BAB III

METODE DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Penelitian ini merupakan *applied project* dengan menggunakan metode sebagai berikut.

1. Studi Pustaka

   Mempelajari artikel, jurnal serta penelitian-penelitian lain yang berhubungan dengan penelitian ini, dan melakukan eksperimen terhadap algoritma ini.

2. Desain dan Implementasi Aplikasi

   Mencatat fungsi-fungsi yang mungkin terdapat pada aplikasi, serta merancang tampilan aplikasi. Dari rancangan tersebut, dilakukan pencarian *library* yang sesuai untuk digunakan dalam pengembangan aplikasi, sehingga selanjutnya aplikasi dapat mulai dibangun.

3. *Testing* dan *Debugging* Aplikasi

   Sejalan dengan pembuatan aplikasi, dilakukan serangkaian pengujian untuk memastikan aplikasi berjalan sesuai dengan tujuan dan memperbaiki berbagai kesalahan-kesalahan pada aplikasi.

4. Penulisan Laporan

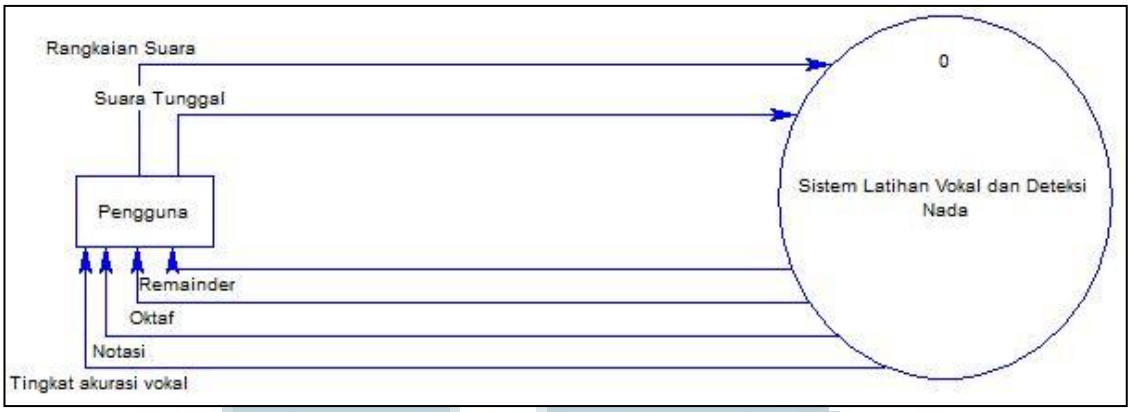
   Setelah aplikasi berhasil dibangun dan telah berjalan sesuai dengan tujuan, laporan ditulis untuk menjelaskan penelitian yang dilakukan serta kesimpulan dari hasil penelitian tersebut.

3.2 Rancangan Aplikasi

Aplikasi yang akan dibangun terdiri dari dua fitur utama, yaitu latihan vokal dan deteksi nada secara *real-time*. Latihan vokal pada aplikasi yang dibangun mencakup tiga dari enam aspek pada bernyanyi, yaitu akurasi nada, *range* vokal dan stabiltias suara.

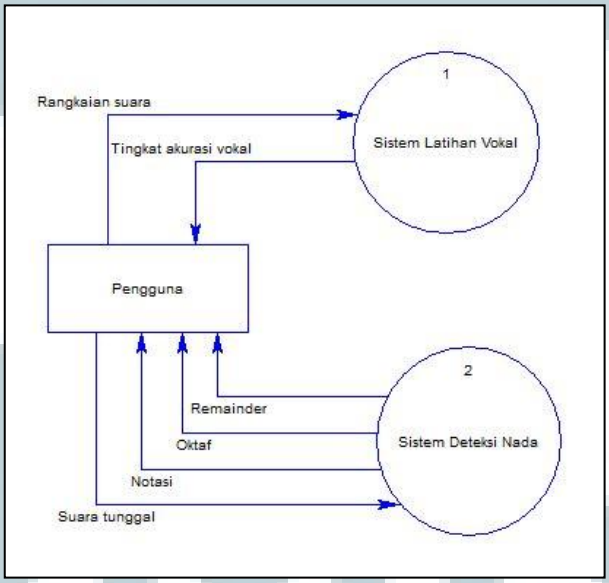
Awalnya aplikasi akan memainkan sebuah set lagu dan menampilkan notasi lagu tersebut dalam bentuk grafik dan tulisan. Akurasi dan stabilitas vokal pengguna aplikasi kemudian dilatih dengan cara pengguna diminta membunyikan ulang nada tersebut dengan nada dan ritme yang tepat. Aplikasi yang dibangun juga dilengkapi dengan fitur pergantian nada dasar, yang mampu membuat nada pada set lagu sebelumnya semakin tinggi ataupun rendah, sehingga memungkinkan pengguna untuk berlatih memperluas jangkauan vokalnya. Untuk mendapatkan hasil yang maksimal, fitur pergantian nada dasar dapat digunakan dengan bantuan alat musik seperti piano atau gitar.

Aliran data pada aplikasi berupa masukan rangkaian suara serta suara tunggal dari pengguna kepada sistem, kemudian sistem akan mengubah masukan tersebut menjadi tingkat akurasi vokal, notasi, oktaf, serta sisa yang menunjukkan jarak antara suara yang dibunyikan dengan nada eksak terdekat. Gambar 3.1 menunjukkan diagram alir data sistem ini.



Gambar 3.1 DFD Level 0 Sistem Latihan Vokal dan Deteksi Nada

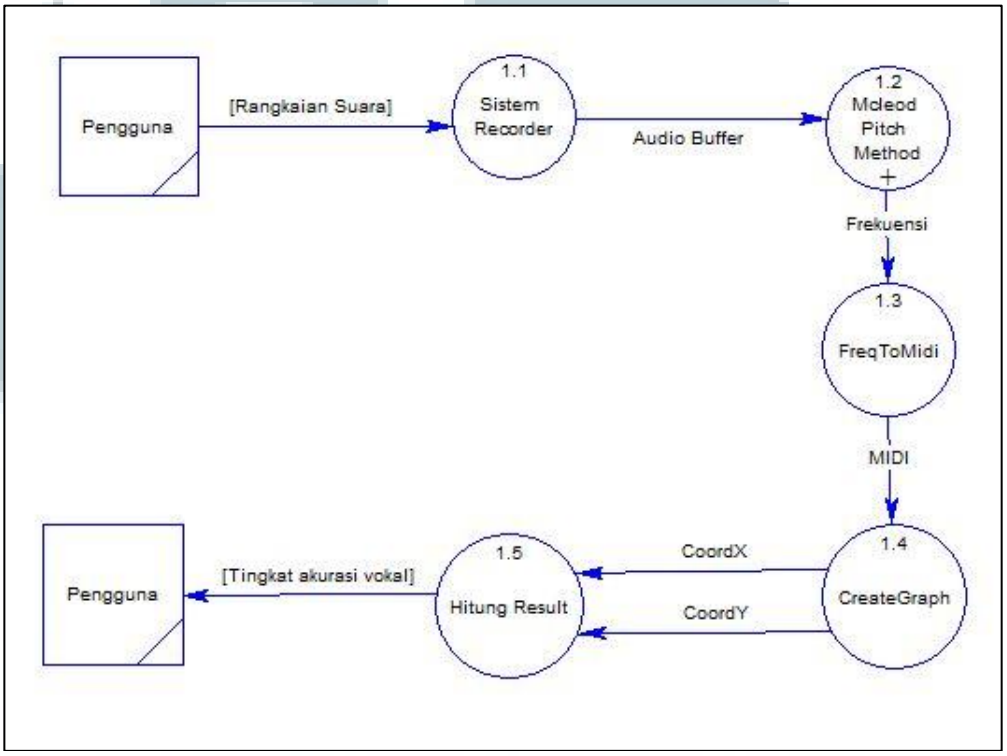
Sistem kemudian dapat dipecah menjadi dua subsistem yang lebih kecil, yaitu sistem latihan vokal yang menerima masukan berupa rangkaian suara dan mengubahnya menjadi tingkat akurasi vokal, serta sistem deteksi nada yang menerima masukan berupa suara tunggal dan mengubahnya menjadi notasi, oktaf, dan sisa yang menunjukkan jarak antara suara yang dibunyikan dengan nada eksak terdekat. Gambar 3.2 menunjukkan diagram alir data selanjutnya.



