BAB III

METODE DAN PERANCANGAN APLIKASI

3.1. Metode Penelitian

Metode berikut ini akan dilakukan oleh penulis dalam penelitian ini.

1. Studi Literatur

   Mempelajari berbagai referensi yang berhubungan dengan perumusan dan pemecahan masalah terkait dengan pengolahan sinyal digital, Algoritma Goertzel dan algoritma lain yang berhubungan dengan pengenalan suara serta metode lain yang lebih umum digunakan.

2. Pengumpulan dan analisa data

   Proses pengumpulan dan analisa ini berguna untuk mempelajari melodi, frekuensi suara pada tangga nada yang berbeda dan teori musik lainnya untuk dikaitkan dengan aplikasi yang akan dibuat penulis.

3. Perancangan aplikasi

   Pembuatan rancangan tampilan aplikasi dan konfigurasi. Desain kerangka antarmuka aplikasi sebelum diimplementasikan ke dalam bahasa pemrograman.

4. Pembangunan Aplikasi

   Mengembangkan dan merealisasikan aplikasi dengan menerjemahkan algoritma ke dalam suatu bahasa pemograman. Semua ini dilakukan di dalam sistem operasi Windows 7 dengan bahasa pemrograman C#.

5. Pengujian Aplikasi dan Implementasi

   Testing dan *debugging* aplikasi. Untuk mencari dan temukan kesalahan yang ada dalam kodingan aplikasi tersebut. Lalu dilanjutkan dengan membandingkan hasil dari aplikasi *music identifier* sejenis lainnya. Aplikasi ini diimplementasikan pada komputer *desktop* ataupun *laptop*.

6. Analisa sampel data

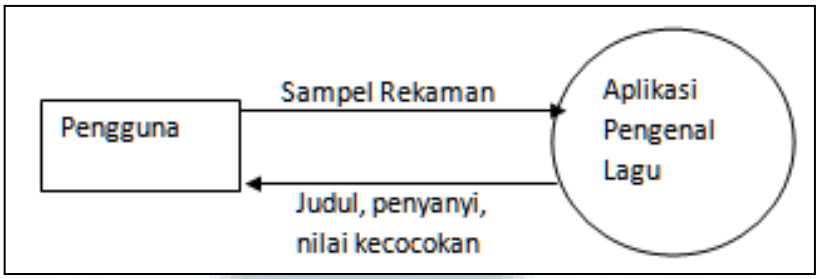
   Sementara pengujian aplikasi dilakukan, sampel data, baik masukan atau hasil keluaran (yang berupa data judul lagu yang memiliki kesesuaian dengan sampel data masukan) akan dianalisa untuk menentukan faktor-faktor yang mempengaruhi kinerja algoritma ataupun aplikasi yang dibuat.

3.2. Data Flow Diagram

Pembuatan Data Flow Diagram (DFD) ini bertujuan untuk menjelaskan hubungan antara aplikasi yang dibuat dengan entitas penggunanya. Perancangan DFD ini dibuat dalam 2 *level*, yaitu *level* konteks dan *level* 1.

3.2.1. DFD Level 0

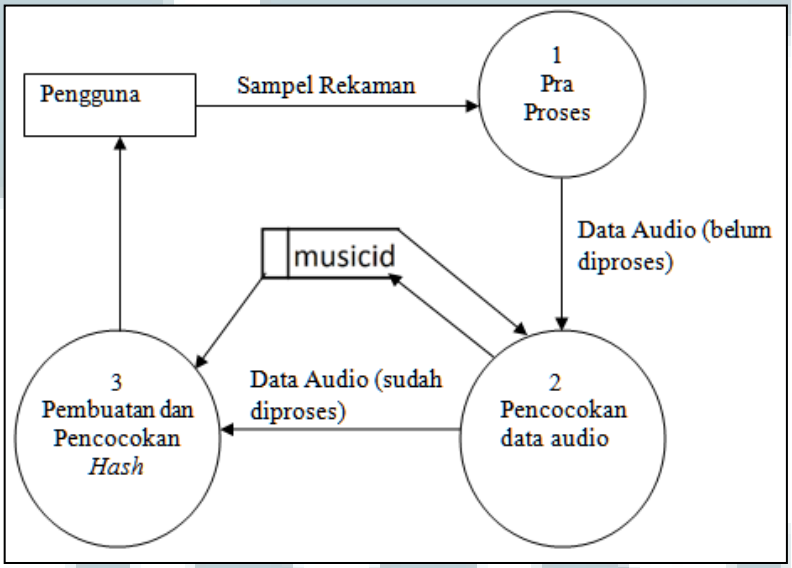
Diagram konteks ini menggambarkan aliran *input* dan *output* antara aplikasi dengan penggunanya. Pengguna cukup mencari sampel rekaman yang akan diproses aplikasi agar mendapatkan data judul dan penyanyi suatu lagu yang memiliki nilai kecocokan dengan sampel yang dipilih.



