_MOOD = "http://10.151.64.169/
    MusicMoo/mood.php";
```

```
19. private String SERVER_GENRE = "http://10.151.64.169
    /MusicMoo/genre.php";
20. private String SERVER_URL = "";
21. private static final String TAG = MainActivity.clas
    s.getSimpleName();
22. String response="";
```

Kode Sumber 4.3 Variabel yang digunakan untuk perekaman

```
1.  @Override
2.  protected void onCreate(Bundle savedInstanceState)
    {
3.      super.onCreate(savedInstanceState);
4.      setContentView(R.layout.activity_rekam);
5.
6.      setButtonHandlers();
7.      enableButtons(false);
8.
9.      String modul = getIntent().getStringExtra("modu
    l");
10.     if(modul.equals("1")){
11.         SERVER_URL = SERVER_SIGNATURE;
12.     }
13.     else if (modul.equals("2")){
14.         SERVER_URL = SERVER_COVER;
15.     }
16.     else if(modul.equals("3")){
17.         SERVER_URL = SERVER_TEMPO;
18.     }
19.     else if (modul.equals("4")){
20.         SERVER_URL = SERVER_MOOD;
21.     }
22.     else if(modul.equals("5")){
23.         SERVER_URL = SERVER_GENRE;
24.     }
25.     Toast.makeText(this,SERVER_URL,Toast.LENGTH_LON
    G).show();
26.
27.     bufferSize = AudioRecord.getMinBufferSize(8000,

28.             AudioFormat.CHANNEL_CONFIGURATION_MONO,
```

```
            AudioFormat.ENCODING_PCM_16BIT);
}
```

Kode Sumber 4.4 Fungsi yang kali pertama dijalankan ketika aplikasi dibuka

```
private View.OnClickListener btnClick = new View.On
ClickListener() {
    @Override
    public void onClick(View v) {
        switch(v.getId()){
            case R.id.btnStart:{
                AppLog.logString("Start Recording")
;

                enableButtons(true);
                startRecording();
                break;
            }
            case R.id.btnStop:{
                AppLog.logString("Stop Recording");
                enableButtons(false);
                stopRecording();
                Toast.makeText(Rekam.this,selectedF
ilePath,Toast.LENGTH_LONG).show();
                break;
            }
            case R.id.btnUpload:{

                AppLog.logString("Start Upload");
                new PostDataTOServer().execute();
                break;
            }
        }
    }
};
```

Kode Sumber 4.5 *Custom Listener* pada aplikasi

```
private void startRecording(){
```

```
2.      recorder = new AudioRecord(MediaRecorder.AudioS
    ource.MIC,
3.              RECORDER_SAMPLERATE, RECORDER_CHANNELS,
    RECORDER_AUDIO_ENCODING, bufferSize);
4.
5.      int i = recorder.getState();
6.      if(i==1)
7.          recorder.startRecording();
8.
9.      isRecording = true;
10.
11.     recordingThread = new Thread(new Runnable() {
12.
13.         @Override
14.         public void run() {
15.             writeAudioDataToFile();
16.         }
17.     },"AudioRecorder Thread");
18.
19.     recordingThread.start();
20. }
```

Kode Sumber 4.6 Fungsi ketika memulai perekaman

```
1.  private void writeAudioDataToFile(){
2.      byte data[] = new byte[bufferSize];
3.      String filename = getTempFilename();
4.      FileOutputStream os = null;
5.
6.      try {
7.          os = new FileOutputStream(filename);
8.      } catch (FileNotFoundException e) {
9.  ODO Auto-generated catch block
10.         e.printStackTrace();
11.     }
12.     int read = 0;
13.     if(null != os){
14.         while(isRecording){
15.             read = recorder.read(data, 0, bufferSiz
    e);
16.
```

```
17.             if(AudioRecord.ERROR_INVALID_OPERATION
    != read){
18.                 try {
19.                     os.write(data);
20.                 } catch (IOException e) {
21.                     e.printStackTrace();
22.                 }
23.             }
24.         }
25.
26.         try {
27.             os.close();
28.         } catch (IOException e) {
29.             e.printStackTrace();
30.         }
31.     }
32. }
```

Kode Sumber 4.7 Fungsi untuk melakukan *writing* data *audio* menjadi sebuah *file*

```
1.  private void stopRecording(){
2.      if(null != recorder){
3.          isRecording = false;
4.
5.          int i = recorder.getState();
6.          if(i==1)
7.              recorder.stop();
8.          recorder.release();
9.
10.         recorder = null;
11.         recordingThread = null;
12.     }
13.
14.     copyWaveFile(getTempFilename(),getFilename());

15.     deleteTempFile();
16. }
```

Kode Sumber 4.8 Fungsi yang dijalankan ketika berhenti merekam

```
1.  private void copyWaveFile(String inFilename,String
    outFilename){
2.      FileInputStream in = null;
3.      FileOutputStream out = null;
4.      long totalAudioLen = 0;
5.      long totalDataLen = totalAudioLen + 36;
6.      long longSampleRate = RECORDER_SAMPLERATE;
7.      int channels = 2;
8.      long byteRate = RECORDER_BPP * RECORDER_SAMPLER
    ATE * channels/8;
9.
10.     byte[] data = new byte[bufferSize];
11.
12.     try {
13.         in = new FileInputStream(inFilename);
14.         out = new FileOutputStream(outFilename);
15.         totalAudioLen = in.getChannel().size();
16.         totalDataLen = totalAudioLen + 36;
17.
18.         AppLog.logString("File size: " + totalDataL
    en);
19.
20.         WriteWaveFileHeader(out, totalAudioLen, tot
    alDataLen,
21.                 longSampleRate, channels, byteRate)
    ;
22.
23.         while(in.read(data) != -1){
24.             out.write(data);
25.         }
26.
27.         in.close();
28.         out.close();
29.     } catch (FileNotFoundException e) {
30.         e.printStackTrace();
31.     } catch (IOException e) {
32.         e.printStackTrace();
33.     }
34. }
```

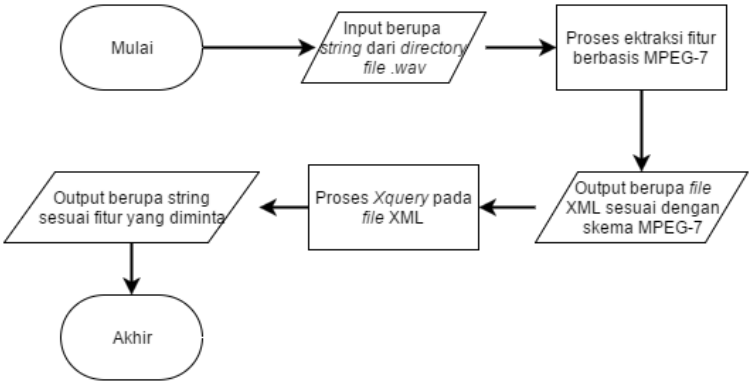
Kode Sumber 4.9 Fungsi untuk melakukan penulisan dari *file raw* menjadi .wav