Gambar 3.2 DFD Level 1 Sistem Latihan Vokal dan Deteksi Nada

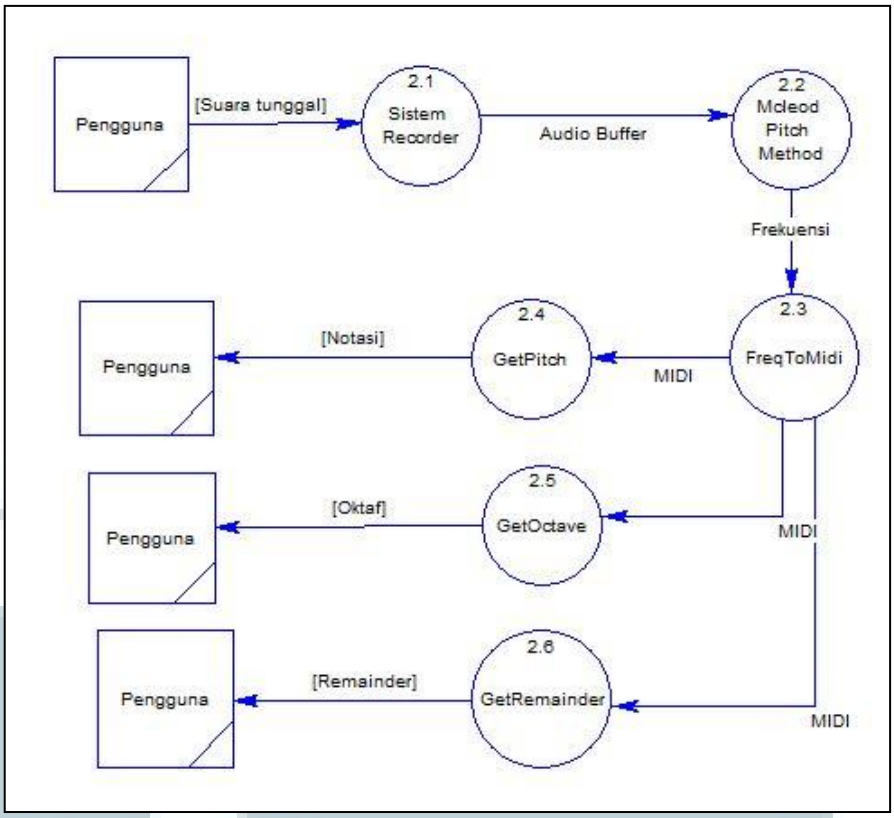
Sistem latihan vokal yang terdapat pada gambar 3.2 dapat dipecah lagi menjadi subsistem yang lebih kecil. Awalnya, rangkaian suara akan diubah menjadi *audioBuffer* melalui sistem *recorder*. Kemudian, algoritma Mcleod akan

mengolah *audioBuffer* tersebut menjadi sebuah frekuensi. Selanjutnya, frekuensi yang didapat akan diubah ke dalam bentuk *MIDI not numbering* melalui proses *FreqToMidi*. Kemudian, *midi* yang didapat tersebut akan dimasukkan ke dalam grafik dan menghasilkan titik koordinat x dan y. Koordinat tersebut akan diolah melalui proses *HitungResult* untuk menghasilkan *result*. Gambar 3.3 menunjukkan diagram alir data dari sistem latihan vokal.



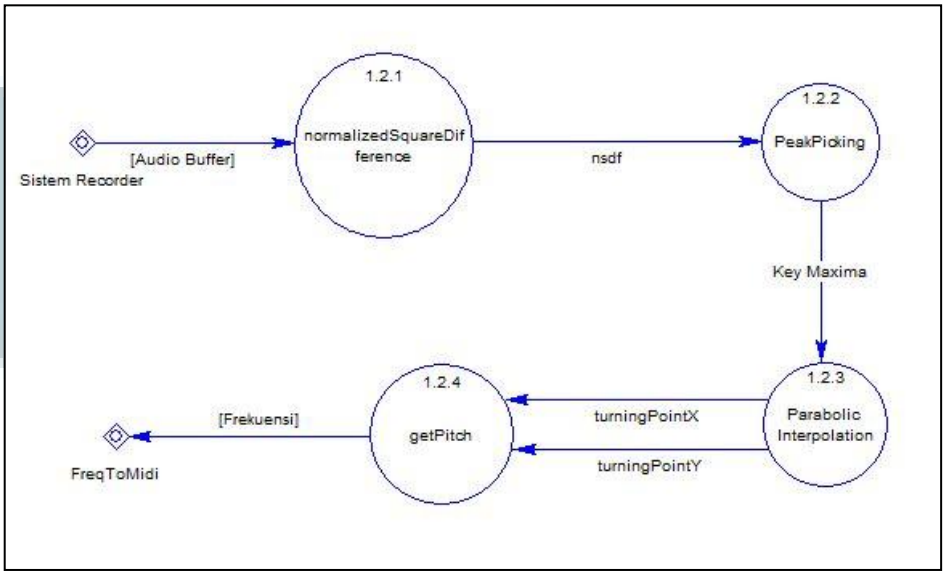
Gambar 3.3 DFD Level 2 Sistem Latihan Vokal

Sistem deteksi nada yang terdapat pada gambar 3.2 juga dapat dipecah ke dalam beberapa subsistem yang lebih kecil. Awalnya, suara tunggal akan dikonversikan ke dalam bentuk *audio buffer* melalui *sistem recorder* dan diubah ke dalam bentuk frekuensi dengan menggunakan algoritma Mcleod. Selanjutnya, frekuensi akan diubah menjadi *midi* melalui proses *FreqToMidi*. Kemudian, dari *midi* tersebut dapat dihasilkan notasi, oktaf, dan sisa yang akan diberikan kepada pengguna. Gambar 3.4 menunjukkan diagram alir data selanjutnya.



Gambar 3.4 DFD Level 2 Sistem Deteksi Nada

Proses *Mcleod Pitch Method* yang terdapat pada gambar 3.3 dapat dipecah lagi menjadi beberapa subproses, yaitu *normalized square difference, peak picking, parabolic interpolation, serta getPitch* yang ditunjukkan pada gambar 3.5.



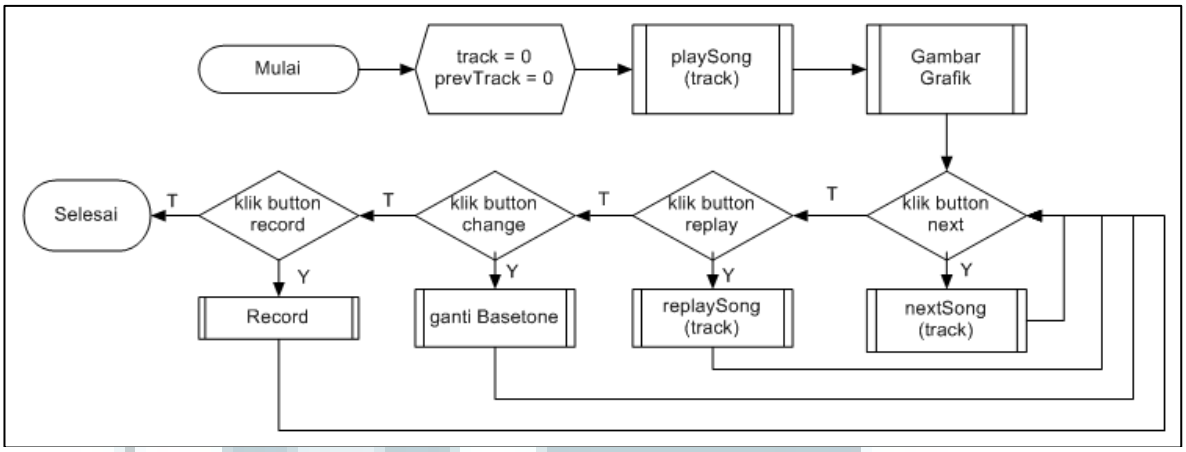
Gambar 3.5 DFD Level 3 Proses *Mcleod Pitch Method*

Proses *Mcleod Pitch Method* dimulai dengan perhitungan *normalized square difference* untuk setiap *audioBuffer*. Dari *normalized square difference* tersebut, kemudian dicari *key maxima*. Selanjutnya, akan dicari *turningPointX* dan *turningPointY* yang merupakan peningkatan akurasi dari titik *key maxima* dengan menggunakan *parabolic Interpolation*. Dari *turningPointX* dan *turningPointY* tersebut, dapat dicari nilai frekuensi.

3.2.1 Tampilan Utama (Latihan Vokal)

Tampilan latihan vokal akan langsung muncul ketika pengguna membuka aplikasi. Pada awalnya, tampilan ini akan memainkan sebuah set lagu dan menampilkan notasi dari set lagu tersebut ke dalam bentuk grafik dan teks. Terdapat 3 buah set lagu pada aplikasi ini, yaitu 1-2-3-4-5-5-4-3-2-1, 1-5-4-5-3-5-2-5-1, dan 1-3-5-3-1-4-6-4-1-5-5-1-1-7-1. Set tersebut didapat dari Aaron Matthew Lim, seorang penyanyi, guru vokal, produser musik, dan penulis lagu dalam situsnya www.your-personal-singing-guide.com.

Setelah memilih set yang diinginkan, pengguna aplikasi dapat mengganti nada dasar ketiga set tersebut sesuai dengan yang dikehendakinya. Setelah semua persiapan dilakukan, pengguna dapat memulai proses rekaman dengan menekan tombol *record*. Setelah proses rekaman selesai, akan ditampilkan hasil berupa persentase yang menunjukkan kemiripan antara nada yang dibunyikan dengan set lagu sebenarnya. Gambar 3.6 menunjukkan diagram alur dari detail proses tampilan utama.



