APLIKASI UNTUK MENGIDENTIFIKASI KEMATANGAN BUAH PISANG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION* BERBASIS ANDROID

SKRIPSI

Oleh

Yani Nur Muslimin

NIM 112410101017

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2015

APLIKASI UNTUK MENGIDENTIFIKASI KEMATANGAN BUAH PISANG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION* BERBASIS ANDROID

SKRIPSI

diajukan guna melengkapi tugas akhir dan memnuhi salah satu syarat untuk menyelesaikan pendidikan di Program Studi Sistem Informasi Universitas Jember dan mendapat gelar Sarjana Sistem Informasi

Oleh

Yani Nur Muslimin

NIM 112410101017

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2015

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Allah SWT;
2. Keluarga;
3. Dosen Pembimbing;
4. Almamater Program Studi Sistem Informasi Universitas Jember.

MOTO

"Man Jadda Wajada"

"Barangsiapa yang bersungguh – sungguh maka dia akan berhasil"

PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Yani Nur Muslimin

NIM : 112410101017

Menyatakan sesungguhnya bahwa karya ilmiah yang berjudul "Aplikasi Untuk Mengidentifikasi Kematangan Buah Pisang Menggunakan *Image Processing* Dengan Metode Jaringan Syaraf Tiruan *Learning Vector Quantization* Berbasis Android" adalah benar - benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan danpaksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 11 Juni 2015

Yang menyatakan,

Yani Nur Muslimin

NIM. 112410101017

PENGESAHAN PEMBIMBING

Skripsi berjudul **"Aplikasi Untuk Mengidentifikasi Kematangan Buah Pisang Menggunakan *Image Processing* Dengan Metode Jaringan Syaraf Tiruan *Learning Vector Quantization* Berbasis Android"**, telah diuji dan disahkan pada :

Hari tanggal : Kamis, 11 Juni 2015

Tempat : Program Studi Sistem Informasi Universitas Jember

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Dr. Saiful Bukhori, ST., M.Kom.	Windi Eka Yulia Retnani S.Kom., MT.
NIP. 196811131994121001	NIP. 198403052010122002

SKRIPSI

APLIKASI UNTUK MENGIDENTIFIKASI KEMATANGAN BUAH PISANG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE JARINGAN SYARAF TIRUAN *LEARNING VECTOR QUANTIZATION* BERBASIS ANDROID

Oleh :
Yani Nur Muslimin
NIM 112410101017

Pembimbing

Pembimbing Utama : Dr. Saiful Bukhori, ST., M.Kom.
Pembimbing Anggota : Windi Eka Yulia Retnani S.Kom., MT.

PENGESAHAN

Skripsi berjudul **"Aplikasi Untuk Mengidentifikasi Kematangan Buah Pisang Menggunakan *Image Processing* Dengan Metode Jaringan Syaraf Tiruan *Learning Vector Quantization* Berbasis Android"**, telah diuji dan disahkan pada :

Hari tanggal : Kamis, 11 Juni 2015

Tempat : Program Studi Sistem Informasi Universitas Jember

Tim Penguji

Ketua,

Dr. Saiful Bukhori, ST., M.Kom
NIP. 196811131994121001

Anggota I,

Prof. Drs. Slamin, M.Comp.Sc.,Ph.D.
19670420 1992011001

Anggota II,

Nelly Oktavia Adiwijaya, S.Si., MT.
NIP. 198410242009122008

Mengesahkan
Ketua Program Studi

Prof. Drs. Slamin, M.Comp.Sc.,Ph.D
NIP. 19670420 1992011001

RINGKASAN

Aplikasi Untuk Mengidentifikasi Kematangan Buah Pisang Menggunakan *Image Processing* Dengan Metode Jaringan Syaraf Tiruan *Learning Vector Quantization* Berbasis Android; Yani Nur Muslimin, 112410101017; 2015; 184 halaman; Program Studi Sistem Informasi Universitas Jember.

Pada pengolahan hasil pertanian buah pisang, proses pemilihan buah pisang salah satunya dapat dilakukan berdasarkan tingkat kematangan buah. Proses tersebut dilakukan dengan melihat perubahan kekerasan, bentuk ujung dan warna kulit pada buah pisang. Petani atau penyeleksi buah pisang umumnya mengidentifikasi tingkat kematangan buah pisang dilihat dari perubahan warna kulit pisang, karena hal tersebut yang paling mudah untuk dilakukan. Walaupun mudah dilakukan, pada kenyataannya hal tersebut menimbulkan permasalahan yang terkadang membuat proses pemilihan buah pisang kurang optimal yang secara tidak langsung akan berdampak pada kepuasan konsumen, apalagi hal tersebut dilakukan secara manual. Hal tersebut bisa terjadi karena perbedaan persepsi dari petani atau penyeleksi buah pisang terhadap fActor komposisi warna pada buah pisang tersebut. Perbedaan itu terjadi karena persepsi dari setiap manusia dalam mengamati komposisi warna atau citra suatu objek berbeda – beda walaupun objek yang dilihat sama persis, hal ini bisa terjadi oleh banyak fActor.

Salah satu alternatif teknologi yang dapat digunakan untuk mengurangi atau mengatasi masalah yang ada adalah dengan memanfaatkan pengolahan citra digital dan metode klasifikasi yaitu *Learning Vector Quantization (LVQ)*. Teknik pengolahan citra digital yang digunakan adalah pengolahan warna. Teknik tersebut digunakan untuk mengekstraksi atau mengambil nilai histogram warna (RGB) citra digital buah pisang yang di *capture* menggunakan kamera *smartphone*. Hasil pengolahan warna akan menjadi *dataset* untuk selanjutnya diolah menggunakan metode klasifikasi *Learning Vector Quantization* agar dapat mengidentifikasi kematangan buah pisang. Pengolahan citra digital dan metode klasifikasi *Learning*

Vector Quantization (LVQ) diimplementasikan ke dalam bentuk aplikasi berplatform android, hal ini dimaksudkan agar aplikasi dapat digunakan secara praktis dan cepat.

Aplikasi ini telah diimplementasikan pada *smartphone Sony Xperia E* dengan nilai *learning rate* = 0,5 dan *eps* = 0,001 serta *max epoch* = 5 kali dan 10 kali pada *dataset* berjumlah 9 data dan 12 data hasil pengolahan citra digital berupa nilai histogram warna (RGB) citra digital buah pisang yang di *capture* pada jarak = 14 cm, dan kemudian data - data tersebut diolah menggunakan metode *Learning Vector Quantization (LVQ*) dengan menghasilkan tingkat akurasi identifikasi yang sama yaitu 90% dengan rincian hasil identifikasi 9 kali benar dan 1 kali salah dari 10 kali percobaan menggunakan data *testing* hasil pengolahan citra digital berupa nilai histogram warna (RGB) pada citra digital buah pisang yang di *capture* pada jarak = 14 cm.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Aplikasi Untuk Mengidentifikasi Kematangan Buah Pisang Menggunakan *Image Processing* Dengan Metode Jaringan Syaraf Tiruan *Learning Vector Quantization* Berbasis Android". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada :

1. Prof. Drs. Slamin, M.Comp.Sc., Ph.D., selaku Ketua Program Studi Sistem Informasi Universitas Jember;
2. Dr. Saiful Bukhori, ST., M.Kom., selaku Dosen Pembimbing Utama dan Windi Eka Yulia Retnani S.Kom., MT., selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi;
3. Dr. Saiful Bukhori, ST., M.Kom., selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Program Studi Sistem Informasi Universitas Jember;
5. Ayah Kasiyanto dan Ibu Asri Asih Andayani serta seluruh pihak keluarga yang telah memberikan dukungan dan doa yang tulus;
6. Teman – teman seperjuangan dan juga teman yang saya perjuangkan.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu penulis mengharapkan adanya masukan yang bersifat membangun dari semua pihak. Penulis berharap skripsi ini dapat bermanfaat bagi semua pihak.

Jember, 11 Juni 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1. PENDAHULUAN

Bab ini merupakan langkah awal penulisan tugas akhir ini. Bab ini berisi latar belakang, perumusan masalah, tujuan dan manfaat, batasan masalah, metodologi penelitian dan sistematika penulisan.

1.1. Latar Belakang

Perkembangan teknologi dewasa ini membuat manusia ingin meningkatkan efektifitas dan efisiensi dalam berbagai bidang, salah satunya dalam bidang pertanian. Dalam dunia pertanian, kemajuan teknologi sangat dibutuhkan untuk menunjang kegiatan – kegiatan yang ada pada bidang pertanian, salah satunya dalam pengolahan hasil pertanian dan perkebunan. Industri pengolahan hasil pertanian dan perkebunan kini semakin berkembang pesat seiring perkembangan teknologi yang ada khususnya untuk produksi buah pisang. Pisang banyak disukai oleh semua kalangan untuk di konsumsi secara langsung sebagai buah atau diolah menjadi produk konsumsi lain seperti kripik pisang, sale pisang, selai pisang dan lain sebagainya. Badan Pusat Statistik Indonesia tahun 2013 menyajikan data bahwa produksi buah pisang di Indonesia mencapai 5.359.126 ton (BPS, 2014). Hal ini dapat dikatakan bahwa Indonesia berpotensi dalam memproduksi buah pisang.

Pada proses pengolahan hasil pertanian dan perkebunan ada beberapa tahapan sebelum produk yang dihasilkan di distribusikan ke konsumen. Salah satu tahapannya adalah pemilihan produk hasil pertanian dan perkebunan sesuai dengan kebutuhan konsumen. Ada banyak cara yang dapat dilakukan dalam tahapan tersebut, pada pengolahan hasil pertanian buah pisang proses pemilihan produk salah satunya dapat dilakukan berdasarkan tingkat kematangan buah. Kematangan adalah stadia perkembangan tanaman atau bagian tanaman yang memiliki persyaratan optimum untuk dapat dimanfaatkan oleh konsumen guna memenuhi tujuan tertentu (Bambang B. Santoso, 2011 : 82). Berdasarkan hal tersebut, kematangan buah dapat digunakan

sebagai parameter ukur pada proses pemilihan produk hasil pertanian dan perkebunan.