Gambar 3.1. DFD *Level* 0

3.2.2. DFD Level 1

Diagram ini menguraikan cara kerja aplikasi menjadi proses-proses yang lebih kecil. Proses-proses ini menyangkut *pre-processing*, pencocokan data audio dan pembuatan *hash*.



Gambar 3.2. DFD *Level* 1

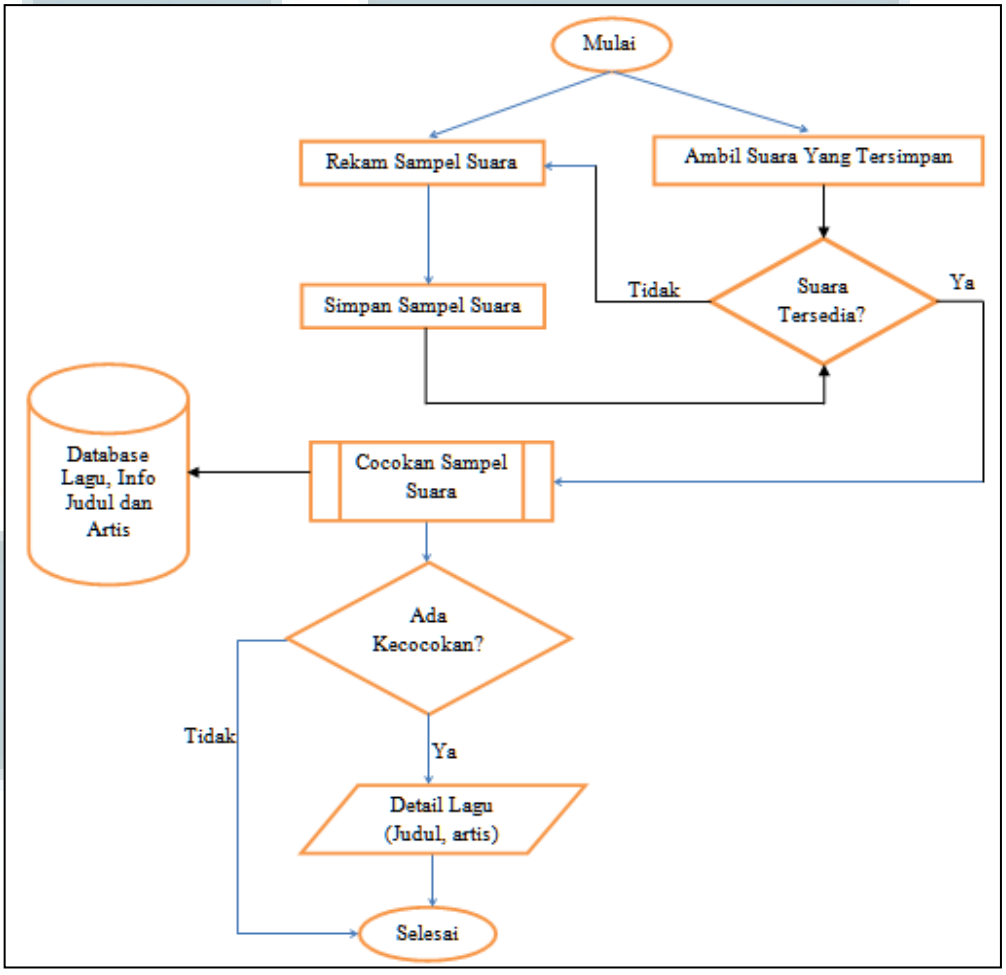
3.3. Cara Kerja

Aplikasi ini memiliki beberapa bagian proses dasar, yaitu proses identifikasi data (pra-proses), pencocokan data sesuai dengan data yang tersimpan

pada *database*, pembuatan dan pencocokan *hash* data audio, serta proses pengisian data audio ke dalam *database*.