Gambar 3.6 *Flowchart* Tampilan Utama

3.2.2 Tampilan Deteksi Nada

Tampilan ini berisi fitur deteksi nada. Ketika tampilan ini dibuka, proses pengolahan audio akan langsung dijalankan dan mengkonversi masukan suara menjadi frekuensi. Kemudian frekuensi tersebut akan diolah menjadi notasi, oktaf, dan penunjuk yang menunjukkan kelebihan atau kekurangan dari nada masukan dengan nada eksak terdekat.

Frekuensi awalnya diubah menjadi *MIDI not number* dengan menggunakan rumus 2.1. Untuk mendapatkan notasi, MIDI yang didapat kemudian dibulatkan ke bilangan bulat terdekat kemudian dimodulus 12. Hasil dari konversi MIDI ke notasi dapat dilihat pada tabel 3.1.

Tabel 3.1 Tabel Konversi Notasi

Hasil	Notasi
0	C
1	C#
2	D
3	D#
4	E
5	F
6	F#
7	G
8	G#
9	A

Tabel 3.1 Tabel Konversi Notasi (Lanjutan)

Hasil	Notasi
10	A#
11	B

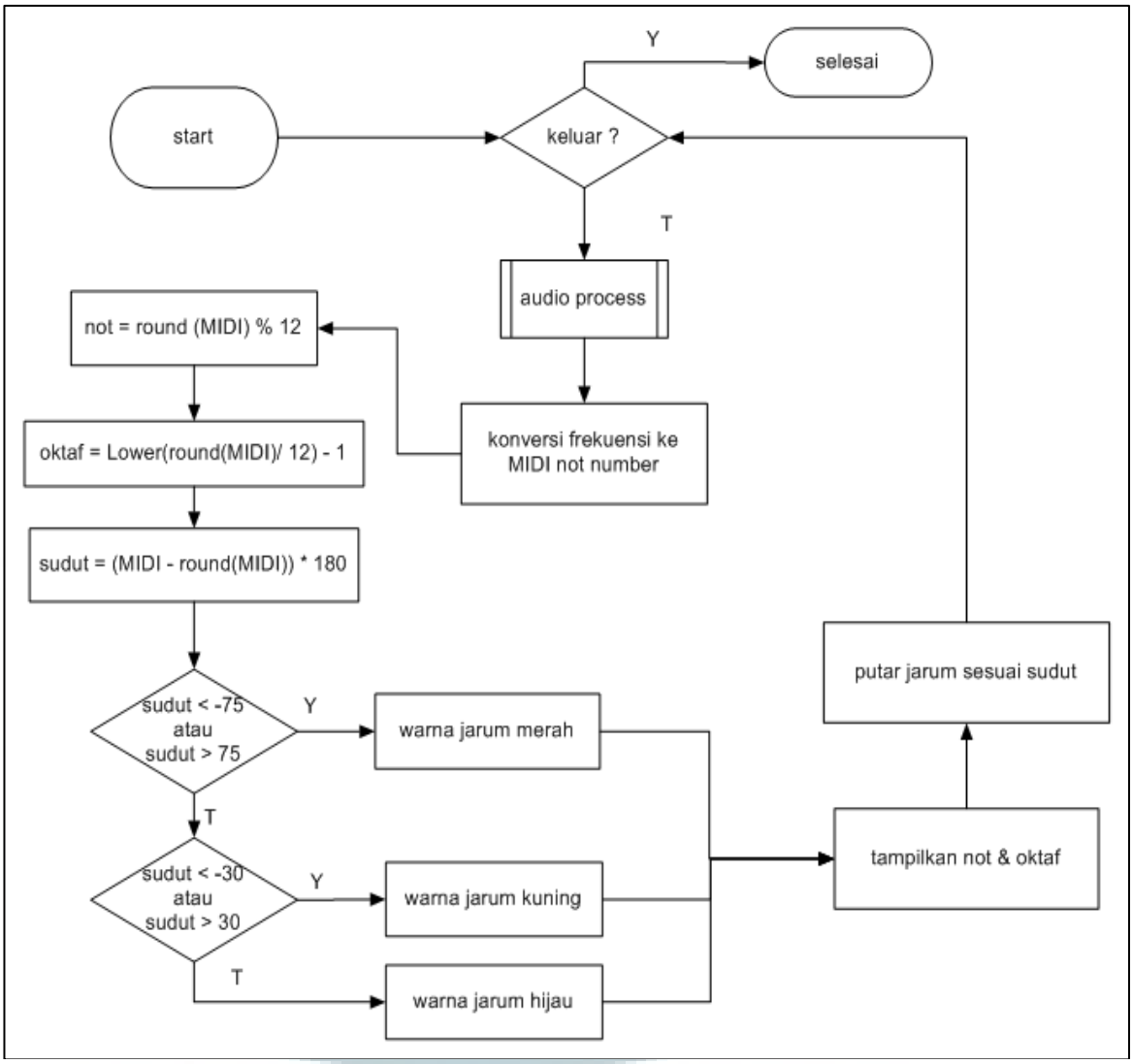
Setelah itu, untuk mendapatkan oktaf, MIDI yang dibulatkan dibagi dengan 12, kemudian dibulatkan ke bawah dan dikurangi 1. Rumus 3.1 menunjukkan notasi matematis dari persamaan di atas.

$$Oktaf = \left\lfloor \frac{[MIDI]}{12} \right\rfloor - 1 \quad \text{...Rumus 3.1}$$

Penunjuk menunjukkan perbandingan antara MIDI yang dibunyikan dengan MIDI yang terdekatnya. Pada awalnya, penunjuk diletakkan tegak lurus secara vertikal pada layar. Putaran penunjuk ke kiri menunjukkan MIDI yang dibunyikan lebih rendah dibandingkan MIDI eksak terdekat, dan sebaliknya, putaran ke kanan menunjukkan lebih tinggi dari nada eksak terdekat, dengan *range* perputaran -90 < x < 90 derajat. Untuk mencapai seperti hasil yang diharapkan, maka digunakan rumus 3.2 untuk mencari sudut putar.

$$Sudut\ Putar = (MIDI - [MIDI]) * 180 \quad \text{...Rumus 3.2}$$

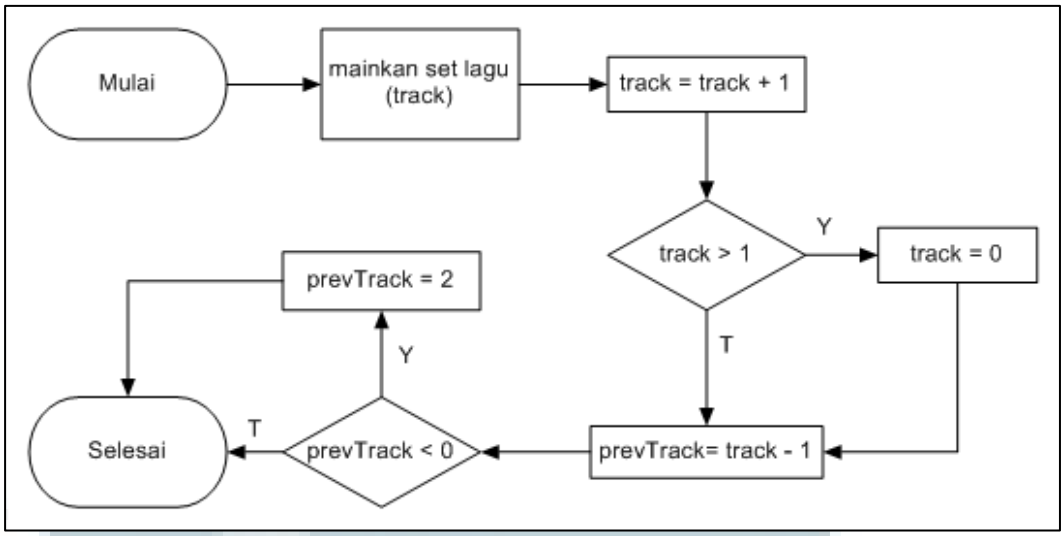
Warna penunjuk akan turut berganti sesuai dengan sudut putar yang dimilikinya, dengan warna merah apabila sudut < -75 atau sudut > 75, kuning apabila sudut < -30 atau sudut > 30, dan hijau apabila sudut dalam *range* -30 < x < 30. Gambar 3.7 menunjukkan diagram alir dari keseluruhan proses tampilan deteksi.