Proses pemilihan buah pisang berdasarkan tingkat kematangan buah dilakukan dengan melihat perubahan kekerasan, bentuk ujung dan warna kulit pada buah pisang. Tujuan dari proses pemilihan buah pisang ini untuk memisahkan buah pisang berdasarkan tingkat kematangan sesuai dengan kebutuhan. Petani atau penyeleksi buah pisang umumnya mengidentifikasi tingkat kematangan buah pisang dilihat dari perubahan warna kulit pisang, karena hal tersebut yang paling mudah untuk dilakukan. Walaupun mudah dilakukan, pada kenyataannya hal tersebut memiliki permasalahan yang terkadang proses pemilihan buah pisang kurang optimal, apalagi hal tersebut dilakukan secara manual. Hal tersebut bisa terjadi karena perbedaan persepsi dari petani atau penyeleksi buah pisang terhadap faktor komposisi warna pada buah pisang tersebut. Perbedaan itu terjadi karena persepsi dari setiap manusia dalam mengamati komposisi warna atau citra suatu objek berbeda – beda walaupun objek yang dilihat sama persis, hal tersebut bisa terjadi oleh banyak faktor. Permasalahan yang ada tersebut terkadang membuat konsumen kurang puas karena kebutuhan yang diharapkan tidak sesuai yang mereka butuhkan. Sehingga, perlu suatu alternatif yang dapat membantu mengurangi atau mengatasi permasalahan yang ada.

Berdasarkan permasalahan tersebut diatas, maka perlu diadakan penelitian tentang implementasi aplikasi untuk mengidentifikasi kematangan buah pisang dengan memanfaatkan pengolahan citra digital *(image processing)* dengan metode jaringan syaraf tiruan *Learning Vector Quantization (LVQ)* berbasis android. Penelitian ini perlu dilakukan karena penelitian ini dapat menjadi salah satu alternatif teknologi untuk dapat membantu mengurangi atau mengatasi permasalahan yang ada. Membangun sebuah aplikasi untuk mengidentifikasi kematangan buah pisang perlu mengkombinasikan beberapa bidang ilmu seperti pengolahan citra digital *(image processing)* dan *datamining*. Pengolahan citra digital *(image processing)* merupakan teknik untuk mengolah citra suatu objek secara digital yang mana setelah melalui

proses pengolahan akan didapatkan suatu informasi yang dapat diolah oleh komputer untuk berbagai kebutuhan. Hal ini nantinya dapat diterapkan pada saat mengolah citra digital dari warna kulit pada buah pisang dalam berbagai tingkat kematangan. Untuk mengidentifikasi objek citra digital buah pisang dapat dikatakan matang, setengah matang atau muda perlu bantuan bidang ilmu yang lainnya yaitu *datamining*. Secara umum *datamining* merupakan suatu bidang ilmu yang mempelajari tentang teknik *mining* atau mengolah data agar dapat digunakan untuk memberikan suatu indikasi tertentu yang bermanfaat. Dalam *datamining* terdapat banyak teknik atau metode dalam mengolah atau me-*mining* suatu data, salah satunya metode jaringan syaraf tiruan *Learning Vector Quantization (LVQ).* Hal ini nantinya diterapkan ketika proses pengidentifikasian kematangan buah pisang. Dimana proses ini menggunakan metode jaringan syaraf tiruan *Learning Vector Quantization (LVQ).* Kedua bidang ilmu tersebut nantinya dikombinasikan kedalam bentuk perangkat lunak atau aplikasi *mobile* berbasis android. Aplikasi ini di implementasikan berbasis android karena saat ini hampir semua *gadget* menggunakan sistem operasi android, sehingga dapat dikatakan aplikasi berbasis android merupakan *trend* saat ini dan cocok untuk digunakan pada saat ini.

Penelitian ini nantinya diharapkan dapat memberikan solusi terhadap permasalahan yang ada pada pengolahan hasil pertanian dan perkebunan. Khususnya pada proses pemilihan buah pisang berdasarkan tingkat kematangannya. Sehingga pengimplementasian aplikasi pada penelitian ini dapat membantu dalam pengolahan hasil pertanian dan perkebunan. Terutama proses pemilihan buah pisang berdasarkan tingkat kematangan buah sesuai kebutuhan konsumen yang dapat dilakukan secara cepat, tepat dan efisien.

1.2. Rumusan Masalah

Dengan mempertimbangkan latar belakang masalah diatas, dapat dirumuskan permasalahan sebagai berikut :

1. Bagaimana membangun aplikasi *Banana Maturity Identification* pada sistem operasi android.
2. Bagaimana mengimplementasikan pengolahan citra digital dan metode jaringan syaraf tiruan *Learning Vector Quantization (LVQ)* pada aplikasi *Banana Maturity Identification*.

1.3. Tujuan dan Manfaat

Berikut merupakan tujuan yang ingin dicapai dan manfaat yang ingin didapat dalam penelitian ini.

1.3.1. Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Membangun aplikasi *Banana Maturity Identification* berbasis android.
2. Mengimplementasikan pengolahan citra digital dan metode jaringan syaraf tiruan *Learning Vector Quantization (LVQ)* pada aplikasi *Banana Maturity Identification*.

1.3.2. Manfaat

Manfaat yang ingin didapatkan dari penelitian ini adalah :

a. Manfaat Bagi Akademis

Hasil penelitian ini diharapkan dapat memberikan kontribusi dan masukan bagi siapa saja yang membutuhkan informasi yang berhubungan dengan judul penelitian ini. Selain itu, hasil penelitian ini merupakan suatu upaya untuk menambah varian judul penelitian yang ada di Program Studi Sistem Informasi Universitas Jember.

b. Manfaat Bagi Peneliti

1. Mengetahui bagaimana proses penerapan pengolahan citra digital dan metode jaringan syaraf tiruan *Learning Vector Quantization (LVQ)* pada aplikasi *Banana Maturity Identification*.

2. Sebagai media bagi penyelesaian Tugas Akhir untuk jenjang S1 pada Program Studi Sistem Informasi Universitas Jember.

c. Manfaat Bagi Objek Penelitian

1. Memberikan inovasi baru kepada instansi tempat penelitian dilakukan mengenai penggunaan aplikasi untuk mengidentifikasi kematangan buah pisang.
2. Membantu instansi untuk melakukan pemilihan buah pisang secara cepat dengan tingkat kesalahan yang minimum.

1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Pisang yang digunakan adalah jenis pisang mas kirana.
2. Identifikasi kematangan buah pisang hanya berdasarkan nilai RGB dari citra digital pada buah pisang.
3. Nilai RGB citra digital pada buah pisang diperoleh melalui teknik pengolahan citra digital dengan mengekstraksi warna citra digital buah pisang pada dimensi atau ukuran 200 x 300 px.
4. *Aspect ratio camera* pada perangkat dalam mengakusisi citra digital adalah 4 : 3.
5. Citra digital yang diambil untuk *dataset training* dan *testing* dalam pencahayaan atau *saturation* yang sama.
6. Metode pengklasifikasian yang digunakan dalam penelitian ini adalah *Learning Vector Quantization (LVQ).*
7. Aplikasi yang dibangun merupakan aplikasi berbasis *mobile android*.

1.5. Sistematika Penulisan

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut :

a. Pendahuluan

Bab ini terdiri atas latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah dan sistematika penulisan.

b. Tinjauan Pustaka

Bab ini berisi tentang kajian pustaka, penelitian terdahulu dan informasi apa saja yang digunakan dalam penelitian ini. Dimulai dari memaparkan penelitian dahulu sampai kajian pustaka mengenai penelitian ini.

c. Metodologi Penelitian

Bab ini menguraikan tentang metode apa yang dilakukan selama penelitian. Dimulai dari tahap pencarian permasalahan hingga pengujian aplikasi *Banana Maturity Identification* akan dibuat.

d. Hasil dan Pembahasan

Bab ini menjelaskan tentang hasil dan pembahasan dari penelitian yang telah dilakukan. Dengan memaparkan hasil penelitian dan hasil percobaan pengimplementasian sistem.

e. Penutup

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Bab ini memaparkan teori - teori dan pustaka yang gunakan dalam penelitian. Teori - teori ini diambil dari buku literatur dan jurnal. Berikut merupakan teori - teori yang digunakan dan dibahas dalam penelitian ini :

2.1. Penelitian Terdahulu

Adapun penelitian terdahulu sehingga penelitian ini muncul adalah sebagai berikut ini :

1. Penelitian yang berjudul "Identifikasi Kematangan Buah Tomat Menggunakan Metoda *Backpropagation*" dilakukan oleh Dila Deswari seorang Mahasiswa Jurusan Sistem Komputer, Fakultas Teknologi Informasi, Universitas Andalas, Hendrick, MT Dosen Jurusan Teknik Elektro, Politeknik Negeri Padang, dan Derisma, MT Dosen Jurusan Sistem Komputer, Fakultas Teknologi Informasi, Universitas Andalas. Penelitian ini meneliti pengaplikasian metode *backpropagation* untuk mengidentifikasi kematangan buah tomat dengan memanfaatkan pengolahan citra digital atau *image processing*. Penggunaan metode *backpropagation* dalam mengklasifikasikan tingkat kematangan buah tomat dapat diterapkan dengan mengkombinasikan pengolahan citra digital *(image processing)*, dimana sebelum proses pengklasifikasian buah tomat dilakukan, tahap pertama adalah mengolah citra kulit tomat yang akan diidentifikasi kematangannya dengan teknik *image processing* yaitu setelah citra diakusisi selanjutnya citra tersebut diambil nilai histogramnya kemudian dinormalisasi RGB warnanya. Setelah proses pengolahan citra selesai dilakukan maka hasil histogram dari tahap pengolahan citra digital tersebut digunakan sebagai inputan proses pengklasifikasian menggunakan metode *backpropagation* sehingga dapat diketahui tingkat kematangan buah tersebut. Salah satu hasil yang di dapat dari penelitian ini adalah sebuah aplikasi yang dapat mengidentifikasi kematangan

buah tomat menggunakan metode *backpropagation* dengan akurasi pengidentifikasian sebesar 71,67 %.

2. Penelitian yang berjudul "Jaringan Syaraf Tiruan *Learning Vector Quantization* Untuk Aplikasi Pengenalan Tanda Tangan" dilakukan oleh Difla Yustisia dan Safrina Rosmalinda seorang Mahasiswa Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penelitian ini meneliti pengenalan tanda tangan dengan menerapkan jaringan syaraf tiruan *learning vector quantization*. Penggunaan metode *learning vector quantization (lvq)* dalam mengidentifikasi tanda tangan seseorang dapat diterapkan dengan mengkombinasikan pengolahan citra digital *(image processing),* sebelum proses pengenalan tanda tangan dilakukan menggunakan jaringan syaraf tiruan *learning vector quantization*, tahap pertamanya adalah mengolah citra tanda tangan dengan teknik *image processing* yaitu menggunakan *Edge Detection Method*. Setelah proses pengolahan citra digital selesai dilakukan maka hasil deteksi tepi *(Edge Detection Method)* digunakan sebagai inputan proses pengklasifikasian atau pengenalan tanda tangan menggunakan jaringan syaraf tiruan *learning vector quantization.* Salah satu hasil yang di dapat dari penelitian ini adalah sebuah aplikasi yang dapat melakukan pengenalan tanda tangan menggunakan jaringan syaraf tiruan *learning vector quantization* akurasi pengidentifikasian sebesar 98 %.

