IMPLEMENTASI ALGORITMA SIDIK JARI AUDIO UNTUK MENDETEKSI DUPLIKAT LAGU

Raka Yusuf[1], Harni Kusniyati[2], Erick Estrada[3]
Universitas Mercu Buana
E-mail: rakayusuf@yahoo.co.id[1], harni246@gmail.com[2], erick_estrada@yahoo.com[3]

ABSTRACT

In a song storage media, often found duplicate songs which resulted in the use of storage media space song not be optimal. Duplicate songs are generally caused by the way the user to store songs on the storage media. Generally, users do not pay attention to whether the storage medium has tracks are the same song or not. In addition, differences in track storage location is one of the factors that make it difficult to determine whether the storage medium contained songs duplicate songs. Therefore, to optimize storage space song, then designed an application that uses audio fingerprinting algorithm to identify duplicate songs in the song storage media. This study uses the waterfall model of software engineering, to produce an application that can overcome the problem of duplicate song files on the storage media.

Keywords: *Fingerprint Algorithm Song, Library Open Fingerprint Architecture*

ABSTRAK

Dalam media penyimpanan lagu, sering ditemukan duplikat lagu yang mengakibatkan penggunaan penyimpanan ruang media lagu tidak optimal. Duplikat Lagu umumnya disebabkan oleh cara pengguna untuk menyimpan lagu pada media penyimpanan. Umumnya, pengguna tidak memperhatikan apakah media penyimpanan memiliki track lagu yang sama atau tidak. Selain itu, perbedaan dalam lokasi penyimpanan lagu adalah salah satu faktor yang membuat sulit untuk menentukan apakah media penyimpanan yang terdapat duplikat lagu. Oleh karena itu, untuk mengoptimalkan ruang penyimpanan lagu, kemudian merancang sebuah aplikasi yang menggunakan algoritma sidik jari audio ke mengidentifikasi duplikat lagu di media penyimpanan lagu. Penelitian ini menggunakan model air terjun rekayasa perangkat lunak, untuk menghasilkan sebuah aplikasi yang dapat mengatasi masalah file lagu duplikat pada media penyimpanan.

Kata kunci*: Sidik Jari Algoritma Lagu, Perpustakaan Terbuka Fingerprint Arsitektur*

1. Pendahuluan

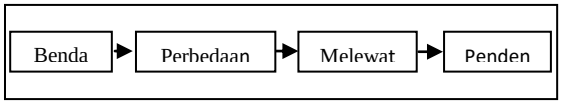
Beban kehidupan akhir-akhir ini terasa bertambah berat, membuat manusia harus berpacu melawan waktu bekerja keras untuk mendapatkan rejeki. Hal inilah yang membuat masyarakat Indonesia, terutama yang hidup di kota-kota besar mengalami tingkat stress yang begitu tinggi. Banyak cara yang dilakukan oleh manusia untuk menghilangkan stress, dimana salah satu cara untuk menghilangkan stress adalah dengan mendengarkan musik. Musik sangat besar pengaruhnya dalam kehidupan manusia, karena dapat membawa emosi bagi pendengarnya. Saat musik mengalun sendu, maka tanpa terasa emosi si pendengarnya terbawa hanyut oleh kesenduan musik tersebut, namun tatkala musik mengalun riang dengan nada semangat, maka emosi si pendengarpun dapat ikut terbawa semangat.

Musik atau lagu yang kita dengarkan sejak kecil hingga dewasa, mungkin sudah berjumlah ratusan lagu. Lagu yang diciptakan manusia mungkin sudah ribuan atau bahkan jutaan lagu. Banyak atau hampir setiap orang menyimpan lagu-lagu sebagai koleksi pribadi, terlebih lagi bagi suatu stasiun radio yang menyimpan ribuan lagu dalam berbagai jenis aliran musik.

2. Landasan Teori

A. Pengertian Suara

Suara adalah gelombang yang dihasilkan oleh getaran benda dengan amplitudo yang berubah secara kontinu terhadap waktu. Suara berhubungan dengan pendengaran yang merambat melalui media terkompresi seperti gas, air atau udara, dan tidak dapat merambat melalui ruang hampa. Lihat Gambar 1.



Gambar 1. Proses Terjadinya Suara

Suara dihasilkan oleh getaran suatu benda dimana selama bergetar, terjadi perbedaan tekanan pada media terkompresi yang dilalui, sehingga menghasilkan pola osilasi yang disebut dengan gelombang. Gelombang yang dihasilkan memiliki pola sama berulang pada interval tertentu, dan dapat dihitung dengan rumus:

$\lambda = c/f$.....(1)

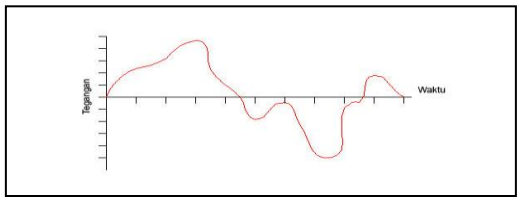
Dimana : λ = Panjang Gelombang suara (m)
c = Kecepatan Rambat Suara (m/s)
f = Frekuensi (Hz)

Berdasarkan tinggi rendahnya frekuensi yang dihasilkan, maka gelombang suara dibagi kedalam empat jenis gelombang suara, yaitu:

1. Infrasonik yang mempunyai rentang frekuensi diantara 0 Hz – 20 Hz.
2. Audiosonik yang mempunyai rentang frekuensi diantara 20 Hz – 20 KHz.
3. Ultrasonik yang mempunyai rentang frekuensi diantara 20 KHz – 1GHz.
4. Hipersonik yang mempunyai rentang frekuensi diantara 1 GHz – 10 THz.