Gambar 3.7 *Flowchart* Tampilan Deteksi Nada

3.2.3 Play Song

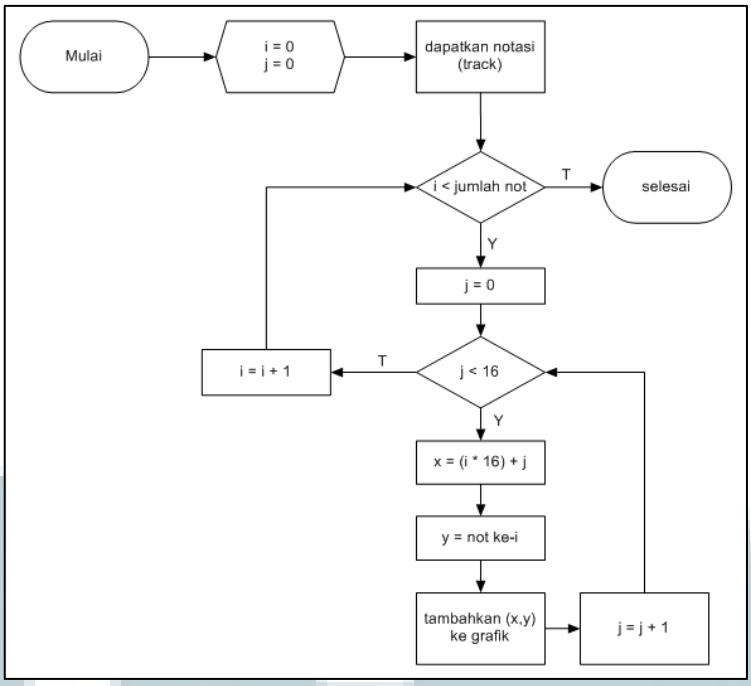
Play Song merupakan proses yang berfungsi untuk memainkan sebuah set lagu. Terdapat dua variabel utama dalam memainkan dan menghentikan set lagu, yaitu *track* dan *prevTrack*. *Track* digunakan untuk memutar lagu, dan akan bertambah setiap lagu diputar, sementara *prevTrack* digunakan untuk menghentikan lagu yang telah diputar sebelumnya. Gambar 3.8 menunjukkan diagram alir dari proses ini.



Gambar 3.8 *Flowchart Play Song*

3.2.4 Gambar Grafik

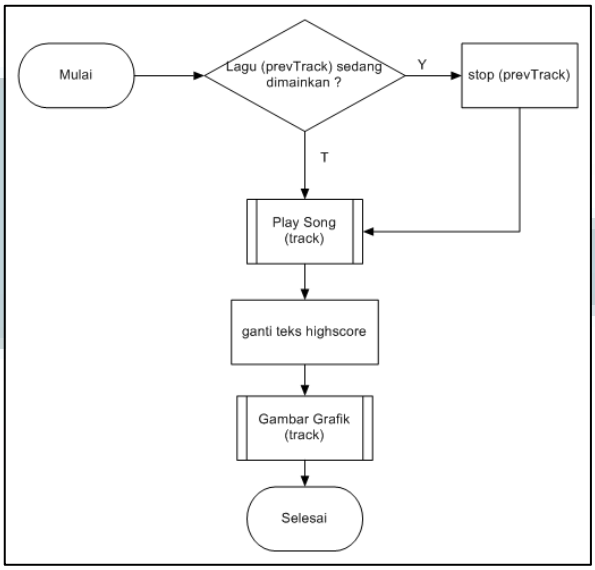
Proses pembuatan grafik dibantu oleh *library graphview4.0*. Koordinat x pada grafik menunjukkan panjang suatu set lagu. Untuk mengubah suatu masukan suara hingga menjadi sebuah titik pada grafik diperlukan waktu sebesar ~0.05 detik. Oleh karena itu, sebuah titik x pada grafik setara dengan 0.05 detik. Seluruh set lagu pada aplikasi ini terdiri dari nada-nada dengan interval 0.8 detik. Oleh dari itu, satu nada direpresentasikan ke dalam 16 titik x (0.8 / 0.05). Koordinat y menunjukkan tinggi nada dalam format *MIDI not numbering*. Gambar 3.9 menunjukkan diagram alur dari proses penggambaran grafik.



Gambar 3.9 *Flowchart* Gambar Grafik

3.2.5 Next Song

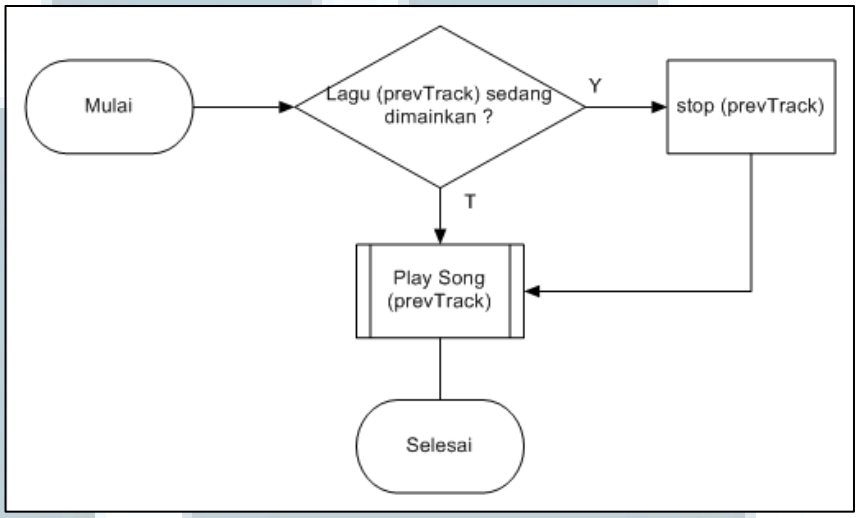
Proses ini berfungsi untuk menghentikan set lagu saat ini apabila sedang dimainkan, kemudian memainkan set lagu berikutnya. Proses ini dilakukan ketika pengguna aplikasi menekan tombol *next*. Gambar 3.10 menunjukkan diagram alir proses ini.



Gambar 3.10 *Flowchart Next Song*

3.2.6 Replay Song

Proses ini berfungsi untuk menghentikan set lagu saat ini apabila sedang dimainkan, kemudian memutar ulang set lagu tersebut. Proses ini dilakukan ketika pengguna aplikasi menekan tombol *replay*. Gambar 3.11 menunjukkan diagram alir proses ini.

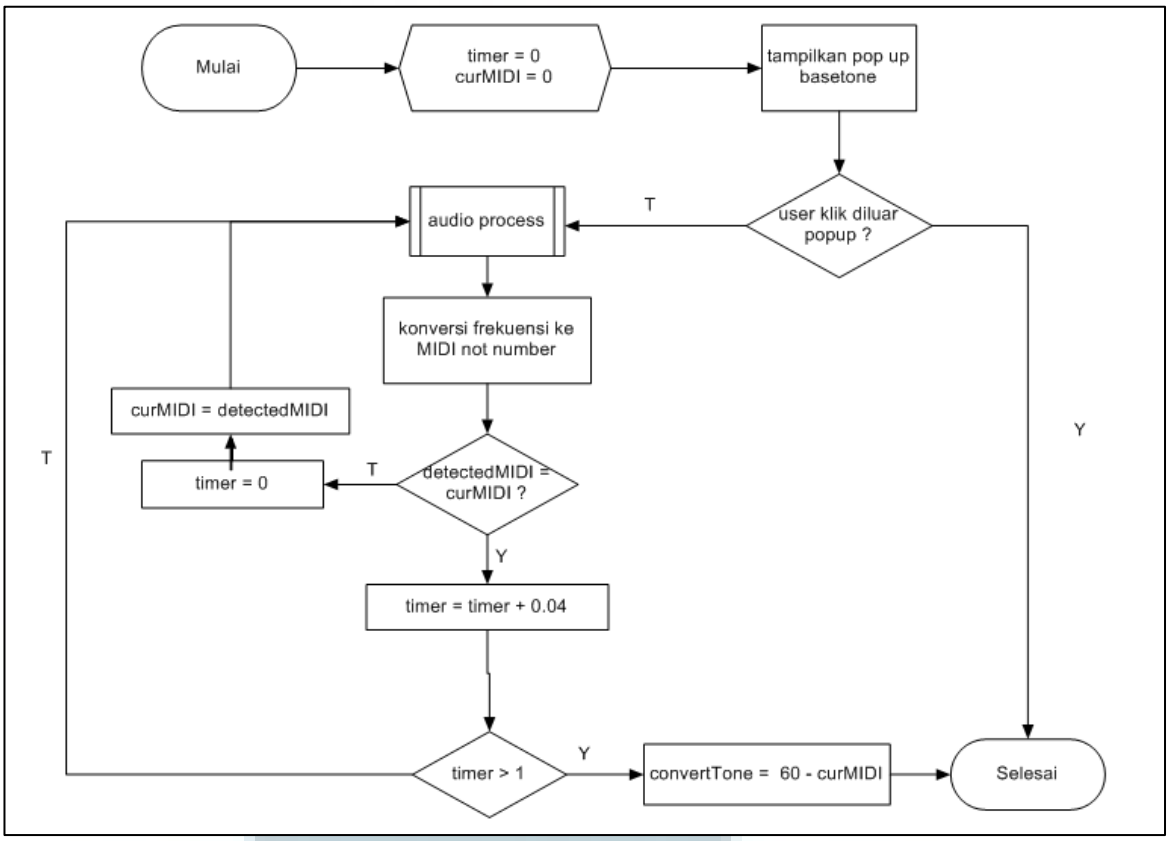


Gambar 3.11 *Flowchart Replay Song*

3.2.7 Ganti Basetone

Pergantian nada dasar memiliki fungsi berdasarkan penggunanya. Untuk pengguna awam, pergantian nada dasar dapat mempermudah proses latihan suara, dengan cara mengubah nada dasar ke nada dasar pengguna. Untuk pelajar musik, pergantian nada dasar dapat digunakan untuk latihan meningkatkan jangkauan suara, maupun untuk latihan ketepatan nada. Menurut McFerron, seorang profesor musik, C4 merupakan *middle C* yang terletak di tengah keyboard dan terdapat di seluruh tingkatan vokal *range (*bass, alto, bariton, mezzo sopran, tenor dan sopran). Oleh karena itu, nada dasar awal yang dipilih adalah C4 (*MIDI not number* : 60). Hasil akhir dari proses ini adalah sebuah bilangan yang bernama