2.2. Pisang

Pisang merupakan tanaman semak yang berbatang semu *(pseudostem),* tingginya bervariasi antara 1 - 4 meter, tergantung varietasnya. Daunnya melebar, panjang, tulang daunnya besar, dan tepi daunnya tidak mempunyai ikatan yang kompak sehingga mudah robek bila terkena angin kencang. Batangnya mempunyai bonggol (umbi) yang besar dan terdapat banyak mata yang dapat tumbuh menjadi tunas anakan. Bunganya tunggal, keluar pada ujung batang dan hanya sekali berbunga selama hidupnya *(monokarpik)* (Sunarjono, 2000). Pisang berasal dari kawasan Asia Tenggara termasuk Indonesia. Tanaman ini kemudian menyebar ke Afrika, Amerika

Selatan dan Tengah. Buah pisang mempunyai kandungan gizi yang baik dan merupakan sumber vitamin, mineral dan juga karbohidrat. Sehingga pisang banyak digemari banyak orang dari semua kalangan.

Pisang memiliki banyak jenis dan kegunaannya, namun tidak semua jenis pisang yang ada dapat diperoleh di pasaran. Dari berbagai jenis pisang, menurut (Prabawati : 2008) ada dua jenis pisang yang dapat dimakan dan dikelompokkan berdasarkan penggunaannya. Pertama, pisang meja *(banana)* yang umum disajikan sebagai buah segar, dan kedua, pisang untuk olahan *(plantain)* yang hanya enak dimakan setelah terlebih dahulu diolah menjadi berbagai produk makanan. Jenis pisang meja yang terkenal antara lain pisang Ambon Kuning, Ambon Lumut, Barangan, Mas, Lampung, Raja Bulu dan Raja Sere, sedangkan jenis pisang olahan yang terdapat banyak di pasaran adalah Kepok, Kapas, Nangka, Siem, Tanduk, dan Uli. Dua kelompok pisang tersebut termasuk dalam keluarga *Musaceae*. Jenis pisang lainnya yaitu pisang Batu dan pisang Klutuk yang banyak dimanfaatkan daunnya, karena buahnya banyak mangandung biji. Terdapat juga jenis pisang yang diambil seratnya yaitu pisang Manila dan Abaca.

Potensi buah pisang yang dapat diperdagangkan untuk pasar dalam negeri bahkan luar negeri perlu penanganan atau pengolahan terlebih dahulu sebelum dipasarkan. Hal ini umumnya disebut penanganan pascapanen atau pengolahan hasil pertanian. Dalam pengolahan hasil pertanian, khususnya buah pisang harus dilakukan dengan baik agar pisang yang diperdagangkan dapat diterima dipasaran dengan mutu yang baik. Gambar 2.1 menunjukkan alur umum penanganan pascapanen buah pisang yang perlu dilakukan para petani pisang.

Gambar 2.1 Alur Penanganan Pascapanen Pisang

(Sumber : Prabawati, 2008)

Pada alur penanganan pascapanen pisang terdapat beberapa tahapan. Adapun penjelasan singkat dari tahapan tersebut, yaitu :

a. Panen

   Proses pengambilan buah pisang dari pohon pisang.

b. Pengangkutan Ke Tempat Pengangkutan

   Proses pengangkutan buah pisang yang sudah dipanen dari kebun ke tempat pengangkutan untuk dikumpulkan.

c. Pemotongan Sisir

   Proses pemisahan buah pisang dari tandan pisang dalam bentuk sisir pisang.

d. Sortasi

   Proses pemilihan pisang yang layak dipasarkan, biasanya disesuaikan dengan kebutuhan konsumen atau pasar. Biasanya dipisahkan berdasarkan tingkat kematangan, bentuk yang tidak normal, kerusakan mekanis dan lain sebagainya. Proses ini juga membantu dalam proses *grading*, oleh karena itu sortasi biasanya dilakukan bersamaan dengan proses *grading.*

e. Pencucian

   Proses pencucian pisang yang telah dipilih atau disortasi agar bersih dari kotoran.

f. Penirisan

Proses pengeringan pisang yang telah dicuci.

g. Pengendalian OPT Pascapanen

Proses pengecekan pisang terhadap organisme penggangu tanaman (OPT) misalnya serangan hama atau penyakit terhadap buah pisang.

h. Pengemasan

Proses *packing* pisang, biasanya setiap sisir pisang akan dimasukkan kedalam kardus kemasan dengan berat tertentu sesuai dengan kebutuhan.

i. Pengangkutan dan Pemeraman

Proses pengangkutan pisang yang sudah dalam bentuk kemasan. Sebelum pisang sampai ke konsumen diperlukan perlakukan khusus yaitu proses pemeraman terhadap pisang agar kematangannya terkontrol, karena menurut (Wills et al., 1999) dalam Prabawati menyatakan bahwa pisang tergolong sebagai buah klimakterik, sehingga setelah dipanen masih melangsungkan proses fisiologi dengan menghasilkan etilen dan karbon dioksida dalam jumlah yang meningkat drastis.

2.3. Aplikasi

Aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan – pekerjaan tertentu (khusus) (Hendrayudi, 2009). Menurut Jogiyanto (2004 : 4), aplikasi merupakan program yang berisikan perintah - perintah untuk melakukan pengolahan data. Jogiyanto menambahkan definisi aplikasi secara umum adalah suatu proses dari cara manual yang ditransformasikan ke komputer dengan membuat sistem atau program agar data diolah lebih berdaya guna secara optimal.

2.4. Pengolahan Citra Digital

Pengolahan citra digital merupakan proses memanipulasi dan menganalisis citra menggunakan bantuan perangkat komputer ataupun perangkat lainnya. Sebagaimana yang dikemukakan oleh Darma Putra bahwa secara umum, pengolahan

citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer (Darma Putra, 2010 : 12). Biasanya kegiatan pengolahan citra digital digunakan untuk memperbaiki kualitas suatu objek gambar sehingga dapat lebih mudah di interpretasikan atau dilihat oleh mata manusia, selain itu juga digunakan untuk mengolah informasi yang terdapat pada suatu gambar sehingga dapat mengenali suatu objek citra secara otomatis.

Secara terminologi pengolahan citra digital berbeda dengan mesin visual walaupun keduanya hampir mirip. Usman Ahmad (2005 : 3) mengemukakan bahwa pengertian pengolahan citra *(Image Processing)* sedikit berbeda dengan pengertian mesin visual *(Machine Vision)*, meskipun keduanya seolah – olah dapat dipergunakan dengan maksud yang sama. Terminologi pengelohan citra dipergunakan bila hasil pengolahan data berupa citra, adalah juga berbentuk citra hasil yang lain, yang mengandung atau memperkuat informasi khusus pada citra hasil pengolahan sesuai dengan tujuan pengolahannya. Sedangkan terminologi mesin visual digunakan bila data hasil pengolahan citra langsung diterjemahkan dalam bentuk lain, misalnya grafik yang siap diinterpretsdiksn untuk tujusn tertentu, gerak peralatan atau bagian dari mekanis, atau aksi lainnya yang berarti bukan merupakan citra lagi.

2.4.1. Akusisi Citra

Pada pengolahan citra digital, akusisi citra merupakan proses awal yang harus dilakukan karena tanpa melakukan akusisi citra maka pengolahan citra digital tidak bisa dilakukan. Proses akusisi citra merupakan pemetaan suatu pandangan *(scene)* menjadi citra kontinu dengan menggunakan sensor (Darma Putra, 2010). Intinya akusisi citra adalah tahap awal untuk mengambil atau mendapatkan citra digital menggunakan suatu perangkat atau alat tambahan tertentu. Tujuannya untuk menentukan data yang diperlukan dan memilih metode perekaman citra digital.

2.4.2. Pengolahan Warna

Warna yang dimiliki suatu obyek merupakan salah satu informasi yang dimiliki oleh obyek tersebut untuk mendeskripsikan salah satu ciri – ciri dari obyek tersebut. Informasi warna yang terkandung dalam suatu citra berwarna dihitung dan dianalisis untuk selanjutnya digunakan dalam suatu proses penilaian atau pengelompokan obyek – obyek dengan warna tertentu. Salah satunya, pengolahan model warna RGB.

Pengolahan warna menggunakan model RGB sangat mudah dan sederhana, karena informasi warna dalam komputer sudah dikemas dalam model yang sama. Hal yang perlu dilakukan adalah bagaimana melakukan pembacaan nilai – nilai R, G, dan B pada suatu *pixel*, menampilkan dan menafsirkan hasil perhitungan tadi sehingga mempunyai arti sesuai dengan yang diinginkan. Salah satu cara yang mudah untuk menghitung nilai warna dan menafsirkan hasilnya dalam model RGB adalah dengan melakukan normalisasi terhadap ketiga komponen warna tersebut. Normalisasi penting dilakukan terutama bila sejumlah citra ditangkap dengan penerangan yang berbeda - beda. Hasil perhitungan tiap komponen warna pokok yang telah dinormalisasi akan menghilangkan pengaruh penerangan, sehingga nilai untuk setiap komponen warna dapat dibandingkan satu sama lainnya walaupun berasal dari citra dengan kondisi penerangan yang tidak sama, tetapi tidak terlalu ekstrim perbedannya. Cara melakukan normalisasi dapat dilihat pada persamaan (1), (2), dan (3).

$$r = \frac{R}{R+G+B} \quad \text{..... pers (1)}$$

$$g = \frac{G}{R+G+B} \quad \text{..... pers (2)}$$

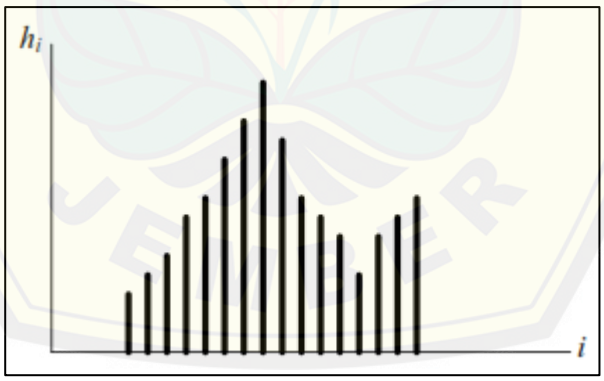
$$b = \frac{B}{R+G+B} \quad \text{..... pers (3)}$$

Nilai warna hasil normalisasi ini kemudian ditafsirkan dengan melihat besarannya. Bila ketiga komponen warna yang telah dinormalkan, jika masing – masing menjadi

indeks warna merah *(r)*, indeks warna hijau *(g)*, indeks warna biru *(b)*, mempunyai nilai yang sama (1/3), maka obyek tidak berwarna. Bila *r* lebih besar dari pada *g* dan *b*, maka obyek berarti berwarna merah, dan seterusnya. Dengan kata lain dominasi warna dapat dilihat dari besaran nilai tiap indeks. Warna merah murni akan mempunyai nilai *r* sama dengan satu, sementara dua indeks lainnya bernilai nol, dan seterusnya (Usman Ahmad, 2005 : 271).

