Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB III

METODE DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Dalam penelitian ini, tahapan-tahapan metode yang digunakan akan dijabarkan dengan perincian sebagai berikut :

1. Tahapan studi literatur

   Proses ini merupakan tahapan paling awal dalam rangkaian penelitian yang dilakukan, yaitu berupa melakukan tinjauan pustaka untuk mendapatkan segala jenis informasi berupa data, metode maupun teori buku, referensi, dan jurnal baik yang bertaraf internasional maupun nasional yang mendukung landasan teori yang dibuat dalam Tugas Akhir ini. Selain itu, pencarian informasi mengenai perkembangan teknologi terkini dan semacamnya dilakukan dengan *browsing* melalui internet untuk mendapatkan data yang akurat dan mengikuti perkembangan masa kini.

2. Tahapan analisis algoritma

   Pada tahap ini, dilakukan proses analisis terhadap metode *Principal Component Analysis (PCA)* dan algoritma *Eigenface*, meliputi teori perhitungan dan karakteristik dari metode algoritma sehingga nantinya dapat digunakan dan diimplementasikan dengan baik ke dalam sistem yang dibuat.

3. Tahapan perancangan program

   Dari hasil analisis yang sudah dilakukan, maka didapatkan beberapa konsep utama dalam sistem. Untuk dapat menggambarkan hubungan fungsionalitas

pada sistem, maka selanjutnya dibuatlah rancangan untuk membangun aplikasi berdasarkan spesifikasi yang sudah dirumuskan. Rancangan yang dibuat berupa tampilan antar muka atau *interface*, gambaran proses yang terjadi dan alur keseluruhan dari program.

4. Tahapan implementasi program

   Dalam tahapan implementasi, dilakukan penggabungan antara konsep dasar rancangan dari program dengan metode dan algoritma untuk menghasilkan fungsionalitas berdasarkan penelitian yang dilakukan. Selain itu, dibuat juga *interface,* modul, sub-modul, kelas, fungsi dan berbagai komponen lainnya untuk melengkapi isi program.

5. Tahapan uji coba dan evaluasi

   Setelah aplikasi selesai, dilakukan uji coba untuk menguji fungsionalitas dan tingkat efisiensi dalam proses yang terjadi pada program. Uji coba dilakukan baik secara internal (dilakukan sendiri) maupun dengan menggunakan kuisioner. Hasil evaluasi didapatkan dengan mengambil kesimpulan dari hasil kuisioner yang telah diisi oleh responden.

6. Penulisan skripsi

   Setelah semua tahapan penelitian selesai dilakukan, maka selanjutnya dilakukan penulisan laporan sebagai bagian dari dokumentasi terhadap aplikasi hasil penelitian yang sudah dibuat.

3.2 Analisis

Dalam penelitian ini, digunakan satu metode dan satu algoritma utama, yaitu *Principal Component Analysis (PCA)* dan *Eigenface*. Metode dan algoritma tersebut didukung dengan penggunaan beberapa algoritma sederhana untuk

melakukan normalisasi. Normalisasi yang dilakukan adalah berupa *cropping*, *scaling*, dan *image color adjustment*. Untuk proses pengenalan wajah, digunakan *library* OpenCV untuk mendapatkan wajah yang akan diidentifikasi.

Aplikasi yang dibuat dalam penelitian ini merupakan aplikasi *mobile* yang ditujukan untuk sistem operasi Android. Aplikasi Android ini dibuat dengan menggunakan Eclipse *Integrated Development Environment*, Android SDK, dan Android NDK. Bahasa pemrograman yang digunakan adalah Java dengan versi Java Development Kit (JDK) 7 *update* 40 untuk perangkat komputer 64-bit. Aplikasi Android ini dibuat untuk dapat digunakan oleh segala jenis perangkat Android mulai dari sistem operasi Gingerbread (2.3, API 9) hingga KitKat (4.4.2 API 19). Seluruh penyimpanan data aplikasi dilakukan oleh internal memori perangkat Android terkait, berupa gambar citra wajah. Gambar citra wajah disimpan pada sebuah *folder* yang di *generate* secara otomatis oleh sistem operasi android dengan *path* /data/data/org.thesis.facerecognition/app_imageDir Storage. Aplikasi Android ini dinamakan Smart Notepad.

3.3 Perancangan Sistem

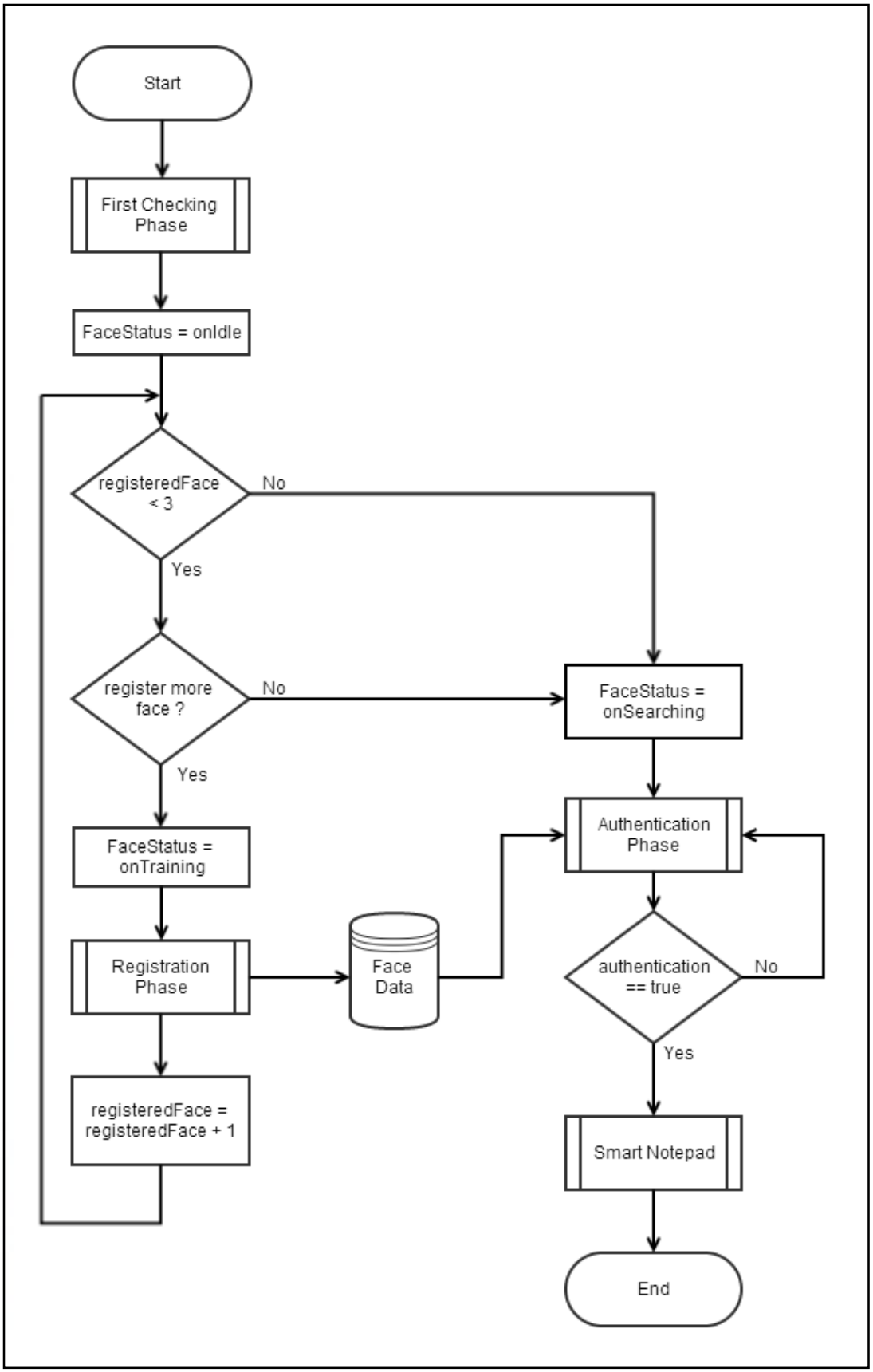
Sistem dari aplikasi yang dikembangkan dalam penelitian ini dapat dijelaskan dalam bentuk diagram alur sistem, fungsionalitas, serta bentuk masukan dan keluaran yang dibutuhkan dengan pemaparan sebagai berikut.

3.3.1 Diagram Sistem

Alur kerja dari sistem yang dibuat akan dipaparkan dengan menggunakan diagram. Proses kerja dari keseluruhan sistem terbagi ke dalam beberapa fase, yang terdiri atas fase registrasi, fase otentikasi, dan fase pengolahan data pada *notepad*. Fase registrasi terbagi lagi menjadi beberapa bagian kecil, yaitu pengambilan

gambar citra wajah dan penyimpanan gambar sebagai objek *training* pada verifikasi. Fase registrasi terdiri atas beberapa tahap antara lain, fase deteksi wajah dan pengambilan gambar wajah, fase normalisasi, dan fase penyimpan data wajah ke dalam *storage.* Ada beberapa hal yang membedakan antara fase registrasi dan fase otentikasi antara lain, dalam fase registrasi, gambar wajah hanya diambil dan disimpan, sedangkan dalam fase otentikasi gambar harus diubah terlebih dahulu menjadi sebuah matriks dengan ukuran pixel yang ditentukan untuk selanjutnya dapat dilakukan perhitungan secara matematis dengan menerapkan metode *Principal Component Analysis* (PCA) dan algoritma *Eigenface*. Gambar wajah yang digunakan berasal dari pengambilan gambar melalui kamera. Dalam penggunaannya, tentu dalam satu *frame* gambar, belum tentu dipenuhi seluruhnya dengan gambar wajah dan akan memiliki ukuran *pixel* yang berbeda-beda tergantung dari ukuran wajah yang terdeteksi itu sendiri. Maka dari itu, gambar wajah harus disesuaikan dengan kriteria perhitungan yang sudah ditentukan. Fase penyesuaian ini termasuk ke dalam bagian *subroutine normalization*.

Data citra wajah yang disimpan pada saat fase registrasi nantinya akan dibandingkan dengan data gambar yang akan diambil pada saat fase otentikasi. Data citra wajah akan disimpan ke dalam memori internal dari perangkat yang bersangkutan. Beberapa penjelasan umum di atas akan dijabarkan secara rinci pada gambar 3.1.



Gambar 3.1 Diagram alir aplikasi

3.3.2 Fungsionalitas Sistem

Beberapa fungsionalitas yang dimiliki oleh sistem adalah sebagai berikut.

1. Pengambilan dan pemilihan gambar citra wajah yang akan disimpan pada saat fase registrasi. Hal ini juga berlaku ketika proses otentikasi dilakukan untuk membatasi semua objek terdeteksi wajah dihitung dan dibandingkan.
2. Pada saat dilakukan fase otentikasi, ditampilkan nilai eigenface dari masing-masing gambar dan pembandingnya untuk melihat persentase kemiripan wajah yang dibandingkan.
3. Terdapat *gallery* untuk melihat wajah-wajah yang sudah disimpan pada memori internal. Fitur ini juga dapat digunakan untuk memodifikasi gambar pembanding jika dinilai lebih akurat oleh *user*.

Sistem yang dirancang ini akan diimplementasikan sebagai sebuah aplikasi utuh dan akan dijalankan pada sistem operasi Android. Aplikasi ini akan memiliki dua fase utama, yaitu fase registrasi dan fase otentikasi.

Untuk lebih rinci, penjelasan dari masing-masing fase akan digambarkan dengan menggunakan diagram alir, hierarki menu, dan rancangan tampilan antarmuka aplikasi.

3.3.3 Desain Modul dan Subroutine

Untuk semua desain modul dan *subroutine* yang digunakan dalam proses internal setiap modul tersebut akan dijelaskan dengan menggunakan diagram alir sebagai berikut.