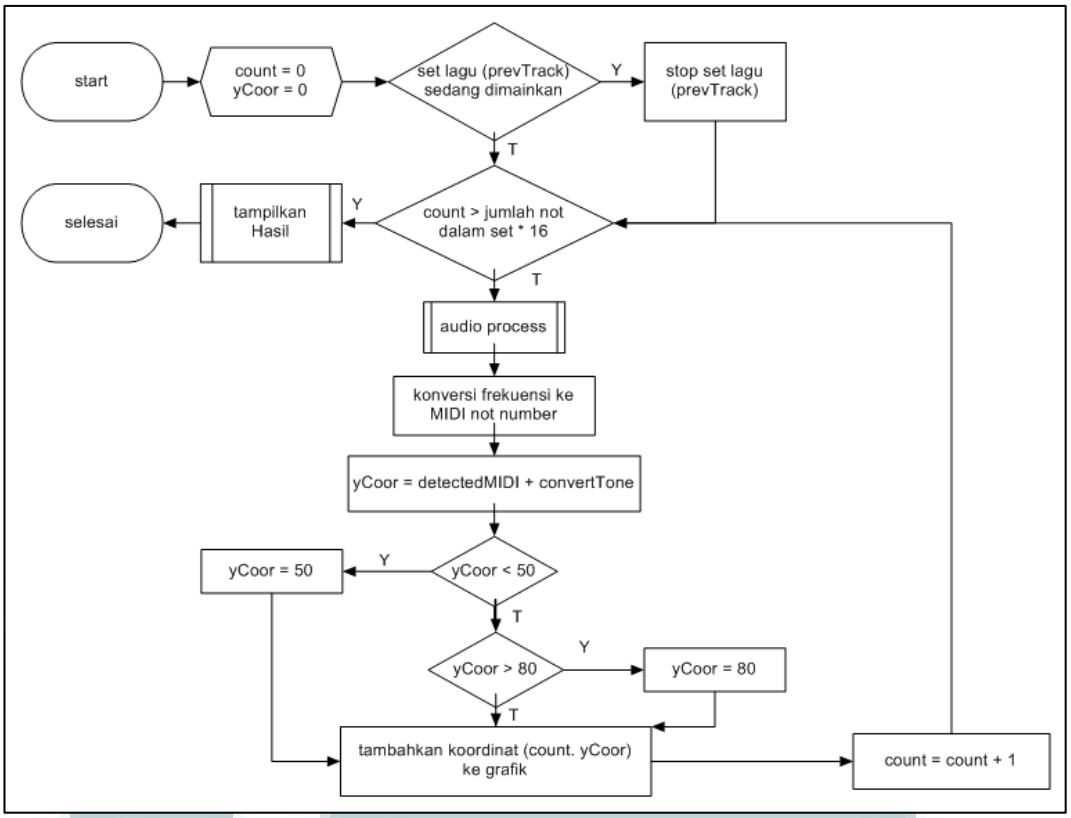
convertTone yang akan bernilai negatif apabila pengguna mengganti ke nada yang lebih tinggi dari C4 dan positif apabila pengguna mengganti ke nada yang lebih rendah dari C4. Gambar 3.12 menunjukkan diagram alir proses ini.



Gambar 3.12 *Flowchart* Ganti *Basetone*

3.2.8 Record

Proses *record* berfungsi untuk mengkonversi masukan suara dari pengguna ke dalam bentuk grafik. Setelah itu, grafik antara set lagu dan suara pengguna akan dibandingkan dan menghasilkan *result* dalam bentuk persentase. Proses *record* akan dijalankan ketika pengguna menekan tombol *record*. Gambar 3.13 menunjukkan diagram alur detail dari proses *record.*



Gambar 3.13 *Flowchart Record*

3.2.9 Tampilkan Hasil

Result merupakan hasil yang menunjukkan tingkat akurasi vokal pengguna, yang didapat dari perbandingan antara set nada dari masukan pengguna dengan set nada sebenarnya yang telah diubah ke dalam bentuk grafik. Perbandingan dilakukan untuk setiap titik pada grafik sampai sejumlah keseluruhan titik pada grafik. Perbandingan tersebut akan menghasilkan nilai dengan rentang 0 hingga 100 untuk masing-masing titik. Semakin besar selisih dari kedua grafik semakin kecil nilai yang didapatkan, sebaliknya semakin kecil selisih dari kedua grafik semakin besar nilai yang didapatkan. Untuk mendapatkan hasil seperti yang diharapkan, maka digunakan rumus 3.3 di bawah.

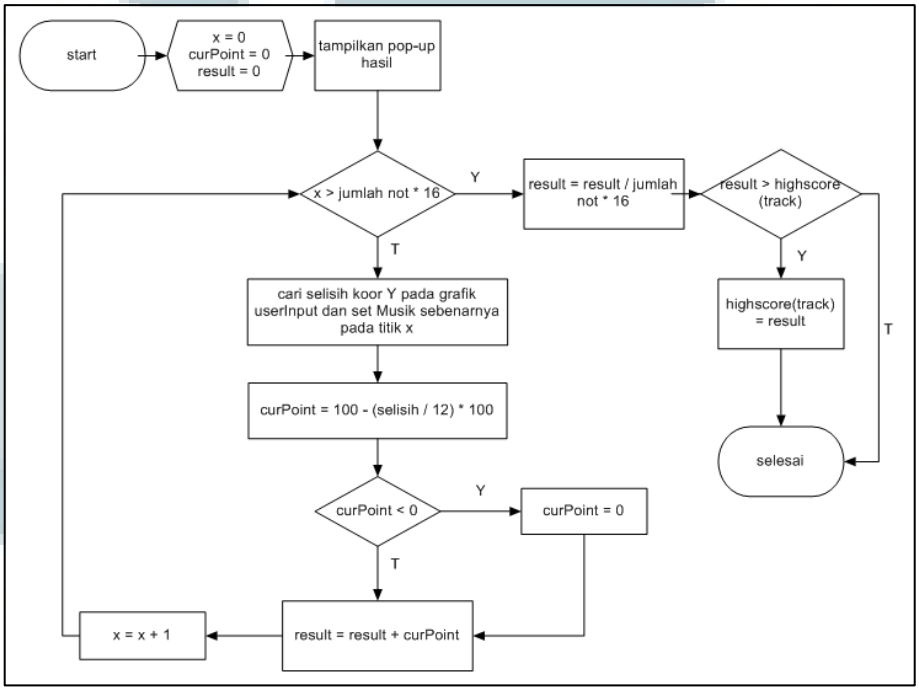
$$Result = \frac{\sum_{k=0}^{jumlah\ titik\ pada\ grafik} 100\% - \left(\frac{selisih(k)}{12}\right) x100\%}{\text{jumlah titik pada grafik}} \quad \text{...Rumus 3.3}$$

dimana

$$selisih(k) = |Y_1(k) - Y_2(k)| \quad \text{...Rumus 3.4}$$

Selisih(k) menunjukkan perbedaan koordinat Y kedua grafik, dimana $Y_1(k)$ merupakan koordinat Y dari grafik suara pengguna pada koordinat X ke-k, dan $Y_2(k)$ merupakan koordinat Y dari grafik set musik sebenarnya. Angka 12 mewakili keseluruhan nada pada satu tangga nada. Angka ini menunjukkan sampai sebatas mana kesalahan masih dapat ditolerir dan diberi nilai. Apabila *selisih*(k) melebihi 12, maka nilai pada titik itu akan bernilai 0.

Setelah itu, nilai akan terus dijumlahkan untuk setiap titik pada grafik. Kemudian, untuk mencari hasil akhir, nilai tersebut akan dibagi sejumlah titik pada grafik, yang merupakan rata-rata dari keseluruhan nilai. Gambar 3.14 menunjukkan diagram alir dari tampilan hasil.