2.4.3. Histogram Citra

Informasi penting mengenai isi citra digital dapat diketahui dengan membuat histogram citra. Histogram citra adalah grafik yang menggambarkan penyebaran nilai – nilai intensitas *pixel* dari suatu citra atau bagian tertentu di dalam citra. Dari sebuah histogram dapat diketahui frekuensi kemunculan nisbi *(relative)* dari intensitas pada citra tersebut (Rinaldi Munir, 2004). Diagram histogram citra dapat dilihat pada Gambar 2.2.



Gambar 2.2 Histogram Citra

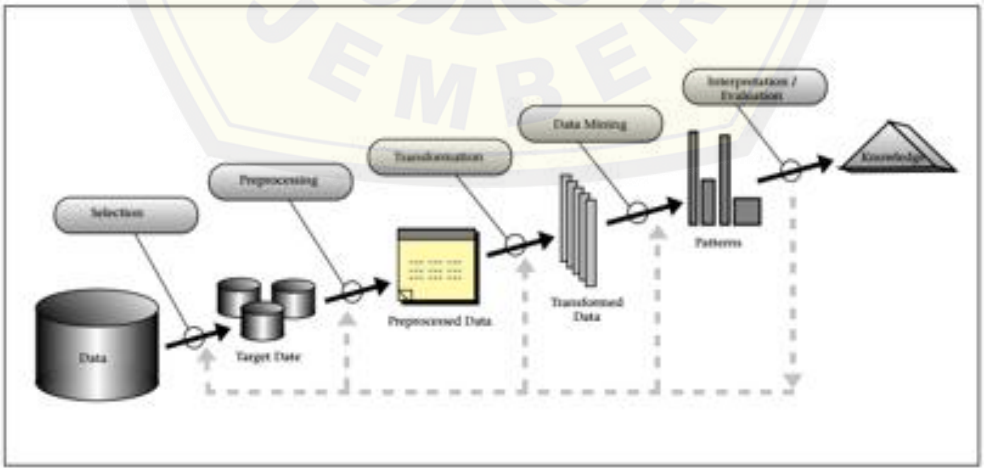
(Sumber : Rinaldi Munir, 2004)

Histogram citra menunjukkan banyak hal tentang kecerahan *(brightness)* dan kontas *(contrast)* dari sebuah gambar. Puncak histogram menunjukkan intensitas *pixel* yang menonjol. Lebar dari puncak menunjukkan rentang kontras dari gambar. Citra yang mempunyai kontras terlalu terang *(overexposed)* atau terlalu gelap

(underexposed) memiliki histogram yang sempit. Histogramnya terlihat hanya menggunakan setengah dari daerah derajat keabuan. Citra yang baik memiliki histogram yang mengisi daerah derajat keabuan secara penuh dengan distribusi yang merata pada setiap derajat keabuan *pixel* (Seminar Nasional Aplikasi Teknologi Informasi (SNATI), 2006). Sehingga, histogram dapat digunakan sebagai alat bantu yang berharga dalam kegiatan pengolahan citra digital baik secara kualitatif maupun kuantitatif.

2.5. Datamining

Datamining merupakan suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan atau *knowledge* di dalam basis data atau *database* dimana nantinya data tersebut dapat digunakan untuk memberikan suatu indikasi yang bermanfaat. Turban mengemukakan bahwa *Datamining* adalah proses yang menggunakan teknik statistik, matematika, kercerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait (Kusrini, dkk., 2009). *Data mining* merupakan bagian dari proses *Knowledge Discovery in Databases (KDD)*. Proses dari KDD tersebut dapat dilihat pada Gambar 2.3.



Gambar 2.3 Tahap - Tahap dalam Proses *Knowledge Discovery*

Sumber : (Fayyad, 1996)

2.5.1. Data *Preprocessing*

Sebelum data diolah menggunakan *datamining*, data perlu melalui tahap *preprocessing*. Menurut Han, tahap ini berhubungan dengan pemilihan dan pemindahan data yang tidak berguna *(data cleaning)*, penggabungan sumber - sumber data *(data integration)*, transformasi data dalam bentuk yang dapat mempermudah proses *(data transformation)*, menampilkan data dalam jumlah yang lebih mudah dibaca *(data reduction)*. Semuanya berasal dari data mentah (data transaksi) dan hasilnya akan menjadi data yang nantinya siap untuk diolah dengan *datamining* (Seminar Nasional Aplikasi Teknologi Informasi (SNATI), 2009). Jadi, data yang akan diolah menggunakan *datamining*, awalnya harus dipersiapkan terlebih dahulu melalui tahap *preprocessing* agar data yang diolah nanti dapat menghasilkan suatu hasil atau informasi yang berguna sesuai yang dibutuhkan.

2.5.2. Klasifikasi

Klasifikasi merupakan proses menemukan sekumpulan model (atau fungsi) yang menggambarkan dan membedakan konsep atau kelas – kelas data, dengan tujuan agar model tersebut dapat digunakan untuk memprediksi kelas dari suatu objek atau data yang label kelasnya tidak diketahui (Han dan Kamber, 2000). Klasifikasi terdiri atas dua tahap, yaitu tahap pelatihan *(training)* dan prediksi (klasifikasi). Pada tahap pelatihan dibentuk sebuah model domain permasalahan dari setiap kasus atau *instance* yang ada. Penentuan model tersebut berdasarkan analisis pada sekumpulan data pelatihan *(training)*, yaitu data yang label kelasnya telah diketahui. Pada tahap klasifikasi, dilakukan prediksi kelas dari kasus atau *instance* baru yang telah dibuat pada tahap pelatihan.

2.6. Jaringan Syaraf Tiruan LVQ

Jaringan Syaraf Tiruan (JST) merupakan representasi buatan dari otak manusia yang selalu mencoba mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini di

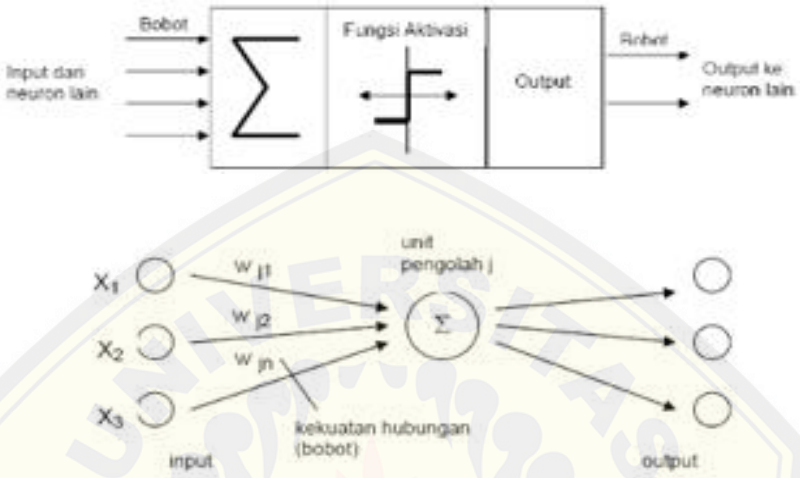
implementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran (Kusumadewi, 2003). Menurut Kusrini (2009 : 199) jaringan syaraf tiruan merupakan sebuah model yang mengadopsi cara kerja *neuron* secara biologi dengan fokus pada cara kerja saraf otak. Pemodelan yang dilakukan hanya di dekati dari sudut komputasinya saja.

Jaringan syaraf tiruan terdiri dari sejumlah simpul *(node)* yang merupakan elemen pemroses. Setiap simpul tersebut memodelkan sebuah sel saraf biologis *(neuron)*. Hubungan antar simpul dicapai melalui bobot koneksi *(weight)*. Bobot koneksi menentukan apakah sinyal yang mengalir bersifat peredam *(inhibitory connection)*. Bobot koneksi yang bersifat meredam dapat dinyatakan, misalnya oleh bilangan negatif, sedangkan yang bersifat merangsang oleh bilangan positif. Selain ditentukan oleh karakteristik bobot koneksinya, besrnya sinyal yang keluar dari sebuah simpul juga ditentukan oleh fungsi aktifasi *(activation function)* yang digunakannya (Jurnal Media Informatika - Volume 4 No. 1, Juni 2006). Jadi, jaringan syaraf tiruan dapat dikatakan sebagai pemodelan atau simulasi yang di buat menyerupai proses pembelajaran pada otak manusia, sehingga pemodelan atau simulasi yang umumnya di implementasikan ke dalam program komputer dapat bekerja menyerupai pembelajaran pada otak manusia yang dapat mempelajari sesuatu secara otomatis berdasarkan inputan yang ada.

2.6.1. Konsep Dasar Jaringan Syaraf Tiruan

Pada jaringan syaraf tiruan setiap pola - pola informasi masukan *(input)* dan keluaran *(output)* yang diberikan ke dalam jaringan syaraf tiruan diproses dalam *neuron*. Neuron - neuron tersebut terkumpul di dalam lapisan - lapisan yang disebut *neuron layers*. Neuron - neuron pada satu lapisan akan dihubungkan dengan lapisan - lapisan sebelum dan sesudahnya. Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan masukan sampai ke lapisan keluaran melalui lapisan tersembunyi *(hidden layer)*. Gambar 2.4 berikut ini

merupakan jaringan syaraf dengan 3 lapisan dan bukanlah struktur umum jaringan syaraf karena beberapa jaringan syaraf ada yang tidak memiliki lapisan tersembunyi (Jurnal Komputer dan Informatika (KOMPUTA) – Edisi I Volume I, Maret 2012).



Gambar 2.4 Model Struktur JST

Sumber : (KOMPUTA, 2012)

Adapun faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Pada umumnya neuron - neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama sehingga pada setiap lapisan yang sama neuron - neuron memiliki fungsi aktivasi yang sama. Jika neuron - neuron pada suatu lapisan (misal lapisan tersembunyi) akan dihubungkan dengan neuron – neuron pada lapisan lain (misal lapisan keluaran) maka setiap neuron pada lapisan tersebut (lapisan tersembunyi) juga harus dihubungkan dengan setiap neuron pada lapisan lainnya (lapisan keluaran).

