BAB III

PERANCANGAN PERANGKAT LUNAK

Pada bab ini dibahas mengenai tahap-tahap perancangan aplikasi sistem pengenal daun untuk klasifikasi spesies tanaman dengan menggunakan algoritma jaringan saraf tiruan berbasis Android. Bab ini dibagi menjadi tiga sub bab, yaitu sub bab kebutuhan perangkat lunak yang berisikan tentang keperluan yang dibutuhkan dalam proses pembuatan, sub bab analisa data mengenai kebutuhan perangkat lunak, dan sub bab yang terakhir yang berisi tentang desain dan gambaran akhir dari perangkat lunak.

3.1 Kebutuhan Perangkat Lunak

Sub bab ini menjelaskan hal-hal mengenai kebutuhan perangkat lunak yang dirancang. Untuk dapat menemukan kebutuhan perangkat lunak, maka dilakukan studi terhadap beberapa literatur-literatur yang membahas masalah pengenalan daun.

3.1.1 Hasil Studi Literatur

Pada bagian sub bab ini akan dijelaskan hal-hal mengenai seluruh kegiatan studi literatur yang sudah dilakukan untuk mendapatkan kebutuhan perangkat lunak. Kebutuhan-kebutuhan tersebut nantinya akan dapat digunakan sebagai bahan masukan dalam pembuatan perangkat lunak.

a. Rancangan Penelitian

Metode penelitian yang digunakan adalah studi literatur, sehingga hasil yang dapat dikumpulkan adalah data – data berupa data deskriptif, yang merupakan hasil dari analisa dan kesimpulan dari penjelasan yang ditemukan di literatur yang dipelajari.

b. Aspek Studi Literatur

Aspek yang akan digali dari studi literatur adalah sistem, fitur dasar, dan masalah dalam sistem pengenalan tanaman yang ditemukan di literatur yang dipelajari.

3.1.2 Analisis Hasil Studi Literatur

Dari hasil studi literatur, perangkat lunak yang dirancang membutuhkan kemampuan untuk dapat mendeteksi bentuk daun tanpa memerlukan kondisi latar belakang yang benar-benar putih polos, maka akan digunakan algoritma yang akan dijelaskan pada bab 3.3.1

Selain itu dari hasil studi literatur yang membahas mengenai pengenalan daun, Fitur dasar yang perlu dimiliki dalam perangkat lunak sistem pengenalan tanaman adalah kemampuan untuk menampilkan informasi dasar berupa nama umum dan nama latin dari tanaman tersebut. Selain itu, karena nama latin dan nama umum dari suatu tanaman dianggap kurang dapat membantu pengguna dalam memperoleh informasi, maka juga akan ditampilkan informasi mengenai deskripsi dan fitur dari tanaman tersebut.

Fitur dasar lainnya yang diperlukan adalah pengondisian keadaan citra *digital* agar dapat dengan mudah dikenali oleh sistem pengenalan daun. Sistem pengenalan daun sangatlah sensitif terhadap posisi dari daun dalam citra *digital*, oleh karena itu diperlukan adanya fitur untuk memposisikan daun sedemikian rupa agar dapat dikenali oleh sistem.[33]

Dalam studi literatur yang dilakukan, ditemukan juga beberapa masalah yang dapat menjadi bahan masukan dalam perancangan perangkat lunak. Salah satunya adalah adanya kemungkinan bahwa ketika akan dilakukan proses pengenalan daun, tidak ada koneksi / jaringan internet, maka untuk dapat mengatasi masalah tersebut, data – data yang diperlukan untuk melakukan pengenalan daun akan disimpan secara lokal di perangkat keras yang digunakan.[34]

Dari analisa hasil studi literatur diatas, maka dapat disimpulkan bahwa perangkat lunak yang dirancang nantinya perlu untuk dapat menampilkan informasi selain nama latin dari tanaman, mengkondisikan citra *digital* daun agar dapat dikenali oleh sistem pengenalan daun, dan juga agar perangkat lunak dapat bekerja walaupun tidak ada koneksi / jaringan internet.

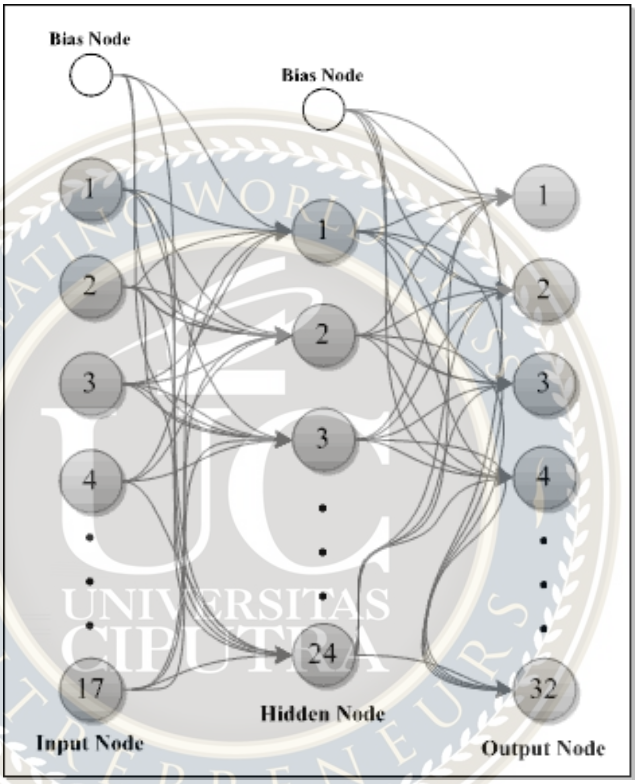
[33] James Painter, "Automatic Plant Leaf Classification for a Mobile Field Guide" (Stanford University), 1.
[34] James Painter, "Automatic Plant Leaf Classification for a Mobile Field Guide" (Stanford University), 3.

3.2 Rancangan Jaringan Saraf Tiruan

Bentuk arsitektur jaringan saraf tiruan yang digunakan, secara umum sama dengan arsitektur jaringan saraf tiruan yang telah dibahas pada bab 2. Jaringan saraf tiruan yang digunakan terdiri dari 3 lapisan (*layer*), yaitu input *layer*, *hidden layer*, dan output *layer*. Input *layer* merupakan lapisan yang terdiri dari *node* masukan yang jumlahnya sesuai dengan jumlah fitur daun yang digunakan, yaitu 17 fitur yang sudah dibahas pada bab 2 yang berarti 17 *node* masukan. Tabel 3.1 berisi 17 fitur morfologi yang menjadi input dari jaringan saraf tiruan. *Hidden layer* merupakan lapisan yang terdiri dari *node* tersembunyi yang jumlahnya adalah 24 untuk 32 spesies tanaman dan 18 untuk 19 spesies tanaman. Dan yang terakhir output *layer* yang merupakan lapisan yang terdiri dari *node* keluaran. Jumlah dari *node* keluaran sendiri merepresentasikan jumlah spesies tanaman yang akan dikenali nantinya oleh jaringan saraf tiruan, pada tugas akhir ini digunakan 19 output *node* (19 spesies tanaman) dan 32 output *node* (32 spesies tanaman). Dalam rancangan arsitektur jaringan saraf tiruan yang digunakan juga terdapat *node* bias yang sebanyak satu, masing-masing diletakkan di input *layer* dan *hidden layer*. Pada gambar 3.1 dapat dilihat contoh model rancangan jaringan saraf tiruan yang digunakan. Representasi dari lapisan output pada rancangan jaringan saraf tiruan menggunakan angka 0 dan 1. Pada gambar 3.1 contohnya terdapat 5 output yang merepresentasikan 5 spesies tanaman. Output untuk spesies tanaman ke-1 adalah 1,0,0,0,0 dan output untuk spesies tanaman ke-2 adalah 0,1,0,0,0 ,begitu pula seterusnya sampai jumlah spesies yang diinginkan.

Fitur	Fitur	Fitur
Aspect Ratio	*Eccentricity*	*Angle Code Histogram* [1]
Rectangularity	*Form Factor*	*Angle Code Histogram* [2]
Convex Hull Area Ratio	*Regional Moments of Inertia* [1]	*Angle Code Histogram* [3]
Convex Hull Perimeter Ratio	*Regional Moments of Inertia* [2]	*Angle Code Histogram* [4]
Sphericity	*Regional Moments of Inertia* [3]	*Angle Code Histogram* [5]
Circularity	*Regional Moments of Inertia* [4]	

Tabel 3.1 Daftar fitur morfologi untuk input jaringan saraf tiruan

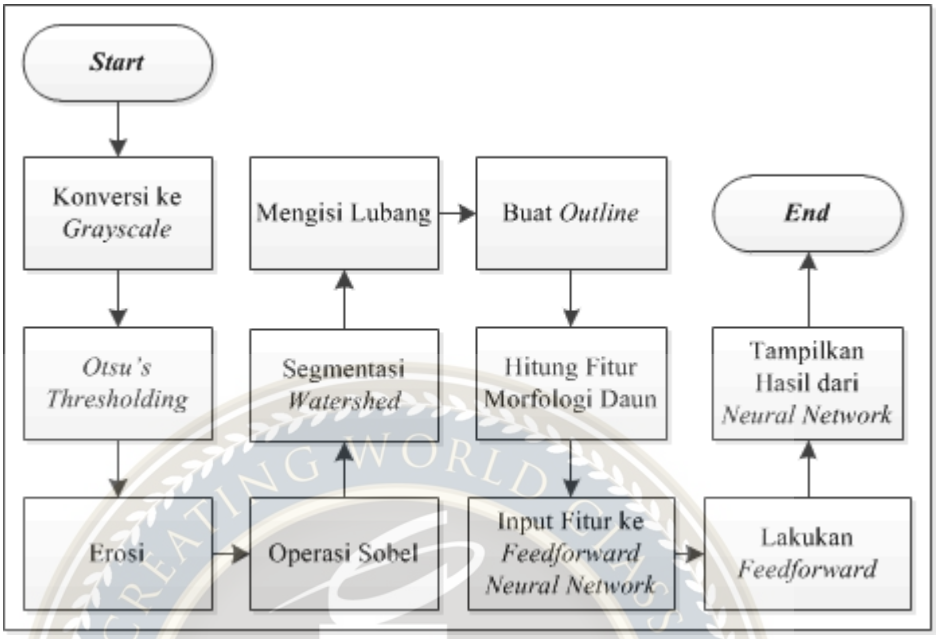


Gambar 3.1 Rancangan Jaringan Saraf Tiruan (32 Spesies Tanaman)

3.3 Algoritma Perangkat Lunak

Pada sub bab ini akan dijelaskan mengenai algoritma pemrosesan citra *digital* dan algoritma jaringan saraf tiruan dari perangkat lunak sistem pengenalan daun ini. Algoritma berikut dibuat sedemikian rupa agar dapat memenuhi kebutuhan dari perangkat lunak. Algoritma pada perangkat lunak yang dirancang, dibagi menjadi 2, yaitu yang berfungsi pada perangkat klien (Android) dan yang berfungsi di *server* (J2SE).