Gambar 3.14 *Flowchart* Tampilkan Hasil

3.2.10 Audio Process

Audio Process merupakan rangkaian proses yang mengubah masukan suara hingga menjadi bentuk frekuensi. Proses ini dibangun dengan mengunakan bantuan *library TarsosDSP*. Secara umum, *audio process* pada aplikasi ini akan mengkonversikan suara ke bentuk *audio buffer*, dan menghitung frekuensi dengan menggunakan *Mcleod Pitch Method*. Gambar 3.15 menunjukkan diagram alur dari *audio process.*

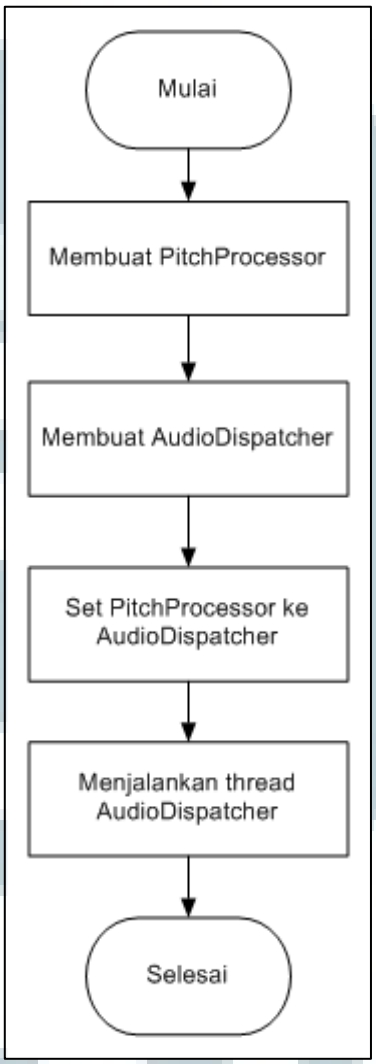
Gambar 3.15 *Flowchart Audio Process*

Pengimplementasian *audio process* membutuhkan 2 *class* utama, yaitu *AudioDispatcher* dan *PitchProcessor*. Menurut dokumentasi dari *TarsosDSP*, *AudioDispatcher* adalah sebuah *class* yang berfungsi untuk memutar suatu *audio file* kemudian mengirimkan *array of float* ke *AudioProcessor* untuk diproses. Pada aplikasi ini, *audio file* yang digunakan berasal langsung dari mikrofon dengan menggunakan fungsi *fromdefaultMicrofon().*

PitchProcessor adalah *class* yang berfungsi untuk memanggil *pitch detection algorithm. Class* ini juga dapat menentukan rangkaian suatu *event* setelah memperoleh hasil deteksi nada. Pada aplikasi ini, algoritma yang digunakan adalah MPM dengan *sample rate* 22050 Hz dan *buffer size* 1024 *sample*, yang merupakan standard untuk MPM (Mcleod, 2008).

Setelah *PitchProcessor* dibuat, *AudioDispatcher* yang sebelumnya telah diinisialisasi ditambahkan dengan menggunakan fungsi *addAudioProcessor().*

AudioDispatcher merupakan *class* yang mengimplementasikan *runnable*, sehingga untuk menjalankannya membutuhkan sebuah *thread*.Gambar 3.16 menunjukkan diagram alur dari implementasi *audio process*.



Gambar 3.16 *Flowchart* Implementasi *Audio Process*

3.2.11 Mcleod Pitch Method

Mcleod Pitch Method atau MPM merupakan suatu algoritma untuk melakukan estimasi frekuensi dari sebuah suara. *TarsosDSP* menyediakan fitur untuk menggunakan MPM dengan k = 0.97, *small_cutoff* = 0.5, dan

lower_pitch_cutoff = 80hz. Detail langkah dari algoritma Mcleod yang diterapkan adalah sebagai berikut.

1. Menghitung *Normalized Square Difference Function* (NSDF)

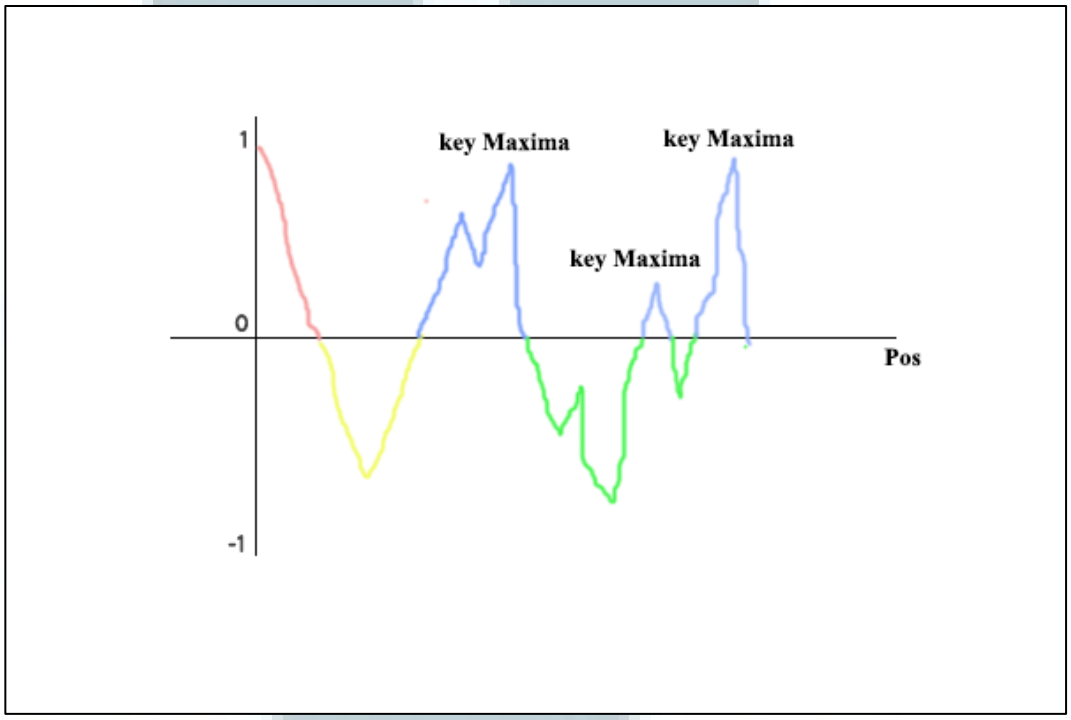
   Perhitungan NSDF menggunakan rumus 2.7. Hasil dari perhitungan ini adalah *array of float* sebanyak *buffer size* (jumlah *sample* yang diproses dalam satu langkah, pada aplikasi ini bernilai 1024) dengan rentang -1 sampai 1 yang bernama nsdf.

2. Mencari *Key Maxima* dengan Algoritma *Peak Picking*

   Pencarian dimulai dengan inisialisasi nilai pos (posisi saat ini) dan curMaxPos (nilai tertinggi pada suatu set gelombang) dengan 0. Nilai pos akan terus ditambah selama nilai nsdf[pos] lebih dari 0. Hal ini bertujuan untuk mencapai titik *negative zero-crossing* yang pertama. Apabila tidak menemukan titik *negative zero-crossing* sampai dengan 1/3 panjang *buffer size*, maka penjumlahan akan dihentikan.Setelah itu untuk mencapai titik *positive zero-crossing*, nilai pos akan terus ditambah selama nilai nsdf[pos] kurang dari 0.