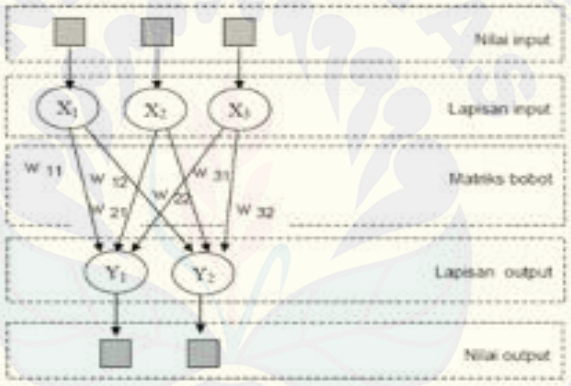
2.6.2. Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai aplikasi. Arsitektur jaringan syaraf tiruan tersebut, antara

lain (Jurnal Komputer dan Informatika (KOMPUTA) – Edisi I Volume I, Maret 2012) sebagai berikut :

1. Jaringan Lapisan Tunggal *(Single Layer Network)*

   Jaringan ini hanya memiliki 1 lapisan dengan bobot - bobot terhubung. Jaringan ini hanya menerima masukan kemudian secara langsung akan mengolahnya menjadi keluaran tanpa harus melalui lapisan tersembunyi. Pada gambar berikut neuron - neuron pada kedua lapisan saling berhubungan. Seberapa besar hubungan antara 2 neuron ditentukan oleh bobot yang bersesuaian. Semua unit masukan akan dihubungkan dengan setiap unit keluaran seperti terlihat pada Gambar 2.5.

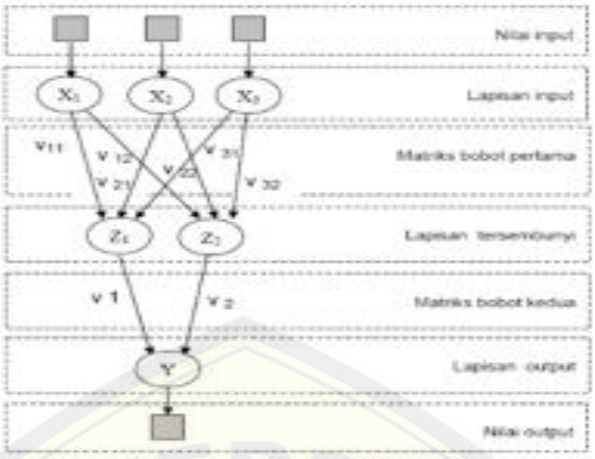


Gambar 2.5 Jaringan Lapisan Tunggal

Sumber : (Kusumadewi, 2003)

2. Jaringan Lapisan Banyak *(Multi Layer Network)*

   Jaringan ini memiliki 1 atau lebih lapisan yang terletak diantara lapisan masukan dan lapisan keluaran. Umumnya ada lapisan bobot - bobot yang terletak antara 2 lapisan yang bersebelahan. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit dari pada lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah. Berikut arsitektur multi layer network dapat dilihat pada Gambar 2.6.

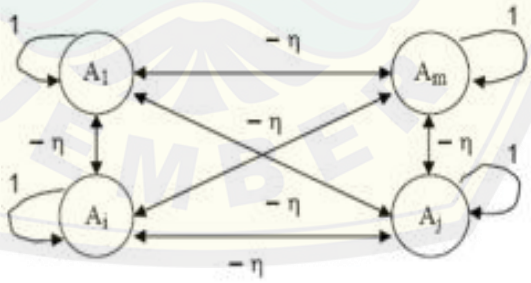


Gambar 2.6 Jaringan Lapisan Banyak

Sumber : (Kusumadewi, 2003)

3. Jaringan Lapisan Kompetitif *(Competitive Layer Network)*

   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Umumnya hubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar 2.7 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot .



Gambar 2.7 Jaringan Lapisan Kompetitif

Sumber : (Kusumadewi, 2003)

2.6.3. Metode Pembelajaran *(Training)* Jaringan Syaraf Tiruan

Pembelajaran atau *training* jaringan syaraf tiruan dapat di kelompokkan menjadi tiga (Jurnal Komputer dan Informatika (KOMPUTA) – Edisi I Volume I, Maret 2012) , yaitu:

1. Pembelajaran Terawasi *(Supervised Learning)*

   Pada pembelajaran ini kumpulan data masukan yang digunakan dan data keluarannya telah diketahui. Perbedaan antara keluaran - keluaran aktual dengan data keluaran yang diinginkan digunakan untuk mengoreksi bobot JST, agar JST dapat menghasilkan hasil sedekat (semirip) mungkin dengan hasil yang benar yang telah diketahui oleh JST.

2. Pembelajaran Tak Terawasi *(Unsupervised Learning)*

   Pada pembelajaran ini, JST mengorganisasi dirinya sendiri untuk membentuk vektor - vektor masukan yang serupa, tanpa menggunakan data atau contoh - contoh pelatihan. Struktur menggunakan dasar data atau korelasi antara pola - pola data yang dieksplorasi. Paradigma pembelajaran ini mengorganisasi pola - pola ke dalam kategori - kategori berdasarkan korelasi yang ada.

3. Pembelajaran Hibrid *(Hybrid Learning)*

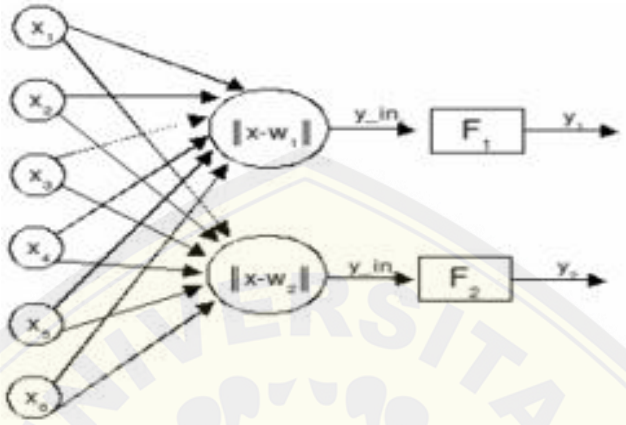
   Merupakan kombinasi kedua pembelajaran *Supervised Learning* dan *Unsupervised Learning*, sebagian dari bobot - bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi.

2.6.4. *Learning Vector Quantization (LVQ)*

Learning Vector Quantization (LVQ) adalah suatu metode untuk melakukan pembelajaran *(training)* pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor - vektor *input*. Kelas - kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor - vektor *input*. Jika 2 vektor *input* mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor *input* tersebut ke dalam kelas yang sama (Kusumadewi,2003).

2.6.4.1. Arsitektur *Learning Vector Quantization (LVQ)*

Adapun arsitektur dari *Learning Vector Quantization (LVQ)* dapat dilihat pada Gambar 2.8.



Gambar 2.8 Arsitektur *Learning Vector Quantization (LVQ)*

Sumber : (Kusumadewi, 2003)

Gambar 7 diatas memperlihatkan bahwa yang bertindak sebagai *dendrit* atau data masukan adalah X_1 sampai dengan X_6, kemudian yang bertindak sebagai *sinapsis* atau bobot adalah W, sedangkan *soma* atau badan sel dari jaringan ini adalah perhitungan $\| x - w_1 \|$ sampai dengan $\| x - w_2 \|$, dan kemudian yang bertindak sebagai *akson* atau data keluaran adalah Y.

2.6.4.2. Algoritma JST *Learning Vector Quantization (LVQ)*

Berikut adalah algoritma dari JST *Learning Vector Quantization (LVQ)*, yaitu :

1. Tetapkan : Bobot (W), Maksimum *Epoch (MaxEpoch)*, *Error* minimum yang diharapkan *(eps)*, *Learning Rate (α)*.
2. Masukkan :

   Input : X(m,n)

   Target : T(1,n)

3. Tetapkan kondisi awal :

   epoch = 0

   err = 1

4. Kerjakan jika : *(epoch < MaxEpoch)* atau *(α > eps)*

   a. *epoch* = *epoch* + 1

   b. Kerjakan untuk *i* = 1 sampai *n*

      1. Tentukan *j* sedemikian hingga || $x - w_j$ || minimum (sebut sebagai C_j)

      2. Perbaiki w_j dengan ketentuan :

         - Jika T = C_j maka : w_j (baru) = w_j (lama) + α (x - w_j (lama))
         - Jika T ≠ C_j maka : w_j (baru) = w_j (lama) - α (x - w_j (lama))

   c. Kurangi nilai α = α – (0,1 * α)

Adapun beberapa penjelasan mengenai variabel atau parameter pada algoritma *Learning Vector Quantization (LVQ)* adalah :

a. Nilai Alfa *(learning rate)*

   Learning rate merupakan nilai laju pembelajaran. Jika nilai *learning rate* terlalu besar, maka algoritma akan menjadi tidak stabil sebaliknya jika alfa terlalu kecil, maka prosesnya akan terlalu lama. Nilai *learning rate* adalah antara 0 dan 1 (Jurnal Media Statistika, Vol. 3, No 1, Juni 2010: 21 - 30).

b. Nilai *Max Epoch*

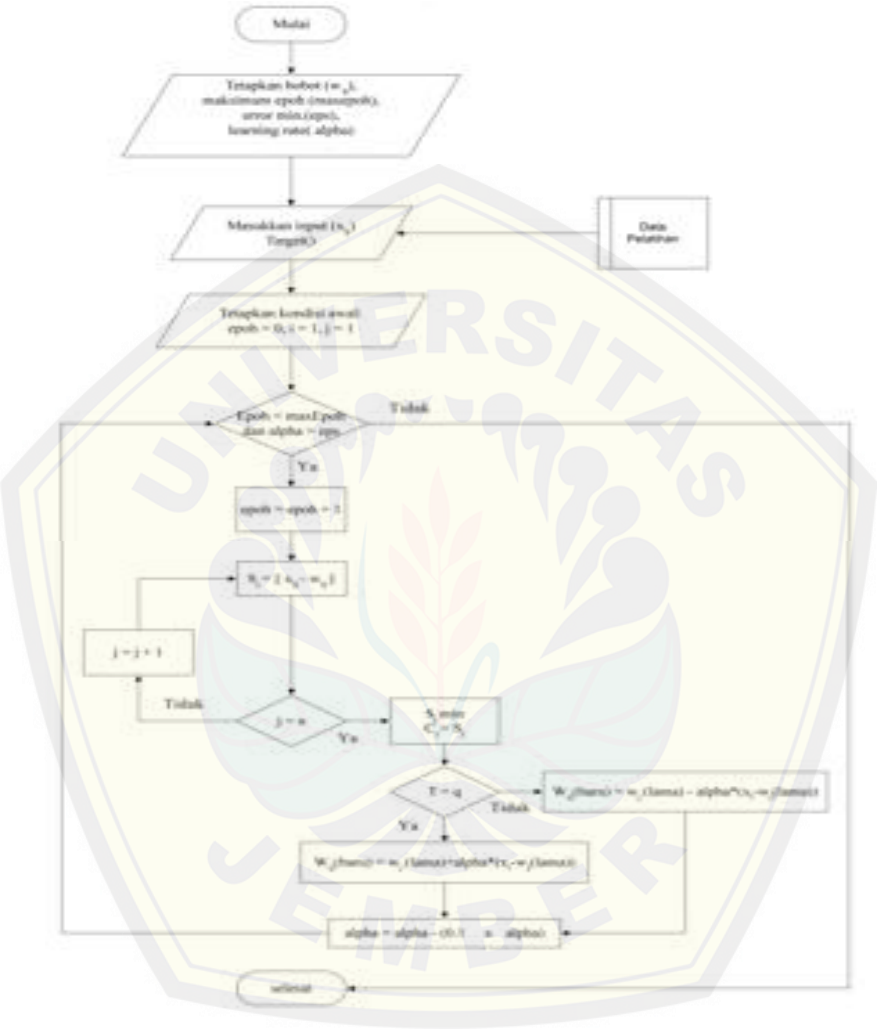
   Max Epoch merupakan nilai atau jumlah iterasi maksimum yang boleh dilakukan selama *training* berlangsung.

c. Nilai Min Alfa *(eps)*

   Eps merupakan nilai minimum *error* yang diharapkan pada proses *training*. Proses training akan terus berjalan jika nilai alfa *(learning rate)* lebih besar dari nilai *eps*.