```
1.  private void WriteWaveFileHeader(
2.          FileOutputStream out, long totalAudioLen,
3.          long totalDataLen, long longSampleRate, in
    t channels,
4.          long byteRate) throws IOException {
5.
6.      byte[] header = new byte[44];
7.      header[0] = 'R'; // RIFF/WAVE header
8.      header[1] = 'I';
9.      header[2] = 'F';
10.     header[3] = 'F';
11.     header[4] = (byte) (totalDataLen & 0xff);
12.     header[5] = (byte) ((totalDataLen >> 8) & 0xff
    );
13.     header[6] = (byte) ((totalDataLen >> 16) & 0xf
    f);
14.     header[7] = (byte) ((totalDataLen >> 24) & 0xf
    f);
15.     header[8] = 'W';
16.     header[9] = 'A';
17.     header[10] = 'V';
18.     header[11] = 'E';
19.     header[12] = 'f'; // 'fmt ' chunk
20.     header[13] = 'm';
21.     header[14] = 't';
22.     header[15] = ' ';
23.     header[16] = 16; // 4 bytes: size of 'fmt ' ch
    unk
24.     header[17] = 0;
25.     header[18] = 0;
26.     header[19] = 0;
27.     header[20] = 1; // format = 1
28.     header[21] = 0;
29.     header[22] = (byte) channels;
30.     header[23] = 0;
31.     header[24] = (byte) (longSampleRate & 0xff);
32.     header[25] = (byte) ((longSampleRate >> 8) & 0
    xff);
33.     header[26] = (byte) ((longSampleRate >> 16) &
    0xff);
34.     header[27] = (byte) ((longSampleRate >> 24) &
    0xff);
```

```
35.      header[28] = (byte) (byteRate & 0xff);
36.      header[29] = (byte) ((byteRate >> 8) & 0xff);
37.      header[30] = (byte) ((byteRate >> 16) & 0xff);
38.      header[31] = (byte) ((byteRate >> 24) & 0xff);
39.      header[32] = (byte) (2 * 16 / 8); // block ali
    gn
40.      header[33] = 0;
41.      header[34] = RECORDER_BPP; // bits per sample
42.      header[35] = 0;
43.      header[36] = 'd';
44.      header[37] = 'a';
45.      header[38] = 't';
46.      header[39] = 'a';
47.      header[40] = (byte) (totalAudioLen & 0xff);
48.      header[41] = (byte) ((totalAudioLen >> 8) & 0x
    ff);
49.      header[42] = (byte) ((totalAudioLen >> 16) & 0
    xff);
50.      header[43] = (byte) ((totalAudioLen >> 24) & 0
    xff);
51.      out.write(header, 0, 44);
52.  }
```

Kode Sumber 4.10 Fungsi yang menuliskan format agar *file* dapat menjadi .wav

4.4.2. Implementasi Ekstraksi Fitur



Gambar 4.4 Diagram Alur Proses Ekstraksi Fitur

Gambar 4.4 merupakan diagram alur mengenai proses ekstraksi fitur. Proses ini menerima *input* berupa *file* musik dengan format .wav dan menghasilkan *output* berupa nilai dari fitur sesuai dengan modul yang diminta. Proses implementasi ekstraksi fitur terjadi pada sisi server Java. Masing-masing kelas akan dijalankan ketika *file routes* dipanggil sesuai dengan *address* pada *file browser* maupun server lain. *File routes* pada server Java dapat dilihat pada Kode Sumber 4.11. Modul *fingerprint* membutuhkan *routes* "/*Descriptor/*:nama". Modul *cover song recognition* membutuhkan *routes "*/*Projection* /:nama*"* dan *routes* "/*Flatness*/:nama". Atribut ":nama" pada akhir dari setiap *routes* menunjukkan *input* yang diterima dari setiap *routes*. Ketika suatu *routes* dipanggil melalui server maka akan menjalankan kelasnya masing-masing. Secara umum setiap kelas pada server Java memiliki tujuan untuk melakukan ekstraksi fitur pada *file* musik dengan format .wav menjadi format dokumen XML sesuai dengan standar MPEG-7. Namun perbedaan yang

paling utama adalah Xquery yang diimplementasikan pada dokumen XML. Kelas yang diimplementasikan untuk modul *fingerprint* terdapat pada Kode Sumber 4.12. Kelas yang diimplementasikan untuk modul *cover song recognition* terdapat pada Kode Sumber 4.13 dan Kode Sumber 4.14. Pada Kode Sumber 4.12 fitur yang dilakukan ekstraksi adalah *Audio Signature Type*. Sedangkan pada Kode Sumber 4.13 dan Kode Sumber 4.14 fitur yang dilakukan ekstraksi adalah *Audio Spectrum Projection* dan *Audio Spectrum Flatness*. Detail dari Xquery yang diimplementasikan untuk dapat dilihat pada Kode Sumber 4.15, 4.16, dan 4.17. Kode Sumber 4.15 adalah Xquery yang dilakukan untuk mengambil fitur *Audio Signature Type*, yang akan digunakan pada modul *fingerprint*. Kode Sumber 4.16 dan 4.17 adalah Xquery yang dilakukan untuk mengambil fitur *Audio Spectrum Flatness* dan *Audio Spectrum Projection*, yang akan digunakan pada modul *cover song*. Hasil keseluruhan dari proses ekstraksi fitur ini adalah nilai *string* yang berisi tentang fitur yang digunakan pada masing-masing modul.

```
1. GET     /Descriptor/:nama           controllers.Descript
   or.index(nama: String)
2. GET     /Projection/:nama           controllers.Descript
   or.index(nama: String)
3. GET     /Flatness/:nama             controllers.Descript
   or.index(nama: String)
```

Kode Sumber 4.11 Routes pada Server Java

```
1.  public class Descriptor extends Controller {
2.
3.      public Result index(String nama){
4.          try {
5.
6.              String configSource = "C:/Users/ponighz
    wa/IdeaProjects/CobaRP/config/config.xml";
```

```
7.              InputStream inputStream = new FileInput
    Stream(configSource);
8.              Reader reader = new InputStreamReader(i
    nputStream);
9.              Config config = ConfigXML.parse(reader)
    ;
10.
11.             //Config config = new ConfigDefault();
12.             //config.enableAll(true);
13.
14.
15.             //System.out.println("1");
16.             String input = "C:/xampp/htdocs/musicmo
    o/uploads/"+nama;
17.             //System.out.println("2");
18.             File file = new File(input);
19.             //System.out.println("3");
20.             AudioInputStream audioInputStream = Aud
    ioSystem.getAudioInputStream(file);
21.             //System.out.println("4");
22.             Document mpeg7 = MP7DocumentBuilder.enc
    ode(audioInputStream,config);
23.             //coba baru
24.             DOMSource domSource = new DOMSource(mpe
    g7);
25.
26.             StringWriter stringWriter = new StringW
    riter();
27.             StreamResult result = new StreamResult(
    stringWriter);
28.
29.             TransformerFactory transformerFactory =
    TransformerFactory.newInstance();
30.             Transformer transformer = transformerFa
    ctory.newTransformer();
31.
32.             transformer.transform(domSource, result
    );
33.             String hasil = stringWriter.toString();
34.             //System.out.println("5");
35.             //file.delete();
```

```
36.             String output = "C:/xampp/htdocs/musicm
    oo/uploads/xml/"+nama.replace(".wav","")+".xml";
37.             File file1 = new File(output);
38.             FileWriter fileWriter = new FileWriter(
    file1);
39.             fileWriter.write(hasil);
40.             fileWriter.flush();
41.             fileWriter.close();
42.
43.             Extract extract = new Extract();
44.             String signature = extract.AudioSignatu
    reType(output);
45.             return ok(signature);
46.         } catch (IOException e) {
47.             System.out.println(e);
48.             return ok("error1");
49.         }catch (UnsupportedAudioFileException e){
50.             System.out.println(e);
51.             return ok("error2");
52.         } catch (ParserConfigurationException e){
53.             System.out.println(e);
54.             return ok("error3");
55.         } catch (TransformerException e){
56.             System.out.println("e");
57.             return ok("error4");
58.         } catch (SAXException e){
59.             System.out.println(e);
60.             return ok("error5");
61.         }
62.     }
63.
64. }
```

Kode Sumber 4.12 Kelas pada Java Server Modul *Fingerprint*

```
1.  public class Projection extends Controller {
2.
3.
4.      public Result index(String nama){
5.          try {
6.
```