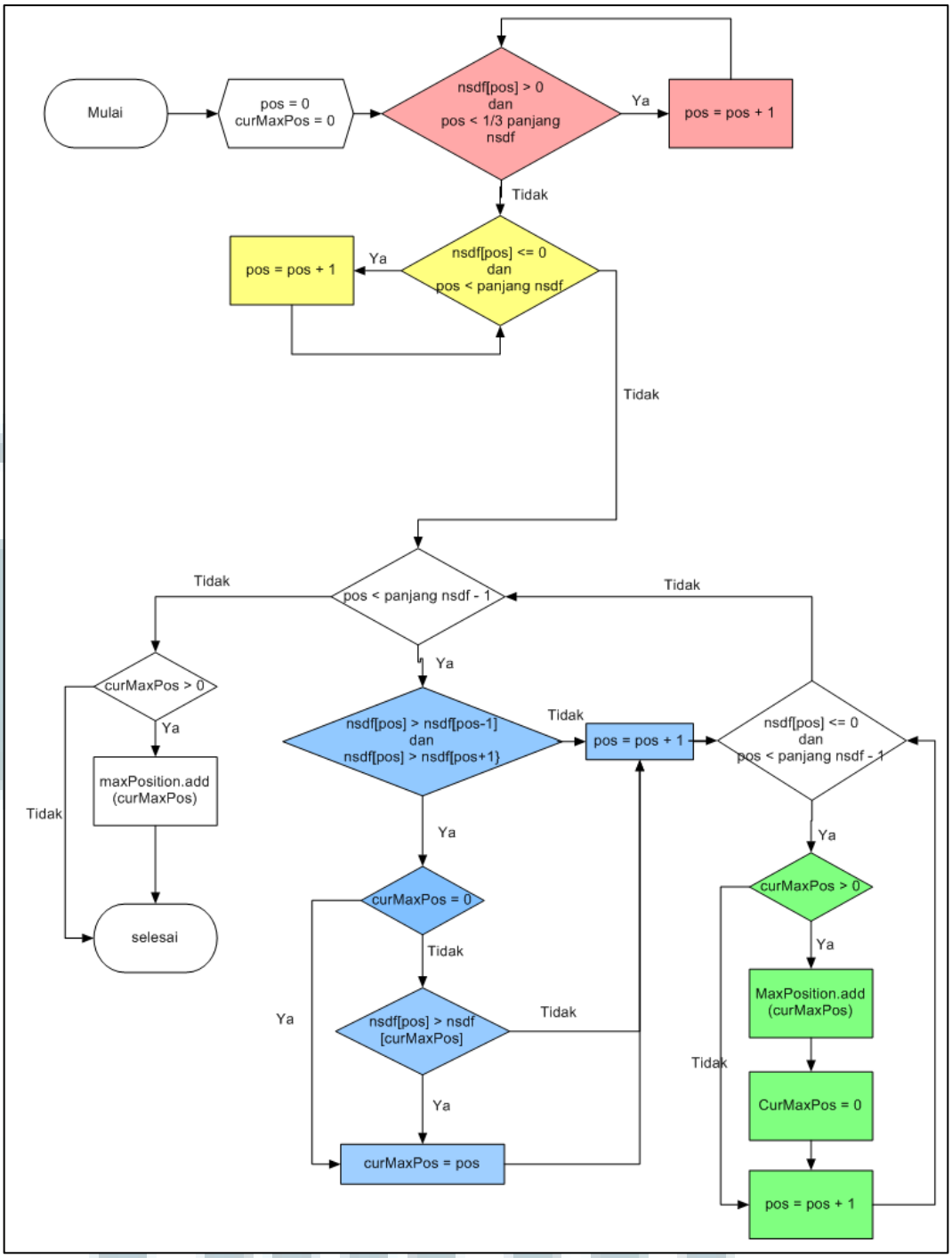
   Pencarian *key maxima* dilakukan dengan cara membandingkan nilai nsdf[pos] dengan nsdf[pos-1] dan nsdf[pos] dengan nsdf[pos+1]. Apabila nsdf[pos] merupakan nilai terbesar, akan dilakukan pengecekan apakah nilai curMaxPos sama dengan 0. Jika ya, maka nilai curMaxPos diubah menjadi nilai pos. Namun, jika tidak tetapi nsdf[pos] lebih besar dari nsdf[curMaxPos], maka nilai curMaxPos akan diubah menjadi pos juga. Hal ini terus dilakukan sampai nilai nsdf[pos] kurang dari atau sama dengan 0 (*negative zero-crossing*). Setelah mencapai nilai di bawah 0, nilai curMaxPos akan dimasukkan ke *list* MaxPosition, dan curMaxPos akan diset menjadi 0 kembali.

Setelah itu, nilai pos akan terus ditambah hingga mencapai titik *positive zero-crossing* berikutnya. Perhitungan akan terus berulang sampai mencapai akhir dari *array* nsdf. Hasil dari tahapan ini berupa bilangan yang merupakan *key maxima* yang ditampung dalam *list* dengan nama *maxPosition*. Contoh *key maxima* pada suatu gelombang dapat dilihat pada gambar 3.17.



Gambar 3.17 *Key Maxima* pada Gelombang

Gambar 3.18 menunjukkan keseluruhan langkah dalam mencari *key maxima* pada suatu gelombang. Warna yang terdapat pada gambar 3.18 melambangkan alur pada titik yang berwarna sama pada gambar 3.17.



Gambar 3.18 *Flowchart* Pencarian *Key Maxima*

3. Peningkatan Akurasi Titik Puncak *Key Maxima* dengan *Parabolic Interpolation*

Kecepatan perhitungan dapat ditingkatkan dengan hanya mengambil *key maxima* yang memiliki nilai nsdf melewati *small_cutoff* (0.5). Untuk setiap *key*

maxima tersebut, kemudian dilakukan perhitungan *parabolic interpolation*, untuk mendapat titik x dan y yang lebih akurat (dari *integer* menjadi *float*).

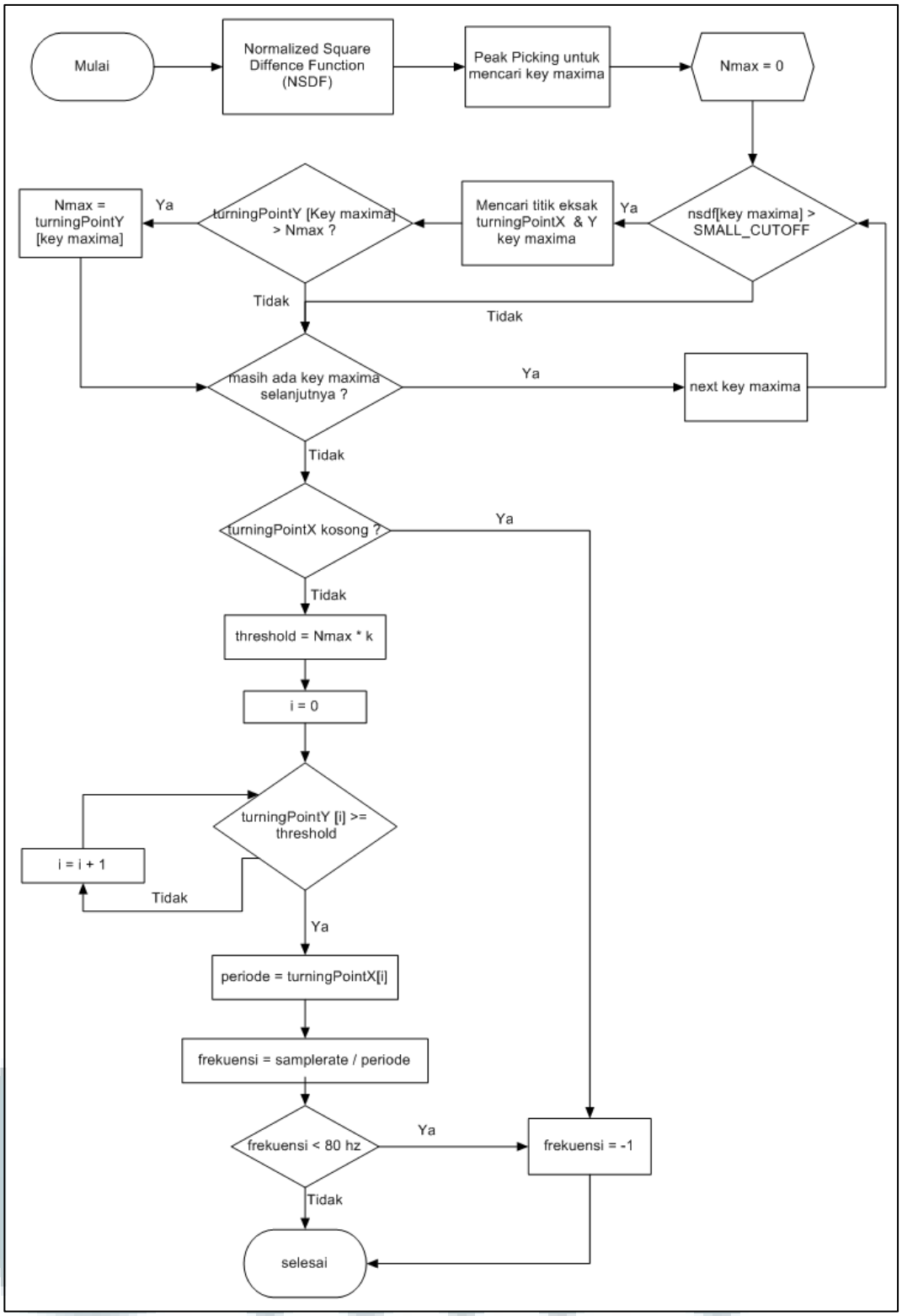
Hasil dari perhitungan *parabolic interpolation* kemudian disimpan pada *list of float* dengan nama *periodEstimate* untuk x, dan *ampEstimate* untuk y. Selain itu, nilai tertinggi *ampEstimate* akan disimpan sebagai Nmax untuk perhitungan berikutnya.

4. Set *threshold* = Nmax * k (0.97).
5. Mencari Periode

   Periode merupakan nilai x dari *ampEstimate* pertama yang melewati *threshold.*
6. Mencari Frekuensi

   Frekuensi didapat dari hasil pembagian antara *sample rate* dengan periode. Apabila frekuensi kurang dari *lower pitch cutoff* (80hz), maka akan dikembalikan nilai -1.

Hasil akhir dari MPM berupa bilangan *real* yang menunjukkan frekuensi yang dipanggil dengan fungsi *getPitch()* dan probabilitas (Nmax) yang dipanggil dengan fungsi *getProbability().* Gambar 3.19 menunjukkan diagram alir dari keseluruhan proses MPM.