2.6.4.3. *Flowchart* JST *Learning Vector Quantization (LVQ)*

Adapun *flowchart* algoritma dari JST *Learning Vector Quantization (LVQ)* dapat dilihat pada Gambar 2.9.



Gambar 2.9 *Flowchart* Algoritma JST *Learning Vector Quantization (LVQ)*

Sumber : (SNATI, 2010)

2.7. Android

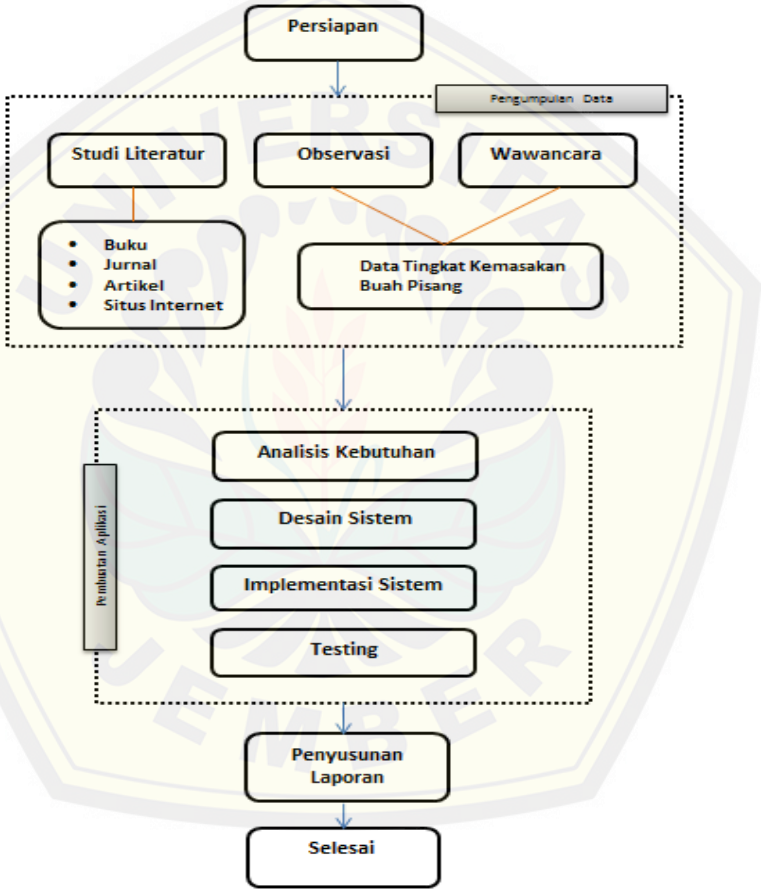
Android adalah sistem operasi bergerak *(mobile operating system)* yang mengadopsi sistem operasi *Linux*, namun telah dimodifikasi. *Android* diambil alih oleh *Google* pada tahun 2005 dari *Android, Inc* sebagai bagian strategi untuk mengisi pasar sistem operasi bergerak. *Google* mengambil alih seluruh hasil kerja *Android*

termasuk tim yang mengembangkan Android. *Google* menginginkan agar *Android* bersifat terbuka dan gratis, oleh karena itu hampir setiap kode program *Android* diluncurkan berdasarkan lisensi *open-source apache* yang berarti bahwa semua orang yang ingin menggunakan Android dapat men-*download* penuh *source code*-nya.

Di samping itu produsen perangkat keras juga dapat menambahkan *extension*-nya sendiri ke dalam *Android* sesuai dengan kebutuhan produk mereka. Model pengembangannya yang sederhana membuat *Android* menarik bagi *vendor – vendor* perangkat keras. Keuntungan utama dari android adalah adanya pendekatan aplikasi secara terpadu. Pengembang hanya berkonsentrasi pada aplikasi saja, aplikasi tersebut bisa berjalan pada beberapa perangkat yang berbeda selama masih ditenagai oleh *Android* dimana pengembang tidak perlu mempertimbangkan kebutuhan jenis perangkatnya (Dodit Supriyanto *et al*, 2012 : 9). Jadi, aplikasi yang dibuat berbasis android saat ini memiliki banyak keuntungan dan selain itu android kini menjadi *trend* yang *booming* di dunia teknologi.

BAB 3. METODOLOGI PENELITIAN

Bab ini akan memaparkan langkah dan prosedur yang akan dilakukan dalam mengumpulkan data atau informasi empiris guna memecahkan permasalahan dalam penelitian ini. Adapun alur penelitian untuk membuat aplikasi *Banana Maturity Identification* berbasis android dapat dilihat pada Gambar 3.1 dibawah ini.



Gambar 3.1 Diagram Alir Penelitian

Sumber : (Hasil Analisis, 2014)

3.1. Jenis Penelitian

Pada penelitian ini digunakan dua jenis penelitian, yaitu penelitian kualitatif dan penelitian kuntitatif. Jenis penelitian kualitatif digunakan karena penelitian ini

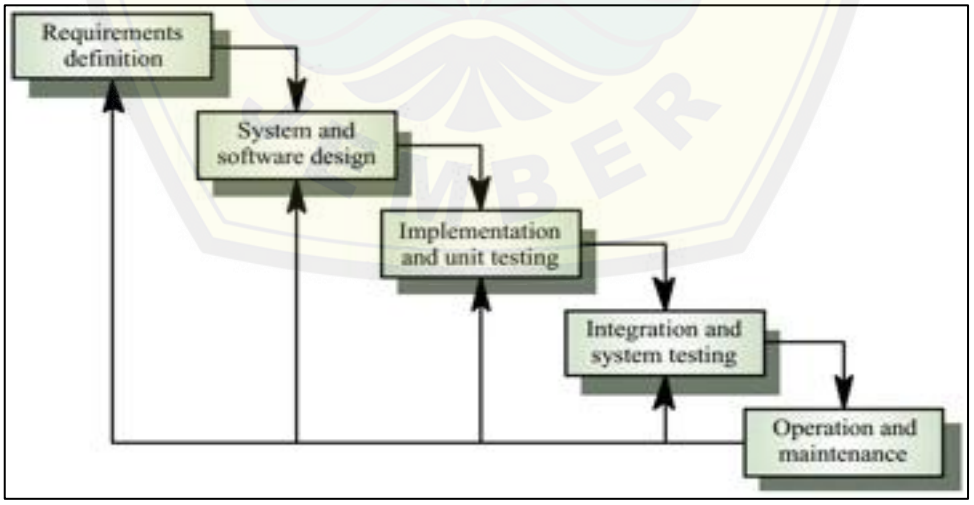
menganalisa studi kasus yang diteliti dan jenis penelitian kuantitatif digunakan karena dalam penelitian ini menerapkan serta mengkaji teori yang sudah ada sebelumnya.

3.2. Waktu dan Tempat

Penelitian ini dilakukan di Kelompok Tani Raja Mas Unit *Packing House* Desa Kandang Tepus, Kecamatan Senduro, Kabupaten Lumajang. Waktu dilaksanakannya penelitian adalah selama lima bulan yaitu ada bulan Oktober 2014 hingga Februari 2015.

3.3. Pengembangan Sistem

Metode yang akan penulis gunakan dalam melakukan pengembangan sistem informasi ini yaitu SDLC (*System Development Life Cycle*) dengan model proses *Waterfall*. Model *waterfall* merupakan metode yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem sampai pada anailisis, desain, kode, test dan pemeliharaan (Roger S. Pressman : 2002). Tahapan dari Paradigma model *Waterfall* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Paradigma *Waterfall Model*

Sumber : (Ian Sommerville, 2000)

3.3.1. Analisis Kebutuhan

Pada proses perancangan perangkat lunak ini tahap pertama yang perlu dilakukan adalah Analisis Kebutuhan. Tahap ini merumuskan solusi dari data dan permasalahan yang diambil dari berbagai sumber. Data dan permasalahan yang diambil diperoleh dari berbagai cara yaitu wawancara, studi literatur yang relevan dengan penelitian dan studi aplikasi atau perangkat lunak yang sejenis. Data primer yang akan diteliti adalah data berupa citra digital dari pisang dengan tingkat kematangan yang berbeda – beda yaitu muda, setengah matang, dan matang. Pengambilan *sample* citra digital yang akan diteliti, nantinya diperoleh dari hasil wawancara dan rekomendasi seorang ahli atau petani pisang yang sudah berpengalaman mengenai tingkat kematangan pisang yang sering dibutuhkan oleh konsumen untuk berbagai keperluan. Selain itu, untuk menunjang data primer yang diteliti, digunakan data sekunder yang diperoleh dari studi literatur yang relevan seperti buku, jurnal, artikel dan lain sebagainya.

3.3.2. Desain Sistem

Proses pembuatan desain sistem pada penelitian ini menggunakan *Unified Modeling Languange (UML)* yang dirancang menggunakan konsep *Object-Oriented Programming (OOP)*. Pemodelan UML yang digunakan adalah sebagai berikut :

1. *Use Case Diagram*

   Use Case Diagram merupakan model atau diagram yang digunakan untuk menggambarkan kebutuhan fungsional yang diharapkan dari suatu sistem. Umumnya *use case diagram* menekankan pada “siapa” melakukan “apa” dalam *enviroment* pada suatu sistem yang dibangun. *Use case diagram* digambarkan dari beberapa *actor, use-case,* dan interaksi diantara komponen – komponen tersebut yang dapat memberikan informasi dari suatu sistem yang akan dibangun.