B. Audio

Watkinson (2002:3) menyatakan, bahwa audio adalah sinyal analog dalam bentuk gelombang listrik yang merepresentasikan gelombang suara. Dalam proses menghasilkan audio, suara asli diubah oleh *transducer* seperti mikrofon menjadi sinyal analog listrik yang merambat secara kontinu pada media terkompresi, dan mengalami perubahan seiring waktu dan media yang dilalui. Lihat Gambar 2.



Gambar 2**.** Contoh Gelombang Sinyal Analog

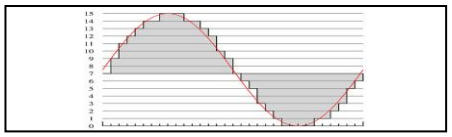
Gelombang sinyal analog dihitung berdasarkan amplitudo sebagai ukuran tinggi rendahnya gelombang sinyal analog dalam satuan *decibel* (db), dan *velocity* sebagai ukuran kecepatan rambat gelombang dalam satuan m/s.

Sebuah sinyal audio analog merupakan akumulasi gabungan dari suara asli dan *noise*. Hal ini merupakan kelemahan dari sinyal audio analog sebab perubahan gelombang listrik yang dihasilkan selama melalui media terkompresi bersifat kontinu sehingga sulit untuk mendeteksi proporsi energi dari suara asli maupun *noise*.

Untuk mengatasi kelemahan sinyal audio analog, maka diperlukan suatu bentuk gelombang sinyal audio yang dapat dideteksi proporsi antara energi suara asli maupun *noise* pada sinyal tersebut. Gelombang sinyal audio ini merupakan gelombang sinyal audio digital.

C. Audio Digital

Fries dan Marty (2005:142) menyatakan, bahwa audio digital adalah representasi dari suara dalam bentuk deret angka berurutan yang mewakili gelombang sinyal selama interval waktu tertentu seperti yang ditunjukan pada Gambar 3.3. Audio digital direpresentasikan oleh bilangan biner (1 dan 0), dimana bilangan biner dinyatakan dalam bit-bit yang mewakili dua jenis tegangan, yaitu tegangan rendah atau *off* yang diwakili oleh bilangan 0, dan tegangan tinggi atau *on* yang diwakili oleh bilangan 1.



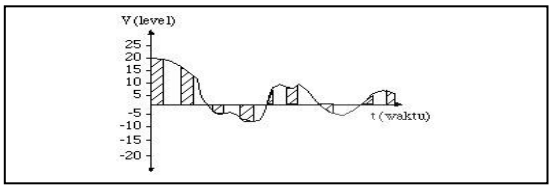
Gambar 3**.** Contoh Gelombang Sinyal Audio Digital

Definisi dari pemrosesan sinyal audio digital adalah analisa dan manipulasi informasi bentuk gelombang audio yang diproses ke dalam sistem komputer. Dalam pemrosesan ke dalam sinyal audio digital, dilakukan pengukuran amplitudo selama interval waktu tertentu pada sinyal analog untuk menghasilkan *sampling*, dimana pada tiap satuan pengukuran dalam *sampling* dinamakan sampel.

D. Audio Sampling

Dalam memproses sinyal analog menjadi bit-bit sinyal digital, digunakan teknik *sampling* yang merupakan teknik untuk mengambil cuplikan-cuplikan dari bagian-bagian sinyal analog, dimana tiap cuplikan yang diambil disebut dengan sinyal-sinyal sampel.

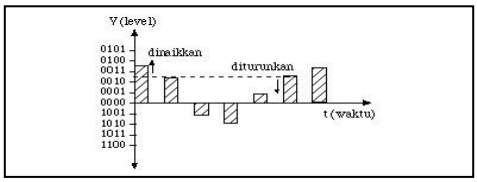
Dalam proses *sampling*, untuk menghasilkan sinyal sampel mengikuti aturan teori Shannon yang menyatakan minimum frekuensi sinyal sampel adalah dua kali frekuensi sinyal analog, sebab frekuensi sinyal sampel yang lebih banyak akan menghasilkan cuplikan yang lebih menggambarkan gelombang sinyal analog yang asli. Setelah proses *sampling* dilakukan, maka akan terbentuk sinyal analog diskrit yang menyerupai sinyal analog asli seperti yang ditunjukan pada Gambar.4.