```
7.              String configSource = "C:/Users/ponighz
    wa/IdeaProjects/CobaRP/config/config.xml";
8.              InputStream inputStream = new FileInput
    Stream(configSource);
9.              Reader reader = new InputStreamReader(i
    nputStream);
10.             Config config = ConfigXML.parse(reader)
    ;
11.
12.             //Config config = new ConfigDefault();

13.             //config.enableAll(true);
14.
15.
16.             //System.out.println("1");
17.             String input = "C:/xampp/htdocs/musicmo
    o/uploads/"+nama;
18.             //System.out.println("2");
19.             File file = new File(input);
20.             //System.out.println("3");
21.             AudioInputStream audioInputStream = Aud
    ioSystem.getAudioInputStream(file);
22.             //System.out.println("4");
23.             Document mpeg7 = MP7DocumentBuilder.enc
    ode(audioInputStream,config);
24.             //coba baru
25.             DOMSource domSource = new DOMSource(mpe
    g7);
26.
27.             StringWriter stringWriter = new StringW
    riter();
28.             StreamResult result = new StreamResult(
    stringWriter);
29.
30.             TransformerFactory transformerFactory =
    TransformerFactory.newInstance();
31.             Transformer transformer = transformerFa
    ctory.newTransformer();
32.
33.             transformer.transform(domSource, result
    );
34.             String hasil = stringWriter.toString();
```

```
            //System.out.println("5");
            //file.delete();
            String output = "C:/xampp/htdocs/musicmoo/uploads/xml/"+nama.replace(".wav","")+".xml";
            File file1 = new File(output);
            FileWriter fileWriter = new FileWriter(file1);
            fileWriter.write(hasil);
            fileWriter.flush();
            fileWriter.close();

            Extract extract = new Extract();
            String signature = extract.AudioSpectrumProjection(output);
            return ok(signature);
        } catch (IOException e) {
            System.out.println(e);
            return ok("error1");
        }catch (UnsupportedAudioFileException e){
            System.out.println(e);
            return ok("error2");
        } catch (ParserConfigurationException e){
            System.out.println(e);
            return ok("error3");
        } catch (TransformerException e){
            System.out.println("e");
            return ok("error4");
        } catch (SAXException e){
            System.out.println(e);
            return ok("error5");
        }
    }

}
```

Kode Sumber 4.13 Kelas pada Java Server Modul *Cover Song*

```
public class Flatness extends Controller {


    public Result index(String nama){
        try {
```

```
6.
7.              String configSource = "C:/Users/ponighzwa/IdeaProjects/CobaRP/config/config.xml";
8.              InputStream inputStream = new FileInputStream(configSource);
9.              Reader reader = new InputStreamReader(inputStream);
10.             Config config = ConfigXML.parse(reader);
11.
12.             //Config config = new ConfigDefault();
13.             //config.enableAll(true);
14.
15.
16.             //System.out.println("1");
17.             String input = "C:/xampp/htdocs/musicmoo/uploads/"+nama;
18.             //System.out.println("2");
19.             File file = new File(input);
20.             //System.out.println("3");
21.             AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(file);
22.             //System.out.println("4");
23.             Document mpeg7 = MP7DocumentBuilder.encode(audioInputStream,config);
24.             //coba baru
25.             DOMSource domSource = new DOMSource(mpeg7);
26.
27.             StringWriter stringWriter = new StringWriter();
28.             StreamResult result = new StreamResult(stringWriter);
29.
30.             TransformerFactory transformerFactory = TransformerFactory.newInstance();
31.             Transformer transformer = transformerFactory.newTransformer();
32.
33.             transformer.transform(domSource, result);
```

```
34.             String hasil = stringWriter.toString();
35.             //System.out.println("5");
36.             //file.delete();
37.             String output = "C:/xampp/htdocs/musicm
    oo/uploads/xml/"+nama.replace(".wav","")+".xml";
38.             File file1 = new File(output);
39.             FileWriter fileWriter = new FileWriter(
    file1);
40.             fileWriter.write(hasil);
41.             fileWriter.flush();
42.             fileWriter.close();
43.
44.             Extract extract = new Extract();
45.             String signature = extract.AudioSpectru
    mFlatness(output);
46.             return ok(signature);
47.         } catch (IOException e) {
48.             System.out.println(e);
49.             return ok("error1");
50.         }catch (UnsupportedAudioFileException e){
51.             System.out.println(e);
52.             return ok("error2");
53.         } catch (ParserConfigurationException e){
54.             System.out.println(e);
55.             return ok("error3");
56.         } catch (TransformerException e){
57.             System.out.println("e");
58.             return ok("error4");
59.         } catch (SAXException e){
60.             System.out.println(e);
61.             return ok("error5");
62.         }
63.     }
64.
65. }
```

Kode Sumber 4.14 Kelas pada Java Server Modul *Cover Song*

```
1.  private static String AudioSignatureType(String str
    ing) throws BaseXException{
2.
```

```
        String query =
                "declare default element namespace \"urn:mpeg:mpeg7:schema:2001\";" +
                        "declare namespace mpeg7 = \"urn:mpeg:mpeg7:schema:2001\";" +
                        "declare namespace xsi = \"http://www.w3.org/2001/XMLSchema-instance\";" +
                        "for $x in doc(\"C:/Users/ponighzwa/Desktop/wav/Testing/XML/"+ string+".xml\")/Mpeg7/Description/MultimediaContent/Audio/" +
                        "AudioDescriptionScheme return if($x/@xsi:type=\"AudioSignatureType\")then data($x/Flatness/SeriesOfVector/Mean) else \"\"";
        String hasil = new XQuery(query).execute(context);
        return hasil;
        //System.out.println(hasil);
    }
```

Kode Sumber 4.15 Xquery pada Modul *Fingerprint*

```
private static String AudioSpectrumFlatnessType(String string) throws BaseXException{

        String query =
                "declare default element namespace \"urn:mpeg:mpeg7:schema:2001\";" +
                        "declare namespace mpeg7 = \"urn:mpeg:mpeg7:schema:2001\";" +
                        "declare namespace xsi = \"http://www.w3.org/2001/XMLSchema-instance\";" +
                        "for $x in doc(\"C:/Users/ponighzwa/Desktop/CoverSong2/Test/XML/"+ string+".xml\")/Mpeg7/Description/MultimediaContent/" +
                        "Audio/AudioDescriptor\n return if($x/@xsi:type=\"AudioSpectrumFlatnessType\") then data($x/SeriesOfVector/Raw) else \"\"";

        //System.out.println(new XQuery(query).execute(context));
        String hasil = new XQuery(query).execute(context);
```

```
        return hasil;
    }
```

Kode Sumber 4.16 *Xquery* pada Modul *Cover Song*

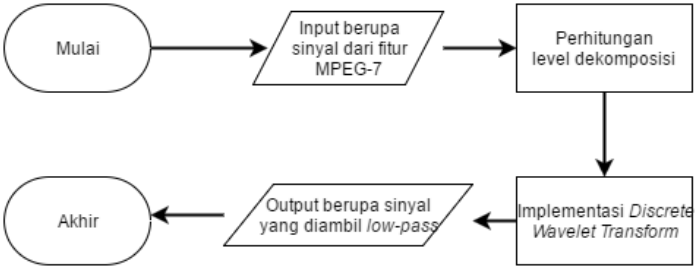
```
private static String AudioSpectrumProjectionType(String string) throws BaseXException{

        String query =
                "declare default element namespace \"urn:mpeg:mpeg7:schema:2001\";" +
                "declare namespace mpeg7 = \"urn:mpeg:mpeg7:schema:2001\";" +
                "declare namespace xsi = \"http://www.w3.org/2001/XMLSchema-instance\";" +
                "for $x in doc(\"C:/Users/ponighzwa/Desktop/yosua/xml/kucing/data traine/"+ string+".xml\")/Mpeg7/Description/MultimediaContent/" +
                "Audio/AudioDescriptor\n return if($x/@xsi:type=\"AudioSpectrumProjectionType\")then data($x/SeriesOfVector/Raw) else \"\"";