3.3.1 Algoritma Klien



Gambar 3.2 Algoritma pada perangkat klien

Gambar 3.2 diatas merupakan gambaran algoritma yang berfungsi pada perangkat lunak di klien (Android). Secara umum, algoritma pada klien berfungsi untuk membaca daun dan menampilkan hasil proses jaringan saraf tiruannya. Algoritma-algoritma yang digunakan tersebut akan dijelaskan lebih lanjut, yaitu antara lain:

1. Pengambilan citra *digital* daun baik melalui kamera ataupun melalui memori dari perangkat keras.
2. Pemrosesan citra *digital* yang terdiri dari beberapa proses, yaitu antara lain:
   a. Mengubah citra *digital* yang berwarna menjadi citra *digital grayscale*.

b. Pemrosesan citra *digital grayscale* menjadi citra yang berwarna hitam dan putih untuk membedakan *foreground* dan *background* dari citra *digital* menggunakan metode *Otsu's Thresholding*.

c. Selanjutnya dilakukan operasi morfologi terhadap citra *digital* dengan "mengurangi" luas dari *foreground* menggunakan metode erosi.

d. Mencari *gradient* dari citra *digital grayscale* dengan menggunakan metode *Sobel's Operation*.

e. Menghitung *pixel - pixel* yang saling berhubungan sehingga dapat ditemukan region yang paling luas dari citra *digital* hasil *Otsu's Thresholding* yang sudah melalui proses erosi.

f. Melakukan segmentasi daun terhadap citra *digital* dengan menggunakan *marker* yang sudah ditentukan sebelumnya.

g. Mengisi dan menutup lubang yang ditemukan pada segmentasi daun di citra *digital* menggunakan metode *connected-component labeling*.

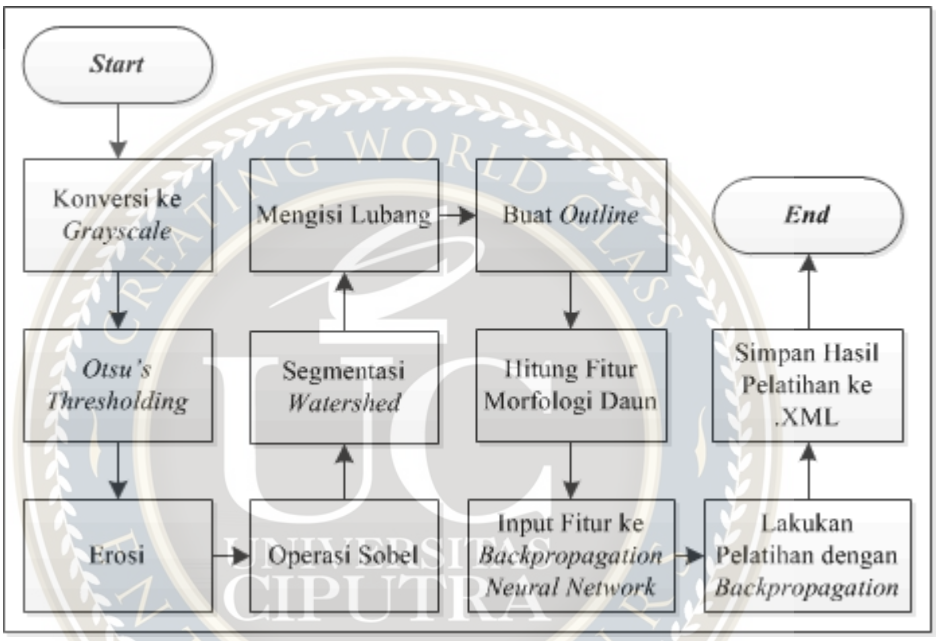
h. Memotong citra *digital* sehingga segmen yang tidak diperlukan dapat dihapus, untuk mempercepat proses selanjutnya.

i. Mencari garis luar dari daun yang sudah tersegmentasi dan menipiskan garis luar tersebut menjadi 1 *pixel*.

3. Menghitung dan mencari fitur-fitur dari daun yang sudah tersegmentasi.

4. Fitur-fitur yang sudah didapatkan sebelumnya diinputkan ke dalam jaringan saraf tiruan yang masing-masing beratnya sudah ditentukan dari hasil pelatihan.
5. Menampilkan hasil dari jaringan saraf tiruan.

3.3.2 Algoritma *Server*



Gambar 3.3 Algoritma pada perangkat *server*

Gambar 3.3 menjelaskan mengenai langkah-langkah algoritma yang berada di *server*. Secara umum, algoritma tersebut berfungsi untuk melakukan pelatihan terhadap jaringan saraf tiruan menggunakan *backpropagation neural network*. Berikut ini adalah urutan proses untuk melakukan pelatihan terhadap jaringan saraf tiruan menggunakan metode *backpropagation neural network*, yaitu antara lain:

1. Melakukan input berupa citra *digital* daun sebagai *training* set.

2. Pemrosesan citra *digital* yang terdiri dari beberapa proses, yaitu antara lain:
   a. Mengubah citra *digital* yang berwarna menjadi citra *digital grayscale*.
   b. Pemrosesan citra *digital grayscale* menjadi citra yang berwarna hitam dan putih untuk membedakan *foreground* dan *background* dari citra *digital* menggunakan metode *Otsu's Thresholding*.
   c. Mencari *gradient* dari citra *digital grayscale* dengan menggunakan metode *Sobel's Operation*.
   d. Menghitung *pixel - pixel* yang saling berhubungan sehingga dapat ditemukan region yang paling luas dari citra *digital* hasil *Otsu's Thresholding* yang sudah melalui proses erosi.
   e. Melakukan segmentasi daun terhadap citra *digital* dengan menggunakan *marker* yang sudah ditentukan sebelumnya.
   f. Mengisi dan menutup lubang yang ditemukan pada segmentasi daun di citra *digital* menggunakan metode *connected-component labeling*.
   g. Memotong citra *digital* sehingga segmen yang tidak diperlukan dapat dihapus, untuk mempercepat proses selanjutnya.
   h. Mencari garis luar dari daun yang sudah tersegmentasi dan menipiskan garis luar tersebut menjadi 1 *pixel*.
3. Menghitung dan mencari fitur-fitur dari daun yang sudah tersegmentasi.

4. Setelah semua fitur dari *training* set didapatkan, maka diinputkan ke dalam *backpropagation neural network* untuk di pelatihan.
5. Pelatihan akan berakhir apabila keadaan / kondisi dari jaringan saraf tiruan sudah memenuhi target.
6. Pelatihan diakhiri dengan menyimpan hasil latihan berupa berat yang didapatkan pada akhir latihan jaringan saraf tiruan.

3.3.3 Algoritma Konversi *Grayscale*

Dari sebuah citra *digital* berwarna, diperlukan citra *digital grayscale* untuk dapat memulai pemrosesan citra *digital*. Secara garis besar, metode yang digunakan untuk melakukan konversi adalah dengan membaca setiap *pixel* dari citra *digital* kemudian menemukan nilai RGB dari *pixel* tersebut. Untuk menemukan nilai *grayscale* dari *pixel* tersebut, digunakan rumus :

$$Grayscale = 0.2989 * Red + 0.5870 * Green + 0.1140 * Blue$$ [35]

Setelah itu *pixel* dengan nilai-nilai *grayscale* yang baru, akan disatukan dan menjadi satu kumpulan *pixel* yang membentuk citra *grayscale*.