1. Desain fase registrasi

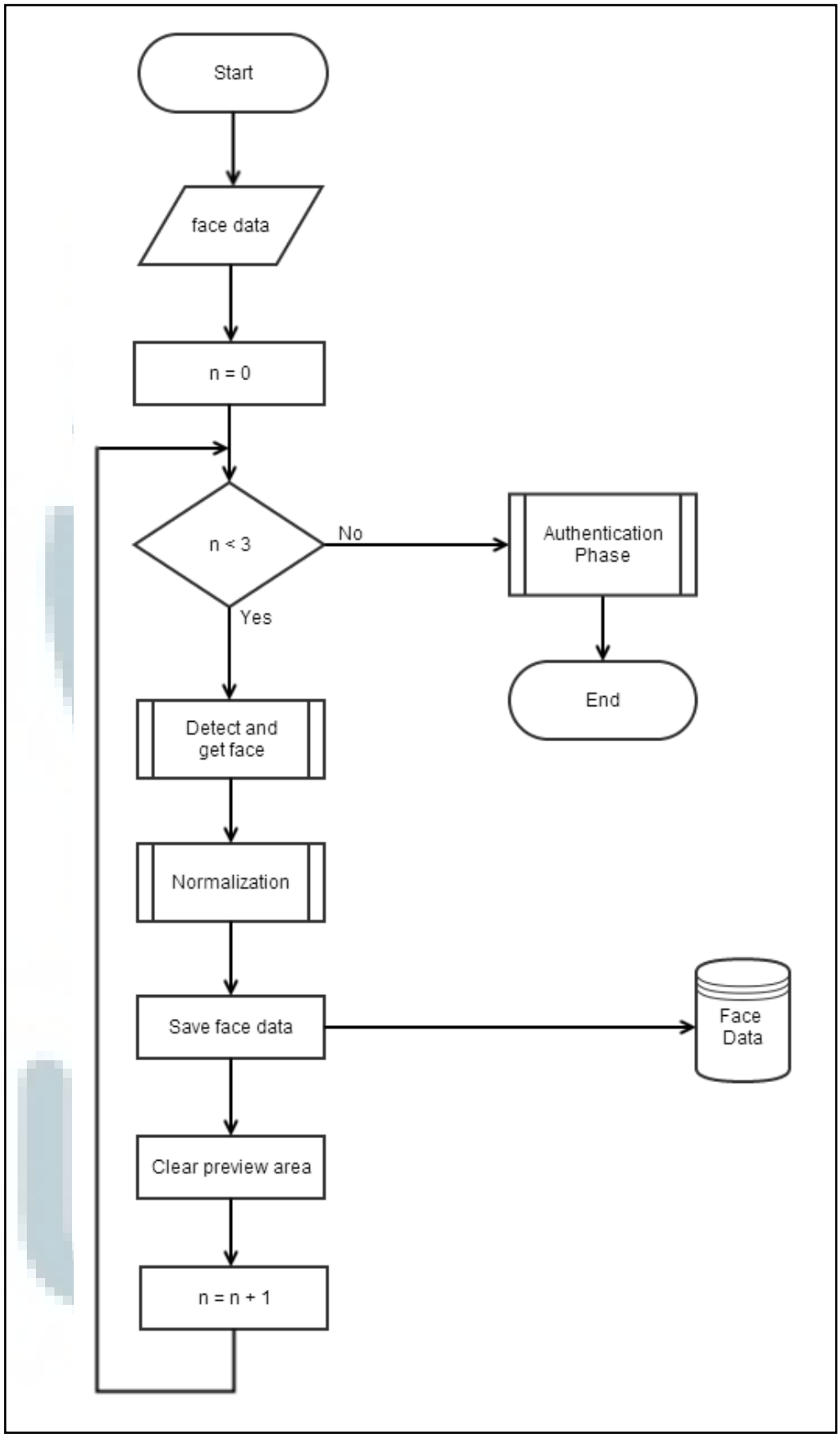
Pada fase registrasi, akan dilakukan pengambilan gambar citra wajah yang nantinya akan digunakan sebagai pembanding dalam fase otentikasi. Sepanjang fase, program akan menampilkan kotak hijau yang menandakan sebagai wajah terdeteksi. Ketika *user* menekan tombol *record*, maka *subroutine detect and get face* akan dipanggil. Pada *subroutine* ini, dilakukan deteksi terhadap semua wajah yang berada dalam satu *frame* kamera.

Dalam proses deteksi wajah, digunakan *library* OpenCV untuk mempermudah dalam mendapatkan informasi wajah yang akurat. Prinsip kerja dari *library* ini adalah dengan menghitung jarak antar mata, hidung, dan mulut yang digambarkan dalam sebuah *file cascade* berbentuk xml. *File* ini berisi informasi perhitungan *frontal face* untuk mengetahui jika ada bentuk wajah dalam sebuah *frame* kamera. Setelah OpenCV berhasil mendapatkan wajah pada *frame*, maka untuk setiap wajah terdeteksi akan ditandai dengan kotak berwarna hijau. Setelah *subroutine* dijalankan, akan ditampilkan lima kali contoh tampilan dari gambar wajah pada *imageview*. Gambar pada *imageview* inilah yang nantinya akan disimpan pada memori internal.

Setelah didapatkan gambar citra wajah yang diinginkan, *user* dapat langsung melakukan penyimpanan terhadap wajah tersebut. Sebelum dilakukan penyimpanan, gambar wajah akan disesuaikan terlebih dahulu agar dapat dilakukan perhitungan pada saat fase otentikasi. Pada saat inilah dilakukan pemanggilan terhadap *subroutine normalization*. Pada *subroutine normalization*, gambar citra wajah akan melalui proses *cropping*, *scaling*, dan *image color adjustment*. Proses *cropping* dilakukan untuk memastikan bahwa hanya *pixel* dengan gambar wajah

saja yang dihitung. Hal ini dilakukan untuk mengantisipasi perhitungan yang tidak diperlukan dan diharapkan dapat meningkatkan efisiensi jalannya *subroutine* selanjutnya sampai selesai. Proses *scaling* dilakukan untuk menyamakan resolusi dari setiap gambar citra wajah, sehingga proses pengerjaan dapat dilakukan lebih optimal tanpa menghilangkan atau merusak bentuk citra wajah yang sudah didapatkan. Proses *scaling* akan menghasilkan gambar sebesar 100x100 *pixels*. Proses *image color adjustment* dilakukan untuk mengurangi isi komponen *Red, Green, Blue* (RGB) pada setiap *pixel* gambar citra wajah. Seluruh komponen RGB akan dikonversikan ke dalam bentuk *grayscale* yang terdiri atas dua warna utama saja, yaitu hitam dan putih. Proses ini dilakukan untuk menghemat waktu perhitungan yang diperlukan jika dilakukan dengan komponen RGB lengkap. Dengan kata lain, proses ini akan mengurangi sekitar 1/3 dari seluruh perhitungan yang harus dilakukan. Setiap *pixel* pada gambar citra wajah nantinya akan memiliki nilai 0 – 255 tergantung dari komposisi warna hitam dan putih yang ada di dalamnya.

Berdasarkan penjelasan yang sudah dipaparkan sebelumnya, diagram alir fase registrasi dapat didefinisikan dengan gambar 3.2 sebagai berikut.



Gambar 3.2 Diagram alir fase registrasi

2. Desain fase otentikasi

Proses pertama yang dilakukan dalam fase otentikasi adalah melakukan pengecekan apakah sudah pernah dilakukan registrasi sebelumnya. Jika ternyata tidak ada gambar citra wajah, maka proses akan diarahkan terlebih dahulu pada modul registrasi dimana *user* harus memasukkan data-data wajah terlebih dahulu guna proses otentikasi. Jika gambar citra wajah sudah tersedia, maka proses pengambilan wajah untuk otentikasi dapat langsung dilakukan.

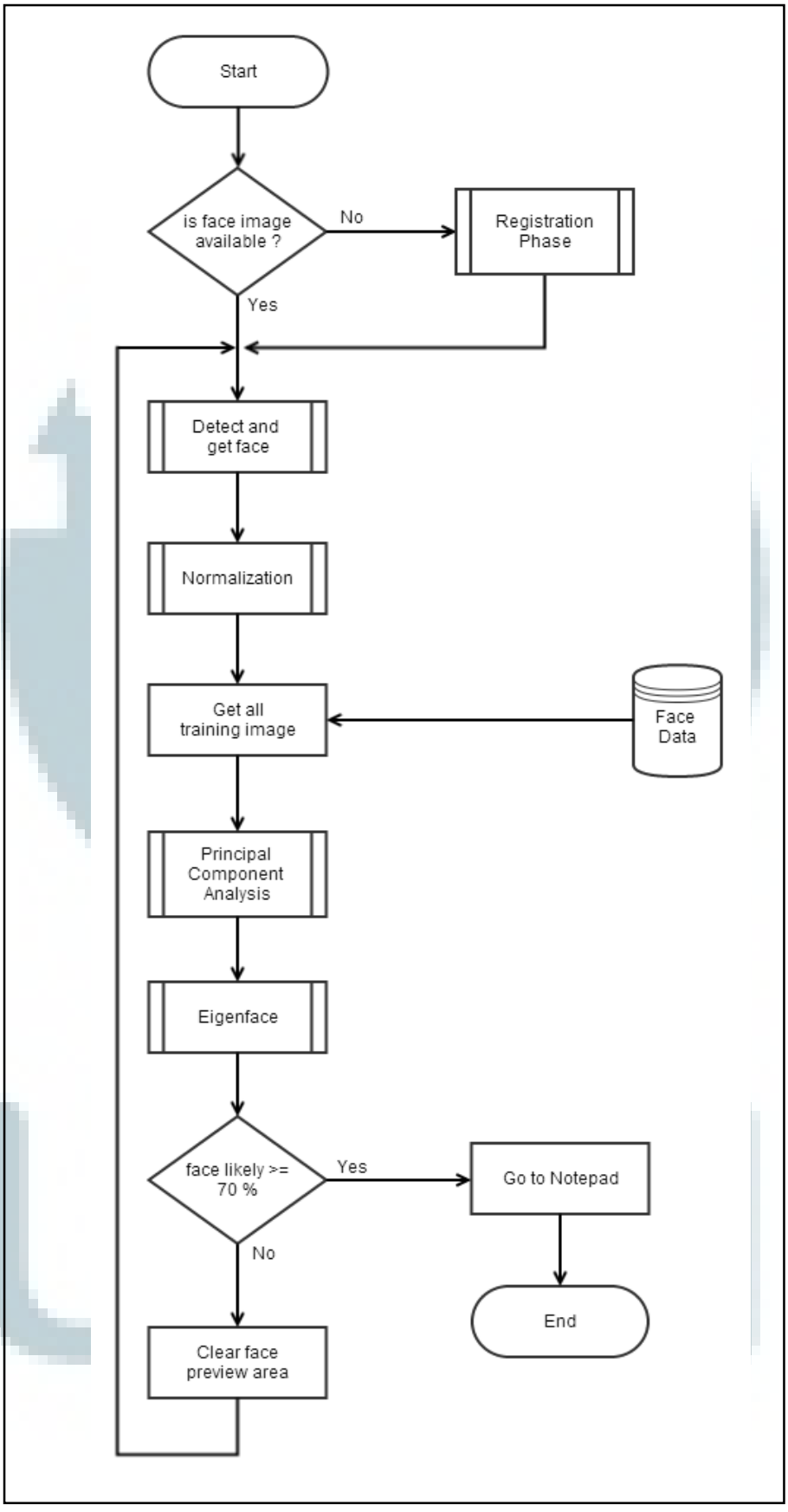
Proses akan terus menerus melakukan deteksi wajah pada *frame* kamera. Ketika *user* menekan tombol *compare*, maka setiap wajah terdeteksi akan dibandingkan dengan gambar-gambar *training* yang sudah diambil sebelumnya. Tidak ada batasan dalam pengambilan gambar wajah. Program tidak akan berhenti sampai menemukan wajah yang tepat untuk membuka *locking system* dan masuk ke dalam aplikasi Smart Notepad.

Setelah ada wajah terdeteksi yang didapatkan, maka proses selanjutnya adalah normalisasi dari setiap gambar yang diambil. Pada tahap inilah *subroutine normalization* dipanggil kembali untuk menyesuaikan wajah terdeteksi dengan format citra wajah yang sudah disimpan sebelumnya. Semua gambar citra wajah akan disimpan dan diambil dari memori internal perangkat yang bersangkutan. Proses ini dilakukan untuk menghindari adanya kesalahan perhitungan dalam setiap *pixel* yang terdapat pada gambar. Setiap wajah terdeteksi akan dibandingkan dengan semua *training image* yang ada dalam *storage*. Setelah itu, wajah-wajah akan diubah ke dalam bentuk matriks guna perhitungan matematis lebih lanjut. Semakin banyak gambar yang digunakan sebagai patokan untuk proses otentikasi, maka kemungkinan kebenaran dalam deteksi wajah yang sesuai akan semakin

meningkat. Dalam sistem ini, gambar wajah yang digunakan maksimal hanya tiga buah untuk menghindari perhitungan berlebih.

Setelah seluruh citra wajah diubah ke dalam bentuk matriks, gambar akan diproses dengan metode *Principal Component Analysis* (PCA) dan dipadukan dengan algoritma *Eigenface* untuk mendapatkan nilai-nilai perbandingan yang diperlukan selama proses otentikasi berjalan. Jika bentuk citra wajah sesuai, maka proses akan dialihkan ke aplikasi penyimpanan catatan sederhana. Dalam penelitian ini, kemiripan wajah yang digunakan adalah 80% dari hasil seluruh perhitungan selama proses otentikasi. Jika selama proses perhitungan ternyata dari semua *training image* tidak ada yang mencapai kemiripan 80%, maka program akan terus menerus mencari gambar lain dan dihitung sampai ada gambar wajah yang cocok.

Berdasarkan penjelasan di atas, semua proses dapat direpresentasikan menjadi sebuah diagram pada gambar 3.3 berikut yang berisi alur kerja dari proses fase otentikasi.