3.3.1. Konteks Kerja Aplikasi Secara Umum

Pada proses identifikasi data, aplikasi ini akan meminta dahulu sampel suara yang akan dicocokan, baik itu merekam langsung maupun mencari sampel yang tersimpan. Namun setelah merekam, rekaman tersebut disimpan dahulu dalam format WAV sebelum dicocokan. Hasil rekaman ini memiliki *sample rate*, yaitu jumlah sampel yang diambil dalam satuan waktu (detik) dari sinyal yang diterima secara terus menerus, sebesar 44100 Hz.

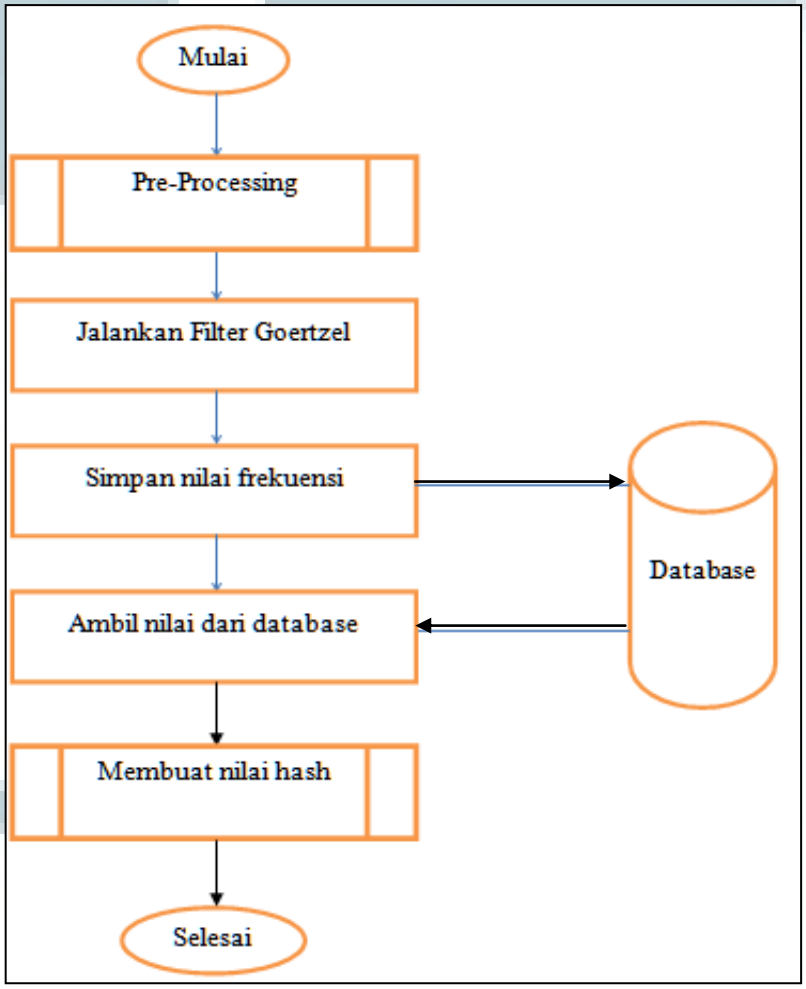


Gambar 3.3. *Flowchart* Konteks Cara Kerja Aplikasi

Setelah pencocokan dilakukan, program ini akan menampilkan data judul dan penyanyi lagu dari sampel yang tersimpan jika ditemukan kecocokan pola frekuensi. Data ini berupa lagu apa saja yang memiliki kecocokan dengan nilai dari sampel, beserta jumlah kecocokannya.

3.3.2. Pencocokkan Sampel Suara

Proses pencocokkan sampel suara ini dilakukan dengan menggunakan sebuah kelas *file reader* yang khusus memproses *file* berformat wav. Proses ini juga dapat berlangsung jika *reader* sudah mendapatkan nilai berupa direktori sampel yang berlaku.