Gambar 4**.** Contoh Sinyal Analog Diskrit

Setelah proses *sampling*, dilakukan proses pembandingan sinyal sampel diskrit dengan tetapan

tetapan level tergantung pada resolusi dari alat konversi. Sinyal-sinyal diskrit yang dihasilkan akan disesuaikan levelnya dengan tetapan-tetapan pada level yang ada. Dengan melakukan pembulatan angka, sinyal-sinyal diskrit yang lebih besar dari median rentang tetapan level akan dinaikan dan jika lebih kecil maka akan diturunkan. Lihat Gambar 5.



Gambar 5. Contoh Gambaran Proses Quantisasi Sinyal Analog

Tiap-tiap sinyal diskrit yang telah mempunyai tetapan tertentu menghasilkan representasi informasi dari sinyal analog. Urutan-urutan dari sinyal-sinyal diskrit disebut sebagai sinyal audio digital karena sudah berbentuk informasi dalam bentuk bit-bit digital.

E. Format Audio Digital

Binanto (2010:56) menyatakan, bahwa Format berkas audio merupakan hasil pemampatan audio yang dilakukan pada saat pembuatan berkas audio dan pada saat distribusi berkas audio tersebut, yang bertujuan untuk mengecilkan ukuran berkas audio. Pendekatan Umum tentang penyimpanan audio digital adalah pengambilan sampel tegangan audio analog menjadi digital yang pada waktu dimainkan kembali *(playback)* akan cocok dengan audio analog aslinya.

Dalam penyimpanannya, berkas audio umumnya dapat mendukung penggunaan beragam *codec*, walaupun terdapat juga berkas audio yang tidak mendukung penggunaan *codec* seperti AVI. Bentuk dari berkas audio sendiri digolongkan ke dalam tiga golongan besar berdasarkan jenis pemampatannya, yaitu:

1. Format audio yang tidak terkompresi *(uncompress)*

Format audio tidak terkompresi disebut juga berkas audio polos/mentah. Formati ini mengkodekan suara dengan jumlah bit per unit waktu yang sama, sebagai contoh sebuah berkas audio berisi simfoni orkestra akan berdurasi sama dengan berkas audio suara hening berdurasi satu menit. Contoh format audio ini adalah WAV, AIFF dan AU.

2. Format audio dengan kompresi *lossless*

Format audio kompresi *lossless* memampatkan suara dengan kualitas yang tidak level tertentu yang disebut quantisasi. level-level pembanding pada proses ini merupakan tetapan-tetapan angka dalam bentuk bilangan biner dimana berkurang, sehingga penggunaannya dapat difokuskan pada kecepatan kompresi dan decompresi, derajat kompresi dan dukungan terhadap perangkat keras dan perangkat lunak. Contoh format audio ini adalah FLAC, Wavpack (WV), Shorten, TTA dan ATRAC.

3. Format audio dengan kompresi *lossy*

Format audio ini memampatkan suara dengan kualitas yang berkurang. Proses kompresi format audio kompresi *lossy* lebih difokuskan untuk menjaga kualitas audio, faktor kompresi, kecepatan kompresi dan decompresi, dan kecepatan *real-time streaming*. Contoh format audio ini adalah MP3, Vorbis, *lossy* Windows Media Audio (WMA) dan AAC.

F. Codec

Binanto (55) menjelaskan, bahwa *codec* adalah program komputer yang memampatkan/menggembungkan data audio digital sesuai dengan format berkas audio yang diberikan. *Codec* audio mempunyai dua arti yang merupakan kombinasi dari *coder-decoder*.

G. Sidik Jari Audio

Untuk menghasilkan sidik jari audio, terdapat hal-hal yang perlu diperhatikan agar sidik jari yang dihasilkan dapat dianggap mewakili audio sumbernya, yaitu:

1. Ketahanan
   Untuk menghasilkan sidik jari audio, sebuah berkas audio harus memiliki tingkat ketahanan yang tinggi.
2. Kehandalan
   Sidik jari audio hasil ekstraksi dari berkas audio harus dapat digunakan untuk mengidentifikasi berkas lagu identik lainnya secara tepat.
3. Ukuran sidik audio
   Sidik jari audio harus berukuran kecil agar dalam penggunaannya sebagai alat identifikasi berkas audio, proses yang dilakukan akan berjalan cepat. Ukuran sidik jari audio sangat berpengaruh pada aplikasi yang berinteraksi dengan server yang menyediakan basis data informasi berkas audio.
4. Terperinci
   Lama durasi berkas audio yang digunakan untuk menghasilkan sidik jari audio harus diperhitungkan secara jelas, sehingga dapat menghasilkan sidik jari audio yang identik dengan berkas audio asli.
5. Kecepatan pencarian dan skalabilitas
   Kecepatan proses dalam satuan waktu yang diperlukan untuk mencocokan sidik jari. Kecepatan sangat diperlukan apabila proses

pencocokan sidik jari audio dilakukan dalam jumlah yang berskala besar.