Gambar 3.3 Diagram alir fase otentikasi

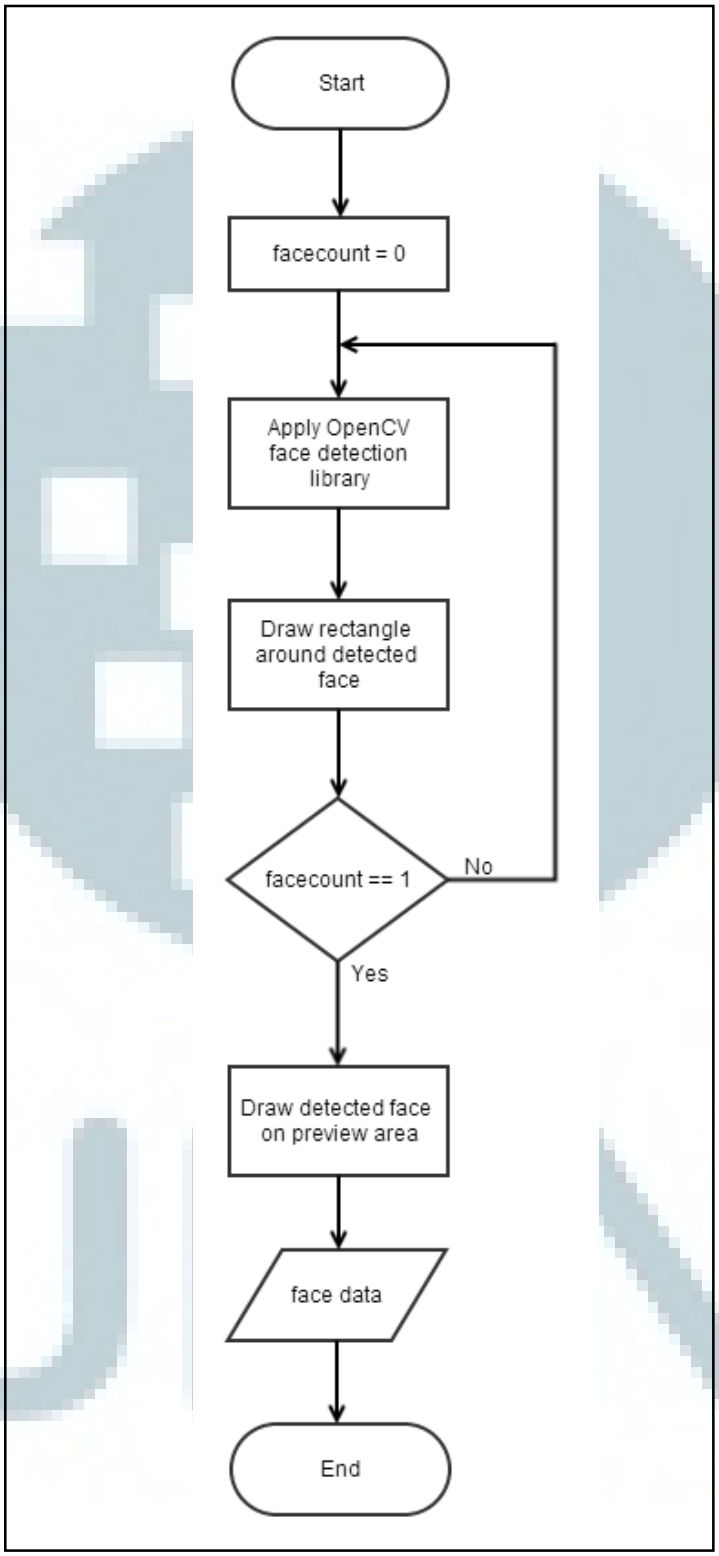
3. Desain *subroutine detect and get face*

Modul ini berfungsi untuk mendeteksi keberadaan wajah dalam *frame* kamera yang tertangkap. Proses yang terjadi dalam modul ini meliputi implementasi *library* OpenCV, membuat *highlight* dari wajah terdeteksi, dan menampilkan wajah pada *image preview area* yang terdapat pada program.

Penggunaan *library* OpenCV pada proses ini berguna untuk mendapatkan spesifikasi wajah manusia yang akurat. Dalam implementasinya, OpenCV menggunakan *cascade file* yang berisi data perhitungan untuk mendeteksi wajah manusia. Dari hasil perhitungan tersebut, maka dicari kecocokkan dalam gambar, apakah ada bentuk wajah yang terdeteksi saat perhitungan masing-masing *pixel* dilakukan. Proses deteksi akan dilakukan secara *real time* menggunakan *camera preview* dari perangkat Android yang bersangkutan.

Setelah ditemukan wajah terdeteksi hasil dari kerja *library* OpenCV, maka selanjutnya dilakukan penandaan terhadap wajah tersebut dengan memberi tanda berupa kotak hijau yang mengelilingi wajah terdeteksi. Jumlah kotak hijau tergantung pada jumlah wajah yang ditemukan. Untuk menghindari bentrok dalam pengambilan wajah, maka program akan dilengkapi dengan pembatasan jumlah wajah pada *camera frame*. Ketika ada lebih dari satu wajah yang terdeteksi, maka program tidak akan dilanjutkan ke proses berikutnya sampai jumlah wajah yang ada benar-benar hanya satu. Proses ini untuk menghindari keadaan ambigu dalam pengambilan wajah. Proses akhir adalah menampilkan wajah ke dalam *preview area*, sehingga *user* dapat melihat bentuk wajah yang akan disimpan sebagai pembanding nantinya.

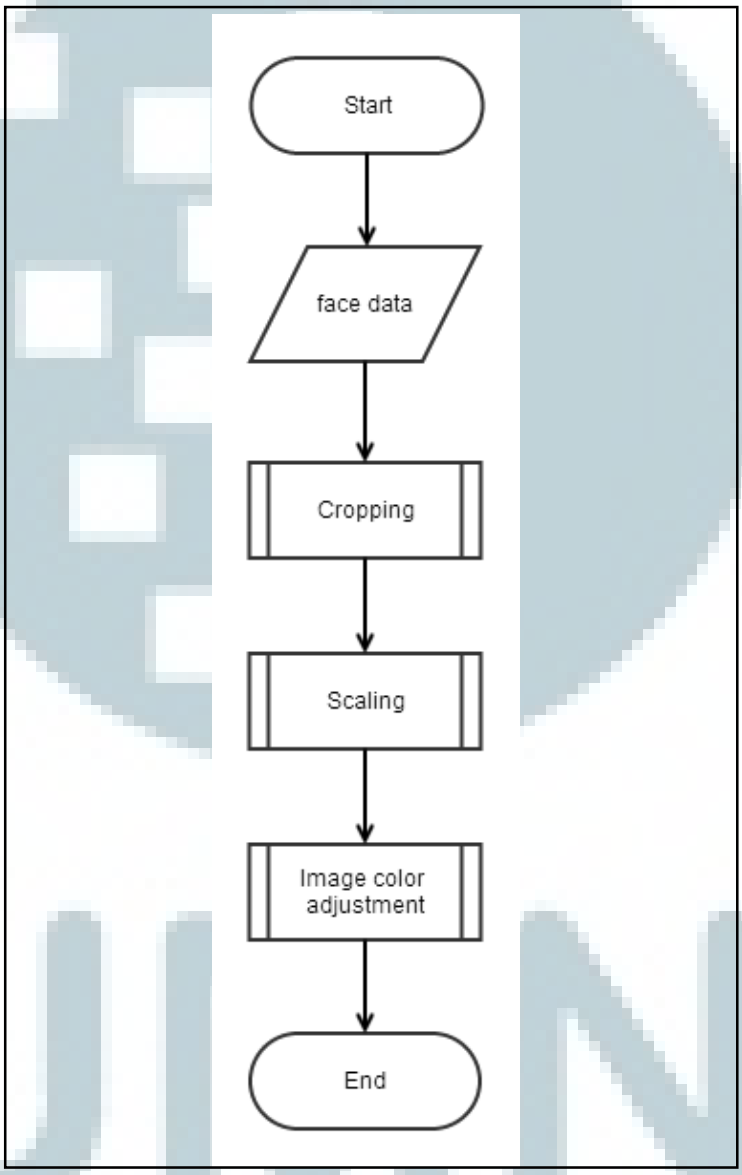
Untuk lebih jelasnya, penjabaran *subroutine detect and get face* dapat dilihat pada gambar 3.4 sebagai berikut.



Gambar 3.4 Diagram alir *subroutine detect and get face*

4. Desain *subroutine normalization*

Modul ini berfungsi untuk melakukan normalisasi pada gambar citra wajah. Normalisasi yang dilakukan di sini berupa *cropping*, *scaling*, dan *image color adjustment*. Berikut ini merupakan gambaran *subroutine normalization* secara umum.



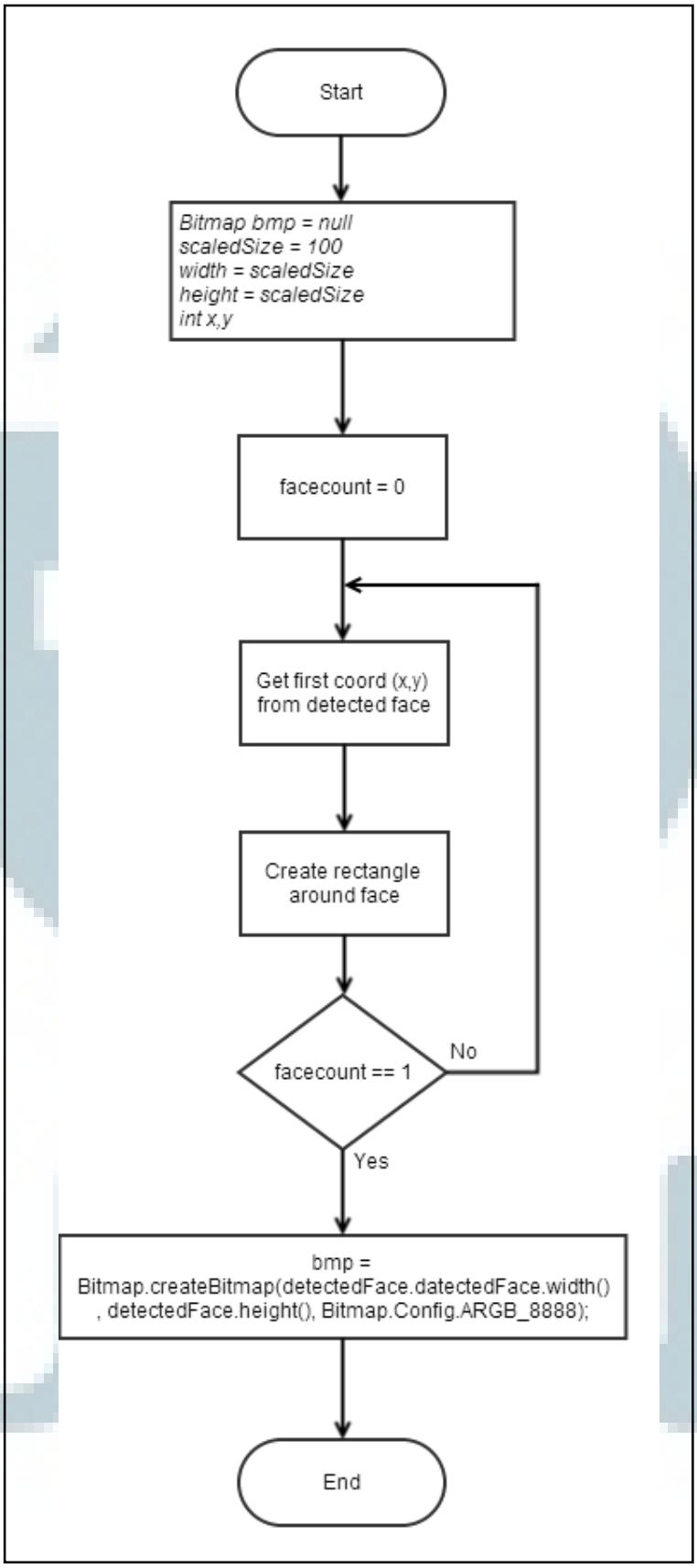
Gambar 3.5 Diagram alir *subroutine normalization*

Pada *subroutine normalization* ini, terdapat tiga *subroutine* utama yang dijalankan, yaitu *cropping*, *scaling*, dan *image color adjustment*. *Subroutine*