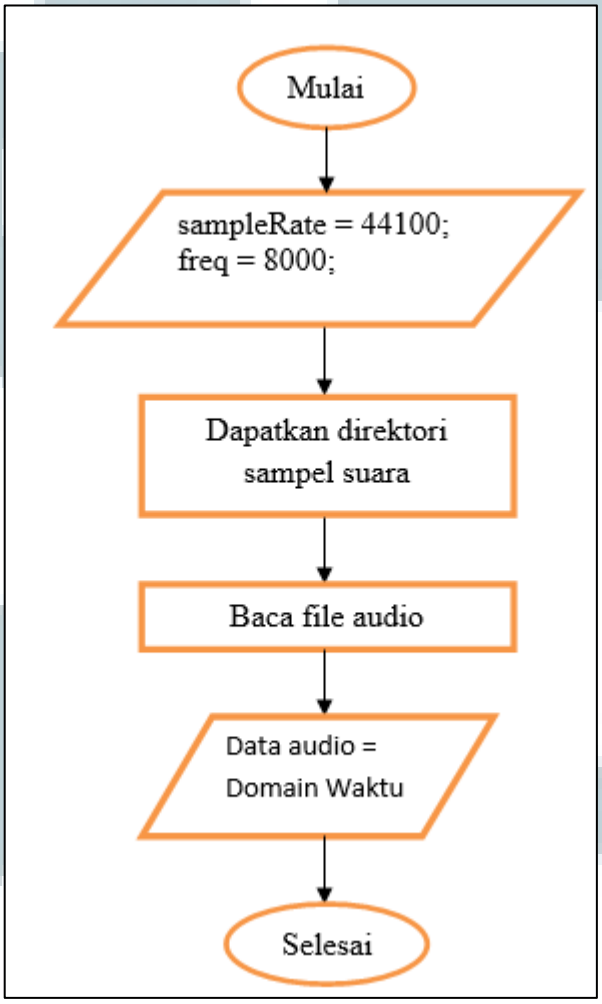


Gambar 3.4. *Flowchart* Proses Pencocokan

Proses ini dimulai dari sebuah proses persiapan untuk mendapatkan data audio dan menetapkan nilai *sample rate*, *buffer size* dan frekuensi yang digunakan dalam proses pengolahan sinyal. Kemudian, Algoritma Goertzel pun dijalankan untuk mengubah data audio tersebut menjadi nilai frekuensi yang akan disimpan ke dalam *database*.

Ketika pencocokan dilakukan, maka nilai yang tersimpan dalam *database* akan dikeluarkan dan dibuat *hash*-nya untuk proses pencocokan.