Dalam mengekstraksi sidik jari audio, terdapat dua bagian yang menjadi proses utama audio mentah, sedangkan *Fingerprint Modeling Block* merupakan bagian dimana proses ekstraksi dari audio mentah yang dihasilkan untuk diubah menjadi sidik jari audio.

Dari kedua proses utama tersebut, terdapat langkah-langkah untuk memproses audio hingga menghasilkan sidik jari audio yang digambarkan ke dalam tahapan-tahapan sebagai berikut:

1. *Front End Module*
   Bagian ini dimulai dari proses awal (*Preprocessing*) yaitu mengkonversi sinyal audio menjadi format standar untuk proses sidik jari audio yaitu 16 bit mono PCM dengan *sample rate* 44,1 kHz. Sinyal audio kemudian dipecah menjadi *frame-frame* overlap berdurasi satuan milidetik *(Framing & Overlap)*. Pada proses selanjutnya, setiap frame yang ada diubah menggunakan metode *fourier transform*, yaitu operasi menguraikan sinyal berdasarkan sinyal penyusunnya ke dalam bentuk blok-blok frekuensi sinyal *(Transform)*. Blok-blok frekuensi sinyal ini disebut sebagai hasil ekstraksi fitur *(feature extraction)* yang selanjutnya akan diproses pada *Fingerprint Model Block (Post-Processing)*.
2. *Fingerprint Modeling Block*
   Bagian ini merupakan tahap proses ekstraksi sidik jari audio, yaitu tahap menghasilkan sidik jari audio dengan mengambil *magnitude* tiap-tiap blok frekuensi sinyal yang diproses pada tahap *front end module*, dan menggunakan algoritma tertentu untuk mengubah blok-blok sinyal tersebut menjadi sidik jari audio.

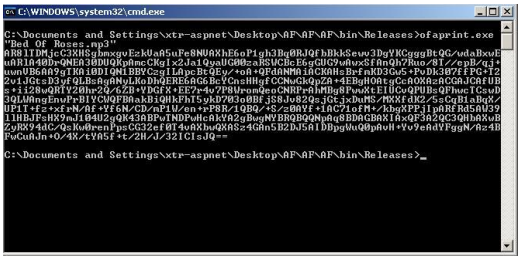
H. Library Open Fingerprint Architecture

Arsitektur *LIBOFA* dapat mengidentifikasi lagu dengan toleransi *bitrate* terendah mulai dari 65 kbps hingga *bitrate* tertinggi yang ada, serta mendukung identifikasi berkas audio dengan algoritma kompresi *lossy* seperti MP3.

Pada Arsitektur *LIBOFA*, terdapat tiga tahap proses utama dimana dua diantaranya merupakan tahap untuk menghasilkan sidik jari audio. Tahap-tahap tersebut yaitu:

1. Konversi berkas audio
   Dalam menciptakan sidik jari audio, arsitektur *LIBOFA* menggunakan *codec* untuk mengubah berkas audio kedalam bentuk audio mentah dengan format 16 bit mono PCM bertipe WAV.
2. Ekstraksi sidik jari audio
   Dari audio mentah yang dihasilkan, sinyal-sinyal audio dibagi menjadi serangkaian *frame* dalam menciptakan sidik jari, yaitu *Front End Module* dan *Fingerprint Modeling Block. Front End Module* merupakan bagian dimana terjadi proses ekstraksi isi berkas audio ke dalam bentuk dengan durasi setiap *frame* berukuran 185 milidetik. Setiap kolom pada matrik awal mewakili pita frekuensi, dan setiap baris pada matrik awal mewakili *frame*.

Dengan menggunakan rumus, digunakan nilai-nilai pada matriks V untuk mengkonstruksi *core print* berukuran 512 *bytes*, digabung dengan empat nilai pita frekuensi paling menonjol *(pitch print)* berukuran 4 *bytes*, lalu dienkripsi dengan enkripsi yang dikembangkan oleh *LIBOFA*, dan dienkripsi kembali menggunakan enkripsi base64 untuk menghasilkan sebuah sidik jari audio berukuran 516 *bytes*, yang divisualisasikan pada Gambar 6.



Gambar 6. Visualisasi Bentuk Sidik Jari Lagu

3. Mengambil informasi-informasi berkas audio
   Arsitektur *LIBOFA* menggunakan *protocol layer* untuk mengirimkan informasi sidik jari audio ke server *MusicDNS* untuk mendapatkan informasi berkas audio yang sesuai dengan sidik jari audio yang dikirimkan. Tahap ini bersifat opsional, karena tidak mempengaruhi proses dalam menghasilkan sidik jari audio.

3. Hasil Dan Pembahasan

A. Analisis Masalah

Teknologi multimedia saat ini mempunyai peranan penting dalam kehidupan sehari-hari, sebagai salah satu contoh adalah kebutuhan untuk mendengarkan lagu sebagai media yang paling sering digunakan untuk menghilangkan stress. Penikmat lagu dapat menikmati lagu yang ia inginkan menggunakan perangkat keras yang memiliki aplikasi pemutar lagu, ataupun mendengarkan langsung melalui sinyal gelombang-gelombang radio.