[35] Forrest Sheng Bao," A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network" (Texas Tech University, 2007), 2.

```
ALGORITMA Grayscale(nilaiRGB:Array,
lebarCitra:Integer, panjangCitra:Integer)
// Fungsi : Mengkonversi citra digital menjadi grayscale
// Input : nilai RGB dari citra digital, panjang dan lebar dari
citra digital
// Variabel yang digunakan : rgbLvl, grayLvl, l dengan tipe
data integer, grayLevel dengan tipe data array integer
// Output : Citra digital grayscale dengan tipe data array
integer

UNTUK n = 0 sampai l
      rgbLvl = nilaiRGB(n)
      grayLvl = 0.3 * rgbLvl >> 16 &0xFF +  0.59 * rgbLvl >> 8
      &0xFF + 0.11 * rgbLvl & 0xFF
      grayLevel(n) = 0xff000000 | grayLvl<<16 | grayLvl<<8 |
      grayLvl
SELESAI
```

Pseudocode 3.1 Algoritma *Grayscale*

3.3.4 Algoritma *Otsu's Threshold*

Proses selanjutnya adalah mengkonversi citra *grayscale* menjadi citra monokrom untuk dapat menentukan bagian yang manakah yang merupakan *foreground* dan *background*. Algoritma Otsu ini pertama-tama membuat histogram yang isinya adalah *pixel-pixel* pada citra *digital* berdasarkan nilai / tingkat intensitas *grayscale*nya. Hal ini dilakukan dengan cara iterasi terhadap seluruh *pixel* yang berada pada citra *digital*. Dari histogram tersebut, dicari nilai thresholdnya dengan cara menghitung between *class variance* dari masing-masing *threshold* (0-255). Untuk mencari nilai tersebut perlu dilakukan iterasi mulai dari *threshold* 0 sampai 255. Nilai *threshold* yang memiliki angka between *class variance* yang paling besarlah yang akan digunakan. Setelah angka *threshold* tersebut ditemukan, maka akan dilakukan iterasi terhadap seluruh *pixel* pada

gambar, apabila nilai *grayscale* dari *pixel* lebih kecil dari *threshold* maka akan di set sebagai *foreground* dan apabila lebih besar sama dengan *threshold* maka akan di set sebagai *foreground*.

Kumpulan *pixel* yang sudah di set *pixel-pixel*nya sebagai *foreground* dan *background* akan dijadikan sebagai output berupa *array integer* yang terdiri dari warna putih dan hitam.

```
ALGORITMA OtsuThreshold(nilaiGrayscale:Array)
// Fungsi : Mengkonversi citra grayscale menjadi monokrom
// Input : nilai grayscale dari citra digital
// Variabel yang digunakan : histogram dengan tipe data array,
sum dengan tipe data integer, wB dan wF dengan tipe data
integer
// Output : Citra digital hitam putih dengan tipe data array
integer

UNTUK n = 0 sampai panjang nilaiGrayscale
      histogram(nilaiGrayscale(n))++
SELESAI
HITUNG SUM histogram
UNTUK t=0 sampai 256
      wB+=histogram(t)
      JIKA wB=0
            LANJUT
      wF=panjang nilaiGrayscale - wB
      JIKA wF=0
            SELESAI
      sumB += t* histogram(t)
      hitung mean background, foreground, dan class variance
      JIKA variance baru > variance maksimum
            Variance maksimum = variance baru
            Threshold = t
SELESAI
UNTUK n=0 sampai panjang nilaiGrayscale
      JIKA (nilaiGrayscale(n)<t) nilaiGrayscale(n)=foreground
      JIKA (nilaiGrayscale(n)>=t) nilaiGrayscale(n)=background
SELESAI
```

Pseudocode 3.2 Algoritma *Otsu's Threshold*

3.3.5 Algoritma Erosi (Morfologi Matematika)

Proses erosi secara garis besar, adalah mencocokkan keadaan suatu *pixel* dan sekelilingnya dengan sebuah *matrix binary*, apabila

keadaan *pixel* tersebut tidak sesuai dengan *matrix binary* yang dipilih, maka *pixel* tersebut akan dihilangkan (diubah menjadi *background*). Erosi dilakukan terhadap semua *pixel foreground* yang berada pada citra *digital* monokrom. Dan di perangkat lunak yang akan dirancang, digunakan *matrix* 5x5 yang berbentuk lingkaran.

	1	1	1	
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
	1	1	1	

Gambar 3.4 *Structuring Element* yang Digunakan

Cara untuk mencocokkan keadaan *pixel foreground* dengan *structuring element* (*kernel*) adalah dengan melakukan pengecekan satu per satu ke sekeliling *pixel* tersebut. Apabila keadaan dari *pixel* tersebut sama dengan keadaan dari kernel, maka *pixel* tersebut tidak berubah, tetapi apabila tidak sama, maka *pixel* tersebut akan diubah menjadi *pixel background*. *Pseudocode* 3.3 menjelaskan mengenai bagaimana urutan kerja dari algoritma erosi ini.

```
ALGORITMA Erosi(binaryMap:Array)
// Fungsi : Melakukan operasi erosi terhadap citra digital
monokrom
// Input : nilai binary dari citra digital monokrom
// Variabel yang digunakan : StructuringElement dengan tipe
data array dua dimensi
// Output : Citra digital monokrom dengan tipe data array
UNTUK n = 0 sampai panjang binaryMap
      JIKA (binaryMap(n) != StructuringElement
      binaryMap(n) = background
SELESAI
```

Pseudocode 3.3 Algoritma Erosi

3.3.6 Algoritma Operasi Sobel

Operasi Sobel bertujuan untuk mencari garis batas dari suatu citra *digital*, metode yang digunakan adalah dengan mencari *gradient*. *Gradient* didapatkan dari menghitung perbedaan warna / intensitas dari suatu *pixel* dengan *pixel* yang berada di sekitarnya. Cara kerja dari operasi sobel ini adalah dengan melakukan iterasi ke semua *pixel* pada gambar, kemudian menghitung nilai *gradient* x dan y. Dari *gradient* x dan y yang didapatkan, dicari nilai *gradient* totalnya. Lalu dicari nilai *gradient* maksimal dan nilai rasio yang nantinya akan digunakan untuk mencari nilai intensitas *grayscale* dari masing-masing *pixel*. Output dari algoritma ini adalah kumpulan *pixel* dalam bentuk *array* satu dimensi yang berisi nilai *grayscale*nya dibagi dengan nilai rasio. *Pseudocode* 3.4 akan menjelaskan mengenai cara kerja dari algoritma operasi sobel ini.

UNIVERSITAS CIPUTRA

```
ALGORITMA OperasiSobel(citraGrayscale:Array)
// Fungsi : Mencari gradient / garis batas dari citraGrayscale
// Input : nilai citraGrayscale dengan tipe array
// Variabel yang digunakan : sobel_x dan sobel_y dengan tipe
data array 2 dimensi; max,rasio, dan Temp dengan tipe data
integer; Total dengan tipe data array
// Output : Citra digital monokrom dengan tipe data array

sobel_x = [[-1,0,1], [-2,0,2], [-1,0,1]]
sobel_y = [[-1,-2,-1], [0,0,0], [1,2,1]]

UNTUK n=1 sampai panjang citraGrayscale
      Menghitung nilai pixel_x dengan sobel_x
      Menghitung nilai pixel_y dengan sobel_y
      Total(n)= pixel_x*pixel_x + pixel_y*pixel_y
      JIKA max < Total(n)
            MAKA max = Total(n)
SELESAI

Rasio = max / 255

UNTUK n=1 sampai panjang citraGrayscale
      Temp=Total(n) / rasio
      Total(n)=0xff00000| Temp<<16 | Temp<<8 | Temp
SELESAI
```

Pseudocode 3.4 Algoritma Operasi Sobel

3.3.7 Algoritma *Connected-component Labeling*

Algoritma *connected-component labeling* bertujuan untuk menghitung segmen – segmen dari citra *digital* monokrom yang saling berhubungan. Algoritma ini menerima input berupa *array* dari citra *digital* dan menghasilkan output berupa pemetaan yang berisikan *integer*. Dalam algoritma ini, segmen yang memiliki luas paling besar akan dianggap sebagai segmentasi dari daun, sedangkan segmen-segmen lainnya akan dianggap sebagai *background*. Cara kerja *connected-component labeling* yang digunakan dalam perangkat lunak ini terdiri dari 2 bagian, yaitu bagian pertama dan bagian kedua. Pada bagian pertama, setiap *pixel* dalam citra *digital* yang saling

berhubungan akan diberi nomor yang identik. Pencarian dilakukan secara iterasi dari bagian kiri atas ke bagian kanan bawah untuk mencari keberadaan *pixel* yang sudah berlabel / belum berlabel yang berada di 8-*pixel* di sekeliling *pixel* yang dicari. Dan kemudian pada bagian kedua, nomor-nomor yang sudah diberikan pada bagian pertama akan disamakan apabila saling berhubungan dengan cara melakukan pengecekan terhadap tabel yang sudah dibuat sebelumnya. Setelah kedua proses tersebut selesai, maka dihitunglah luas setiap segmentasi, dan segmentasi yang paling luas akan dianggap menjadi satu-satunya *foreground* pada gambar, sedangkan segmentasi lainnya akan dianggap sebagai *background* / objek lain selain daun itu sendiri.

Output dari fungsi ini sendiri berupa *array* yang terdiri dari penomoran yang sudah diberikan sebelumnya dan akan digunakan lagi ketika melakukan proses segmentasi *watershed*.