        String hasil = new XQuery(query).execute(context);
        return hasil;
    }
```

Kode Sumber 4.17 Xquery pada Modul *Cover Song*

4.4.3. Implementasi *Discrete Wavelet Transform*



Gambar 4.5 Diagram Alur Proses *Discrete Wavelet Transform*

Gambar 4.5 merupakan diagram alur proses yang terjadi pada saat implementasi proses *Discrete Wavelet Transform*. Proses ini menerima *input* berupa fitur yang telah dilakukan ekstraksi fitur dalam bentuk variabel *list* pada bahasa pemrograman Python dan memberikan *output* berupa sinyal yang telah diambil *low-pass filter* dari metode *Wavelet*. Implementasi *Discrete Wavelet Transform* terjadi pada sisi server dan dijalankan ketika menerima sinyal dalam bentuk *list* satu dimensi pada Python. Sebelum melakukan implementasi *wavelet* pada sinyal, sebelumnya perlu menentukan level dekomposisi yang sesuai. Untuk melakukan level dekomposisi yang sesuai, sinyal akan diolah oleh fungsi "*frekuensi"* (Kode Sumber 4.18) dan hasilnya akan digunakan *input* oleh fungsi "*nilaiDekomposisi*". Sehingga *output* dari process ini adalah nilai *integer* yang merupakan level dekomposisi yang tepat pada sinyal tersebut. Setelah itu, sinyal yang sama akan diimplementasikan *wavelet* dengan level dekomposisi yang sama dengan nilai *integer output* dari fungsi "*frekuensi*". Setelah itu nilai level dekomposisi akan dijadikan sebaga parameter pada Kode Sumber 4.19 untuk mengaplikasikan *Discrete Wavelet Transform* sehingga

didapatkan informasi sinyal tanpa merusak informasi dari sinyal asli.

```
def nilaiDekomposisi(fre):
    if (fre > 512):
        return 0
    elif(256 <= fre and fre <= 512):
        return 1
    elif(128 <= fre and fre <= 256):
        return 2
    elif(64 <= fre and fre <= 128):
        return 3
    elif(32 <= fre and fre <= 64):
        return 4
    elif(16 <= fre and fre <= 32):
        return 5
    elif(8 <= fre and fre <= 16):
        return 6
    elif(4 <= fre and fre <= 8):
        return 7
    elif(2 <= fre and fre <= 4):
        return 8
    elif(1 <= fre and fre <= 2):
        return 9
    elif(0.5 <= fre and fre <= 1):
        return 10
    elif(0.25 <= fre and fre <= 0.5):
        return 11
    elif(0.125 <= fre and fre <= 0.25):
        return 12
    elif(0.0625 <= fre and fre <= 0.125):
        return 13

def frekuensi(arr):
    npArr = np.array(arr)
    mean = np.mean(npArr)
    fin = npArr - mean

    hasil = max(abs(np.fft.fft(fin.transpose(), axis = 0) ) )
    fft = np.array(abs(np.fft.fft(fin.transpose(), axis=0 )))
```

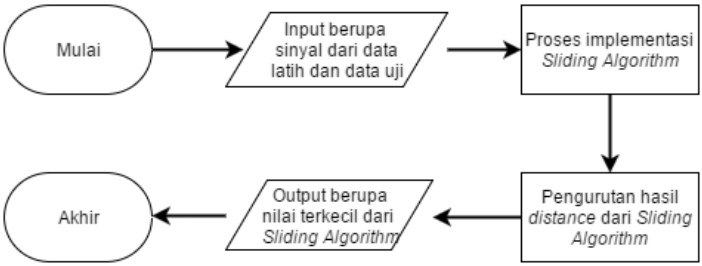
```
38.     maxIndex = np.where(fft == hasil)[0][0]
39.
40.     fre = float(maxIndex) * 1024 / len(arr)
41.
42.     return nilaiDekomposisi(fre)
```

Kode Sumber 4.18 Menentukan level dekomposisi *wavelet*

```
1.  def waveletTransform(lagu, n):
2.      w = pywt.Wavelet('bior2.8')
3.      maks = pywt.dwt_max_level(data_len=len(lagu), f
    ilter_len=w.dec_len)
4.      if(n > maks):
5.          n = maks
6.      hasil = pywt.wavedec(lagu, "bior2.8", level = n
    )
7.
8.      return hasil[0]
```

Kode Sumber 4.19 *Wavelet* Transform

4.4.4. Implementasi *Sliding Algorithm*



Gambar 4.6 Diagram Alur pada *Sliding Algorithm*

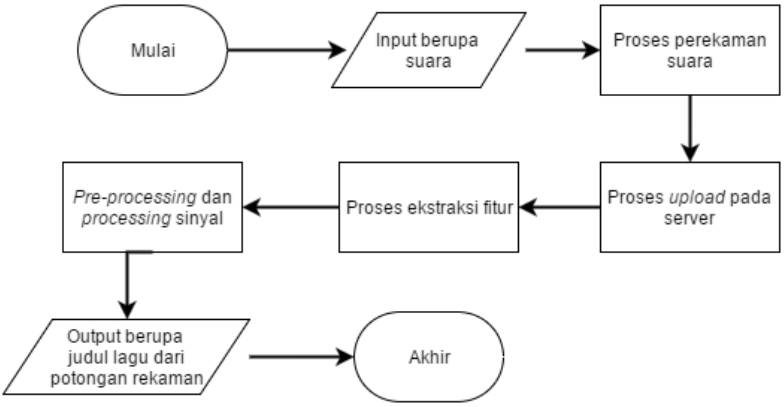
Gambar 4.6 merupakan diagram alur dari implementasi *Sliding Algorithm*. Implementasi ini menerima 2 *input* yaitu sinyal dari data latih dan data uji. *Output* dari proses ini adalah nilai terkecil dari *Sliding*

Algorithm. Implementasi *Sliding Algorithm* terletak pada sisi server dan dijalankan ketika menerima *input* 2 sinyal yang akan dibandingkan. Perhitungan pada *Sliding Algorithm*, menggunakan perhitungan *Euclidian Distance* pada setiap *subband* dari sinyal. *Output* dari fungsi *sliding* (Kode Sumber 4.20) adalah nilai kesamaan terkecil dari beberapa *subband* yang telah dibandingkan. Nilai kesamaan terkecil ini diasumsikan sebagai nilai kesamaan suatu sinyal terhadap sinyal lain.