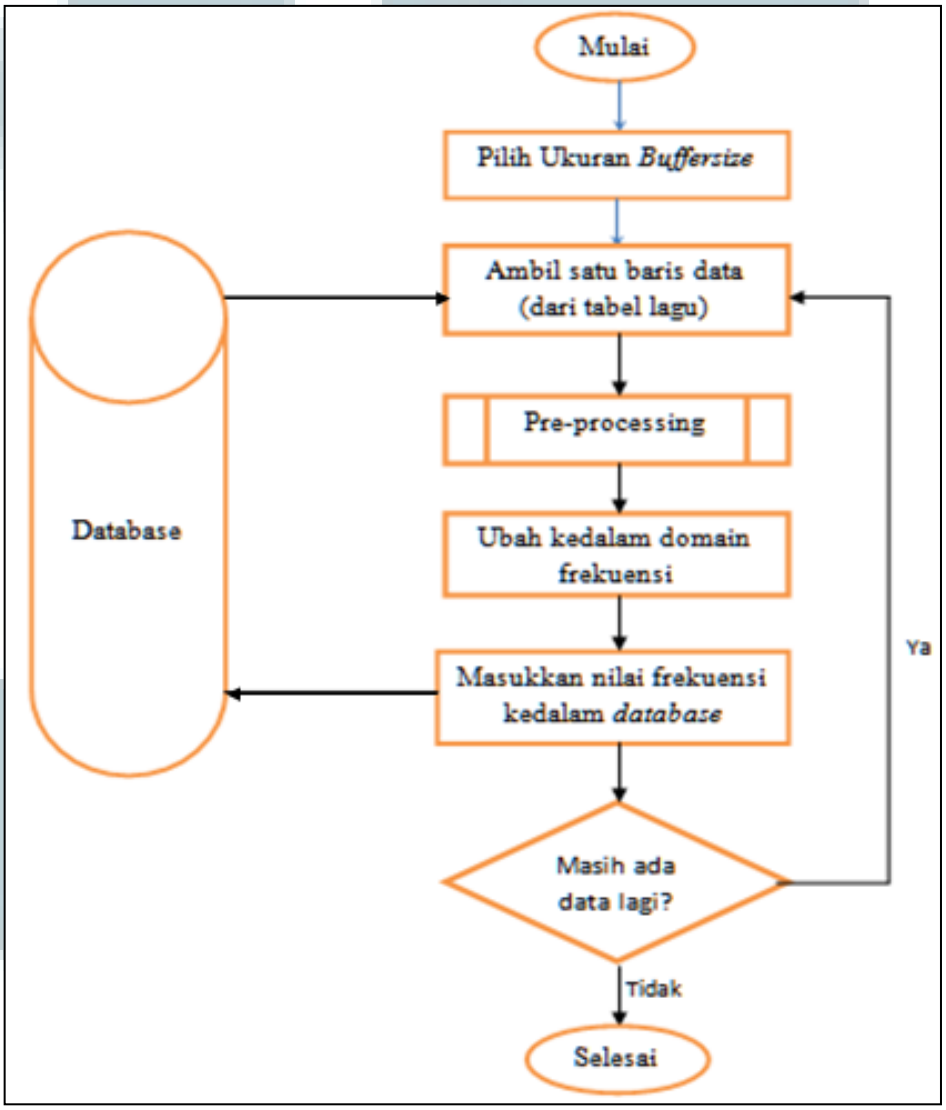
3.3.3. Pre-Processing



Gambar 3.5. *Flowchart* Proses Persiapan (*Pre-processing*)

Pre-processing dalam hal ini adalah proses persiapan pengenalan suara. Fungsinya untuk memproses sampel audio yang sudah didapatkan dengan menyesuaikan tingkat frekuensi, *buffer size* dan *sample rate* untuk mendapatkan nilai data audio sebelum diproses dengan algoritma. Dalam hal tersebut, *sample rate* 44100 dipilih karena angka 44100 merupakan *sampling rate* yang paling sering digunakan dalam media penyimpanan audio pada umumnya.

3.3.4. Proses Pengisian Data Audio ke dalam Database



Gambar 3.6. *Flowchart* Proses Pengisian ke dalam *Database*

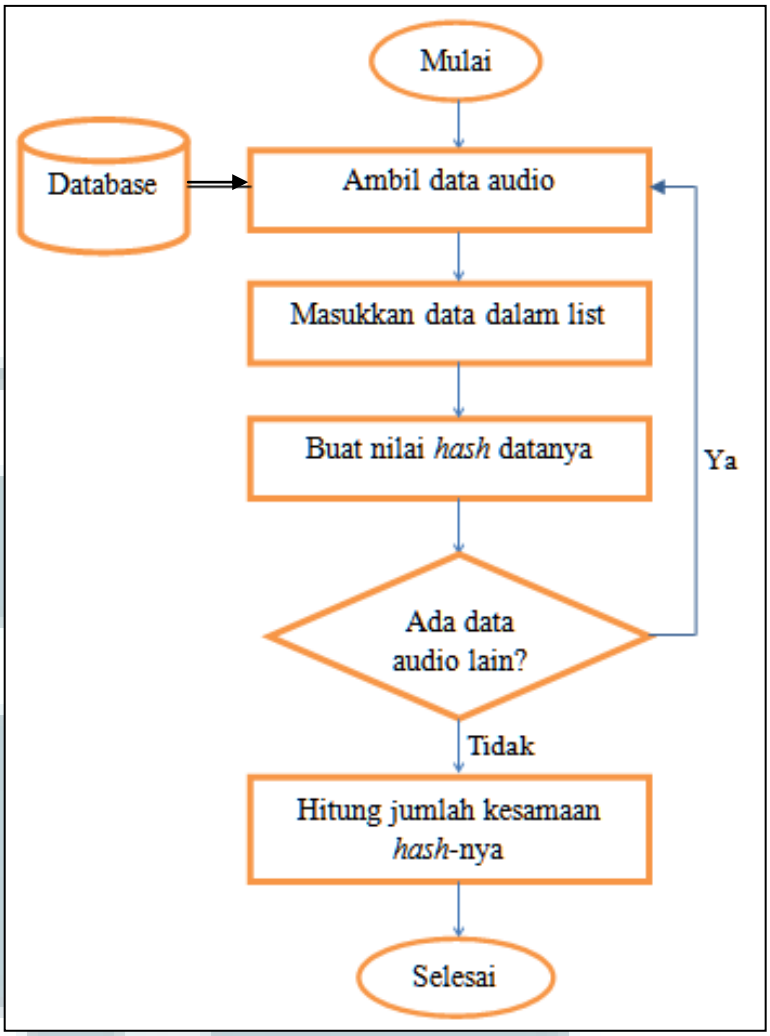
Proses ini dimaksudkan untuk memasukkan data-data audio dari lagu-lagu asli yang direktorinya tersimpan dalam sebuah tabel di *database* yang bersangkutan. Seperti pada proses awal pencocokan, proses pengisian ini dimulai dari *pre-processing*. Setelah data audio (domain waktu) didapatkan, maka data tersebut diolah ke dalam domain frekuensi yang lebih praktis dalam proses pencocokannya.