3.3.8 Algoritma Segmentasi *Watershed*

Algoritma segmentasi *watershed* berfungsi untuk melakukan *watershed segmentation* terhadap citra *gradient* hasil operasi sobel dengan citra penanda dari hasil *connected-component labeling*. Cara kerja dari algoritma ini adalah dengan melakukan iterasi ke semua *pixel* dalam gambar *marker* yang merupakan *background*, lalu melakukan pengecekan apakah ada *pixel foreground* di *pixel* sekelilingnya. Apabila ada, maka masukkan *pixel* tersebut ke dalam *priority queue* berdasarkan nilai dari *grayscale*nya. Lalu dari *priority*

queue tersebut setiap *pixel* dikeluarkan berdasarkan nilai *grayscale*nya yang paling kecil. Apabila *pixel* yang dikeluarkan dari *queue* memiliki hanya satu label, maka *pixel* tersebut pun diberikan label yang sama. Tetapi apabila label tersebut memiliki lebih dari satu label, maka *pixel* tersebut akan dijadikan *pixel* batas dari *watershed*. Pada *pseudocode* 3.6 akan dijelaskan bagaimana urutan algoritma dari segmentasi *watershed* ini.

```
ALGORITMA ConnectCompLabeling(citraMonokrom:Array)
// Fungsi : Mencari segmen dari citra digital yang paling luas
// Input : nilai citraMonokrom dengan tipe data array
// Variabel yang digunakan : binaryMap dengan tipe data array,
table dengan tipe data array.
// Output : Citra digital monokrom dengan tipe data array

UNTUK n=1 sampai panjang citraMonokrom
      JIKA citraMonokrom(n) = 1
            JIKA (8Connectivity binaryMap(n) !=0 & hanya satu)
                  binaryMap(n) = 8Connectivity binaryMap(n)
            JIKA (8Connectivity BinaryMap(n) =0)
                  binaryMap(n) = segmen++
            JIKA (8Connectivity binaryMap(n) !=0 & tidak sama)
                  binaryMap(n) = MIN(8Connectivity binaryMap)
                  masukkan 8Connectivity binaryMap ke table
                  sebagai parent dan child
SELESAI
UNTUK n=1 sampai panjang citraMonokrom
      JIKA binaryMap(n) != 0
            JIKA binaryMap(n) = child dari parent di table
                  binaryMap(n)= parent
                  hitung luas
SELESAI
UNTUK n=1 sampai jumlah segmen
      segmenDaun = MAX(luas segmen)
      segmenMax = n
SELESAI
UNTUK n=1 sampai panjang citraMonokrom
      JIKA binaryMap(n) = segmenMax
            binaryMap(n)=1
      JIKA TIDAK
            binaryMap(n)=2
SELESAI
```

Pseudocode 3.5 Algoritma *Connected-component Labeling*

```
ALGORITMA watershedSegmen(citraGradient:Array,
citraMonokrom:Array)
// Fungsi : Mencari hasil segmentasi akhir dari suatu citra
gradient.
// Input : nilai citraGradient dan citraMonokrom dengan tipe
data array.
// Variabel yang digunakan : pixelQueue dengan tipe data
PriorityQueue, citraHasil dengan tipe data Array.
// Output : Citra digital yang sudah tersegmentasi dengan tipe
data array.

UNTUK n=1 sampai panjang citraMonokrom
      JIKA citraMonokrom(n) = 0 && ada pixel 1 di 8Connect-nya
            Masukkan citraGradient(n) ke pixelQueue
SELESAI

UNTUK n=1 sampai panjang citraMonokrom
      JIKA citraMonokrom(n) = 1
            citraHasil(n) = 1
SELESAI

SELAMA !pixelQueue.isEmpty()
      Pixel = pixelQueue.remove()
      JIKA 8Connect Pixel = 1 || 2
            citraHasil(Pixel.get(n)) = 1 || 2
      JIKA 8Connect Pixel = 1 && 2
            citraHasil(Pixel.get(n)) = 3
      JIKA 8Connect Pixel = 0
            Masukkan ke pixelQueue
SELESAI
```

Pseudocode 3.6 Algoritma Segmentasi *Watershed*

3.3.9 Algoritma Menutup Lubang pada Segmentasi Daun

Untuk menutup dan mengisi lubang pada segmentasi daun, sebelumnya harus ditemukan dulu posisi dari lubang tersebut. Metode yang digunakan untuk menemukan posisi dari lubang pada daun adalah dengan menggunakan metode *connected-component labeling* seperti yang digunakan ketika membuat citra penanda untuk proses segmentasi *watershed*. Hanya saja perbedaannya adalah yang dicari

adalah *pixel background* dan luas segmentasi yang paling besar diacuhkan. Berikut adalah *pseudocode* dari algoritma tersebut.

```
ALGORITMA FillHoles(citraMonokrom:Array)
// Fungsi : Mencari dan mengisi lubang yang ditemukan pada
citra monokrom
// Input : nilai citraMonokrom dengan tipe data array
// Variabel yang digunakan : binaryMap dengan tipe data array,
table dengan tipe data array
// Output : Citra digital monokrom dengan tipe data array

UNTUK n=1 sampai panjang citraMonokrom
      JIKA citraMonokrom(n) = 0
            JIKA (8Connectivity binaryMap(n) !=0 & hanya satu)
                  binaryMap(n) = 8Connectivity binaryMap(n)

            JIKA (8Connectivity BinaryMap(n) =0)
                  binaryMap(n) = segmen++

            JIKA (8Connectivity binaryMap(n) !=0 & tidak sama)
                  binaryMap(n) = MIN(8Connectivity binaryMap)
                  masukkan 8Connectivity binaryMap ke table
                  sebagai parent dan child
SELESAI

UNTUK n=1 sampai panjang citraMonokrom
      JIKA binaryMap(n) != 0
            JIKA binaryMap(n) = child dari parent di table
                  binaryMap(n)= parent
                  hitung luas
SELESAI

UNTUK n=1 sampai jumlah segmen
      segmenDaun = MAX(luas segmen)
      segmenMax = n
SELESAI

UNTUK n=1 sampai panjang citraMonokrom
      JIKA binaryMap(n) != segmenMax
            ISI dengan pixel foreground
SELESAI
```

Pseudocode 3.7 Algoritma Mengisi Lubang pada Segmentasi Daun

3.3.10 Algoritma Pembuatan *Outline*

Algoritma pembuatan *outline* ini bertujuan untuk membuat garis dari ROI (*Region of Interest*) yang didapat dari hasil segmentasi

watershed. Cara kerjanya adalah dengan membuat garis *pixel* dan menghilangkan titik dari garis yang tidak dibutuhkan sehingga didapatkan garis luar dengan ketebalan 1 *pixel*. Algoritma ini akan melakukan iterasi pertama kali untuk mencari *pixel background* yang memiliki *pixel foreground* di sekelilingnya, kemudian iterasi beberapa kali untuk mengecek keadaan suatu *pixel* yang menjadi keliling dari segmentasi daun, dan menghilangkan *pixel* yang tidak dibutuhkan.

```
ALGORITMA outline(citraROI:Array)
// Fungsi : Mencari garis luar dari ROI suatu citra
// Input : citraROI dengan tipe data Array
// Variabel yang digunakan : imageOutline dengan tipe data
array, PROI dengan tipe data integer, ROI dengan tipe data
integer.
// Output : Jumlah Perimeter ROI dengan tipe data integer,
imageOutline dengan tipe data array, ROI dengan tipe data
integer.

imageOutline = citraROI

UNTUK n = 1 sampai panjang citraROI
      JIKA citraROI(n)=2 && 8Connect citraROI(n)=1
            imageOutline(n)=2
            PROI++
      JIKA citraROI(n)=1
            imageOutline(n)=1
            ROI++
SELESAI
UNTUK n = 1 sampai panjang imageOutline
      JIKA imageOutline(n)=1
            JIKA imageOutline(n).get8ConnectNeighbour < 2
                  imageOutline(n)=1
                  PROI—
                  ROI++
SELESAI
UNTUK n = 1 sampai panjang imageOutline
      JIKA imageOutline(n)=1
            JIKA imageOutline(n).get4ConnectNeighbour > 1
                  imageOutline(n)=1
                  PROI—
                  ROI++
SELESAI
```

Pseudocode 3.8 Algoritma Pembuatan *Outline*

3.3.11 Algoritma *Feature Extraction*

Algoritma ini berfungsi sebagai penghitung fitur dari citra *outline* dan parameter – paremeter yang sudah didapatkan sebelumnya. Algoritma ini akan mengambil input berupa *array* dari citra *digital* yang sudah diproses dan hasil output fitur-fitur tersebut dihitung menjadi output berupa *array* dengan tipe data *double* untuk memperoleh hasil yang akurat.

CREATING WORLD CLASS ENTREPRENEURS
UC
UNIVERSITAS CIPUTRA

```
ALGORITMA featureExtract(imageOutline:Array,
ROI:Integer, PROI:Integer, Lebar:Integer,
Panjang:Integer)
// Fungsi : Mencari dan menghitung fitur-fitur dari input
// Input : imageOutline dengan tipe data array;
// Variabel yang digunakan : MinX,Max,MinY,MaxY,FitX,dan FitY
dengan tipe data Integer; convexHull, PconvexHull, RI, RC,
UR,SR,EA, dan EB dengan tipe data Double; RMI dan ACH dengan
tipe data array double.
// Output : Fitur dengan tipe data array double.

MinX,MaxX,MinY,MaxY,FitX,FitY
Fitur
UNTUK n=1 sampai panjang imageOutline
      JIKA imageOutline(n)=1
            MinX = MIN(MinX, x);
            MaxX = MAX(MaxX, x);
            MinY = MIN(MinY, y);
            MaxY = MAX(MaxY, y);
SELESAI

FitX = MaxX-MinX
FitY = MaxY-MinY
Fitur(0)=FitY/FitX
Fitur(1)=ROI/(FitX * FitY)
Fitur(2)=ROI/convexHull
Fitur(3)=PROI/PconvexHull
Fitur(4)=RI/RC
Fitur(5)=UR/SR
Fitur(6)=EA/EB
Fitur(7)= 4 * pi * ROI / PROI.
Fitur(8)=RMI(0)
Fitur(9)=RMI(1)
Fitur(10)=RMI(2)
Fitur(11)=RMI(3)
Fitur(12)=ACH(0)
Fitur(13)=ACH(1)
Fitur(14)=ACH(2)
Fitur(15)=ACH(3)
Fitur(16)=ACH(4)
```

Pseudocode 3.9 Algoritma *Feature Extraction*

3.3.12 Algoritma Jaringan Saraf Tiruan (*Backpropagation*)

Algoritma jaringan saraf tiruan berfungsi sebagai metode latihan bagi sistem untuk dapat mengenali fitur-fitur dari daun. Input dari algoritma ini berupa kumpulan fitur-fitur dan akan menghasilkan *output* berupa *weight* hasil latihan. Algoritma ini merupakan penerapan

dari dasar teori tentang jaringan saraf tiruan yang sudah dibahas pada sub bab 2.