```
1.  def sliding(fulllagu, potongan):
2.      distance = []
3.      i = 0
4.      k = 0
5.      n = len(fulllagu) / len(potongan)
6.      while (i < n):
7.          j = 0
8.          jumlah = 0.0
9.          while(j < len(potongan)):
10.             jumlah += (potongan[j] - fulllagu[k]) *
    (potongan[j] - fulllagu[k])
11.             j = j+1
12.             k = k+1
13.         distance.append(jumlah)
14.         i = i +1
15.
16.     n = len(fulllagu) % len(potongan)
17.     if(n > len(potongan)/2):
18.         i = 0
19.         jumlah = 0.0
20.         while (i < n):
21.             jumlah += (potongan[i] - fulllagu[k]) *
    (potongan[i] - fulllagu[k])
22.             i = i + 1
23.             k = k + 1
24.
25.         distance.append(jumlah)
26.     return min(distance)
```

Kode Sumber 4.20 *Sliding Algorithm*

4.4.5. Implementasi Modul *Fingerprint*



Gambar 4.7 Diagram Alur pada Modul *Fingerprint*

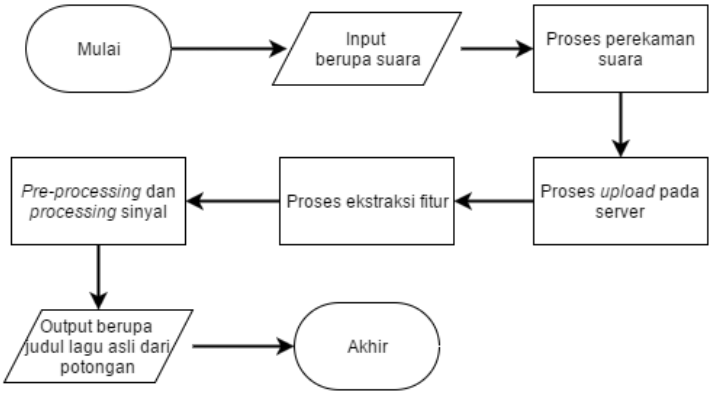
Gambar 4.7 merupakan diagram alur pada modul *fingerprint*. Implementasi pada modul ini menerima *input* berupa suara dan menghasilkan *output* berapa judul asli dari potongan rekaman lagu. Implementasi modul *fingerprint* terletak pada sisi komputer server. Modul *fingerprint* akan dijalankan ketika *route* "/slidingSignature/" akan dijalankan (Kode sumber 4.21). Modul *fingerprint* akan menerima *input* dari SQL *database* dan melakukan perhitungan. Perhitungan dilakukan dengan cara *Sliding Algorithm* dan menerima beberapa nilai. Nilai tersebut akan diurutkan dari yang terkecil hingga terbesar. Nilai terkecil dianggap sebagai lagu yang dicari.

```
1.  @app.route("/slidingSignature/")
2.  def slidingSignature():
3.      conn = MySQL.connect()
4.      cursor = conn.cursor()
5.
6.      cursor.execute('Select * from Lagu')
```

```
7.      conn.commit()
8.      data1 = cursor.fetchall() #isinya semua data
9.
10.     cursor.execute('Select * from Testing')
11.     conn.commit()
12.     testing = cursor.fetchall()
13.
14.     sinyalT = stringToList(testing[0][1])
15.
16.     sinyalTraining = []
17.     i = 0
18.     j = len(data1)
19.
20.     while(i < j):
21.         sinyal = stringToList(data1[i][2])
22.         #level = frekuensi(sinyal)
23.         hasil = (data1[i][1],sliding(sinyal, sinyal
    T))
24.         sinyalTraining.append(hasil)
25.         i = i+1
26.
27.     classifier = sorted(sinyalTraining, key=lambda
    x:x[1])
28.     i = 0
29.     ret = ""
30.     while(i < 5):
31.         ret += classifier[i][0] + " "
32.         #ret += classifier[i][0] + " "  +str(classi
    fier[i][1]) + "<br>"
33.         i = i+1
34.     return ret
```

Kode Sumber 4.21 Implementasi Modul *Fingerprint*

4.4.6. Implementasi Modul *Cover Song Recognition*



Gambar 4.8 Diagram Alur pada Modul *Cover Song Recognition*

Gambar 4.8 merupakan diagram alur pada modul *cover song recognition*. Implementasi pada modul ini menerima *input* berupa suara dan menghasilkan *output* berapa judul asli dari rekaman lagu *cover*. Implementasi modul *cover song recognition* terletak pada sisi komputer server. Modul *cover song recognition* akan dijalankan ketika *route* "/slidingCoverSong/" akan dijalankan (Kode Sumber 4.22). Modul *cover song recognition* akan menerima *input* dari SQL *database* dan melakukan perhitungan. Perhitungan dilakukan dengan cara implementasi metode *Wavelet*, kemudian hasil dari *wavelet* akan dilakukan *Sliding Algorithm*. Nilai *wavelet* akan diambil hanya aproksimasinya saja dan kemudian dilakukan perhitungan. Nilai hasil dari *Sliding Algorithm* akan diurutkan dari yang terkecil hingga terbesar. Nilai terkecil dianggap sebagai lagu asli yang dicari.

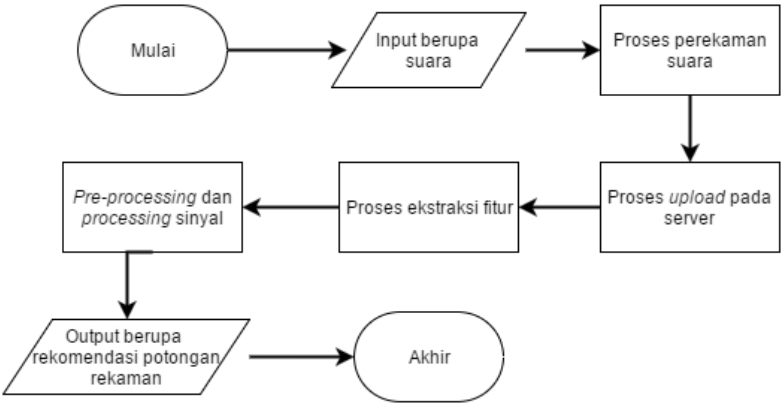
```
1.  @app.route("/slidingCoverSong/")
2.  def slidingCoverSong():
```

```
3.      conn = MySQL.connect()
4.      cursor = conn.cursor()
5.
6.      cursor.execute('Select * from CoverSong')
7.      conn.commit()
8.      data1 = cursor.fetchall() #isinya semua data
9.
10.     cursor.execute('Select * from testC')
11.     conn.commit()
12.     testing = cursor.fetchall()
13.
14.     sinyalT = []
15.     sinyalT.append(stringToList(testing[0][1]))
16.     sinyalT.append(stringToList(testing[0][2]))
17.
18.
19.     sinyalTraining = []
20.     i = 0
21.     j = len(data1)
22.
23.     while(i < j):
24.         sinyal = []
25.         sinyal.append(stringToList(data1[i][2]))
26.         sinyal.append(stringToList(data1[i][3]))
27.         levelf = frekuensi(sinyal[0])
28.         levelp = frekuensi(sinyal[1])
29.         hasil= (data1[i][1],math.sqrt(sliding(wv.wa
    veletTransform(sinyal[0],levelf),wv.waveletTransfor
    m(sinyalT[0],levelf))+sliding(wv.waveletTransform(s
    inyal[1],levelp),wv.waveletTransform(sinyalT[1],lev
    elp))))
30.         sinyalTraining.append(hasil)
31.         i = i+1
32.
33.     classifier = sorted(sinyalTraining, key=lambda
    x:x[1])
34.     i = 0
35.     ret = ""
36.     while(i < 10):
37.         ret += classifier[i][0] + " " +str(classifi
    er[i][1]) + "<br>"
38.         #ret += classifier[i][0] + " "  +str(classi
    fier[i][1]) + "<br>"
```

```
39.         i = i+1
40.
41.
42.     return ret
```

Kode Sumber 4.22 Implementasi Modul *Cover Song Recognition*

4.4.7. Implementasi Modul *Song Recommendation*



Gambar 4.9 Diagram Alur Modul *Song Recommendation*

Gambar 4.9 merupakan diagram alur pada modul *song recommendation*. Implementasi pada modul ini menerima *input* berupa suara dan menghasilkan *output* berupa judul lagu yang menjadi rekomendasi berdasarkan potongan rekaman lagu saat ini. Modul *song recommendation* merupakan modul yang *include* dari modul *fingerprint*. Pada Modul *fingerprint*, nilai kemiripan dikembalikan lebih dari satu. Nilai terkecil diasumsikan sebagai potongan lagu yang dicari dan menjadi *output* pada modul tersebut. Modul *song recommendation* akan memilih lagu terkecil kedua yang dijadikan rekomendasi. Pemilihan tersebut dinilai sebagai kemiripan yang kedua dari lagu yang sedang dicari, sehingga dapat dijadikan sebagai

rekomendasi. Implementasi dari modul ini dapat dilihat pada Kode Sumber 4.23.