2. *Use Case Scenario*

   Use Case Scenario merupakan deskripsi atau penjabaran alur kinerja (*step – step* dari tiap *use-case*) dari *use case diagram* yang telah dibuat. Umumnya *use case scenario* digambarkan dalam bentuk tabel yang dapat menggambarkan penjabaran alur kinerja dari tiap *use-case* yang ada.

3. *Activity Diagram*

   Activity Diagram merupakan model atau diagram yang menggambarkan aktivitas *(activity)* dari suatu sistem yang akan dibangun. Sehingga, dengan *activity diagram*, *process* dari sistem yang akan dibangun dapat diketahui dengan jelas berdasarkan aktivitasnya saat adanya suatu aksi atau *action* pada sistem.

4. *Sequence Diagram*

   Sequence Diagram merupakan model atau diagram yang menggambarkan interaksi antar objek yang mengindikasikan komunikasi diantara obyek - obyek tersebut di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut terdiri dari pengguna *(user)*, tampilan *(display)*, dan lain sebagainya berupa pesan *(message)*. Sehingga, dengan *squence diagram*, aliran logika dalam sebuah sistem dapat dimodelkan secara visual dalam bentuk diagram.

5. *Class Diagram*

   Class Diagram merupakan model statis yang menggambarkan struktur dan deskripsi class serta hubungannya antara class. *Class diagram* mirip dengan ERD pada perancangan *database*, bedanya pada ERD tidak terdapat operasi atau *method* tetapi hanya atribut saja. *Class Diagram* terdiri dari nama kelas, atribut dan operasi atau *method.*

3.3.3. Implementasi Sistem

Implementasi sistem merupakan tahap untuk mengimplementasikan atau mengubah desain sistem yang telah dibuat kedalam kode program. Tahap pertama

yang dilakukan dalam implementasi adalah penulisan kode program *(coding)* menggunakan bahasa pemrograman *Java*. Kemudian untuk tahap kedua adalah melakukan manajemen basis data menggunakan DBMS *Sqlite.*

3.4. Pengujian

Pada tahap pengujian ini dilakukan uji coba sistem yang telah dibuat dengan pengujian *white box* dan *black box*.

3.4.1. Pengujian *White Box*

White Box Testing merupakan cara pengujian dengan melihat modul yang telah dibuat dengan progam – progam yang ada. Menurut (Ayuliana, 2009) *White Box Testing* merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kasus - kasus uji. Dengan menggunakan metode pengujian *white box,* para pengembang *software (developer)* dapat menghasilkan kasus uji seperti berikut ini :

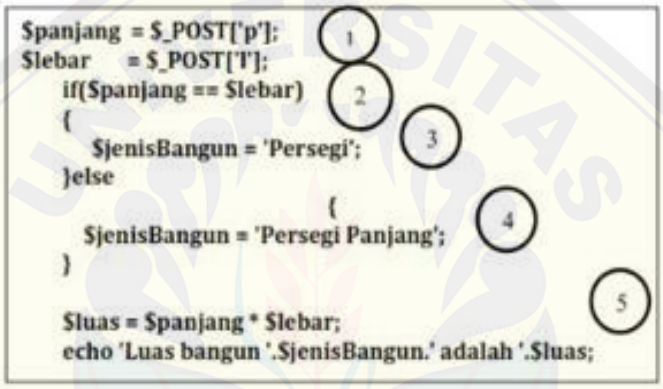
1. Menjamin bahwa seluruh *independent paths* dalam modul telah dilakukan sedikitnya satu kali.
2. Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah.
3. Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya.
4. Menguji struktur data internal untuk memastikan validitasnya.

Pada pengujian *white box* ini, aplikasi yang dibangun pada penelitian ini akan diuji menggunakan teknik pengujian berbasis alur *(basis path testing)* dimana kompleksitas dari aplikasi yang dibangun akan dihitung menggunakan *Cyclomatic Complexity*. Pengujian berbasis alur *(basis path testing)* merupakan teknik pengujian *white box* pertama yang diusulkan oleh Tom McCabe. Pengujian berbasis alur memungkinkan perancang kasus uji untuk menghasilkan ukuran kompleksitas logikal dari desain prosedural dan menggunakan ukuran ini untuk mendefinisikan himpunan basis dari alur eksekusi. Kasus uji dihasilkan untuk melakukan sekumpulan

basis yang dijamin untuk mengeksekusi setiap perintah dalam program, sedikitnya satu kali selama ujicoba (Ayuliana, 2009).

3.4.1.1. *Listing Program*

Listing Program merupakan baris - baris kode yang nantinya akan diuji. Setiap langkah dari kode - kode yang ada diberi nomor baik menjalankan *statement* biasa atau penggunaan kondisi dalam program. Contoh penerapan tahapan ini dapat dilihat pada Gambar 3.3 di bawah ini.

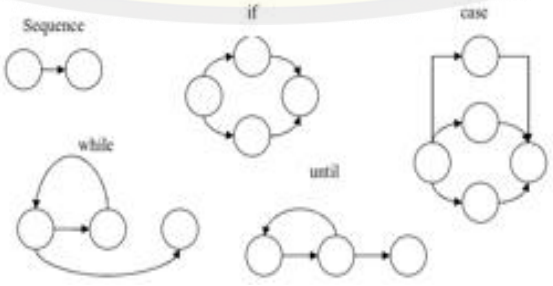


Gambar 3.3 Contoh *Listing Program*

Sumber : (Pressman, 2012)

3.4.1.2. Notasi Graf Alur *(Path Graph Notation)*

Graf Alur *(Flow Graph)* merupakan notasi sederhana untuk merepresentasikan alur kontrol, seperti Gambar 3.4 dibawah ini.

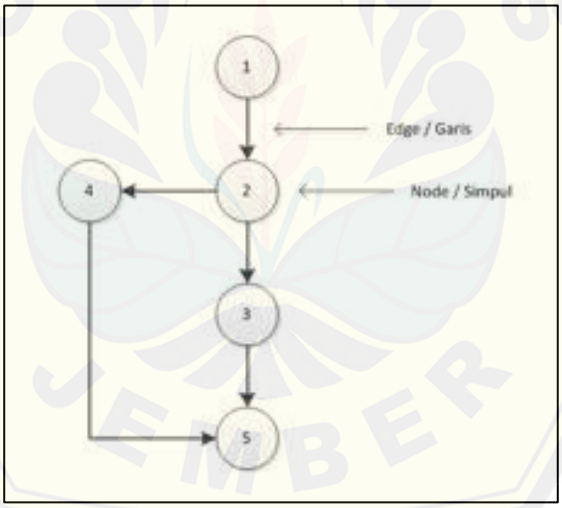


Gambar 3.4 Notasi *Flow Graph*

Sumber : (Pressman, 2012)

Urutan dari simbol proses dan simbol keputusan dapat digambarkan menjadi sebuah node, sedangkan anak panah disebut *edges*, menggambarkan aliran dari kontrol sesuai dengan diagram alir. Sebuah *edge* harus berakhir pada sebuah *node* walaupun tidak semua *node* merepresentasikan perintah prosedural. Area yang dibatasi oleh *edge* dan *node* disebut *region*, area diluar *graph* juga dihitung sebagai *region*.

Menurut Pressman (2012) grafik alir merupakan sebuah notasi sederhana yang digunakan untuk merepresentasikan aliran kontrol. Aliran kontrol yang digambarkan merupakan hasil penomoran dari listing program. Grafik alir digambarkan dengan *node* - *node* (simpul) yang dihubungkan dengan *edge* – *edge* (garis) yang menggambarkan alur jalannya program. Contoh penggambaran diagram alir dapat dilihat pada Gambar 3.5 di bawah ini.



Gambar 3.5 Contoh Diagram Alir

Sumber : (Pressman, 2012)

3.4.1.3. *Cyclomatic Complexity*

Cyclomatic complexity merupakan *software metric* yang menyediakan ukuran kuantitatif dari komplesitas logikal suatu program. Ketika digunakan dalam konteks metode pengujian berbasis alur, nilai yang dikomputasi untuk kompleksitas *cyclomatic* mendefinisikan jumlah *independent path* dalam himpunan basis suatu

program dan menyediakan batas atas untuk sejumlah ujicoba yang harus dilakukan untuk memastikan bahwa seluruh perintah telah dieksekusi sedikitnya satu kali (Ayuliana, 2009). Rumus yang digunakan untuk menghitung kompleksitas siklomatika yaitu :

$$V(G) = E - N + 2$$

Keterangan :

V(G) : Kompleksitas Siklomatik

E : Jumlah *Edge* / Garis

N : Jumlah *Node* / Simpul

Berdasarkan grafik alir yang ada pada Gambar 13, diketahui jumlah *edge* adalah 5 dan jumlah *node* adalah 5, sehingga dapat dihitung kompleksitas siklomatisk V(G) = E – N + 2 = 5 – 5 + 2 = 2. Jadi jumlah jalur independen adalah 2 jalur.

3.4.1.4. Jalur Program Independen

Jalur Program Independen atau *Independent path* adalah alur dari manapun dalam program yang memperkenalkan sedikitnya satu kumpulan perintah pemrosesan atau kondisi baru (Pressman, 2012). Bila dinyatakan dalam grafik alir, jalur independen harus bergerak setidaknya sepanjang satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisi (Pressman, 2012). Dari perhitungan kompleksitas siklomatik *Basis Set* pada Gambar 13 yang dihasilkan dari jalur independen secara linier adalah 2 jalur, yaitu :

Jalur / *Path* 1 : 1-2-3-5

Jalur / *Path* 2 : 1-2-4-5

3.4.1.5. Pengujian *Basis Set*

Pada bagian ini diberikan contoh data yang akan memaksa pelaksanaan jalur di *basis set*. Data yang dieksekusi dimasukkan ke dalam grafik alir apakah sudah melewati *basis set* yang tersedia. Sistem telah memenuhi syarat kelayakan software

jika salah satu jalur yang dieksekusi setidaknya satu kali. Dari tahap sebelumnya telah diketahui 2 *basis set* Jika kemudian diuji dengan memasukkan data panjang = 5 dan lebar = 3, maka *basis set* jalur yang digunakan adalah 1-2-4-5. Dapat dilihat bahwa jalur telah dieksekusi satu kali. Berdasarkan ketentuan tersebut dari segi kelayakan *software*, sistem ini telah memenuhi syarat.