CREATING WORLD CLASS
UC
UNIVERSITAS
CIPUTRA
ENTREPRENEURS

```
ALGORITMA JST(Fitur:array, JumlahInputNode:Integer,
JumlahOutputNode:Integer, JumlahHiddenNode:Integer)
// Fungsi : Melatih JST dan mencari weight dari hasil latihan
// Input : Kumpulan Fitur dengan tipe data array,
JumlahInputNode, JumlahOutputNode, dan JumlahHiddenNode dengan
tipe data Integer.
// Variabel yang digunakan : InputNodes, HiddenNodes,
OutputNodes
// Output : Weight yang disimpan kedalam file xml

UNTUK n = 1 sampai JumlahInputNode
      UNTUK m = 1 sampai JumlahHiddenNode
            ACAK WEIGHT(0.5 - -0.5)
SELESAI
UNTUK n = 1 sampai JumlahHiddenNode
      UNTUK m = 1 sampai JumlahOutputNode
            ACAK WEIGHT(0.5 - -0.5)
SELESAI

SELAMA ERROR>0.01
      UNTUK n = 1 sampai Fitur.size()
            UNTUK m = 1 sampai JumlahInputNode
                  InputNode(m).input=Fitur(n).get(m)
                  UNTUK o = 1 sampai JumlahHiddenNode
                        InputNode(m).afterWeight(o) =
                        InputNode(m).input *
                        InputNode(m).weight(o)
                  SELESAI
            SELESAI
            UNTUK m = 1 sampai JumlahHiddenNode
                  TOTAL Input ke HiddenNode(m)
                  HiddenNode(m).output =
                  Sigmoid(HiddenNode(m).input)
                  UNTUK o = 1 sampai JumlahOutputNode
                        HiddenNode(m).afterWeight(o) =
                        HiddenNode(m).output *
                        HiddenNode(m).weight(o)
                  SELESAI
            SELESAI
            UNTUK m = 1 sampai JumlahOutputNode
                  TOTAL Input ke OutputNode(m)
                  OutputNode(m).output =
                  Sigmoid(OutputNode(m).input)
                  Hitung error dari OutputNode(m).output
            SELESAI
            UNTUK m = 1 sampai JumlahHiddenNode
                  UNTUK o = 1 sampai JumlahOutoutNode
                        HiddenNodes(m).delta(o) = LR *
                        OutputNodes(o).error *
                        HiddenNodes(m).output
                  SELESAI
            SELESAI
```

Pseudocode 3.10 Algoritma Jaringan Saraf Tiruan (*Backpropagation*)

```
            UNTUK m = 1 sampai JumlahHiddenNodes
                  UNTUK o = 1 sampai JumlahOutputNodes
                        TotalError +=outputNodes(o).Error *
                        HiddenNodes(m).weight(o)
                  SELESAI
                  hiddenNodes(m).Error = totalError
                  Hitung error HiddenNodes(m) dengan turunan
                  sigmoid
            SELESAI
            UNTUK m = 1 sampai JumlahInputNode
                  UNTUK o = 1 sampai JumlahHiddenNode
                        InputNodes(m).delta(o) = LR *
                       HiddenNodes(o).error *
                       InputNodes(m).output
            SELESAI
            HITUNG WEIGHT BARU DENGAN DELTA
      SELESAI
SELESAI

SIMPAN KEADAAN WEIGHT TERAKHIR
```

Lanjutan *Pseudocode* 3.10 Algoritma Jaringan Saraf Tiruan (*Backpropagation*)

3.3.13 Algoritma Jaringan Saraf Tiruan (*Feedforward*)

Algoritma jaringan saraf tiruan *feedforward* berfungsi sebagai metode yang digunakan untuk melakukan pengenalan terhadap suatu pola fitur morfologi daun berdasarkan bobot yang sudah dilatih sebelumnya menggunakan *backpropagation*.

```
ALGORITMA JST(XMLBobot:file, Fitur:array,
JumlahInputNode:Integer, JumlahOutputNode:Integer,
JumlahHiddenNode:Integer)
// Fungsi : Melatih JST dan mencari weight dari hasil latihan
// Input : File XML yang berisikan bobot, Kumpulan Fitur dengan
tipe data array, JumlahInputNode, JumlahOutputNode, dan
JumlahHiddenNode dengan tipe data Integer.
// Variabel yang digunakan : InputNodes, HiddenNodes,
OutputNodes
// Output : result dengan tipe array

UNTUK n = 1 sampai JumlahInputNode
      UNTUK m = 1 sampai JumlahHiddenNode
            AMBIL Bobot dari XMLBobot
SELESAI
UNTUK n = 1 sampai JumlahHiddenNode
      UNTUK m = 1 sampai JumlahOutputNode
            AMBIL Bobot dari XMLBobot
SELESAI

UNTUK n = 1 sampai Fitur.size()
      UNTUK m = 1 sampai JumlahInputNode
            InputNode(m).input=Fitur(n).get(m)
            UNTUK o = 1 sampai JumlahHiddenNode
                  InputNode(m).afterWeight(o) =
                  InputNode(m).input * InputNode(m).weight(o)
            SELESAI
      SELESAI
      UNTUK m = 1 sampai JumlahHiddenNode
            TOTAL Input ke HiddenNode(m)
            HiddenNode(m).output = Sigmoid(HiddenNode(m).input)
            UNTUK o = 1 sampai JumlahOutputNode
                  HiddenNode(m).afterWeight(o) =
                  HiddenNode(m).output *
                  HiddenNode(m).weight(o)
            SELESAI
      SELESAI
      UNTUK m = 1 sampai JumlahOutputNode
            TOTAL Input ke OutputNode(m)
            OutputNode(m).output = Sigmoid(OutputNode(m).input)
            Result(m)= error dari OutputNode(m).output
      SELESAI
```

Pseudocode 3.11 Algoritma Jaringan Saraf Tiruan (*Feedforward*)

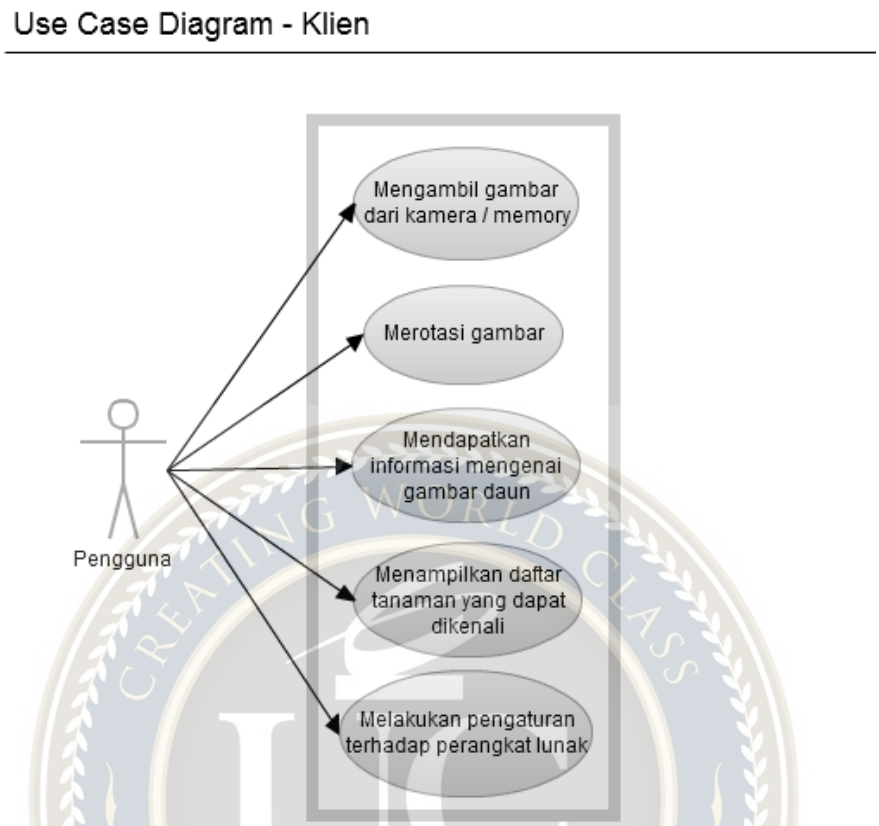
3.4 *Use Case Diagram*

Sub bab *use case diagram* berisikan tentang *use case* yang ada pada perangkat lunak yang dirancang. *Use case diagram* dari perangkat lunak ini terbagi menjadi 2 bagian, yaitu bagian klien dan *server*.

3.4.1 *Use Case Diagram* Klien

Use case ini adalah *use case* yang berisikan tentang aktivitas yang dapat dilakukan oleh pengguna dalam perangkat lunak yang dibuat pada perangkat klien (Android). Pengguna dapat mengambil gambar daun yang ingin dikenali dari kamera / memori (SD *Card*). Selain itu pengguna dapat langsung melakukan rotasi terhadap gambar dalam perangkat lunak. Pengguna juga dapat menggunakan fitur-fitur tambahan lainnya seperti melakukan pengaturan dan melihat daftar tanaman yang dapat dikenali.