1) Faktor-Faktor Penyebab Duplikasi Lagu

Duplikasi berkas lagu pada media penyimpanan lagu merupakan kondisi dimana terdapat 2 atau lebih berkas lagu yang sama pada media penyimpanan lagu tersebut. Duplikasi tersebut dapat terjadi dikarenakan beberapa faktor yang disebabkan oleh cara pemilik media

penyimpanan lagu dalam melakukan penyimpanan berkas lagu kedalam media penyimpanan lagu, seperti:

Menyimpan berkas lagu dengan nama A.MP3 ke dalam direktori A pada media

2) Solusi-Solusi Untuk Mengatasi Duplikasi Lagu Berdasarkan faktor-faktor penyebab duplikasi berkas lagu, maka dapat didefinisikan kondisi-kondisi yang disebut sebagai duplikasi berkas lagu, yaitu:

1. Kondisi dimana terjadi duplikasi berkas lagu pada lokasi penyimpanan yang sama, karena terdapat 2 atau lebih berkas lagu yang sama dengan penamaan berkas lagu yang berbeda-beda.
2. Kondisi dimana terjadi duplikasi berkas lagu pada lokasi penyimpanan yang berbeda, walaupun berkas-berkas lagu tersebut memiliki penamaan yang sama.
3. Kondisi terjadinya duplikasi berkas lagu pada lokasi penyimpanan yang berbeda, dengan penamaan berkas-berkas lagu yang berbeda pada tiap lokasi penyimpanannya.

Berdasarkan kondisi-kondisi diatas, dapat disolusikan beberapa cara untuk mencari duplikasi berkas-berkas lagu pada media penyimpanan lagu, yaitu:

1. Melakukan pencarian secara manual ke dalam lokasi media penyimpanan lagu, dengan cara memutar tiap-tiap berkas lagu untuk mengidentifikasi apakah berkas-berkas lagu tersebut dapat dinyatakan sama.
2. Menggunakan fitur pencarian yang dimiliki oleh sistem operasi, dimana berkas-berkas lagu dengan penamaan yang mirip pada lokasi yang sama atau penamaan yang sama pada lokasi yang berbeda akan diputar agar dapat teridentifikasi apakah berkas-berkas lagu tersebut dapat dinyatakan sama.
3. Menggunakan algoritma khusus untuk melakukan ekstraksi sidik jari untuk setiap berkas lagu pada media penyimpanan lagu, kemudian melakukan pencarian duplikasi berkas lagu dengan cara membandingkan setiap sidik jari berkas lagu satu dengan lainnya, dan menampilkan hasil pencarian berkas-berkas lagu yang di teridentifikasi sama berdasarkan perbandingan antara sidik jari berkas-berkas lagu yang sama, serta menampilkan lokasi berkas-berkas lagu yang sama tersebut.

3) Analisis Program Aplikasi Setelah menganalisa permasalahan dan menganalisa solusi terbaik dari solusi-solusi yang ada, maka penulis memutuskan untuk merancang sebuah program aplikasi yang dapat melakukan pencarian duplikasi berkas lagu pada suatu media penyimpanan lagu.

penyimpanan lagu, dimana pada direktori A dalam media penyimpanan lagu tersebut sudah terdapat berkas lagu yang sama namun memiliki nama B.MP3.

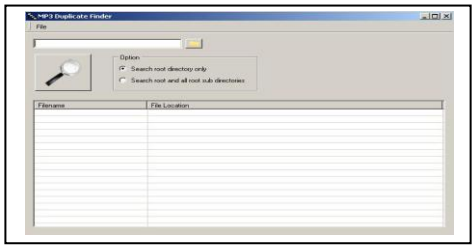
B. Implementasi Program Aplikasi

Pada tahap implementasi program aplikasi, proses pengkodean program akan menggunakan bahasa pemrograman Visual basic.NET, dan Microsoft Visual Studio 2005 sebagai antar muka pengembang aplikasi yang digunakan.

Berdasarkan Rancangan diagram alir program, ditentukan tahapan-tahapan implementasi yang akan dilakukan yaitu implementasi antar muka aplikasi, implementasi fitur lokasi pencarian, implementasi fitur opsi pencarian, dan implementasi fitur pencarian.

1. Implementasi Antar Muka Aplikasi

Pada diagram alir program, setelah program dijalankan, pengguna akan disuguhkan tampilan layar utama program. Pada pembuatan layar utama digunakan beberapa komponen yang terdapat pada *toolbar* Microsoft Visual Studio 2005 yaitu *menu*, *textbox*, tombol *(button)*, *optionbox*, *radio*, dan *listview* sebagai komponen pendukung antar muka program aplikasi. Berikut digambarkan pada Gambar 7. tampilan layar utama.