```
while(i < j):
        sinyal = stringToList(data1[i][2])
        #level = frekuensi(sinyal)
        hasil = (data1[i][1],sliding(sinyal, sinyalT))
        sinyalTraining.append(hasil)
        i = i+1

    classifier = sorted(sinyalTraining, key=lambda x:x[1])
    i = 0
    ret = ""
    while(i < 5):
        ret += classifier[i][0] + " "
        #ret += classifier[i][0] + " "  +str(classifier[i][1]) + "<br>"
        i = i+1
    return ret
```

Kode Sumber 4.23 Implementasi Modul *Song Recommendation*

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem yang telah dijabarkan pada Bab III dan terhadap tujuan dibuatnya aplikasi ini, yakni melakukan implementasi sesuai dengan modul-modul yang telah dijabarkan sebelumnya.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan server sebagai berikut:

Prosesor : Prosesor Intel® Core™ i3-CPU
RAM : 4 GB
Jenis *Device* : Personal Computer
Sistem Operasi : Ubuntu 16.04 LTS.

5.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pada modul *fingerprint* skenario pengujian terdiri dari 2 langkah, yaitu pengujian dengan data uji diacak dari musik dan pengujian dengan data uji yang dilakukan sesuai pada desain Bab III subbab Perancangan Data Latih. Modul *cover song recognition* juga memiliki 2 skenario dalam pengujian, yaitu pengujian pertama menggunakan data uji berupa musik yang dinyanyikan laki-laki dan pengujian kedua dengan musik yang dinyanyikan perempuan. Rincian dari skenario pengujian disajikan pada Tabel 5.1.

Tabel 5.1 Skenario Pengujian

Kode Pengujian	Scenario Pengujian
SP-UC1	Pengujian Modul *fingerprint*
SP-UC2	Pengujian Modul *cover song*
SP-UC3	Pengujian Modul *song recommendation*

5.2.1. Pengujian Fungsionalitas

Pengujian fungsionalitas aplikasi dilakukan secara mandiri dengan melakukan skenario yang sama dengan rancangan alur proses aplikasi sebagai tolok ukur keberhasilan pengujian, dan mengacu pada kasus penggunaan yang sebelumnya telah dijelaskan pada Bab III. Pengujian pada kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut.

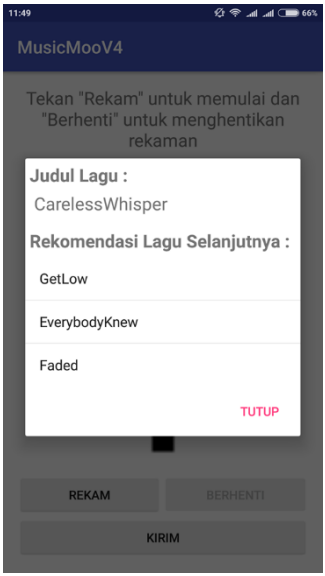
5.2.1.1. Pengujian Modul *Fingerprint*

Pengujian modul *fingerprint* dimulai ketika pengguna melakukan perekaman pada suatu sumber suara. Perekaman dilakukan pada perangkat bergerak Android yang telah dilakukan instalasi aplikasi tugas akhir ini. Aplikasi akan mengeluarkan nilai *string* yang mengandung informasi dari judul lagu yang dicari. Rincian dari pengujian modul dapat dilihat pada Tabel 5.2. Hasil dari modul *fingerprint* pada aplikasi dapat dilihat pada Gambar 5.1.

Tabel 5.2 Pengujian Modul *Fingerprint*

Nomor	**SP-UC01**
Nama	Pengujian Modul *Fingerprint*
Tujuan	Mengecek apakah aplikasi dapat mendeteksi lagu dari potongan suara
Kondisi Awal	Pengguna membuka halaman modul *fingerprint*
Skenario	Pengguna mendengarkan kepada aplikasi berupa potongan lagu
Langkah pengujian	1. Pengguna menekan tombol “rekam” untuk memulai proses perekaman

	2. Pengguna menekan tombol “berhenti” untuk menghentikan proses perekaman 3. Pengguna menakan tombol “upload” untuk mendapatkan hasil sesuai dengan modul.
Masukan	Potongan lagu
Keluaran yang Diharapkan	Judul lagu asli dari potongan lagu yang didengarkan kepada aplikasi
Hasil Pengujian	Berhasil



Gambar 5.1 Hasil dari Modul *Fingerprint*

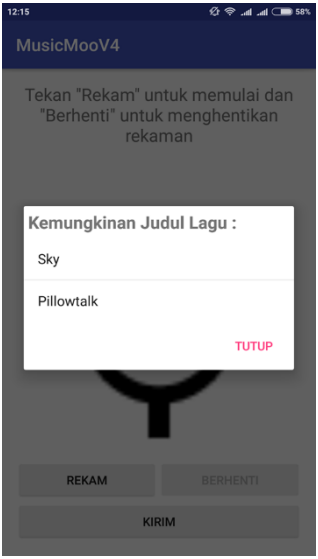
5.2.1.2. Pengujian Modul *Cover Song Recognition*

Pengujian modul *cover song recognition* dimulai ketika pengguna melakukan perekaman pada suatu sumber suara. Perekaman dilakukan pada perangkat bergerak Android yang telah dilakukan instalasi aplikasi tugas akhir ini. Aplikasi akan mengeluarkan nilai *string* yang mengandung informasi dari judul lagu asli yang dicari.

Rincian dari pengujian modul dapat dilihat pada Tabel 5.3. Hasil dari modul *cover song* pada aplikasi dapat dilihat pada Gambar 5.2.

Tabel 5.3 Pengujian Modul *Cover Song Recognition*

Nomor	**SP-UC02**
Nama	Pengujian Modul *Cover Song*
Tujuan	Mengecek apakah aplikasi dapat mendeteksi lagu asli
Kondisi Awal	Pengguna membuka halaman modul *cover song recognition*
Skenario	Pengguna mendengarkan kepada aplikasi berupa potongan lagu *cover*.
Langkah pengujian	1. Pengguna menekan tombol "rekam" untuk memulai proses perekaman 2. Pengguna menekan tombol "berhenti" untuk menghentikan proses perekaman 3. Pengguna menakan tombol "upload" untuk mendapatkan hasil sesuai dengan modul.
Masukan	Potongan Suara
Keluaran yang Diharapkan	Judul lagu asli dari potongan lagu *cover* yang didengarkan kepada aplikasi
Hasil Pengujian	Berhasil



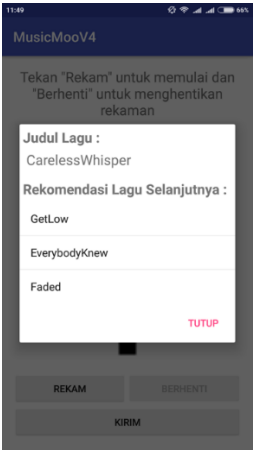
Gambar 5.2 Hasil dari Modul *Cover Song*

5.2.1.3. Pengujian Modul *Song Recommendation*

Pengujian modul *song recommendation* dimulai ketika pengguna melakukan perekaman pada suatu sumber suara. Perekaman dilakukan pada perangkat bergerak Android yang telah dilakukan instalasi aplikasi tugas akhir ini. Aplikasi akan mengeluarkan nilai *string* yang mengandung informasi dari rekomendasi lagu berdasarkan potongan lagu saat ini. Rincian dari pengujian modul dapat dilihat pada Tabel 5.4. Hasil dari modul *fingerprint* pada aplikasi dapat dilihat pada Gambar 5.3.

Tabel 5.4 Pengujian Modul *Song Recommendation*

Nomor	**SP-UC03**
Nama	Pengujian Modul *Song Recommendation*
Tujuan	Memberikan rekomendasi lagu dari potongan lagu yang dicari pada saat ini
Kondisi Awal	Pengguna membuka halaman modul *cover song recognition*
Skenario	Pengguna mendengarkan kepada aplikasi berupa potongan lagu
Langkah pengujian	1. Pengguna menekan tombol “rekam” untuk memulai proses perekaman 2. Pengguna menekan tombol “berhenti” untuk menghentikan proses perekaman 3. Pengguna menakan tombol “upload” untuk mendapatkan hasil sesuai dengan modul.
Masukan	Potongan Suara
Keluaran yang Diharapkan	Judul lagu yang mirip dengan lagu yang sekarang diperdengarkan
Hasil Pengujian	Berhasil



Gambar 5.3 Hasil pada Modul *Song Recommendation*

5.3. Akurasi Pengujian

Pada subbab ini diberikan hasil evaluasi dari masing-masing modul dalam bentuk sebuah akurasi. Masing-masing modul dihitung akurasinya dan detail dari perhitungan akurasi akan dijelaskan pada subbab berikut beserta rincian dari masing-masing modul.

5.3.1. Akurasi Pengujian Fungsionalitas

Pada subbab ini akan menjelaskan akurasi dari hasil yang telah didapatkan dari modul *fingerprint* dan modul *cover song recognition*. Untuk modul *song recommendation*, tidak dihitung akurasinya, sebab masing-masing pengguna berhak untuk menerima atau tidak.

$$PPV = \frac{TP}{TP + FP} * 100\% \quad (5.1)$$

Akurasi pada setiap modul akan dihitung dengan Persamaan 5.1 dengan menggunakan metode *Positive Preditive Value* (*PPV*). *TP* adalah *True Positif* yaitu jumlah data uji yang dapat dideteksi benar oleh sistem. Pada modul *fingerprint*, *True Positif* adalah jumlah potongan lagu yang dapat dideteksi judulnya oleh sistem. *True Positif* pada modul *cover song* adalah jumlah potongan lagu *cover* yang dapat dideteksi oleh sistem. *FP* merupakan *False Positif* yaitu data uji yang seharusnya benar namun dideteksi salah oleh sistem. Sehingga *TP* + *FP* adalah jumlah data keseluruhan yang digunakan pada sistem.

5.3.1.1. Akurasi Modul *Fingerprint*

Tingkat akurasi modul *fingerprint*, didapat ketika hasil judul lagu dari sistem sesuai dengan potongan judul yang diperdengarkan. Data *training*

diambil dari internet dengan format .wav sebanyak 11 judul. Data untuk *testing* diacak dari 11 judul tersebut untuk dipotong 30 detik. Pada pengujian pertama data uji diambil secara acak dari bagian manapun musik pada data latih dan dilakukan perhitungan. Detail hasil dapat dilihat pada Tabel 5.5.