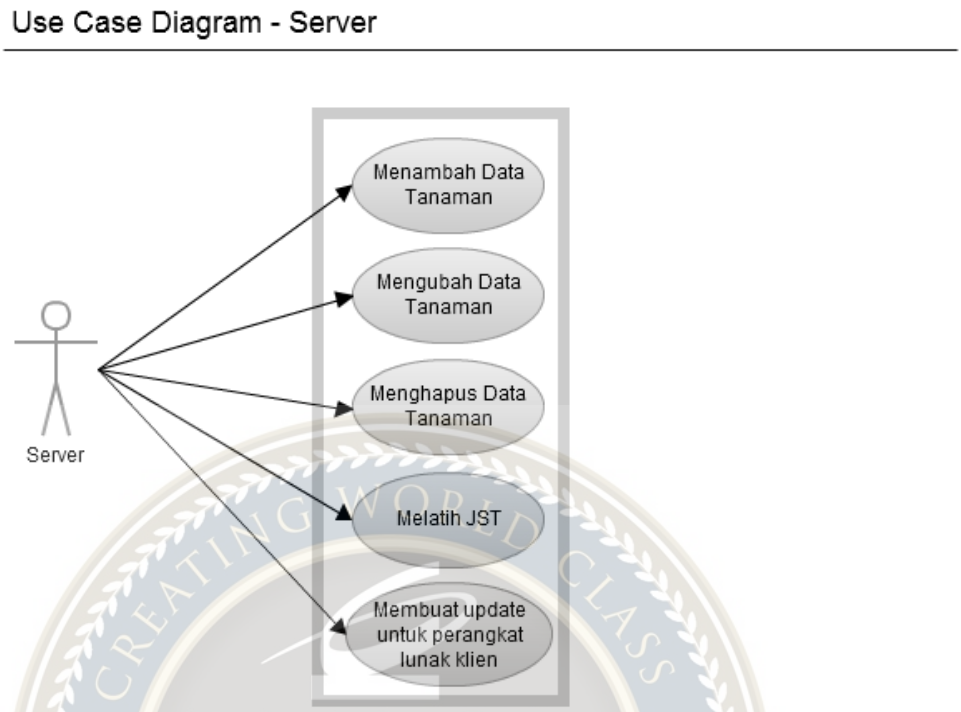
UNIVERSITAS CIPUTRA



Gambar 3.5 *Use Case Diagram* Klien

3.4.2 *Use Case Diagram Server*

Use case ini adalah *use case* yang berisikan tentang aktivitas yang dapat dilakukan oleh *server* dalam sistem perangkat lunak yang dibuat untuk *server*. Pengguna *server* dapat menambah, mengubah, dan juga menghapus data tanaman. Selain itu juga dapat melatih JST dan memberikan *update* baru untuk perangkat lunak klien.

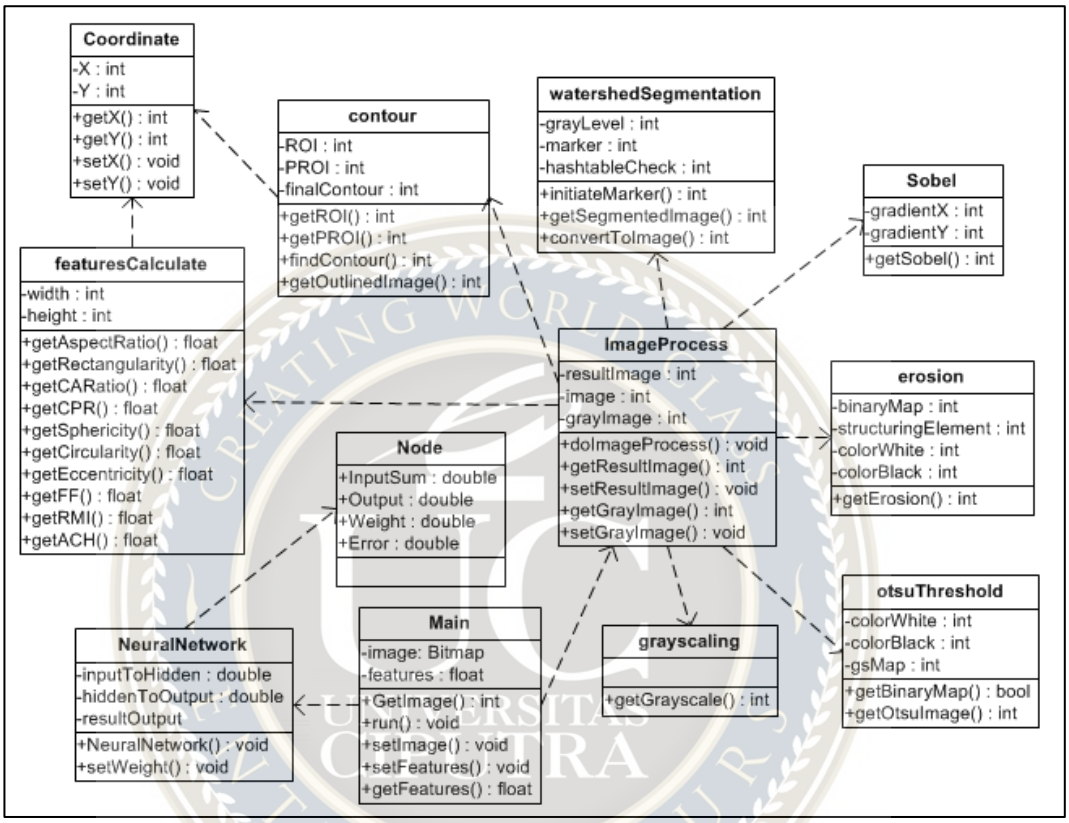


Gambar 3.6 *Use Case Diagram Server*

3.5 *Class Diagram*

Pada sub bab ini akan ditunjukkan rancangan dari *class diagram* secara umum dari proses pengolahan citra *digital* dan jaringan saraf tiruan yang ada dalam perangkat lunak ini. *Class diagram* ini nantinya akan digunakan dan disesuaikan lagi dengan penerapannya pada perangkat lunak klien dan *server*. Gambar 3.7 adalah gambar *class diagram* yang dirancang. Dari gambar tersebut, proses diawali dengan kelas main yang memanggil kelas ImageProcess, kemudian ImageProcess memanggil beberapa kelas lainnya yaitu grayscaling, erosion, sobel, otsuThreshold, waterSegmentation, contour dan featureCalculate. Semua kelas yang dipanggil oleh ImageProcess, mengambil objek gambar dari kelas ImageProcess untuk di proses sesuai kebutuhan. Kelas contour dan featuresCalculate apabila dipanggil akan menggunakan kelas coordinate untuk

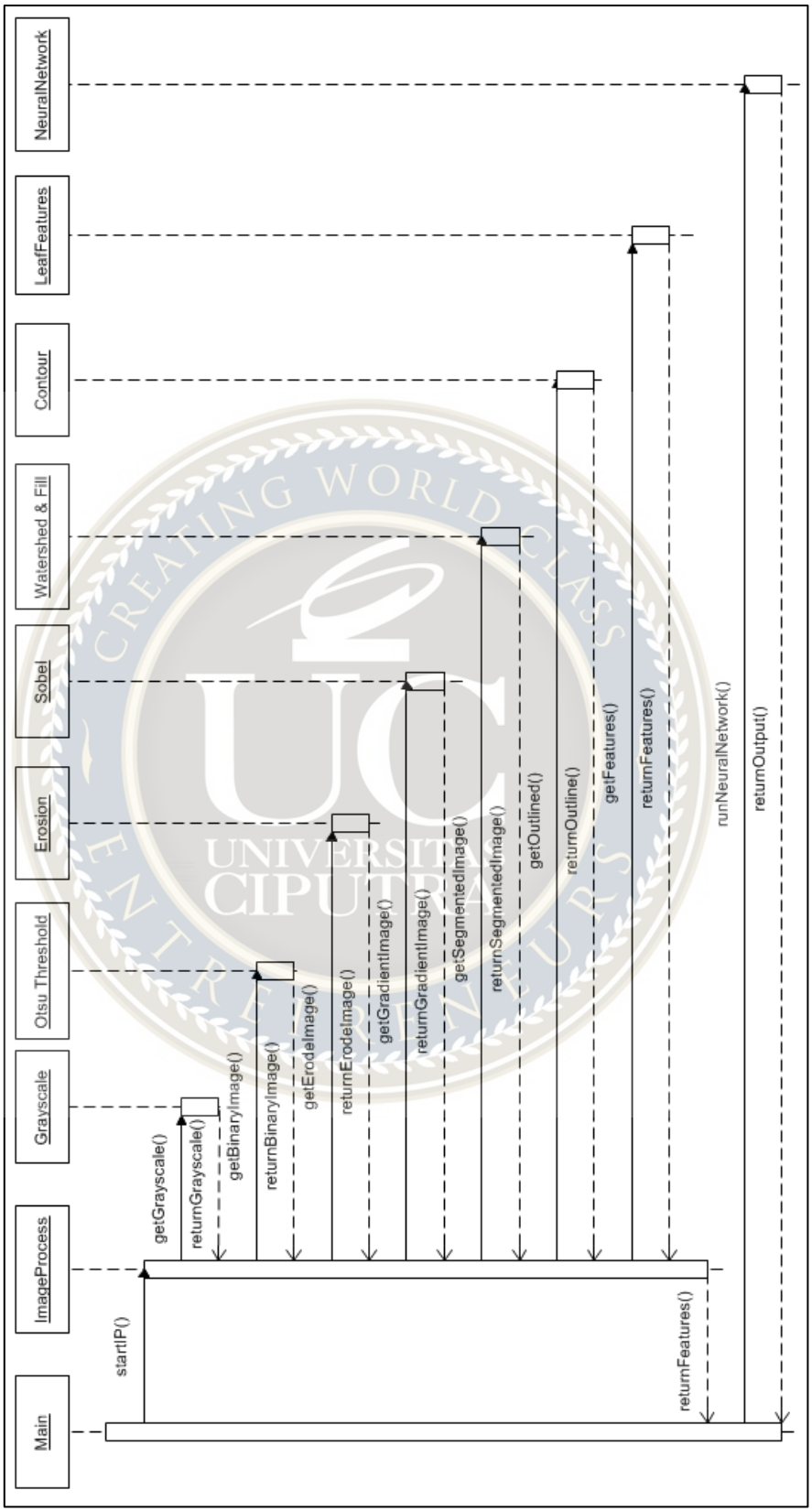
menyimpan posisi titik. Kelas main setelah memanggil kelas ImageProcess, memanggil kelas NeuralNetwork yang memiliki beberapa objek kelas *Node* untuk melakukan proses jaringan saraf tiruan.