Gambar 7. Tampilan Layar Utama Program Aplikasi

2. Implementasi Fitur Lokasi Pencarian

Fitur lokasi pencarian merupakan fitur untuk menentukan lokasi pencarian duplikasi lagu atau merupakan fitur untuk menentukan lokasi media penyimpanan lagu. Dalam fitur ini digunakan 1 komponen *textbox* dan 1 komponen *button* pada antar muka program aplikasi. Implementasi Fitur Opsi Pencarian

Fitur Opsi pencarian merupakan fitur untuk menentukan cara pencarian duplikasi lagu pada media penyimpanan lagu. Pada antar muka aplikasi, diberikan opsi pencarian duplikasi berkas lagu pada direktori utama atau direktori dan sub direktori utama media penyimpanan lagu. Dalam fitur ini digunakan 1 komponen *Optionbox* dan 2 komponen *radio* pada antar muka program aplikasi

C. Pengujian Program Aplikasi

Pengujian program aplikasi merupakan tahapan yang dilakukan untuk penilaian kesesuaian program yang dibuat dengan analisa yang

dilakukan dan hasil yang diharapkan. Dalam tahap pengujian aplikasi ini, dilakukan pengujian dengan metode *Black Box* yang merupakan uji spesifikasi dan fungsionalitas program, tanpa adanya pengetahuan tentang struktur *internal* dari *source* pengujian, hasil pengujian dan analisis hasil pengujian.

1. Lingkungan Pengujian

Dalam pengujian program aplikasi, diperlukan perangkat lunak dan perangkat keras untuk melakukan pengujian, sehingga diperlukan persiapan awal untuk mempersiapkan kebutuhan akan perangkat keras dan perangkat lunak tersebut. Untuk pengujian program aplikasi ini, dilakukan pengujian pada lingkungan pengujian sebagai berikut:

a. Perangkat Keras Yang Dibutuhkan

Berdasarkan fungsi atau kegunaannya, perangkat keras dibagi ke dalam tiga jenis, yaitu:

A. Alat Masukan (*Input Device*)

Dalam pengujian program aplikasi ini, digunakan 2 alat masukan yaitu papan ketik atau *keyboard* dan *mouse*.

B. Alat Pemroses (*Processing Unit*)

Alat pemroses yang digunakan dalam pengujian program aplikasi ini adalah sebuah CPU (*Central Processing Unit*) yaitu prosesor Intel Core 2 Duo T6400 @2.00GHz, 1 Gb RAM memory, dan sebuah *harddisk* dengan kapasitas penyimpanan sebesar 250 Gb.

C. Alat Keluaran (*Output Device*)

Dalam pengujian program aplikasi ini digunakan alat keluaran berupa layar monitor bertipe SVGA.

b. Perangkat Lunak Yang Digunakan

c. Perangkat lunak merupakan sekumpulan baris perintah atau program yang digunakan untuk memberikan instruksi-instruksi pengolahan data kepada perangkat keras komputer. Pada pengujian program ini, dilakukan pada sistem operasi Windows XP.

2. Skenario Pengujian

Dalam melakukan pengujian program aplikasi ini, ditetapkan skenario pengujian yang disusun sebagai berikut:

Tabel 1. Skenario Pengujian

	Yang Diuji	Data Pengujian	Cara Menguji	Hasil Yang Diharapkan
1	Layar Utama	2 berkas lagu dengan nama yang sama, isi yang sama, *bitrate* yang sama, dengan kedua berkas lagu berada pada lokasi berbeda	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama dan sub direktori dari direktori utama 4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	Program akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian
2	Layar Utama	2 berkas lagu dengan nama yang sama, isi yang sama, *bitrate* yang berbeda, dengan kedua berkas lagu berada pada lokasi berbeda	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser.* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama dan sub direktori dari direktori utama	Progrm akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian

			4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	
3	Layar Utama	2 berkas lagu dengan nama yang berbeda, isi yang sama, *bitrate* yang sama, dengan kedua berkas lagu berada pada lokasi yang sama	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama 4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	Program akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian
4	Layar Utama	2 berkas lagu dengan nama yang berbeda, isi yang berbeda, *bitrate* yang sama, dengan kedua berkas lagu berada pada lokasi yang sama	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama 4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	Program tidak akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian
5	Layar Utama	2 berkas lagu dengan nama yang berbeda, isi yang berbeda, *bitrate* yang sama, dan kedua berkas lagu berada pada lokasi yang berbeda	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama dan sub direktori dari direktori utama 4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	Program tidak akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian

3. Hasil Pengujian

Dari skenario pengujian yang dilakukan, didapatkan hasil pengujian sebagai berikut:

Tabel 2. Hasil Pengujian

No	Yang Diuji	Data Pengujian	Cara Menguji	Hasil Yang Diharapkan	Hasil Pengujian
1	Layar Utama	2 berkas lagu dengan nama yang sama, isi yang sama, *bitrate* yang sama, dengan kedua berkas lagu berada pada lokasi berbeda	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama dan sub direktori dari direktori utama 4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	Program akan menampilkan kedua berkas lagu yang diuji, berikut lokasi dari masing-masing berkas lagu, yang akan ditampilkan pada tabel hasil pengujian	Sesuai
2	Layar Utama	2 berkas lagu dengan nama yang sama, isi yang sama, *bitrate* yang berbeda, dengan kedua berkas lagu berada pada lokasi berbeda	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK 3 Pilih opsi pencarian pada direktori utama dan sub direktori dari direktori utama 4 Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu	Program akan menampilkan kedua berkas lagu yang diuji, berikut lokasi dari masing-masing berkas lagu, yang akan ditampilkan pada tabel hasil pengujian	Sesuai
3	Layar Utama	2 berkas lagu dengan nama yang berbeda, isi yang sama, *bitrate* yang sama, dengan kedua	1 Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser* 2 Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK	Program akan menampilkan kedua berkas lagu yang diuji, berikut lokasi dari masing-masing berkas lagu, yang akan ditampilkan pada tabel hasil pengujian	Sesuai

		berkas lagu berada pada lokasi yang sama	3	Pilih opsi pencarian pada direktori utama		
			4	Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu		
4	Layar Utama	2 berkas lagu dengan nama yang berbeda, isi yang berbeda, *bitrate* yang sama, dengan kedua berkas lagu berada pada lokasi yang sama	1	Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser*	Program tidak akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian	Sesuai
			2	Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK		
			3	Pilih opsi pencarian pada direktori utama		
			4	Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu		
5	Layar Utama	2 berkas lagu dengan nama yang berbeda, isi yang berbeda, *bitrate* yang sama, dan kedua berkas lagu berada pada lokasi yang berbeda	1	Pilih lokasi dengan menekan tombol bergambar direktori, maka akan muncul jendela *directory browser*	Program tidak akan menampilkan kedua berkas lagu yang diuji, pada tabel hasil pengujian	Sesuai
			2	Pada jendela *directory browser*, pilih lokasi berkas lagu yang akan diuji, kemudian tekan tombol OK		
			3	Pilih opsi pencarian pada direktori utama		
			4	Tekan tombol cari untuk memulai proses pengujian kedua berkas lagu		

4. Analisis Hasil Pengujian

Dari hasil skenario pengujian terhadap aplikasi, dilakukan analisa sebagai berikut:

1. Hasil pengujian pada skenario pengujian nomor 1 menunjukkan kesesuaian dengan hasil yang diharapkan, karena data pengujian yang digunakan merupakan 2 berkas lagu dengan nama, isi dan *bitrate* yang sama, dimana dalam prosesnya aplikasi akan menghasilkan *sampling* dan nilai-nilai quantisasi yang sama untuk setiap berkas lagu data pengujian, dan dalam proses ekstraksi sidik jari akan menghasilkan sidik jari berkas lagu yang dapat dipastikan sama. Perbedaan lokasi tidak akan mempengaruhi dalam proses identifikasi duplikasi berkas lagu, selama data pengujian berada pada direktori utama atau sub-sub direktori dari direktori utama lokasi yang dipilih.
2. Hasil pengujian pada skenario pengujian nomor 2 menunjukkan kesesuaian dengan hasil yang diharapkan. Dalam penggunaan data pengujian 2 berkas lagu dengan nama dan isi yang sama, namun dengan *bitrate* yang berbeda menunjukkan bahwa perbedaan *bitrate* pada berkas lagu dengan nama dan isi yang sama akan menghasilkan audio sampel dengan ukuran yang berbeda karena perbedaan jumlah bit-bit pembentuk kedua lagu tersebut. Dalam proses ekstraksi sidik jari lagu, algoritma enkripsi *LIBOFA* dapat menghasilkan sidik jari lagu yang sama untuk kedua berkas lagu Hal ini menunjukkan bahwa proses quantisasi sinyal pada kedua lagu dapat dihasilkan perbandingan tetapan nilai yang sama, sehingga pada proses *transform* hingga membentuk sidik jari lagu, dapat dihasilkan sidik jari yang sama.
3. Hasil pengujian pada skenario pengujian nomor 3 menunjukkan kesesuaian dengan hasil yang diharapkan. Dalam penggunaan data pengujian 2 berkas lagu dengan isi dan *bitrate* yang sama namun dengan penamaan berkas lagu yang berbeda, ditunjukkan bahwa pada proses pencocokan berkas lagu tidak dilakukan pencocokan berdasarkan nama berkas-berkas lagu, melainkan dari ekstraksi isi berkas lagu tersebut. Dijelaskan pula bahwa kesamaan lokasi data pengujian tidak akan mempengaruhi dalam proses identifikasi duplikasi berkas lagu, sama seperti keadaan apabila berkas lagu data pengujian terdapat pada lokasi yang berbeda-beda.
4. Hasil pengujian pada skenario pengujian nomor 4 menunjukkan kesesuaian dengan hasil yang diharapkan. Dalam proses ekstraksi sidik jari lagu dapat dipastikan bahwa kedua sidik jari lagu data pengujian yang dihasilkan merupakan sidik jari yang berbeda, karena merupakan 2 lagu dengan isi yang berbeda.
5. Hasil pengujian pada skenario pengujian nomor 5 menunjukkan kesesuaian dengan hasil yang diharapkan, karena selain perbedaan lokasi tidak mempengaruhi hasil dari identifikasi duplikasi berkas lagu, hasil ekstraksi sidik jari lagu dari kedua berkas lagu data pengujian akan menunjukkan perbedaan antara kedua sidik jari berkas lagu tersebut, sebab kedua berkas lagu tersebut hanya memiliki kesamaan *bitrate*, tapi sebenarnya merupakan 2 jenis berkas lagu yang berbeda.