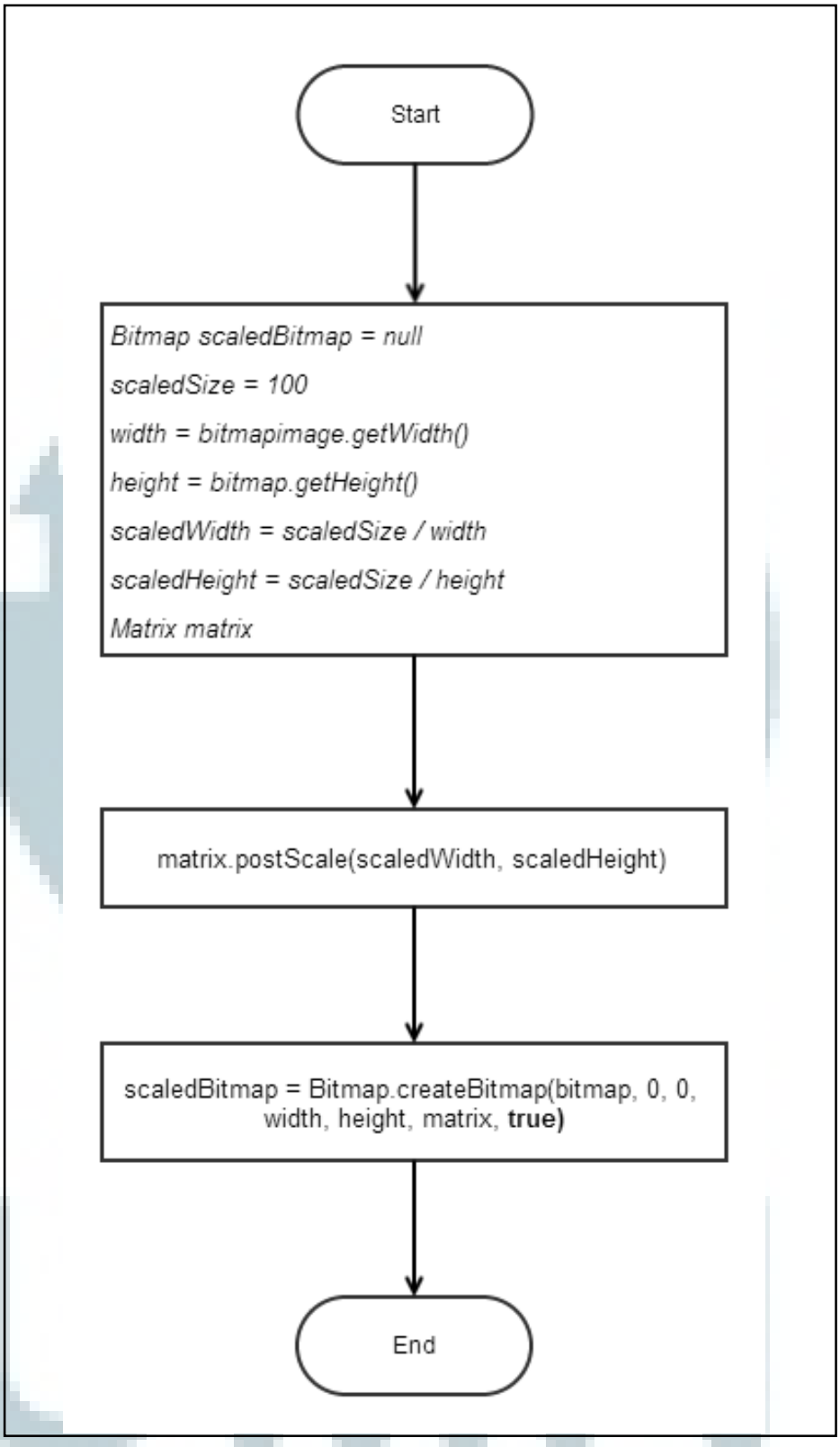
cropping dijalankan untuk memastikan bahwa hanya *frame* yang berisi wajah saja yang akan ditampilkan. Semua *frame* kosong di luar gambar wajah akan dihilangkan untuk mengurangi beban kerja dari perangkat.

Pada *subroutine scaling*, dilakukan modifikasi ukuran gambar citra wajah agar semua gambar memiliki ukuran yang sama. Proses ini dilakukan agar perhitungan matriks yang ada dapat dilakukan dengan jumlah *pixel* yang akurat. Dalam aplikasi ini, gambar citra wajah disesuaikan dengan ukuran 100x100 *pixel*, baik gambar citra wajah pada fase registrasi maupun fase otentikasi. Hal ini akan sangat berguna untuk menunjang proses perhitungan *Principal Component Analysis* dan implementasi algoritma *Eigenface*. Proses yang dilakukan dalam *subroutine scaling* ini sudah tersedia dalam *library* Android dan siap digunakan.

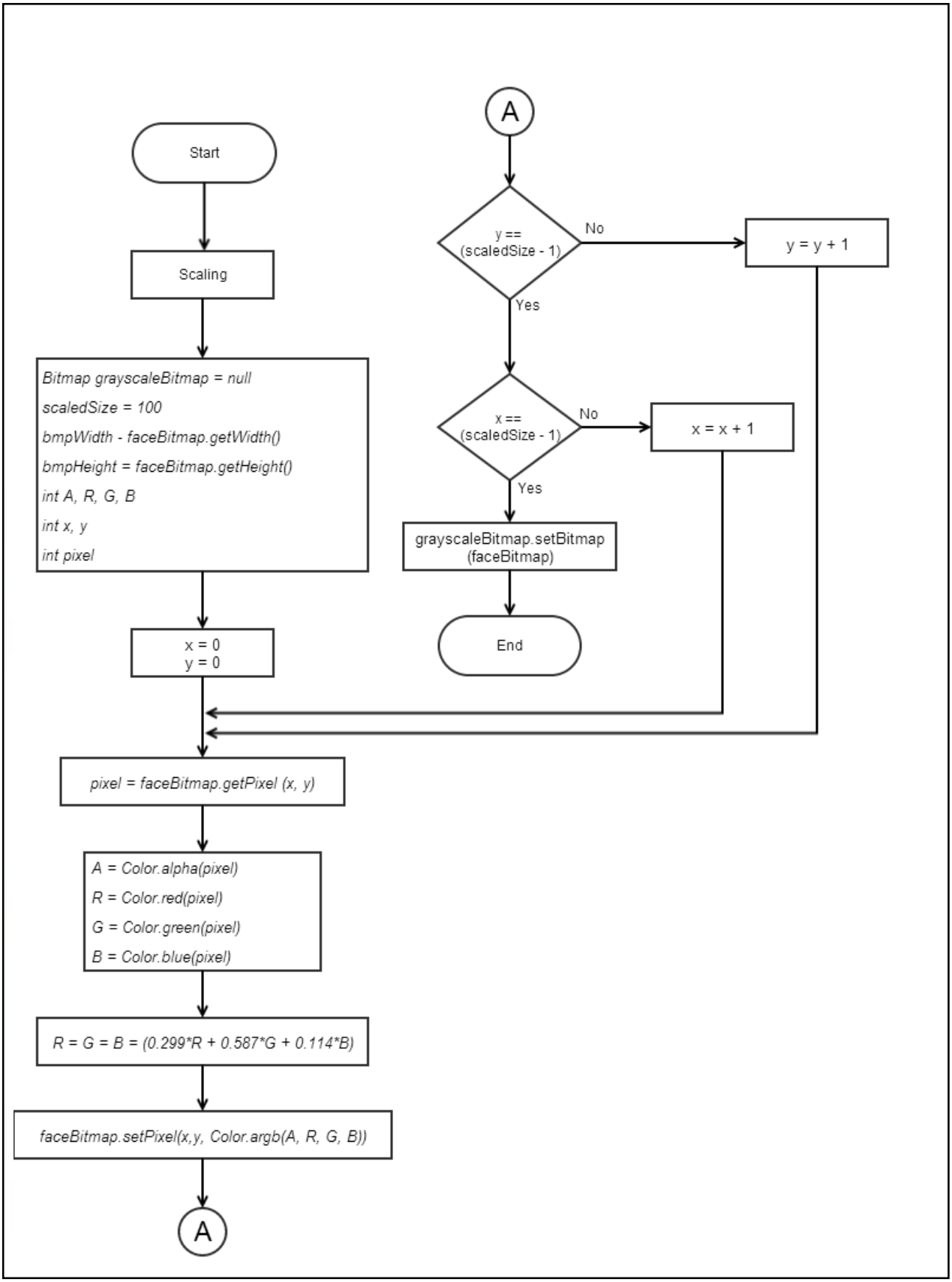
Setelah gambar citra wajah didapatkan dan disesuaikan dengan ukuran perhitungan, maka proses selanjutnya adalah menyamakan format warna pada gambar dengan menjalankan *subroutine image color adjustment*. *Subroutine* ini berfungsi untuk mengubah gambar dengan warna lengkap menjadi bentuk *grayscale image*. *Grayscale image* merupakan gambar yang hanya terdiri dari dua warna utama saja, yaitu hitam dan putih. Proses yang terjadi adalah setiap warna *pixel* pada gambar diambil sehingga didapatkan komponen *Red, Green, Blue* (RGB) dari masing-masing *pixel*. Kemudian dengan perhitungan kombinasi warna, seluruh RGB diubah menjadi hitam putih yang nantinya akan membentuk format *grayscale*. Untuk setiap *pixel*, nantinya hanya akan berisi angka dari 0 – 255, tergantung dari tingkat warna hitam yang ada. Hal ini dilakukan untuk mengurangi beban kerja perhitungan warna *pixel*. Untuk lebih jelasnya, semua *subroutine* pada proses normalisasi akan digambarkan dalam diagram sebagai berikut



Gambar 3.6 Diagram alir *subroutine cropping*



Gambar 3.7 Diagram alir *subroutine scaling*



Gambar 3.8 Diagram alir *subroutine image color adjustment*

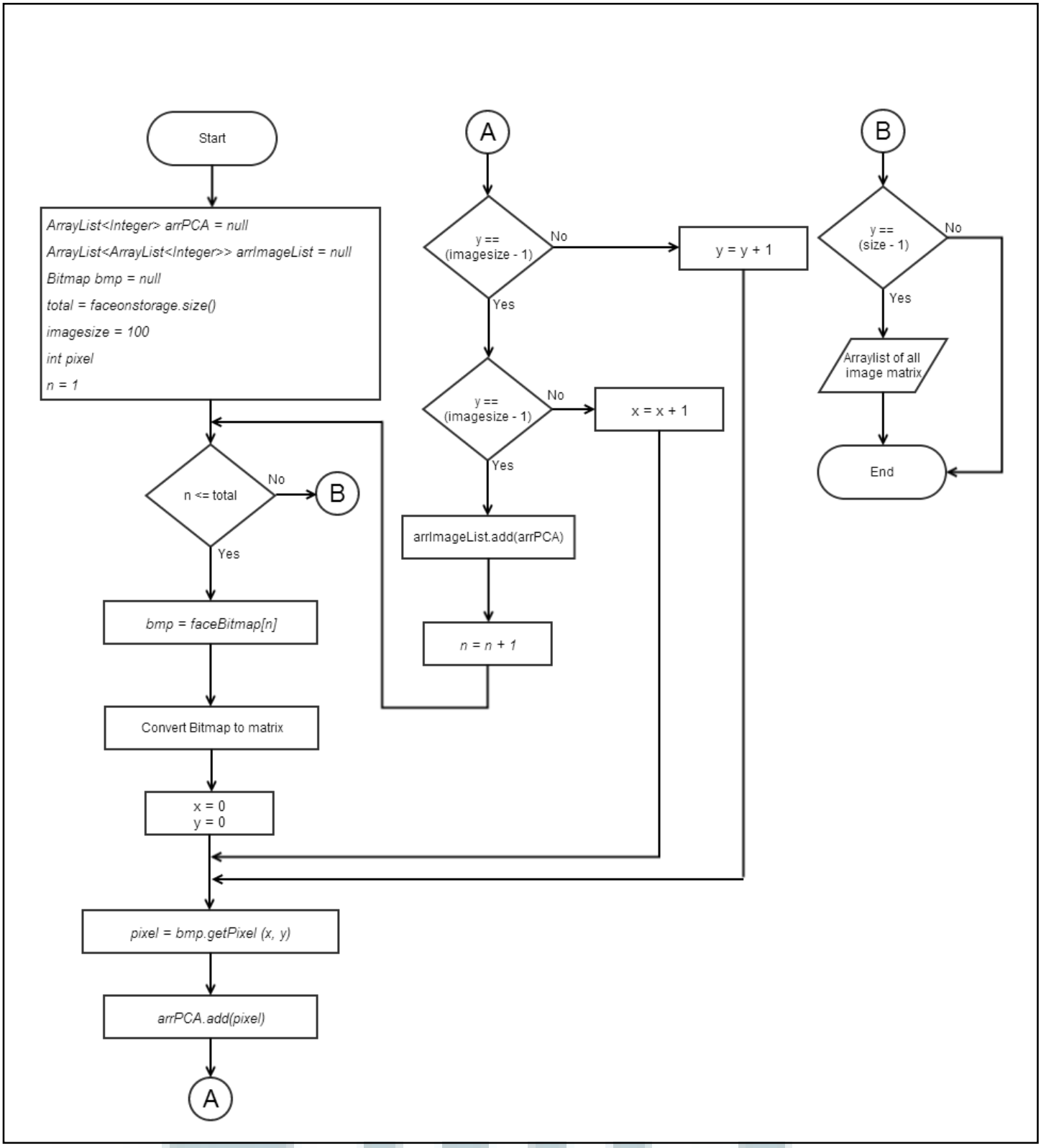
5. Desain *subroutine Principal Component Analysis*

Proses selanjutnya yang dilakukan adalah melakukan perhitungan dengan menggunakan metode *Principal Component Analysis* (PCA). Prinsip kerja dari metode ini cukup sederhana. Fungsi utama dari PCA ini adalah untuk menyederhanakan bentuk matriks dengan tujuan mengurangi dimensi yang ada pada gambar.