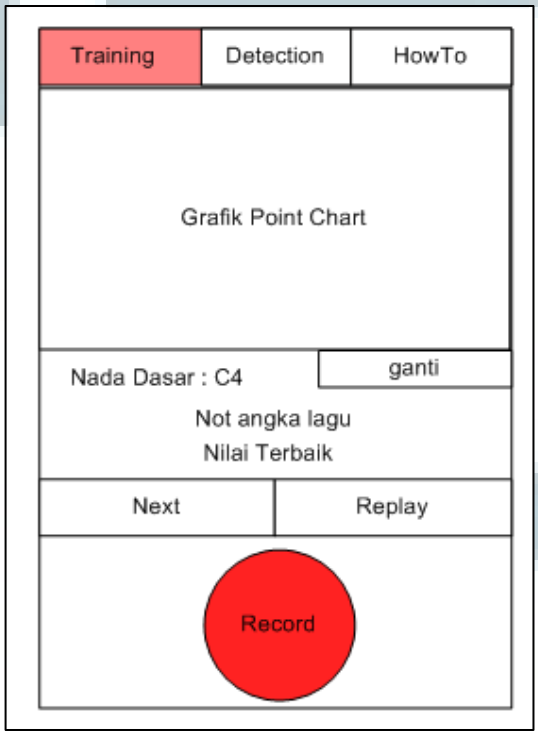


Gambar 3.19 *Flowchart Mcleod Pitch Method*

3.3 Rancangan Antarmuka Aplikasi

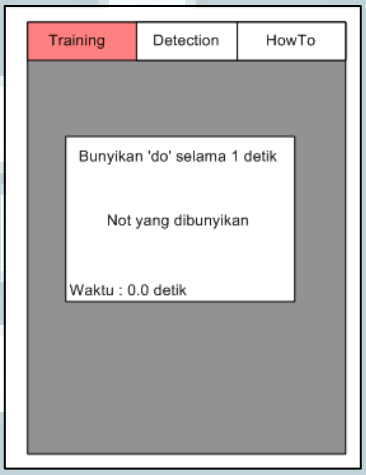
Aplikasi yang dibuat terdiri dari satu *activity* tunggal yang terbagi ke dalam tiga *tab* dengan fungsi *swipe gesture.* Masing-masing *tab* mewakili satu *fragment*, yaitu *Training, Detection,* dan *HowTo.* Berikut rancangan masing-masing tampilan dan keterangannya.

Pada *Training tab* yang merupakan tampilan utama, aplikasi akan memainkan sebuah set lagu dan menampilkan nada lagu tersebut ke dalam bentuk grafik dan tulisan. Aplikasi juga akan menampilkan nilai terbaik untuk lagu tersebut, beserta nada dasar untuk menyanyikan lagu tersebut dengan nada awal C_4. Pada halaman ini, terdapat empat buah tombol, yaitu *next* untuk mengganti set lagu, *replay* untuk memainkan ulang suatu lagu, *change basetone* untuk mengganti nada dasar, dan *record.* Rancangan antarmuka latihan vokal dapat dilihat pada gambar 3.20.



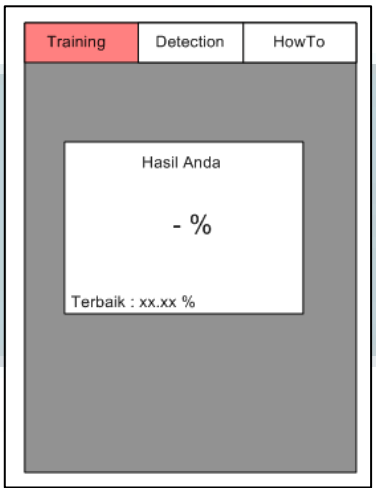
Gambar 3.20 Rancangan Antarmuka Latihan Vokal

Saat tombol ganti nada dasar ditekan, maka akan muncul sebuah *pop-up* yang meminta pengguna membunyikan nada yang diinginkannya menjadi ‘do’ selama 1 detik. *Pop-up* akan menghilang apabila pengguna aplikasi menekan di luar tampilan kotak *pop-up*. Rancangan antarmuka *pop-up* pergantian *basetone* dapat dilihat pada gambar 3.21.



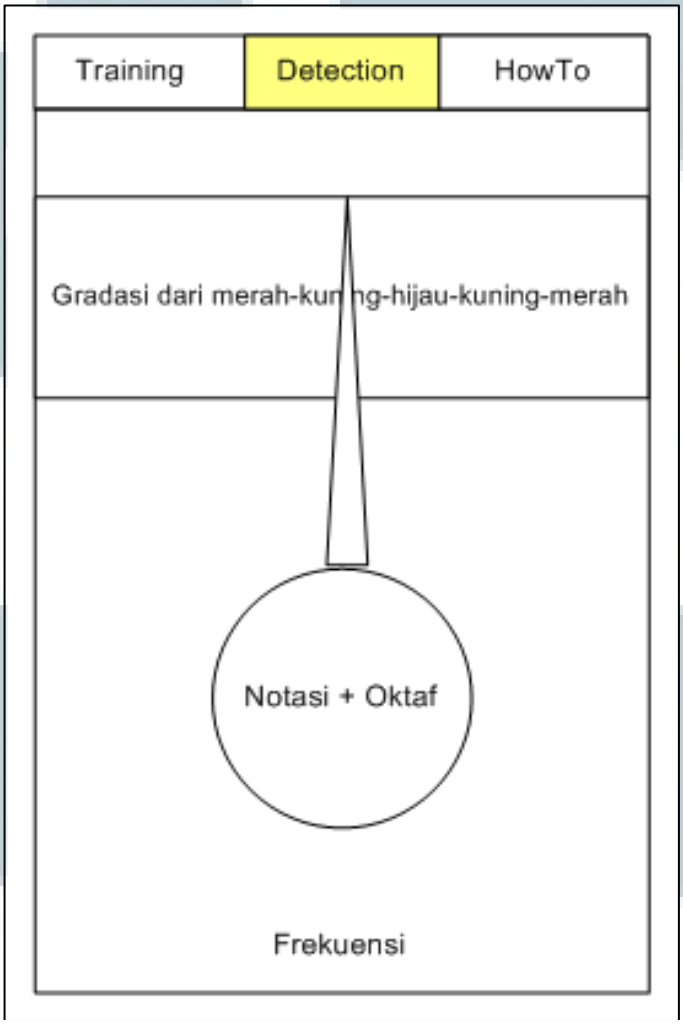
Gambar 3.21 Rancangan Antarmuka *Pop-up* Pergantian *Basetone*

Setelah pengguna menekan tombol *record*, aplikasi akan memulai mengisi grafik dengan nada yang dibunyikan pengguna. Setelah grafik mencapai panjang set lagu, maka akan ditampilkan sebuah *pop-up* yang berisi hasil perbandingan antara nada dari suara pengguna dengan nada sebenarnya dari set lagu.



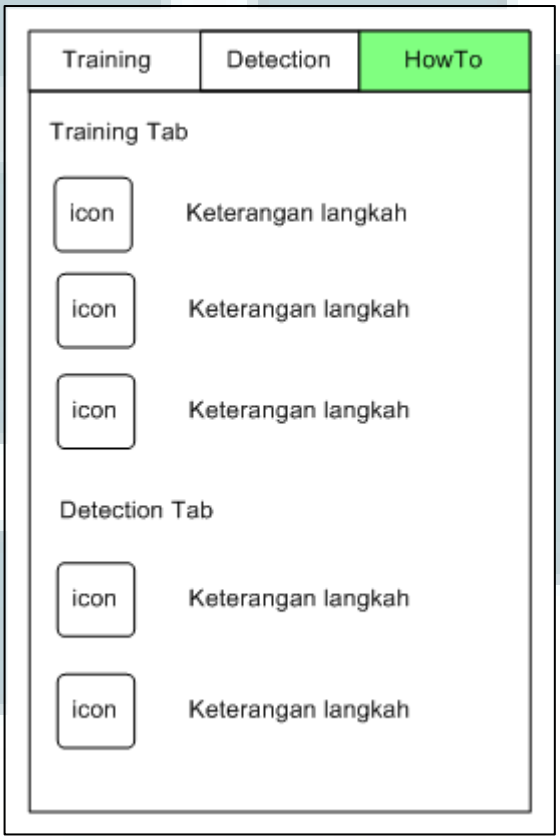
Gambar 3.22 Rancangan Antarmuka *Pop-up Result*

Pada *Detection Tab,* aplikasi akan mengkonversi frekuensi suara yang diterima ke dalam bentuk notasi dan oktaf terdekat dari nada yang dibunyikan. Kelebihan atau kekurangan nada tersebut akan direpresentasikan dengan sebuah penunjuk. Semakin ke tengah ujung penunjuk tersebut, semakin akurat nada yang dibunyikan. Penunjuk ke kiri menunjukkan nada yang dibunyikan lebih rendah dari nada eksak terdekat dari nada yang dibunyikan. Sementara penunjuk ke kanan menunjukkan nada yang dibunyikan lebih tinggi dari nada eksak terdekat dari nada yang dibunyikan. Rancangan antarmuka deteksi nada dapat dilihat pada gambar 3.23.



Gambar 3.23 Rancangan Antarmuka Deteksi Nada

How To tab berisikan langkah-langkah penggunaan aplikasi, baik pada *training tab* maupun *detection tab*. *How To tab* dilengkapi dengan *icon-icon* yang digunakan pada kedua *tab* sebelumnya untuk mempercantik sekaligus meningkatkan pemahaman pengguna aplikasi. Rancangan antarmuka cara penggunaan aplikasi dapat dilihat pada gambar 3.24.



Gambar 3.24 Rancangan Antarmuka Cara Penggunaan