Kemudian, hasil data audio yang telah diolah tersebut disimpan ke dalam sebuah tabel terpisah dalam *database*. Pengisian ini dilakukan untuk semua lagu yang tersimpan dalam sebuah tabel dalam *database*.

3.3.5. Proses Pembuatan Hash

Proses ini mencakup pembuatan *hash* dari nilai-nilai audio yang telah diproses dalam *database*. Proses ini dilakukan terhadap nilai data audio dari sampel audio dan semua data audio dari lagu asli yang tersimpan dalam *database* setelah diolah oleh algoritma.

Dalam pembuatan *hash*, digunakanlah fungsi *hash* MD5 yang biasa dipakai dalam kriptografi, seperti pengamanan *password* dan pengujian integritas data yang bersifat rahasia. Sedangkan pencocokan hasil dilakukan dengan mencocokkan karakter-karakter dari hasil *hash* tersebut.



Gambar 3.7. *Flowchart* Proses Pembuatan Nilai Hash dan Pencocokan

3.4. Database

Aplikasi ini terhubung dengan *database* MySQL. Ada 6 tabel yang dibuat untuk mendukung kerja dari aplikasi ini, yaitu

- Songs

  Tabel ini menyimpan *file* lagu-lagu beserta judul dan penyanyinya. Rancangannya adalah sebagai berikut:

Tabel 3.1. Struktur Tabel Songs

Nama Kolom	Tipe Data	Karakter	Keterangan
Song_id	int	20	
Judul	varchar	30	
Penyanyi	varchar	40	
Genre	varchar	15	
Direktori	varchar	200	

- Hashed

Tabel ini menyimpan nilai perhitungan data audio dari sampel suara yang ditentukan. ID lagu dalam tabel ini bertipe data karakter, karena akan diisikan data non-angka ketika dimasukkan hasil *hash* dari audio sampel yang tidak tersimpan dalam *database*. Dalam tabel ini, hashed_value merupakan nilai hasil pengolahan data audio sebelum dilakukan *hashing*, sedangkan rekaman_ke merupakan urutan nilai blok audio yang diproses.

Tabel 3.2. Struktur Tabel Hashed

Nama Kolom	Tipe Data	Karakter	Keterangan
hashed_value	varchar	20	
rekaman_ke	int	10	

- hashed_sample

Tabel ini menyimpan data audio yang telah diproses dari direktori audio yang tersimpan dalam *database* sesuai dengan ID lagu yang tersimpan dan nomor urut blok sampelnya. Dalam tabel ini, ID lagu berbentuk int karena

isinya hanya nomor urutan song_id dari lagu yang tersimpan dalam *database*. Tabel ini dibuat untuk kepentingan pencocokan dengan *buffer size* sebesar 1000 satuan.

Tabel 3.3. Struktur Tabel hashed_sample

Nama Kolom	Tipe Data	Karakter	Keterangan
hashed_value	varchar	20	
song_id	int	10	
blok_ke	int	10	

- hashed_sample_full

  Tabel ini memiliki fungsi dan struktur yang sama dengan hashed_sample, namun dipakai untuk pencocokan dengan *buffer size* sebesar 8820 satuan (Seperti yang ditetapkan oleh program).

- hashed_sample_halfthousand

  Tabel ini memiliki fungsi dan struktur yang sama dengan hashed_sample, namun dipakai untuk pencocokan dengan *buffer size* sebesar 500 satuan.

- hashed_sample_hundred

  Tabel ini memiliki fungsi dan struktur yang sama dengan hashed_sample, namun dipakai untuk pencocokan dengan *buffer size* sebesar 100 satuan.