Gambar 3.7 *Class Diagram*

3.6 *Sequence Diagram* (Pengolahan Citra *Digital* – JST)

Pada sub bab ini akan ditunjukkan rancangan dari *sequence diagram* yang dirancang untuk perangkat lunak klien dan *server*. *Sequence diagram* yang dirancang secara umum berfungsi untuk proses pengolahan citra *digital* dan jaringan saraf tiruan. Dimulai dari main yang memanggil proses pengolahan citra *digital*, kemudian setelah proses tersebut selesai, dilanjutkan dengan proses jaringan saraf tiruan. Gambar 3.8 adalah gambar *sequence diagram* yang dirancang.



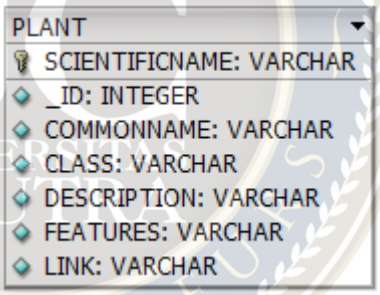
Gambar 3.8 *Sequence Diagram*

3.7 *Database*

Sebelum membuat dan mengisi *database*, ditentukan terlebih dahulu data apa saja yang akan dimasukkan ke dalam *database*. Setelah menemukan data yang akan dimasukkan ke *database*, barulah kemudian *database* dari perangkat lunak ini akan dibuat.

3.7.1 *Physical Design Model*

Database yang akan dirancang hanyalah terdiri dari satu tabel, sehingga tidak ada relasi antar tabel dalam desain *database* untuk perangkat lunak ini. Gambar 3.9 Menggambarkan *Physical Design Model* yang digunakan dalam perangkat lunak ini.



Gambar 3.9 *Physical Design Model*

3.7.2 Struktur Tabel

Struktur tabel digunakan untuk menjelaskan struktur dari tabel yang digunakan dalam perangkat lunak ini. Satu-satunya tabel yang digunakan adalah tabel PLANT.

Tabel PLANT ini berfungsi sebagai tabel penyimpan data tanaman yang dapat dikenali oleh perangkat lunak ini. Tabel ini berisi

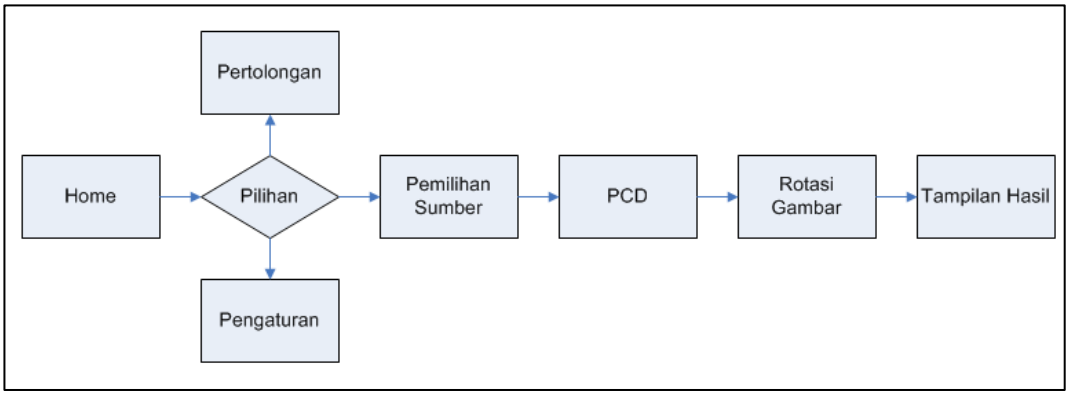
data nama biasa, nama latin, kelas, deskripsi, fitur, dan informasi tambahan dari tanaman.

No	Nama Kolom	Tipe	*Constraint*	Keterangan
1	SCIENTIFICNAME	VARCHAR	PRIMARY KEY, NOT NULL	Nama latin dari spesies tanaman tersebut
2.	_ID	INTEGER	NOT NULL, AUTO INCREMENT	Kode dari spesies tanaman yang dapat dikenali
3.	COMMONNAME	VARCHAR	NULL	Nama umum dari spesies tanaman tersebut
4.	CLASS	VARCHAR	NULL	Kelas dari spesies tanaman tersebut.
5.	DESCRIPTION	VARCHAR	NULL	Deskripsi / informasi mengenai spesies tanaman
6.	FEATURES	VARCHAR	NULL	Fitur / ciri-ciri dari spesies tanaman tersebut
7.	LINK	VARCHAR	NULL	Informasi tambahan dari spesies tanaman tersebut

Tabel 3.2. Tabel PLANT

3.8 Antarmuka Pengguna *(User Interface)*

Ketika akan membuat suatu *user interface*, sebelumnya diperlukan desain gambaran *low-fidelity* dari *user interface*. Dalam perancangan *user interface* sendiri dilakukan beberapa pertimbangan, salah satunya adalah agar *user interface* dapat digunakan dengan baik oleh pengguna atau *user-friendly*. Desain yang ditampilkan dalam bentuk *low-fidelity* ini masih dapat berubah lagi ketika proses pembuatan perangkat lunak, karena perlunya menyesuaikan desain yang sudah dirancang dengan keterbatasan ketika melakukan pembuatan perangkat lunak. Berikut adalah pilihan halaman dari *user interface* dari perangkat lunak yang akan dirancang.



Gambar 3.10 Langkah-langkah *User Interface*

Dari menu halaman utama, user dapat memilih 3 pilihan halaman berikutnya, yaitu halaman pertolongan, halaman pemilihan sumber gambar, dan halaman pengaturan. Dari halaman pemilihan sumber gambar, *user* dapat melanjutkannya ke halaman pemrosesan citra *digital*, kemudian ke halaman untuk merotasi gambar dan diakhiri dengan halaman tampilan hasil. Pada sub-bab dibawah akan dijelaskan desain setiap halaman pada perangkat lunak ini.

3.8.1 Desain Tampilan Awal (*Low – Fidelity*)

Berikut adalah gambar dari desain *low-fidelity* tampilan awal ketika program pertama kali dijalankan.

3.8.2 Desain UI Halaman Pemilihan Sumber Gambar (*Low – Fidelity*)

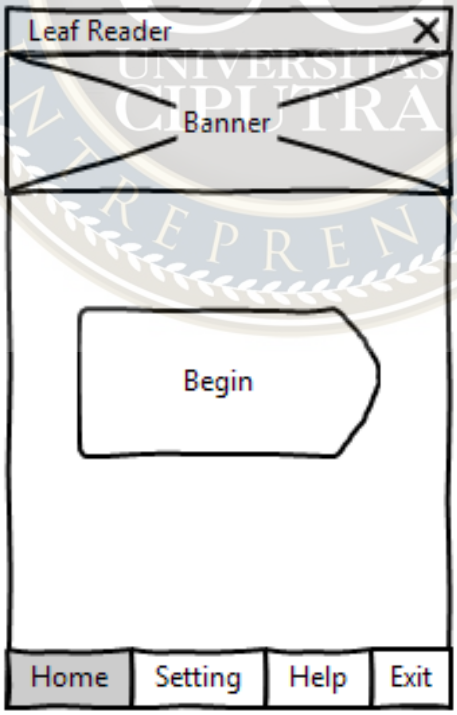
Ketika user menekan tombol “Begin” pada halaman awal, maka *user* akan dibawa ke halaman berikutnya yang berisi tentang menu pilihan sumber dari gambar yang akan diproses. Di bagian atas dari halaman, diberi penanda keberadaan dari pengguna.

3.8.3 Desain UI Halaman Proses Gambar (*Low – Fidelity*)

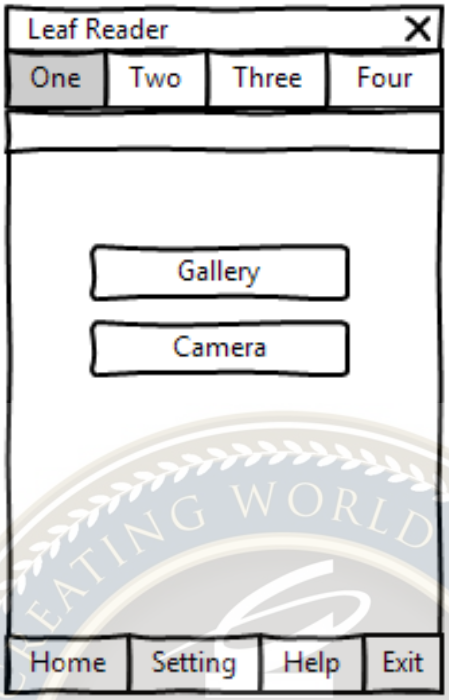
Setelah *user* memilih sumber gambar, maka gambar akan ditampilkan di halaman berikutnya dimana *user* juga akan dapat melihat progres dari pemrosesan citra *digital* yang dilakukan oleh perangkat lunak di halaman ini.