Sebagai contoh adalah implementasi pada program ini. Matriks yang digunakan untuk perhitungan kecocokkan wajah adalah matriks dengan ukuran 100x100 *pixel*. Tujuan akhir dari proses ini adalah untuk mengubah matriks dengan dimensi tertentu menjadi matriks dengan susunan horizontal. Di akhir implementasi PCA ini, bentuk matriks 100x100 *pixel* akan dikonversikan menjadi matriks ukuran 1x10000 *pixel*. Kemudian selanjutnya matriks inilah yang akan digunakan untuk perhitungan dengan algoritma *Eigenface.* Setiap wajah yang tersimpan dalam memori akan dikonversikan ke dalam bentuk matriks horizontal untuk memudahkan perhitungan saat dilakukan otentikasi terhadap kecocokkan wajah.

Sama halnya dengan wajah terdeteksi. Setiap ada wajah yang tertangkap pada *frame* kamera, wajah tersebut akan secara cepat dikonversi dan dihitung kecocokkannya dengan wajah *training* lainnya. Maka dari itu, penggunaan ukuran *pixel* pada aplikasi ini tidak terlalu besar dikarenakan banyaknya perhitungan yang harus dilakukan. Penyesuaian gambar dan dimensi pada matriks tidak akan mengganggu atau mengurangi efisiensi dan keakuratan dari proses perhitungan dalam fase otentikasi.

Berikut ini adalah gambaran proses *subroutine Principal Component Analysis* yang direpresentasikan dalam bentuk diagram alir.



Gambar 3.9 Diagram alir *Principal Component Analysis*

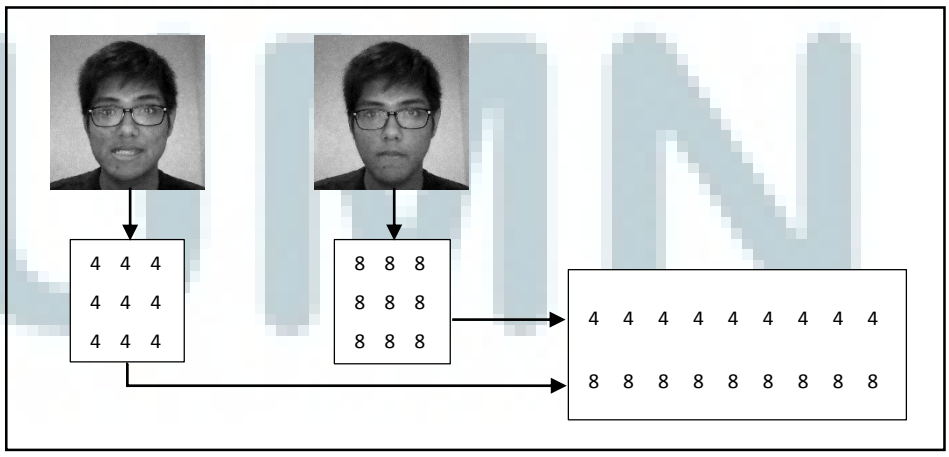
6. Desain *subroutine Eigenface*

Prosedur penyesuaian *Principal Component Analysis* berhubungan langsung dengan implementasi berikutnya, yaitu algoritma *Eigenface.* Proses ini membutuhkan hasil tranformasi matriks dari proses sebelumnya untuk digunakan dalam proses perhitungan. Tahap awal dalam perhitungannya adalah dengan mencari rata-rata matriks dari setiap *training image* yang telah disimpan sebelumnya. Kemudian matriks *vector* dikurangi dengan nilai rata-rata yang sudah didapat. Hal ini dilakukan untuk mencari selisih nilai perbandingan dari setiap *training image* yang ada dalam rangka memperkecil skala proses perhitungan matriks horizontal.

Untuk lebih jelasnya, perhitungan *Eigenface* akan dijabarkan secara bertahap sebagai berikut.

1. Menyusun *flat vector* dari semua *training image*

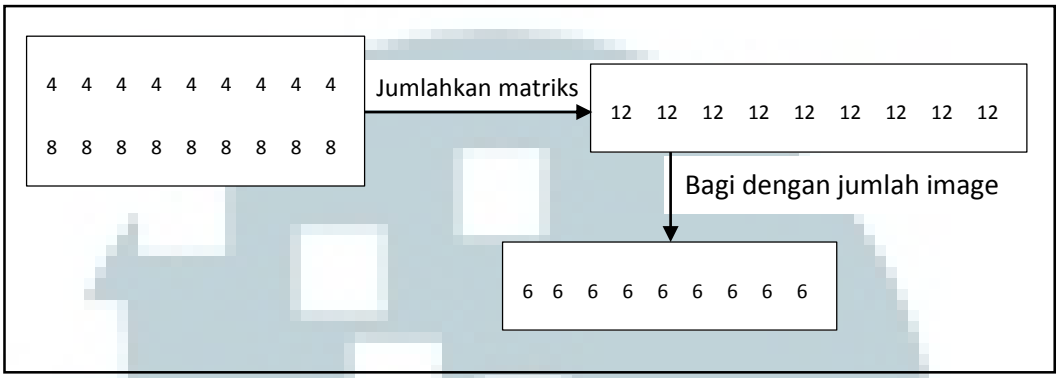
Langkah pertama yang harus dilakukan adalah menyusun *training image* menjadi satu matriks tunggal. Misalnya gambar citra wajah yang disimpan berukuran W x H *pixel* dan berjumlah N buah. Maka matriks yang dihasilkan pada perhitungan menjadi N x (W x H). Ilustrasinya sebagai berikut.



Gambar 3.10 Contoh penyusunan *flat vector*

2. Cari nilai rataan dari *flat vector*

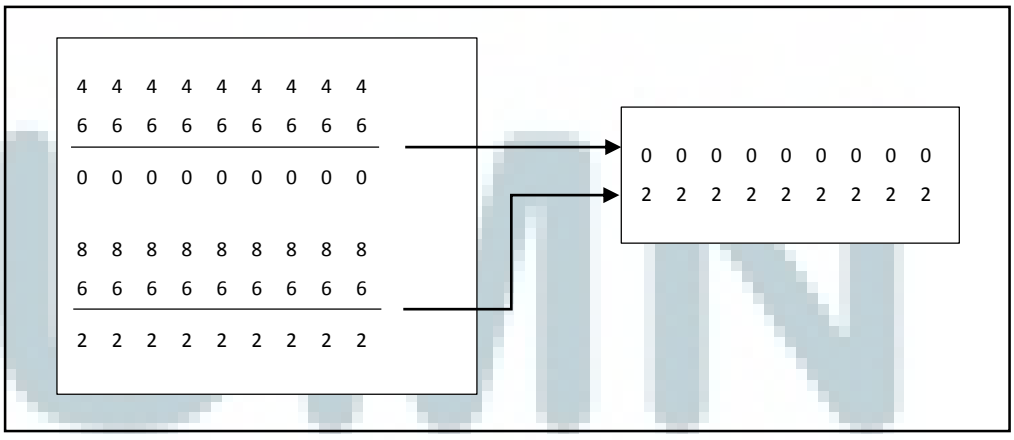
Jumlahkan hasil *flat vector* yang diperoleh, kemudian bagi hasil penjumlahan dengan N sesuai dengan jumlah flat *vector* yang ada.



Gambar 3.11 Contoh perhitungan rataan *flat vector*

3. Tentukan nilai *eigenface* dari tiap *flat vector*

Dengan menggunakan nilai rataan *flat vector* yang sudah didapat, akan dihitung nilai *eigenface* untuk setiap matriks *flat vector* yang disusun sebelumnya. Caranya sederhana, kurangi baris-baris untuk setiap matriks *flat vector* dengan rataan *flat vector*. Ketika nilai yang dihasilkan di bawah nol, maka ganti nilainya dengan nol sehingga tidak ada nilai minus dalam matriks *eigenface*.

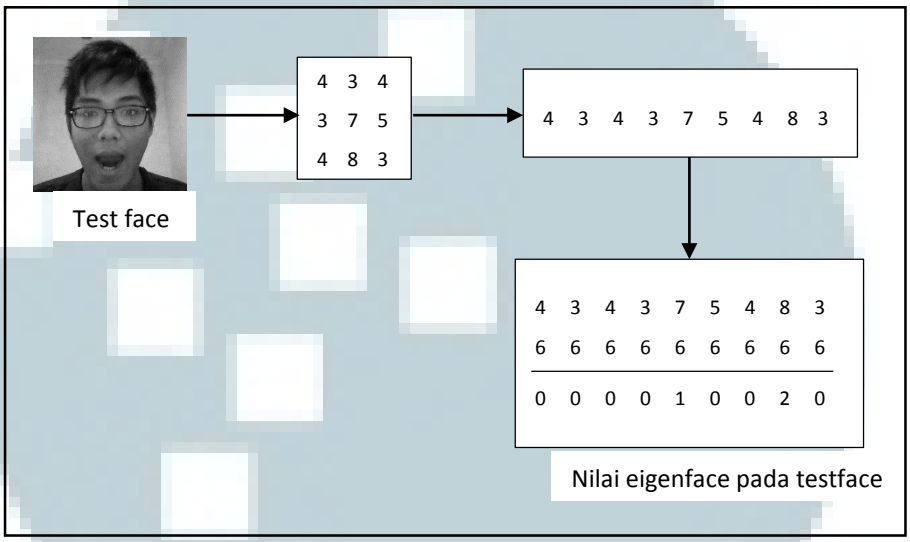


Gambar 3.12 Contoh perhitungan nilai *eigenface* dari *training image*

4. Proses identifikasi gambar citra wajah

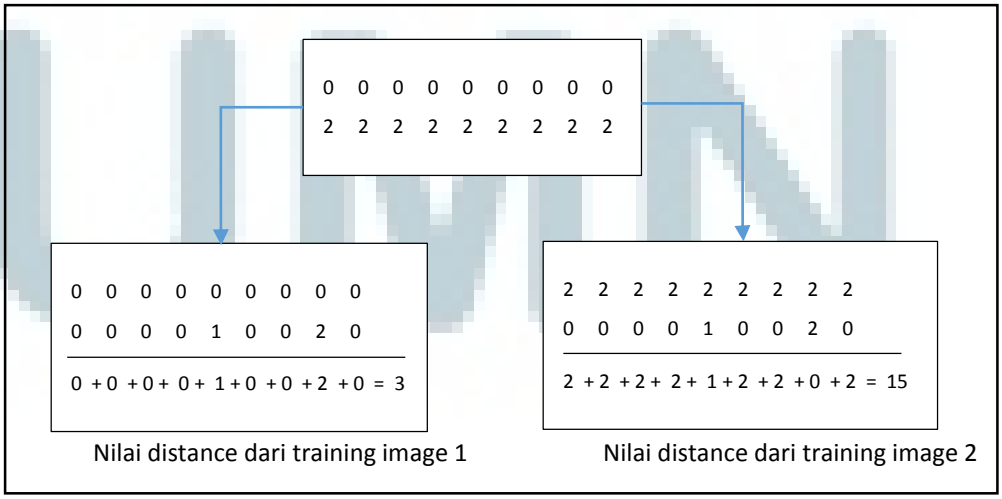
Ketika didapatkan citra wajah yang akan diidentifikasi (*test face*), maka langkah selanjutnya adalah sebagai berikut.

- Hitung nilai *eigenface* untuk matriks *test face* dengan cara yang sama dengan perhitungan pada *flat vector*.



Gambar 3.13 Perhitungan *eigenface* untuk *test face*

- Setelah nilai *eigenface* pada *test face* diperoleh, kemudian dapat dilakukan proses identifikasi dengan menentukan jarak (*distance*) terpendek hasil perbandingan *test face* dan masing-masing nilai *eigenvector training image*.