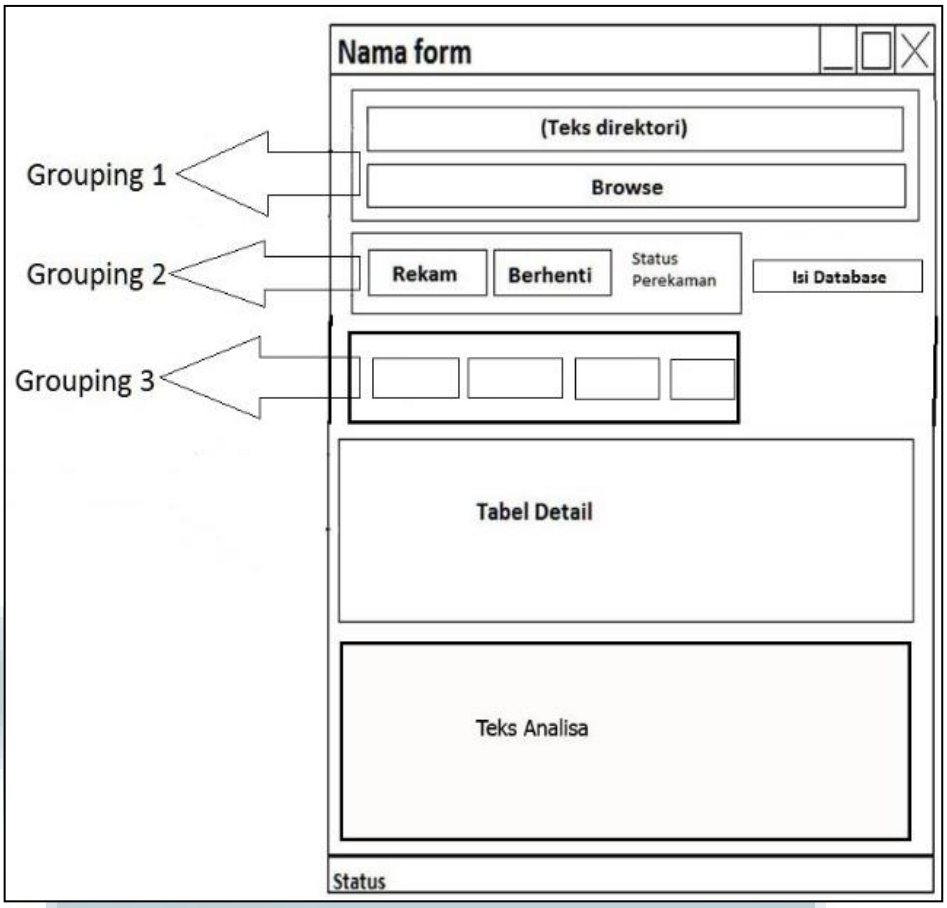
Tahap pengisian nilai dalam *database* ini adalah sebagai berikut:

- Tabel songs mula-mula sudah berisi sejumlah lagu yang akan digunakan sebagai bahan pencocokan dengan sampel.

- Tabel hashed mula-mula tidak berisi (kosong), dan akan diisikan dengan hasil olahan data audio sampel ketika pencocokan dilakukan. Namun ketika pencocokan sudah selesai, maka tabel ini akan dikosongkan kembali isinya.
- Tabel hashed_sample, hashed_sample_full, hashed_sample_halfthousand dan hashed_sample_hundred berisi data olahan audio dari lagu-lagu yang tersimpan dalam tabel songs sesuai dengan *buffer size* yang digunakan. Jika lagu yang dimasukkan belum pernah diambil nilainya, maka akan diisi juga nilai audio dari lagu yang baru masuk tersebut.

3.5. Antarmuka Aplikasi

Antarmuka aplikasi ini dibuat dengan cukup sederhana, karena aplikasi ini hanya berperan dalam hal merekam sumber bunyi dan mencocokkannya untuk mendapatkan judul lagu dan penyanyi yang sesuai. Semua judul lagu yang menemui kecocokan akan ditampilkan detailnya di sebuah *listview* beserta jumlah kecocokannya.



Gambar 3.8. Kerangka Rancangan Aplikasi

Dengan penjelasan bagian-bagian komponen sebagai berikut:

1. Grouping 1. Berupa sebuah komponen *grouping* yang berfungsi untuk mengelompokkan komponen-komponen yang berkaitan dengan pemilihan *input* direktori *file* audio yang akan dicocokkan. Isi dari group ini mencakup:
    a. Teks direktori

        Komponen ini berupa *textview* yang menampilkan direktori sampel suara yang ditentukan

b. Browse

Komponen ini akan membuat dialog *open file* untuk memilih sampel suara. Sampel yang bisa digunakan adalah *file* berformat wav.