Tabel 5.5 Rincian Hasil Modul *Fingerprint*

Judul Lagu	**Hasil**
Careless Whisper	x
Everybody Knew	v
Faded	v
Get Low	v
Its My Life	v
Livin on A Prayer	x
Middle	v
Saxobeats	v
Thing Will Get Better	v
Tremor	v
Turn Up the Speakers	v

Keterangan:
x: Potongan lagu yang tidak terdeksi.
v: Potongan lagu yang berhasil dideteksi.

Dari Tabel 5.5 dapat disimpulkan bahwa dari 11 percobaan terdapat 9 hasil yang benar dan 2 yang salah. Kesalahan 2 data *testing* tersebut dapat dihasilkan karena gangguan sumber suara (sumber suara yang buruk). Contoh dari sumber suara yang buruk adalah kualitas *speaker* yang menjadi sumber suara. Ketika kualitas *speaker* yang digunakan kurang baik, maka akan besar kemungkinan aplikasi tidak dapat mendeteksi secara akurat. Sehingga untuk akurasi modul *fingerprint* menurut Persamaan 5.1

adalah 81,81%. Pada pengujian kedua data uji diambil dari bagian 30 detik terakhir sesuai dengan subbab pada Bab III mengenai Perancangan Data Latih. Kualitas *speaker* yang dalam pengujian kedua menggunakan laptop dengan kualitas *speaker* yang baik. Hasil dari pengujian kedua dapat dilihat pada Tabel 5.6. Sehingga untuk akurasi modul *fingerprint* skenario pengujian kedua menurut Persamaan 5.1 adalah 97,852%.

Tabel 5.6 Rincian Hasil Modul *Fingerprint*

Judul Lagu	Potongan Ke	Hasil
Careless Whisper	1	v
	2	x
	3	v
	4	v
	5	v
Everybody Knew	1	v
	2	v
	3	v
	4	v
Faded	1	v
	2	v
	3	v
	4	v
Get Low	1	v
	2	v

	3	v
	4	v
Its My Life	1	v
	2	v
	3	v
	4	v
Living On a Prayer	1	v
	2	v
	3	v
	4	v
Middle	1	v
	2	v
	3	v
	4	v
Saxobeats	1	v
	2	v
	3	v
	4	v
Thing Will Get better	1	v
	2	v
	3	v
	4	v
	5	v

Tremor	1	v
	2	v
	3	v
	4	v
Turn Up The speaker	1	v
	2	v
	3	v
	4	v

Keterangan:

x: Potongan musik tidak dapat dideteksi.

v: Potongan musik dapat dideteksi.

5.3.1.2. Akurasi Modul *Cover Song Recognition*

Tingkat akurasi modul *cover song recognition*, didapat ketika hasil judul lagu dari sistem sesuai dengan potongan judul *cover* yang diperdengarkan. Data *training* diambil dari internet dengan format .wav sebanyak 5 judul. Dengan masing-masing judul mengambil 5 laki-laki dan 5 perempuan. Sehingga total data *training* adalah 10 untuk masing-masing judul. Untuk percobaan pada penelitian kali ini, maka nilai K yang diujikan adalah nilai K = 5.

Tabel 5.7 Hasil *Testing Cover Song* laki-laki

Judul lagu *cover*	Nilai K terdekat				
	K=1	K=2	K=3	K=4	K=5
Sky	Sky	Sky	Sky	Sky	Sky
Stitches	Stitches	Pillowtalk	Sky	Stitches	Heathens
Treat	Stitches	Pillowtalk	Pillowtalk	Pillowtalk	Pillowtalk
Heathens	Heathens	Treat	Heathens	Sky	Stitches
Pillowtalk	Pillowtalk	Sky	Sky	Sky	Pillowtalk

Tabel 5.8 Hasil *Testing Cover Song* perempuan

Judul lagu *cover*	Nilai K terdekat				
	K=1	K=2	K=3	K=4	K=5
Sky	Sky	Sky	Pillowtalk	Stitches	Sky
Stitches	Sky	Heathens	Pillowtalk	Heathens	Sky
Treat	Sky	Sky	Heathens	Pillowtalk	Heathens
Heathens	Heathens	Sky	Sky	Heathens	Heathens
Pillowtalk	Sky	Sky	Pillowtalk	Sky	Pillowtalk

Dari Tabel 5.7 dan Tabel 5.8 dapat disimpulkan bahwa hasil percobaan pada laki-laki hanya judul Treat yang buruk dan sisanya memiliki hasil yang cukup baik. Pada perempuan, hasil yang buruk terletak pada judul Stitches dan Treat. Hasil percobaan disebabkan oleh banyak faktor. Salah satu faktor yang mungkin terjadi adalah, pada rekaman lagu tersebut data *testing* maupun data *training* masih mengandung banyak suara instrumen, sehingga ketika dibandingkan hasil menjadi buruk. Sehingga untuk akurasi modul *cover song* menurut Persamaan 5.1 adalah 70%.

5.4. Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5.9. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik kesimpulan bahwa fungsionalitas dari aplikasi telah dapat bekerja sesuai dengan yang diharapkan.

Tabel 5.9 Rangkuman Hasil Pengujian

ID	Nama	Hasil
SP-UC1	Pengujian Modul *Fingerprint*	Berhasil
SP-UC2	Pengujian Modul *Cover Song*	Berhasil
SP-UC3	Pengujian Modul *Song Recommendation*	Berhasil

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. *Fingerprint* dapat didapatkan dengan fitur *Audio Signature Type* pada fitur MPEG-7. Klasifikasi *fingerprint* dapat menggunakan KNN, namun harus dimodifikasi. Hal tersebut sangatlah penting, sebab terjadi perbedaan dimensi antara sinyal untuk *testing* dan sinyal untuk *training*.
2. *Cover Song Recognition* dapat diperoleh melalui fitur *Audio Spectrum Flatness* dan *Audio Spectrum Projection* pada fitur MPEG-7. Kedua fitur ini tidak dapat langsung diaplikasikan, melainkan perlu diambil informasi untuk yang *vocal* saja. Pengambilan informasi tersebut dapat dicapai dengan mengimplementasikan *Discrete Wavelet Transform* pada masing-masing fitur. Setelah itu, baru fitur dapat dilakukan klasifikasi.
3. *Song recommendation* lagu dapat dicapai hanya dengan melihat modul *fingerprint*. Sebab pada modul *fingerprint* sudah menghitung kemiripan dari suatu lagu. Rekomendasi lagu dapat diberikan ketika lagu tersebut mirip dengan lagu yang lain.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-

saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Penambahan *multi processing* ketika melakukan perhitungan, sehinga waktu yang dibutuhkan tidak terlalu lama.
2. Melakukan pergantian *database* menjadi *database* NoSQL seperti MongoDB atau Hadoop.
3. Untuk hasil yang lebih optimal, perhitungan pada saat klasifikasi dapat dengan cara mempertahankan fitur dalam bentuk matriks. Pengambilan fitur dapat dengan cara implementasi *wavelet* pada fitur. Hasil dari masing-masing level dekomposisi akan dianggap sebagai fitur baru.

DAFTAR PUSTAKA

[1] S. D. You, W.-H. Chen, and W.-K. Chen, "Music Identification System Using MPEG-7 Audio Signature Descriptors," *Sci. World J.*, vol. 2013, pp. 1–11, Mar. 2013.

[2] T. Bertin-Mahieux and D. P. W. Ellis, "Large-scale cover song recognition using hashed chroma landmarks," in *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2011, pp. 117–120.

[3] "ISO/IEC 15938-1:2002 - Information technology -- Multimedia content description interface -- Part 1: Systems," *ISO*. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34228. [Accessed: 24-Dec-2016].

[4] "Copyright." [Online]. Available: http://mpeg7audioenc.sourceforge.net/copyright.html. [Accessed: 15-Jan-2017].

[5] H.-G. Kim, N. Moreau, and T. Sikora, "MPEG-7 Audio and Beyond Audio Content Indexing and Retrieval," p. 13.

[6] H.-G. Kim, N. Moreau, and T. Sikora, "MPEG-7 Audio and Beyond Audio Content Indexing and Retrieval," pp. 13–58.

[7] H.-G. Kim, N. Moreau, and T. Sikora, "MPEG-7 Audio and Beyond Audio Content Indexing and Retrieval," p. 80.

[8] M. N. Munawar, R. Sarno, D. A. Asfani, T. Igasaki, and B. T. Nugraha, "Significant preprocessing method in EEG-Based emotions classification," *J. Theor. Appl. Inf. Technol.*, vol. 87, no. 2, pp. 176–190, May 2016.

[9] D. R. Wijaya, R. Sarno, and E. Zulaika, "Information Quality Ratio as a novel metric for mother wavelet selection," *Chemom. Intell. Lab. Syst.*, vol. 160, pp. 59–71, 2016.

[10] "XQuery Tutorial." [Online]. Available: http://www.w3schools.com/xml/xquery_intro.asp. [Accessed: 18-Dec-2016].

[11] “why_mir.” [Online]. Available: http://musicinformationretrieval.com/why_mir.html. [Accessed: 21-Dec-2016].
[12] H.-G. Kim, N. Moreau, and T. Sikora, “MPEG-7 Audio and Beyond Audio Content Indexing and Retrieval,” p. 217.
[13] “Play Framework - Build Modern & Scalable Web Apps with Java and Scala.” [Online]. Available: https://www.playframework.com/. [Accessed: 31-Dec-2016].
[14] “Welcome | Flask (A Python Microframework).” [Online]. Available: http://flask.pocoo.org/. [Accessed: 21-Dec-2016].
[15] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer Science & Business Media, 2009.

LAMPIRAN A