Gambar 3.14 Contoh proses identifikasi wajah

Gambar 3.15 Diagram alir *subroutine Eigenface*

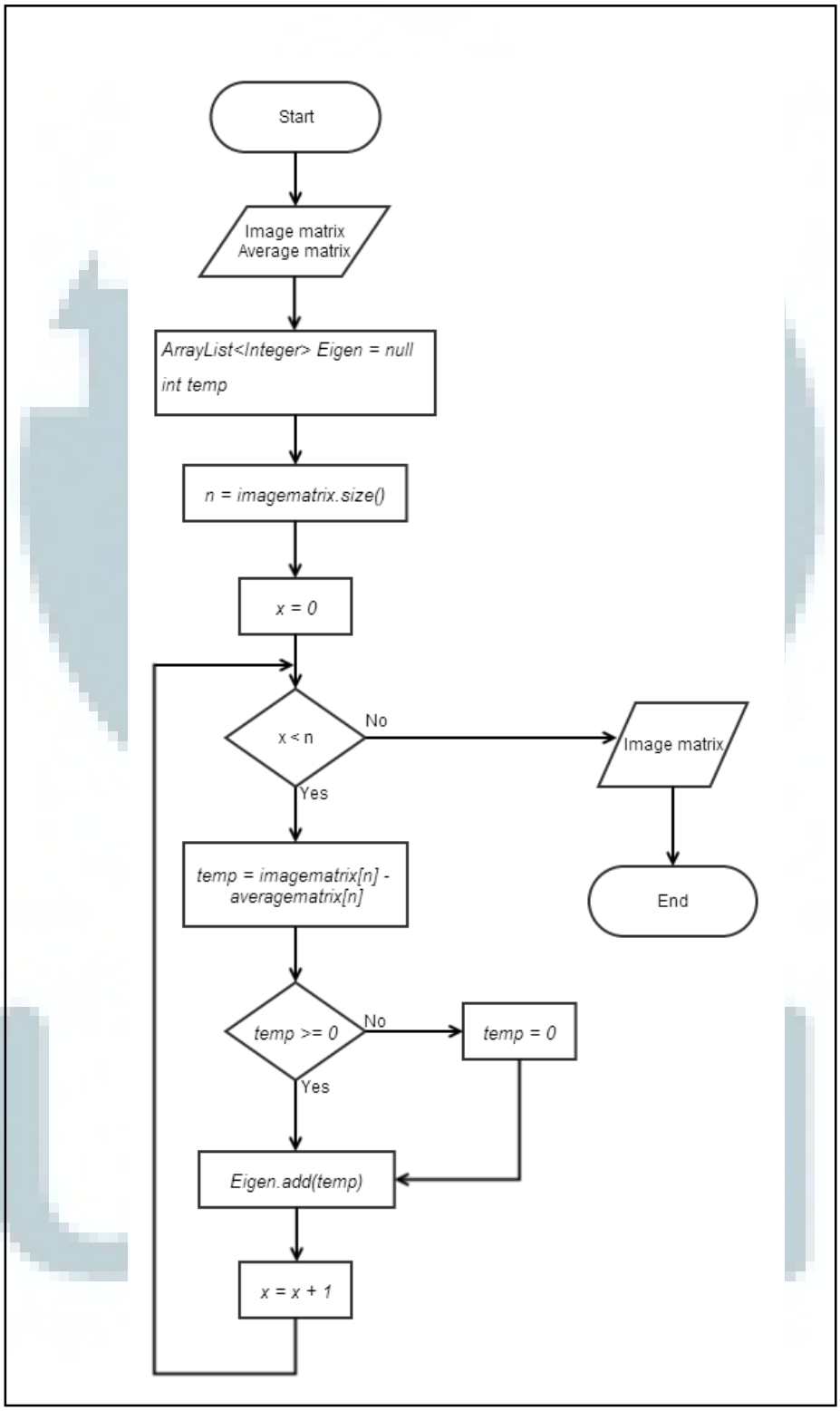
7. Desain *subroutine get eigenvalue*

Proses ini merupakan sebagian proses dari identifikasi *eigenface*. Dalam *subroutine* ini, digambarkan bagaimana mendapatkan *eigenvalue* berupa *flat vector* yang nantinya akan digunakan dalam proses perhitungan. Proses ini dilakukan dengan metode konversi matriks dari gambar citra wajah menjadi matriks satu dimensi yang kemudian direpresentasikan dalam sebuah *array list of array list of integer*. Dalam kumpulan *array list*, didapatkan sebuah wadah perhitungan tunggal, sehingga dengan mudah dapat dilakukan penjumlahan maupun pencarian selisih yang nantinya akan dilakukan pada proses akhir dalam identifikasi *eigenface*.

Banyaknya isi dalam *array list* tergantung dari banyak jumlah gambar citra wajah yang selanjutnya akan disebut sebagai *training image*. Semakin banyak *training image* yang tersimpan, tentunya perhitungan akan semakin rumit dan waktu proses yang dibutuhkan akan semakin lama. Namun, fakta ini berbanding lurus dengan tingkat akurasi dari perhitungan *eigenface* ini. Dengan semakin banyaknya citra wajah, maka tingkat akurasi dalam proses identifikasi akan semakin meningkat.

Dalam *subroutine* ini, ada dua proses berbeda yang menggunakannya. Pertama adalah dalam perhitungan *training image*. Dalam proses pertama ini, dibutuhkan *input* berupa matriks dari kumpulan *training image* dan hasil rataan *eigenvector*, kemudian dihitung sesuai dengan prosedur yang ada. Proses kedua adalah ketika identifikasi dilakukan saat ada *test face* yang masuk. Proses akan terus menerus melakukan konversi matriks dan melakukan perhitungan matematis dalam proses identifikasi sampai didapatkan gambar citra wajah dengan tingkat akurasi

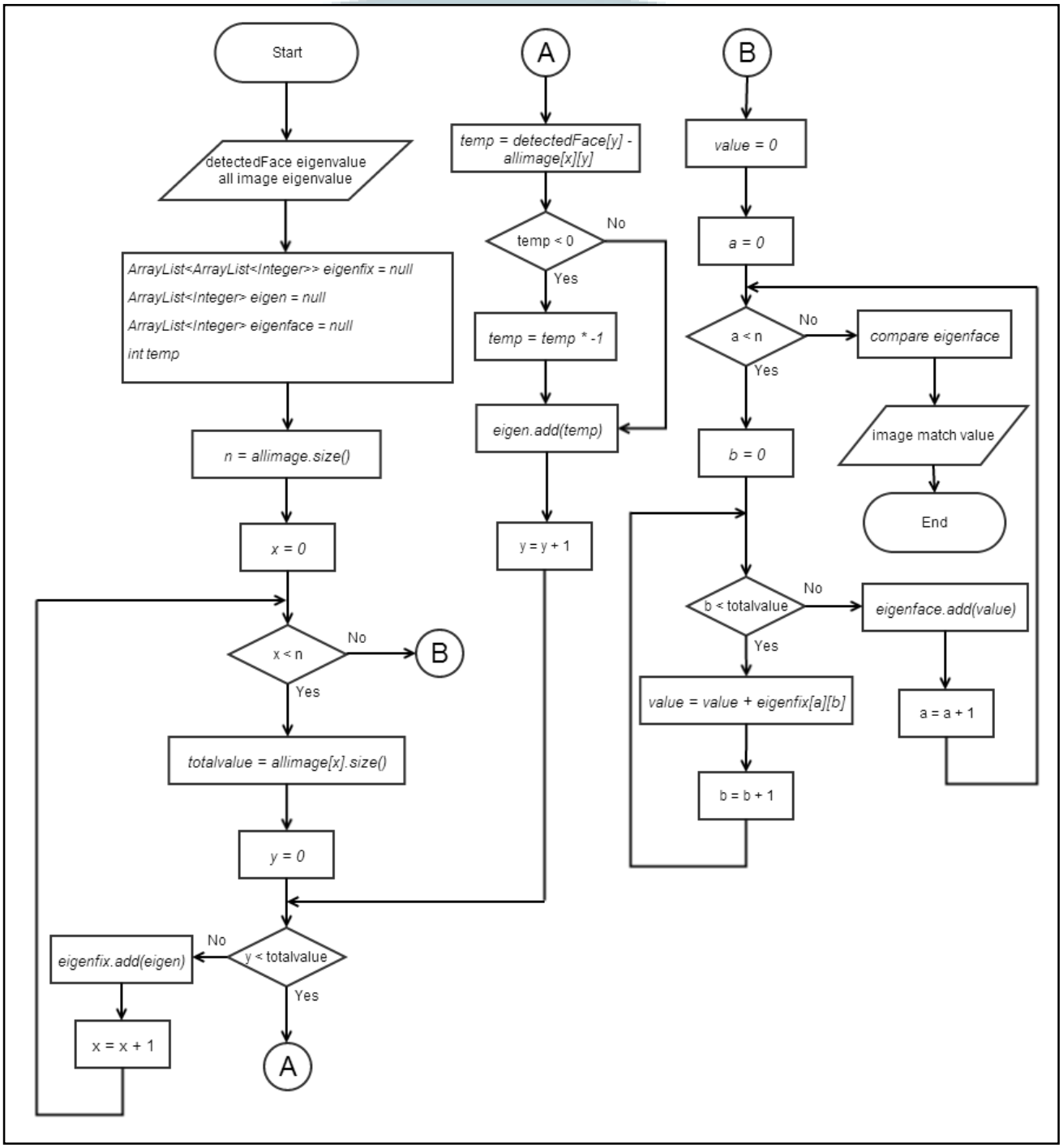
yang cukup. Setelah itu, maka proses akan dilanjutkan melalui perhitungan persentase kemiripan wajah. Berikut adalah gambaran dari *subroutine* tersebut.



Gambar 3.16 Diagram alir *subroutine get eigenvalue*

8. Desain *subroutine identify eigenvalue*

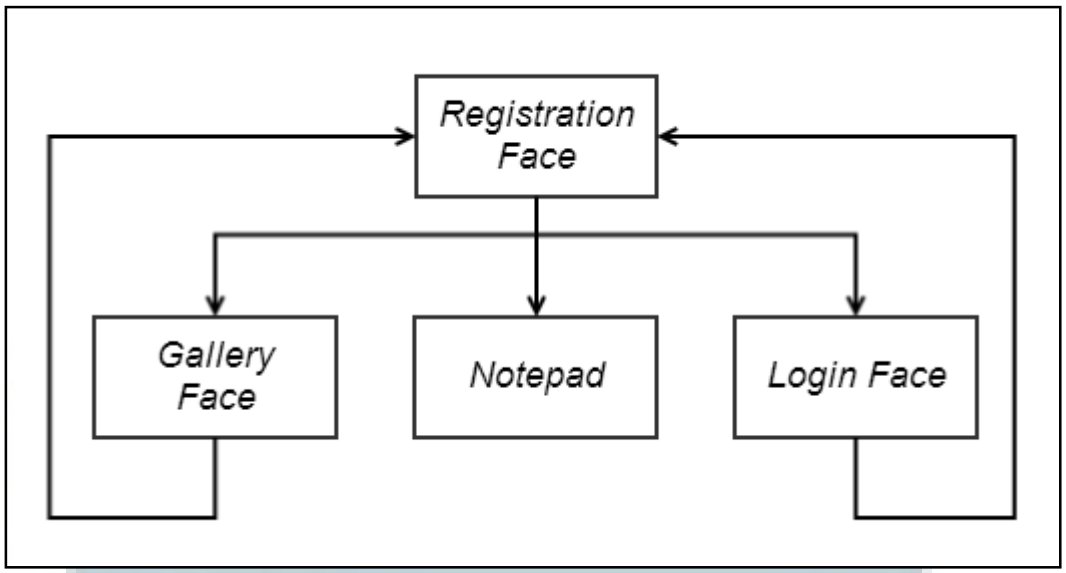
Dalam proses perhitungannya, *subroutine* ini sudah dijelaskan dalam pemaparan algoritma *eigenface* sebelumnya. Berikut ini adalah terusan hasil perhitungan nilai *eigenface* untuk mendapatkan persentase kemiripan citra wajah.



Gambar 3.17 Diagram alir *subroutine identify eigenvalue*

3.3.4 Hirarki Menu

Hirarki menu dari pengembangan aplikasi ini dijabarkan dalam diagram pada gambar 3.18



Gambar 3.18 Hirarki menu

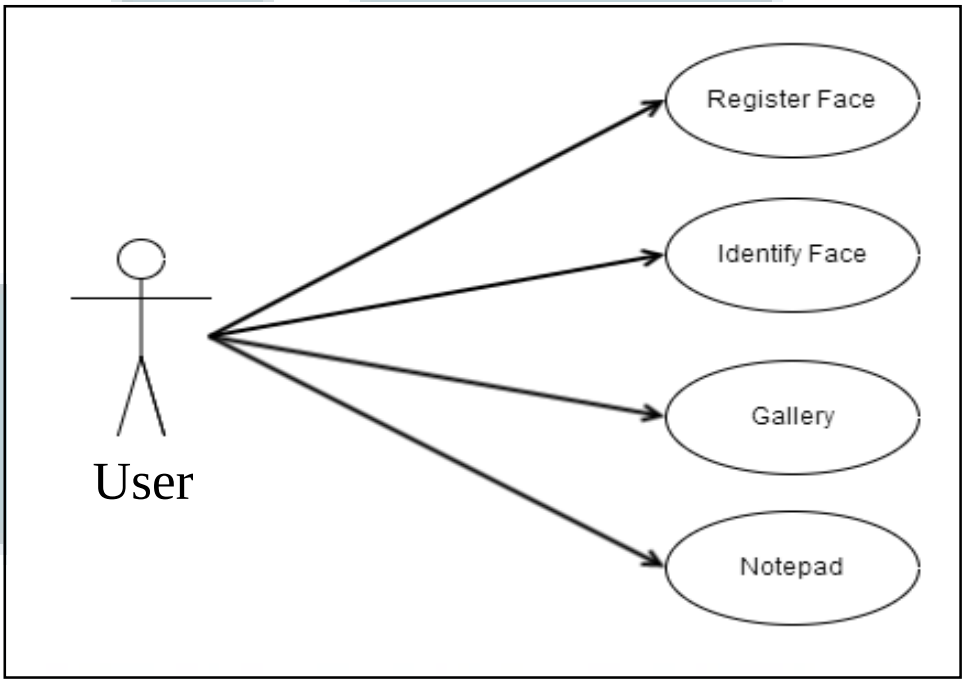
Proses pengembangan aplikasi ini dilakukan pada sistem operasi *mobile* Android dengan menggunakan Bahasa pemrograman Java. Dalam implementasi pembuatan antar muka dari aplikasi, digunakan XML sebagai dasar dan Java sebagai logika pembuatan program. Setiap halaman aktif yang terdapat pada sistem Android disebut dengan istilah *activity*. Dalam *activity* inilah *user* berinteraksi dengan aplikasi. Setiap komponen yang digambarkan pada diagram hirarki diatas akan diwakilkan dengan masing-masing satu *activity*.