2. Grouping 2. Berupa sebuah komponen *grouping* yang berfungsi untuk mengelompokkan komponen-komponen yang berkaitan dengan proses merekam suara. Ketika suara tersebut direkam, suara akan langsung disimpan ke dalam sebuah *file* yang ditentukan sebelum dianalisa datanya. Isinya mencakup:

   a. Rekam

      Berupa sebuah tombol untuk merekam sampel suara, sekaligus langsung menyimpannya. Selama proses perekaman berlangsung, semua tombol (selain tombol berhenti) tidak bisa diklik sampai proses rekaman berakhir.

   b. Berhenti

      Tombol ini berguna untuk mengakhiri proses perekaman sampel suara.

   c. Status perekaman

      Berupa sebuah label yang menampilkan sebuah teks keterangan bahwa proses perekaman sedang berlangsung.

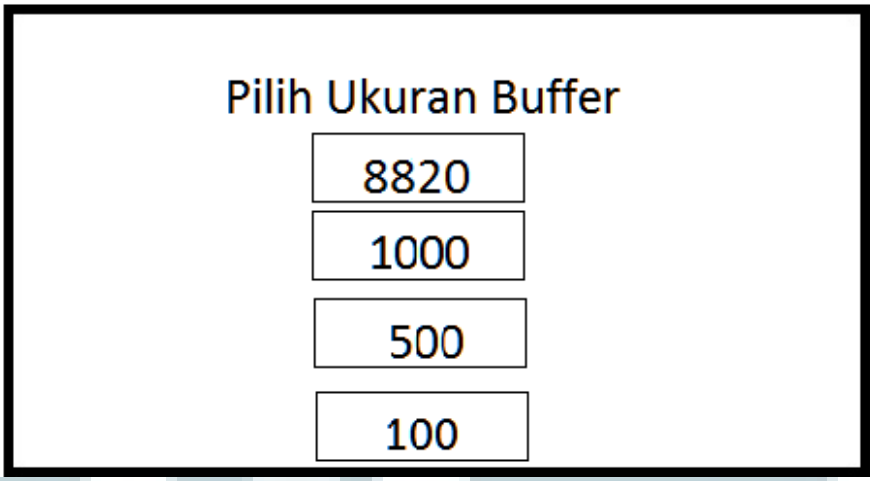
3. Grouping 3. Berupa komponen *grouping* yang berisi 4 tombol untuk memulai proses pencocokan sampel. Tombol ini berupa angka yang bertuliskan jumlah *buffer size* (jumlah area memori yang dialokasikan untuk menyimpan data maupun informasi yang diproses) yang digunakan

untuk pemrosesan sampel. *Buffer size* yang digunakan dalam aplikasi ini ada yang berjumlah 8820, 1000, 500 dan 100 satuan. Jika ukuran ini semakin kecil, maka kinerja perangkat akan lebih cepat, namun akan menambah beban perangkat pada proses pencocokannya karena sampel yang lebih kecil jumlahnya.

4. Tabel Detail, yang berguna untuk menampilkan daftar lagu-lagu beserta penyanyinya yang mengalami kecocokan dengan sampel suara yang diambil.
5. Isi *Database*. Berupa sebuah tombol yang digunakan untuk mengisi data audio dari lagu-lagu yang tersimpan dalam *database*. Tombol ini dipakai jika ada pembaharuan isi dari tabel songs dalam *database*. Saat tombol ini ditekan, maka isi yang lama akan dihapus terlebih dahulu untuk digantikan dengan data audio yang baru. Pengisian ke dalam *database* ini juga terdiri atas 4 pilihan berdasarkan *buffer size*-nya, yaitu pilihan pengisian sebanyak 8820 satuan, 1000 satuan, 500 satuan dan 100 satuan.
6. Teks Analisa. Berupa sebuah *textbox* yang dipakai untuk menjelaskan proses yang terjadi selama pengenalan lagu, pengenalan dan pengisian data audio berlangsung. Dalam teks ini, dijelaskan berapa nilai data audio yang didapatkan dan nilai pencocokannya.
7. Status, yang berfungsi sebagai *status bar* untuk memberikan keterangan tentang proses yang terjadi selama audio diolah.

Selain *form* utama tersebut, ada satu lagi *form* kecil yang dipakai untuk menentukan pilihan pengisian data audio sesuai dengan *buffer size*-nya. *Form* ini

hanya berisi 4 tombol yang menentukan pemilihan *buffer size* yang akan diisi dengan data audionya



Gambar 3.9. Rancangan *Form* Pilihan *Buffer size* dalam Pengisian *Database*