```
1.  public class AppLog {
2.
3.      private static final String APP_TAG = "AudioRec
    order";
4.
5.      public static int logString(String message){
6.          return Log.i(APP_TAG,message);
7.      }
8.  }
```

Kode Sumber A.1 Kelas *AppLog*

```
1.  <manifest xmlns:android="http://schemas.android.com
    /apk/res/android"
2.      package="com.example.ponighzwa.musicmoov4">
3.      <uses-
    permission android:name="android.permission.WRITE_E
    XTERNAL_STORAGE"></uses-permission>
4.      <uses-
    permission android:name="android.permission.RECORD_
    AUDIO"></uses-permission>
5.      <uses-
    permission android:name="android.permission.INTERNE
    T"/>
6.      <uses-
    permission android:name="android.permission.ACCESS_
    NETWORK_STATE"/>
7.      <uses-
    permission android:name="android.permission.READ_EX
    TERNAL_STORAGE"/>
8.      <application
9.          android:allowBackup="true"
10.         android:icon="@mipmap/ic_launcher"
11.         android:label="@string/app_name"
12.         android:supportsRtl="true"
13.         android:theme="@style/AppTheme">
14.         <activity android:name=".MainActivity">
15.             <intent-filter>
```

```
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Rekam" />
    </application>

</manifest>
```

Kode Sumber A.2 *Manifest* pada Android

```
<resources>
    <string name="app_name">MusicMooV4</string>

    <string name="app_info">Tekan \"Rekam\" untuk memulai dan \"Berhenti\" untuk menghentikan rekaman</string>
    <string name="choose_format_title">Choose Audio Format</string>
    <string name="start_recording">Rekam</string>
    <string name="stop_recording">Berhenti</string>

    <string name="audio_format">Format</string>
</resources>
```

Kode Sumber A.3 *String Values* pada Android

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "23.0.3"

    defaultConfig {
        applicationId "com.example.ponighzwa.musicmoov4"
        minSdkVersion 21
        targetSdkVersion 25
```

```
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:25.0.1'
}
```

Kode Sumber A.4 *Gradle* pada Android

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Kode Sumber A.5 *Colour Values* pada Android

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Mochammad Faris Ponighzwa Rizkanda dilahirkan pada tanggal 20 Maret 1995 di Gresik. Pada tahun 2013, sesudah lulus dari SMAN 1 Gresik melanjutkan menimba ilmu di jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya. Penulis aktif pada beberapa kepanitiaan seperti ITS Expo, Schematics, dan kegiatan lainnya. Dalam melakukan penulisan Tugas Akhir, penulis memiliki ketertarikan pada bidang Manajemen Informasi. Untuk menghubungi penulis, dapat menghubungi melalui *email*: ponighzwa13@mhs.if.its.ac.id.