Login face merupakan *activity* dimana *user* akan melakukan identifikasi wajah untuk masuk ke aplikasi *Smart Notepad. Gallery face* adalah tempat *user* dapat melihat *training image* yang sudah dikumpulkan pada saat fase registrasi. *Registration face* merupakan *activity* dimana *user* akan melakukan registrasi

dengan mengambil citra wajah guna proses identifikasi nantinya. *Notepad* sendiri adalah aplikasi yang ditampilkan jika proses identifikasi wajah berhasil dilakukan.

3.3.5 Use Case Diagram

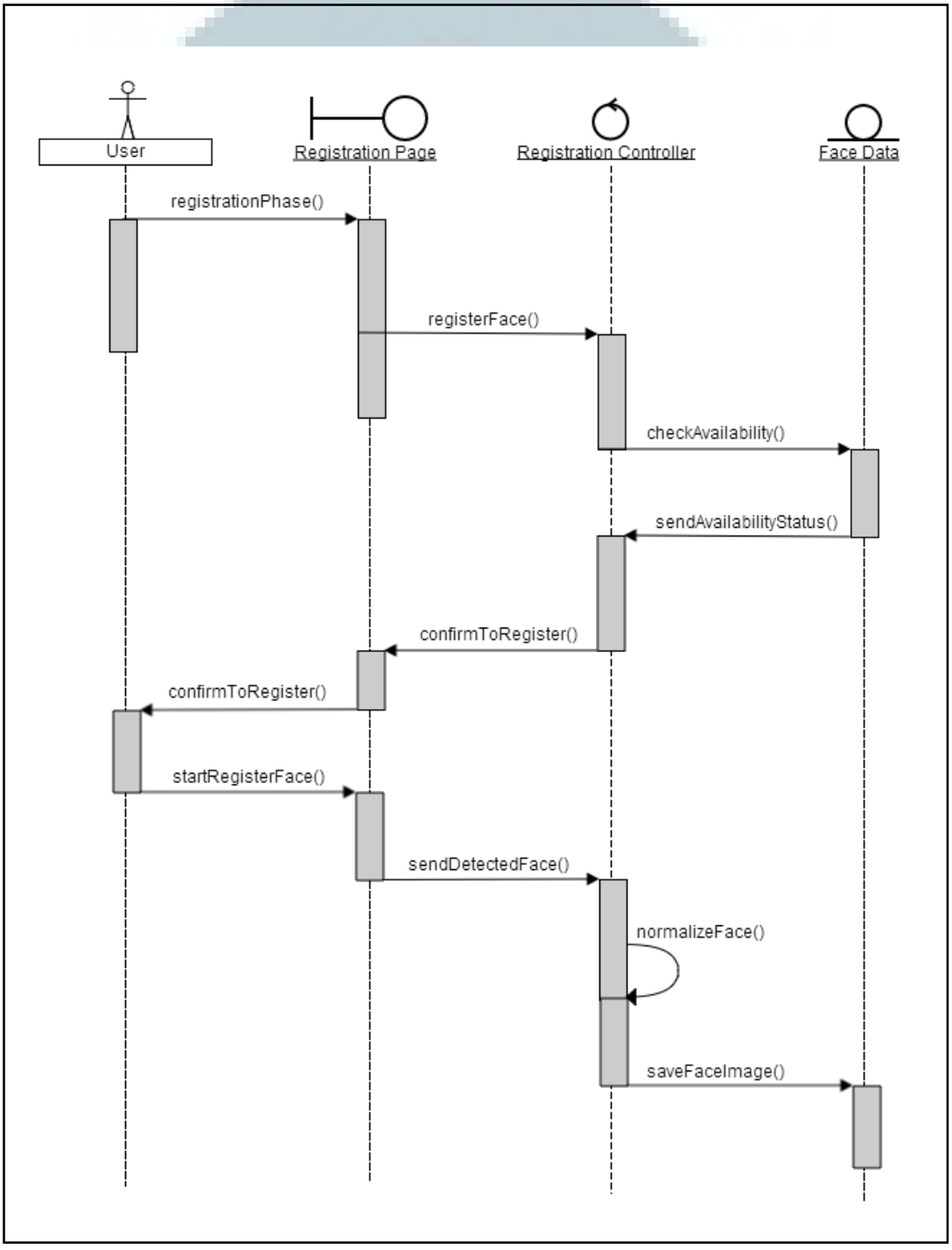
Dalam aplikasi ini, *user* memiliki otoritas untuk melakukan berbagai hal, seperti registrasi dan otentikasi gambar citra wajah. Hal-hal yang dapat dilakukan *user*, antara lain *register face*, yang merupakan aktivitas dimana *user* dapat melakukan registrasi gambar citra wajah untuk digunakan pada proses otentikasi, *identify face*, yang menjadi aktivitas dimana identifikasi wajah dilakukan sehingga *user* bisa masuk ke dalam aplikasi yang sesungguhnya, *notepad* yang menjadi aplikasi yang dapat digunakan oleh *user* ketika proses otentikasi berjalan dengan lancar, dan yang terakhir adalah *gallery*, dimana ditampilkan gambar-gambar hasil registrasi dan *user* dapat dengan mudah melakukan perubahan terhadap gambar-gambar tersebut. Gambar 3.19 ini merupakan desain diagram *use case* dari sistem.



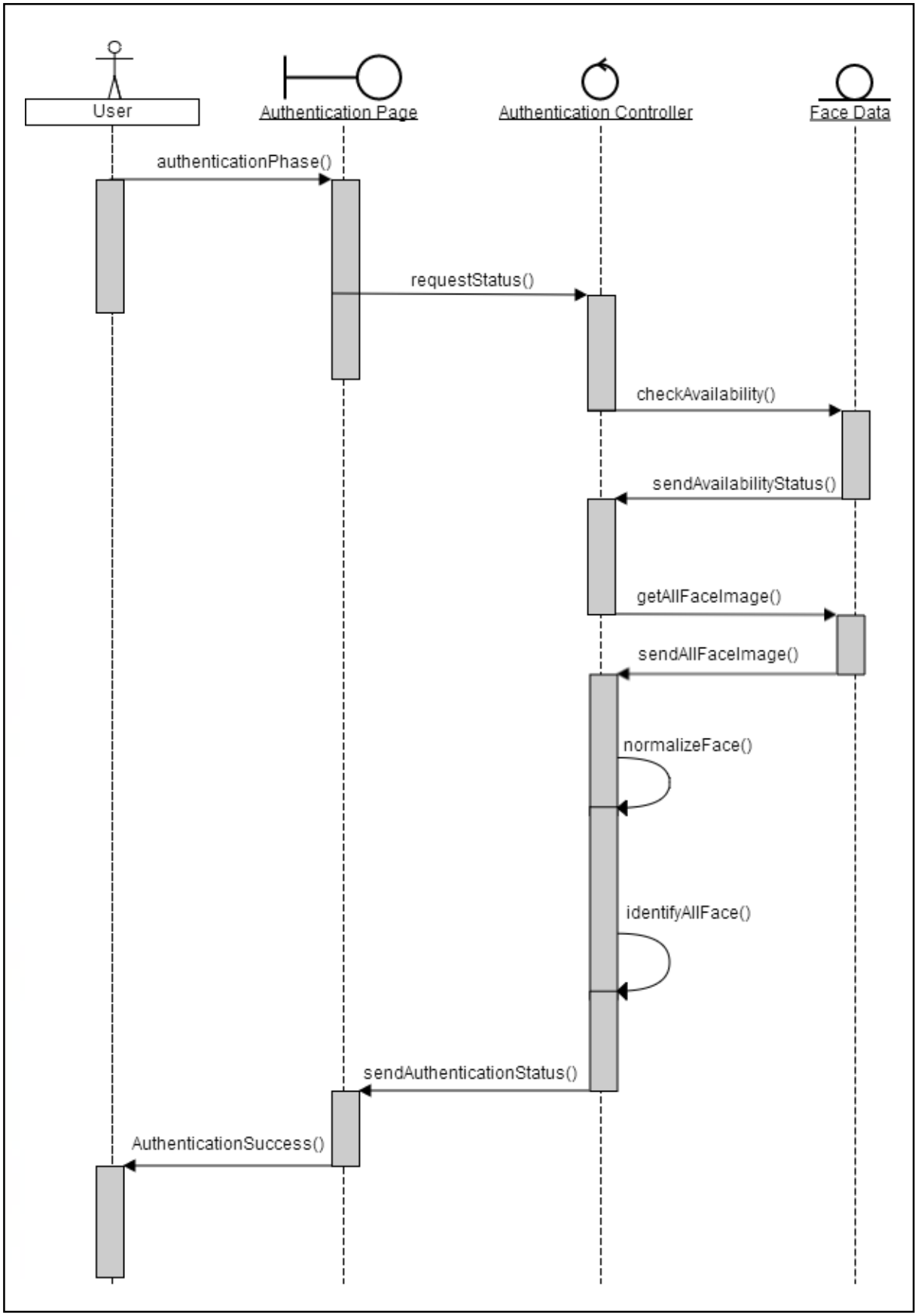
Gambar 3.19 *Use case diagram*

3.3.6 Sequence Diagram

Dalam sistem ini, ada dua aktivitas utama yang dilakukan, yaitu fase registrasi dan fase otentikasi. Gambar 3.20 dan 3.21 merupakan rancangan *sequence diagram* dari kedua proses utama tesebut.



Gambar 3.20 *Sequence diagram* fase registrasi



Gambar 3.21 *Sequence diagram* fase otentikasi

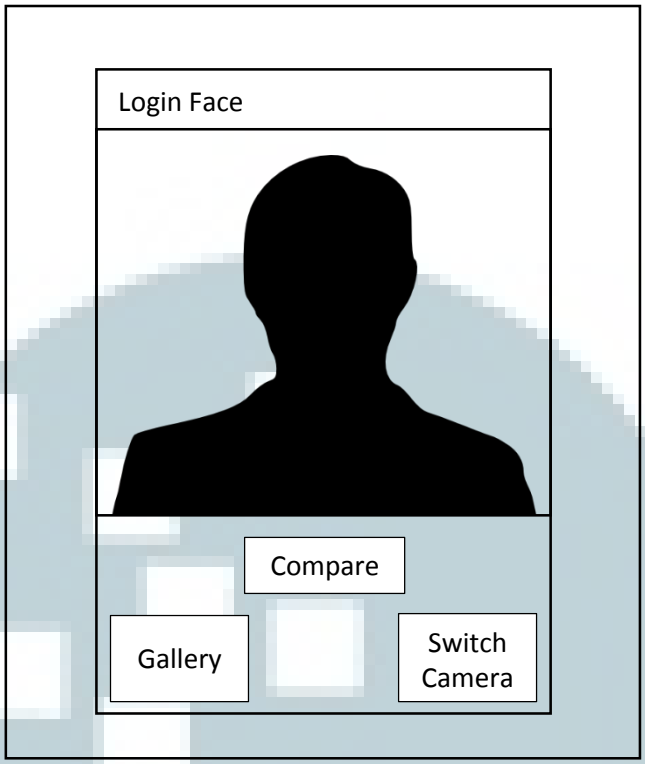
3.3.7 Tampilan Antarmuka

Sebagai tahap awal pembuatan sistem, dirancang terlebih dahulu segala macam bentuk antarmuka yang digunakan dalam aplikasi. Berikut ini adalah rancangan antarmuka yang digunakan.

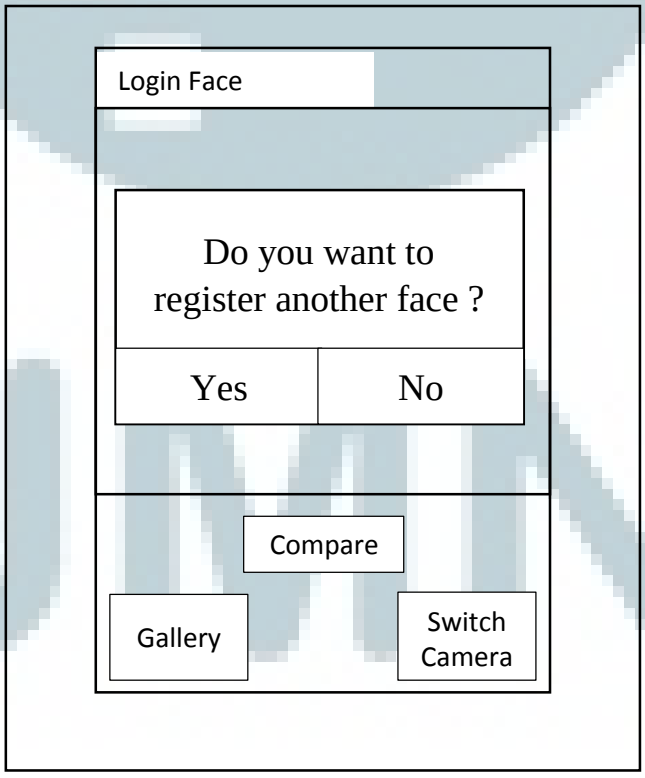
1. *Login Face*

   Activity untuk *login* ini merupakan *activity* yang pertama kali dijalankan ketika aplikasi dimulai dan digunakan untuk proses identifikasi sebelum masuk ke aplikasi *Smart Notepad*. Jika ternyata *user* belum melakukan registrasi wajah, maka halaman aplikasi akan langsung diarahkan kepada *activity registration face*. Dan jika wajah yang ada dalam *storage* belum maksimal (dalam kasus ini tiga buah pengambilan gambar), maka aplikasi akan melakukan konfirmasi apakah akan dilakukan pengambilan gambar lagi atau langsung menuju proses *login*.