3.8.4 Desain UI Halaman Rotasi Gambar (*Low – Fidelity*)

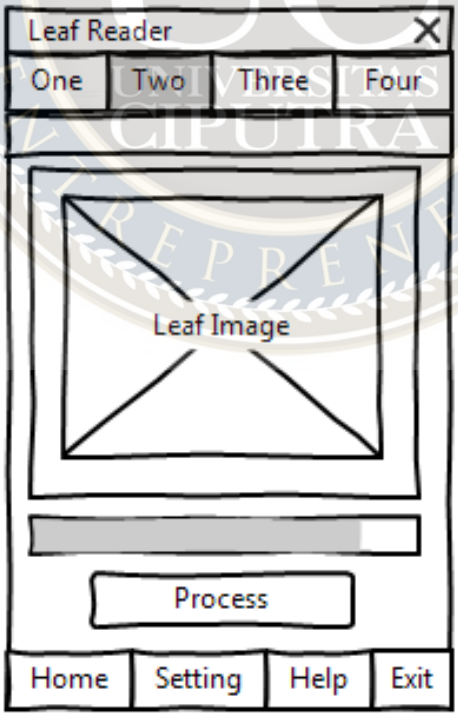
Halaman berikutnya adalah halaman dimana *user* akan diminta untuk merotasi gambarnya yang sudah melewati proses oleh perangkat lunak. Di halaman ini juga akan terdapat tombol untuk melakukan pengenalan terhadap gambar yang sudah di proses dan di rotasi oleh pengguna.



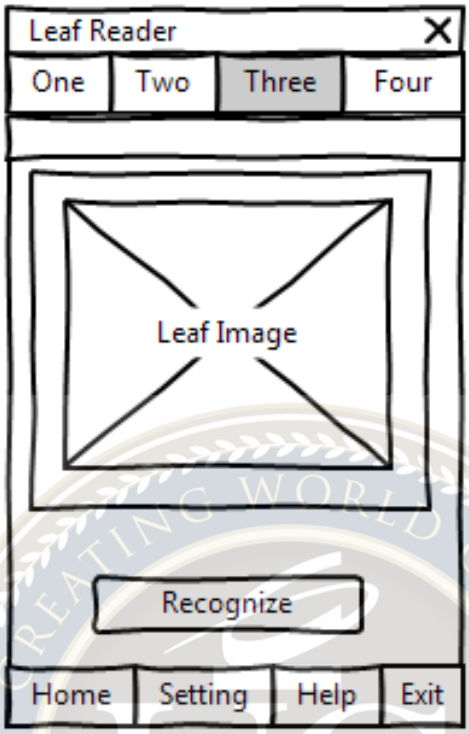
Gambar 3.11 UI Home (*Low – Fidelity*)



Gambar 3.12 UI Menu Sumber (*Low – Fidelity*)



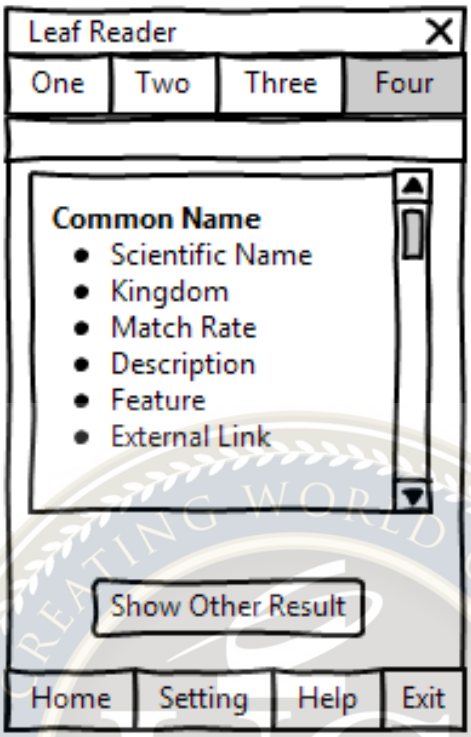
Gambar 3.13 UI PCD (*Low – Fidelity*)



Gambar 3.14 UI Rotasi Gambar (*Low – Fidelity*)

3.8.5 Desain UI Halaman Tampilan Hasil (*Low – Fidelity*)

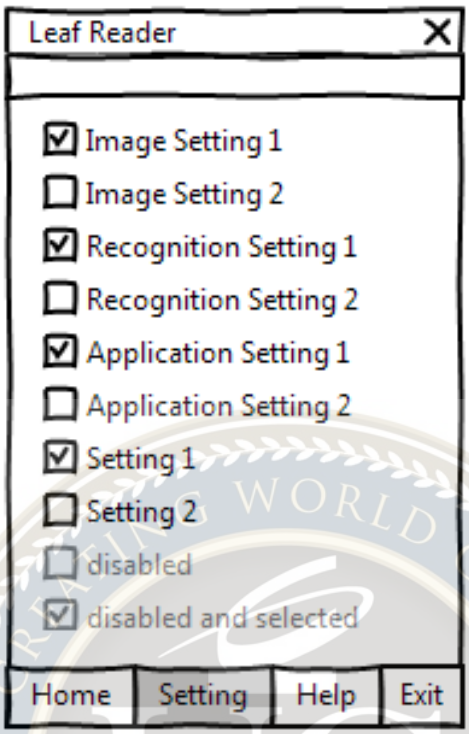
Halaman ini merupakan halaman terakhir dari seluruh proses pengenalan bentuk daun yang dilakukan oleh perangkat lunak. Di halaman ini, akan ditampilkan hasil output dari proses pengenalan bentuk daun. Berikut Gambar 3.5 yang menunjukkan bentuk / desain dari halaman tampilan hasil secara sekilas.



Gambar 3.15 UI Tampilan Hasil (*Low – Fidelity*)

3.8.6 Desain UI Halaman Pengaturan (*Low – Fidelity*)

Halaman ini merupakan halaman yang dapat diakses oleh *user* menggunakan menu dibagian bawah. Halaman ini berisikan tentang pengaturan-pengaturan yang dapat dilakukan oleh pengguna sesuai preferensinya mengenai perangkat lunak sistem pengenalan bentuk daun ini.



Gambar 3.16 UI Tampilan Pengaturan (*Low – Fidelity*)

3.8.7 Desain UI Halaman Pertolongan (*Low – Fidelity*)

Halaman ini merupakan halaman yang dapat diakses oleh *user* menggunakan menu dibagian bawah. Dan di halaman ini pengguna dapat membaca mengenai beberapa hal, salah satunya adalah mengenai pertolongan tentang cara penggunaan perangkat lunak ini.

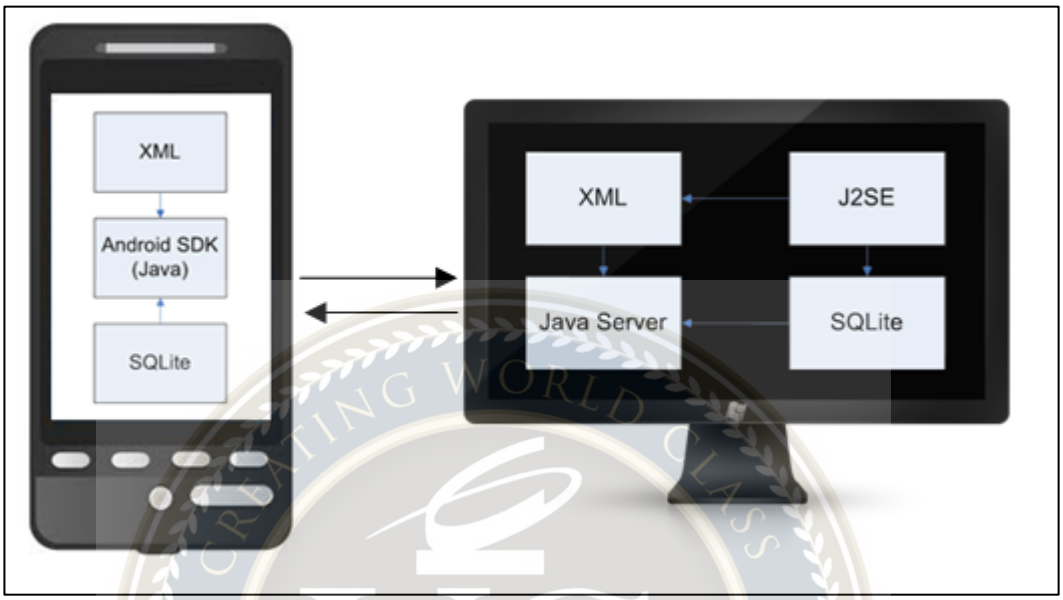
Gambar 3.17 UI Tampilan Pertolongan (*Low – Fidelity*)

3.9 Sistem Arsitektur

Perangkat lunak sistem pengenalan bentuk daun diawali dengan proses melatih jaringan saraf tiruan yang dilakukan pada suatu perangkat komputer menggunakan bahasa pemrograman J2SE kemudian proses pengenalan daunnya akan dilakukan menggunakan bahasa pemrograman Java (Android SDK).

Untuk dapat melakukan proses latihan jaringan saraf tiruan, diperlukan data *training* set terlebih dahulu. *Training* set disini berupa citra *digital* dari daun yang sudah dipersiapkan sebelumnya. Kemudian dari hasil latihan jaringan saraf tiruan tersebut, akan disimpan ke dalam bentuk XML yang dapat dibaca oleh sistem operasi Android.

Gambar 3.18 menunjukkan gambar sistem arsitektur dari perangkat lunak sistem pengenalan bentuk daun.



Gambar 3.18 Sistem Arsitektur Perangkat Lunak Sistem Pengenalan Bentuk Daun

UNIVERSITAS CIPUTRA