1. Kesimpulan Dan Saran

A. Kesimpulan

Dari hasil analisis yang dilakukan dapat ditarik kesimpulan sebagai berikut:

1. Aplikasi yang dibuat dapat melakukan identifikasi berkas-berkas lagu yang terduplikasi pada media penyimpanan lagu, baik untuk berkas lagu dengan *bitrate* yang sama maupun berbeda. Hal ini menunjukkan bahwa dalam proses quantisasi untuk menghasilkan audio sampel, algoritma *LIBOFA* dapat melakukan penyesuaian quantisasi sinyal sampel terutama pada lagu dengan *bitrate* yang berbeda, dan mengubahnya kedalam *sampling rate* yang sama. Kemudian pada proses *transform* yang dilakukan, algoritma sidik jari *LIBOFA* dapat melakukan penyesuaian perhitungan untuk mengubah nilai matriks awal dari perbandingan nilai-nilai pada pita-pita frekuensi terhadap nilai-nilai waktu, sehingga dapat dihasilkan sidik jari yang sama baik untuk lagu dengan *bitrate* yang sama maupun berbeda.
2. Aplikasi dapat digunakan untuk membantu pengguna dalam menemukan duplikasi berkas-berkas lagu yang terdapat pada media penyimpanan lagu, sehingga pengguna dapat menghapus duplikasi-duplikasi lagu tersebut.

B. Saran

Beberapa saran yang diajukan dengan kemungkinan untuk dilakukan pengembangan aplikasi lebih lanjut adalah sebagai berikut:

1. Pengembangan ke depannya disarankan untuk menggunakan komponen ekstraksi sidik jari lagu lainnya, sehingga dapat dilakukan perbandingan mana komponen yang lebih baik.
2. Pengembangan lebih lanjut dapat dicoba untuk melakukan ekstraksi sidik jari dari berkas lagu dengan kualitas yang sangat baik, seperti berkas lagu dengan kualitas orkestra atau *live music*.
3. Pengembangan lebih lanjut juga dapat dicoba dengan tipe format lagu yang berbeda seperti MP4, MIDI, DLL. Hal ini ditujukan untuk

4. menguji kemampuan *LIBOFA* dalam ekstraksi sidik jari dari berkas lagu selain MP3.
5. Faktor kecepatan dalam ekstraksi dan besarnya ukuran sidik jari lagu sebaiknya menjadi pertimbangan untuk pengembangan lebih lanjut, karena kedua hal tersebut cukup mempengaruhi waktu proses pencarian duplikasi berkas lagu.
6. Penambahan fitur-fitur baru pada pengembangan lebih lanjut seperti penambahan konfigurasi *threeshold* dalam mencocokan sidik jari, atau menggunakan ID3 Tag sebagai tambahan untuk membantu proses identifikasi berkas lagu.

2. Referensi

[1] Binanto, Iwan. Multimedia Digital Dasar Teori dan Pengembangannya. Yogyakarta: CV. Andi Offset, 2010

[2] Cano, Pedro, Eloi Batlle, Emilia Gomez, Leandro de C.T Gomes, and Madeleine Bonnet, 2005 "*Audio Fingerprinting: Concepts And Applications" Studies in: Computational Intelligence for Modeling and Prediction* Halgamuge, S.K, Lipo Wang. Berlin: Springer, 2005

[3] Elektro Indonesia, Audio Sampling, http://www.elektroindonesia.com/elektro/elek35a.html, diakses pada tanggal 1 mei 2011

[4] Fries, Bruce, and Marty Fries. Digital Audio Essentials Sebastopol: O`Reilly Media, Inc, 2005

[5] Kusumo, Ario Suryo. Buku Latihan Pemrograman Visual Basic 2005 Jakarta: PT.Elex Media Komputindo, 2006

[6] Music IP. MusicDNS Developers Guide Monrovia: Music IP Corporation, 2006

[7] Rainer, R.Kelly, Turban, Efraim, Introduction to Information Systems Danvers: John Wiley & Sons Inc, 2009

[8] Serrao, Carlos, Clara, Marco, 2007 "*Describing Acoustic Fingerprint Technology Integration For Audio Monitoring Systems" Studies in: Innovation And Advanced Techniques In Computer And Informatio*

[9] *Sciences And Engineering* Sobh, Tarek Dordrecht: Springer, 2007

[10] Watkinson, John, An Introduction to Digital Audio Oxford: Focal Press, 2002