   Orientasi layar perangkat yang digunakan adalah berupa *portrait* untuk mempermudah *user* dalam proses pengambilan gambar citra wajah. *Activity* ini berisi *surface view* untuk pengambilan citra wajah melalui kamera, tombol untuk memulai proses identifikasi wajah, tombol untuk masuk ke *activity gallery face*, dan tombol untuk *switch camera* yang digunakan untuk mengubah penggunaan kamera depan atau belakang. Gambar 3.22 dan 3.23 berikut ini memperlihatkan rancangan tampilan dari antarmuka *activity login face*.



Gambar 3.22 Rancangan antarmuka *activity login face* 1

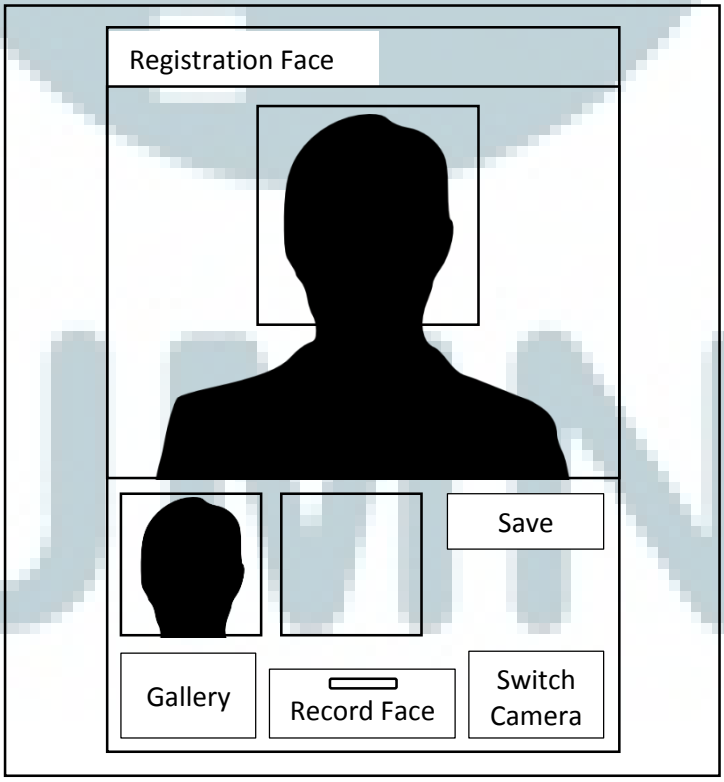


Gambar 3.23 Rancangan antarmuka *activity login face* 2

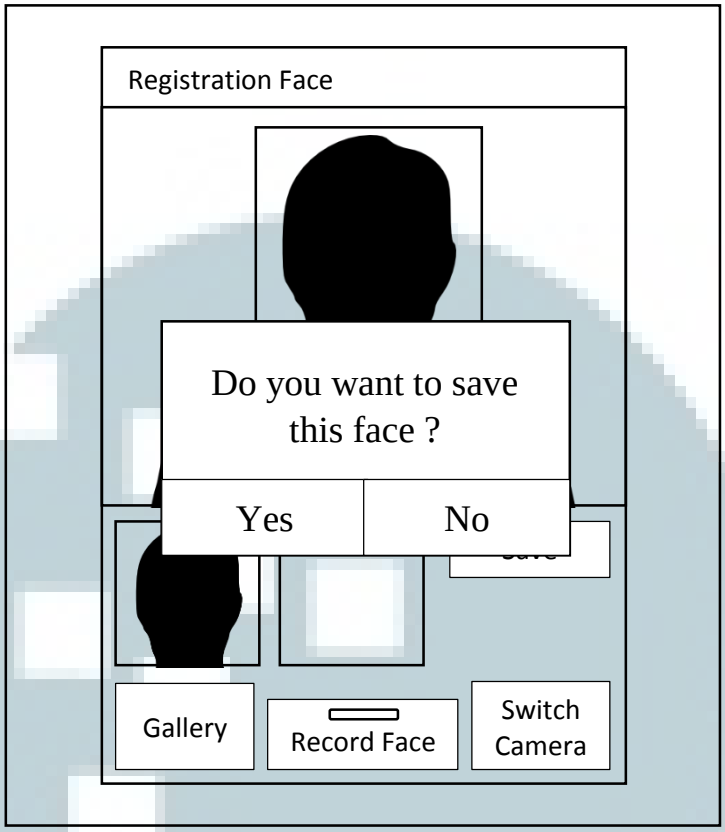
2. *Registration Face*

Dalam *activity registration face*, dilakukan pendaftaran gambar citra wajah sebagai alat bantu dalam proses otentikasi. Jumlah gambar yang diambil dapat bervariasi sesuai dengan keinginan *user*. Jumlah gambar citra wajah yang bisa disimpan maksimal tiga buah gambar. Semakin banyaknya gambar yang disimpan, maka tingkat akurasi dalam perhitungan identifikasi wajah akan semakin meningkat.

Activity registration face ini terdiri dari *surface view* untuk pengambilan citra wajah, *image view* yang digunakan sebagai *preview* gambar wajah yang akan disimpan, tombol *save* untuk menyimpan gambar, tombol *gallery* untuk masuk ke daftar citra wajah yang sudah diambil, *toggle button record face* digunakan untuk mengambil gambar wajah yang akan disimpan, dan tombol *switch camera* untuk penyesuaian kamera depan atau belakang.



Gambar 3.24 Rancangan antarmuka *activity registration face* 1

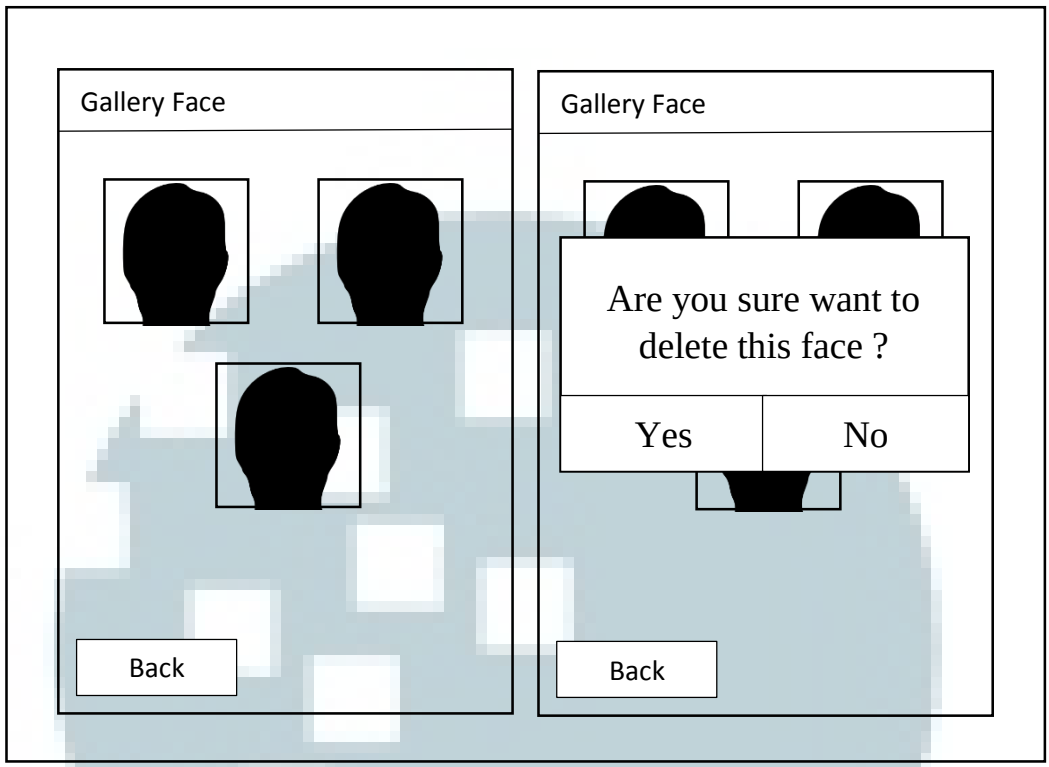


Gambar 3.25 Rancangan antarmuka *activity registration face* 1

3. Gallery Face

Activity gallery face ini digunakan untuk menampilkan gambar-gambar citra wajah yang sudah diambil sebelumnya. Rangkaian isi dari *activity* ini terdiri atas tiga buah *image view* yang masing-masing menjadi tempat untuk ditampilkannya gambar citra wajah, tombol *back* untuk kembali ke menu sebelumnya, dan *user* memiliki otoritas untuk menghilangkan data-data wajah yang dirasa kurang cocok dan dapat dilakukan pengambilan gambar wajah lagi dengan melalui proses registrasi.

Gambar 3.26 menggambarkan rancangan dasar dari *activity gallery face* yang akan dibuat.

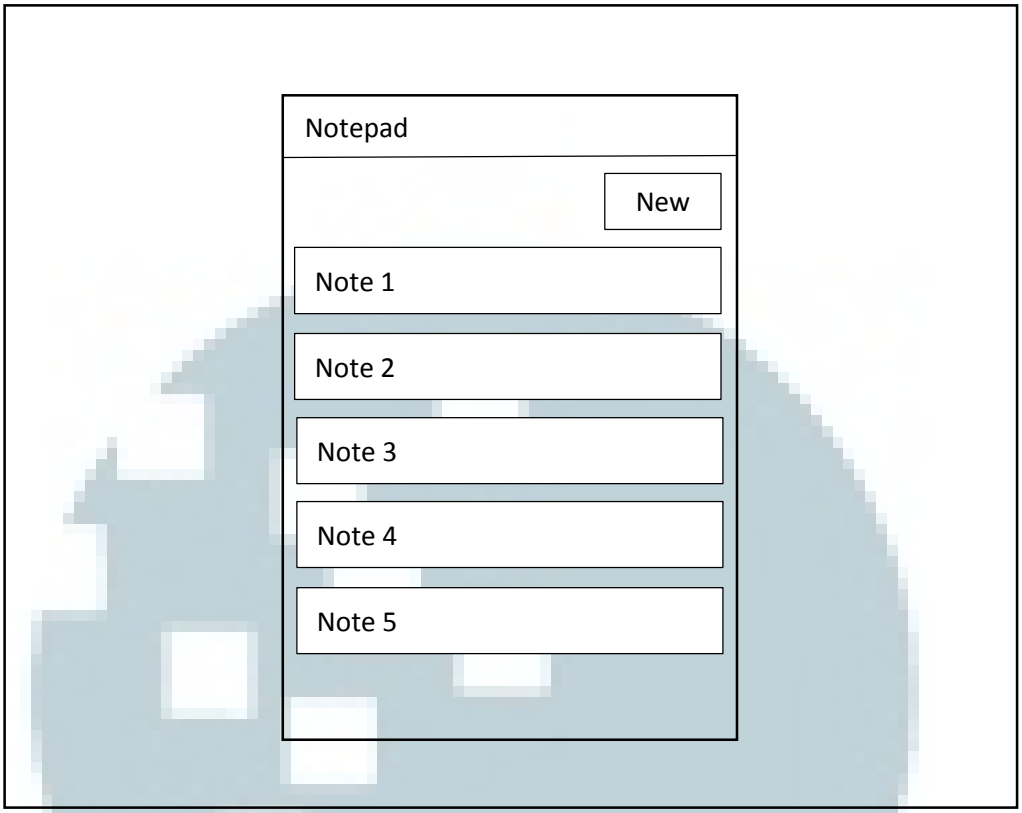


Gambar 3.26 Rancangan antarmuka *activity gallery face*

4. Notepad

Activity notepad ini merupakan sebuah aplikasi kecil yang akan muncul, jika proses identifikasi wajah berjalan dengan benar, dalam artian citra wajah yang digunakan tepat dan cocok dengan tingkat kemiripan tertentu dengan membandingkan pada seluruh *training image*.

Notepad berisi catatan penting dan diasumsikan menyimpan data-data pribadi dari *user* yang menggunakan otentikasi ini. Notepad ini menjadi aplikasi *dummy* sebagai bahan percobaan untuk dapat masuk ke suatu aplikasi melalui proses otentikasi telebih dahulu. Gambar rancangan *activity* notepad ini dibuat berdasarkan contoh yang didapat dari Android SDK. Gambar 3.27 berikut merupakan rancangan antarmuka dari *activity notepad*.



Gambar 3.27 Rancangan antarmuka *activity notepad*