3.4.2. Pengujian *Black Box*

Black Box Testing adalah metode pengujian perangkat lunak yang memeriksa fungsionalitas dari aplikasi yang berkaitan dengan struktur internal atau kerja. Pengetahuan khusus dari kode aplikasi atau struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Metode ini memfokuskan pada keperluan fungsionalitas dari *software* (Wildan Agissa, 2013).

Pada pengujian *black box* ini, aplikasi yang dibangun pada penelitian ini akan diuji dengan mengujikan langsung *running* aplikasi atau program dan melakukan kegiatan pengujian dengan menganalisis proses *input* dan *output* yang dihasilkan aplikasi. Adapun Tabel 3.1 pengujian *balack box* yang disusun sebagai berikut ini.

Tabel 3.1 Pengujian *Black Box*

(Hasil Analisis, 2014)

No	Menu	Fungsi	Kasus	Hasil	Ket.
...	...	...	...	...	...

Keterangan Tabel :

a. **No** **:** Digunakan untuk menuliskan nomor.
b. **Menu** **:** Menu aplikasi yang diujikan
c. **Fungsi** **:** Merupakan fungsi aplikasi yang diujikan.
d. **Kasus** **:** Rincian fitur yang diuji dari fungsi yang terdapat pada aplikasi.

e. **Hasil** : Hasil pengujian yang dilakukan

f. **Ket** **:** Digunakan untuk keterangan hasil diterima atau tidak.

Dalam melakukan kegiatan pengujian dengan menganalisis proses *input* dan *output* yang dihasilkan, dilakukan dengan cara menginputkan data normal dan data yang sengaja disalahkan, dari penginputan tersebut nantinya akan dilakukan analisis terhadap reaksi yang muncul pada saat *running* aplikasi atau program. Adapun contoh tabel pengujian untuk *event* yang terjadi ketika ada data masukan, dapat dilihat pada Tabel 3.2 dibawah ini.

Tabel 3.2 Pengujian *Black Box* Data Normal dan Salah

(Hasil Analisis, 2014)

No	Menu	Fungsi	Kasus	Hasil	Ket.
1.	*Training Data*	Untuk melihat *list dataset*.	Ketika *dataset* masih kosong.	Menampilkan pesan "Dataset Kosong, Silahkan Isi Dataset Terlebih Dahulu !"	OK.
		Untuk melakukan *training data*.	Ketika *user* menekan tombol *Training* tetapi *dataset* masih kosong.	Menampilkan pesan "Dataset Kosong, Silahkan Isi Dataset Terlebih Dahulu !".	OK.
			Ketika *user* menekan tombol *Training* tetapi jumlah *dataset* pada masing - masing kelas	Menampilkan pesan "Pastikan Jumlah Dataset Lebih Dari 3 Data !".	OK.

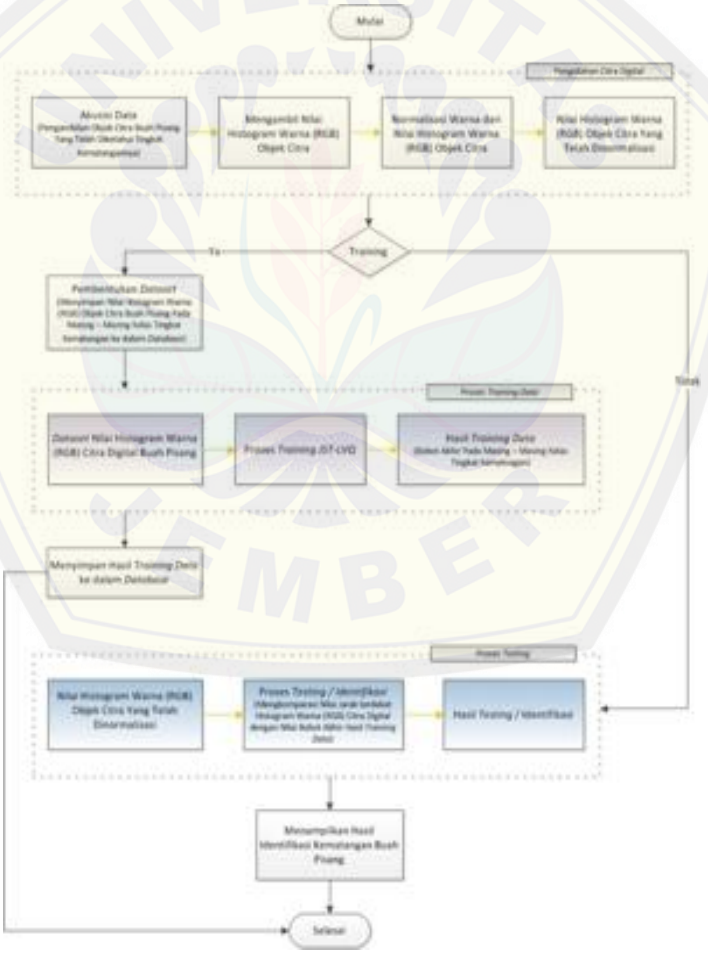
			ada tetapi jumlah data hanya 3.		
			Ketika *user* menekan tombol *Training* tetapi *dataset* pada salah satu kelas tidak ada.	Menampilkan pesan "Dataset Pada Masing - Masing Kelas Harus Ada !".	OK.
			Ketika menekan tombol *Training*.	Menampilkan *dataset* kelas matang pada halaman *Initial Vartrain CW1*.	OK.

3.5. Gambaran Sistem

Banana Maturity Identification merupakan aplikasi atau perangkat lunak *mobile* berbasis *android* yang digunakan untuk mengidentifikasi kematangan pada buah pisang dengan memanfaatkan pengolahan citra digital dan *datamining*. Aplikasi ini dapat melakukan identifikasi kematangan buah pisang berdasarkan citra digital yang didapat atau diakusisi. Pada proses akusisi citra digital, aplikasi ini dapat meng-*capture* buah pisang menggunakan *camera* dan menginputkan citra digital buah pisang dari *sd card* atau media penyimpanan pada gadget yang digunakan. Aplikasi ini dapat menampilkan dan mengambil nilai histogram warna (RGB) citra digital dari objek yang dinputkan ke dalam aplikasi. Nilai histogram warna (RGB) citra digital ini yang nantinya dinormalisasi dan kemudian disimpan ke dalam *database* berdasarkan *class* tingkat kematangan buah pisang sebagai *dataset* pada proses *training data*.

Pada proses *training data*, aplikasi ini menggunakan metode *Learning Vector Quantization (LVQ)*. Hasil dari klasifikasi pada proses *training data* adalah nilai

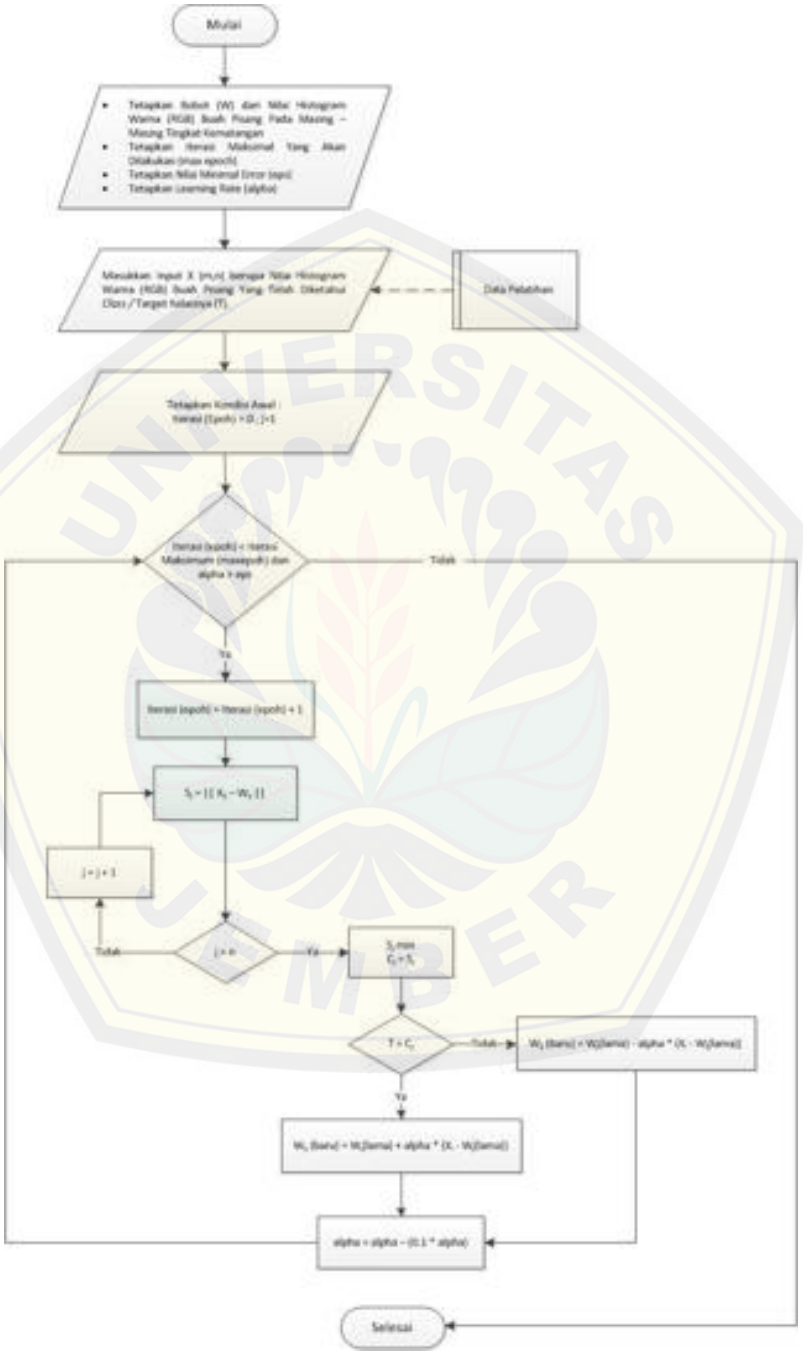
bobot akhir tiap masing – masing *class* tingkat kematangan buah pisang, dimana bobot - bobot tersebut disimpan ke dalam *database* sebagai acuan pada proses identifikasi kematangan buah pisang. Bobot – bobot akhir tersebut merupakan nilai histogram warna (RGB) citra digital tiap masing – masing *class* tingkat kematangan buah pisang yang telah diolah. Pengidentifikasian kematangan buah pisang yang dilakukan aplikasi didapat dengan cara mengkomparasi nilai jarak terdekat histogram warna (RGB) dari citra digital yang tidak diketahui *class* tingkat kematangannya dengan nilai bobot akhir yang ada pada *database*. Diagram alir dari gambaran sistem dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram Alir Gambaran Sistem

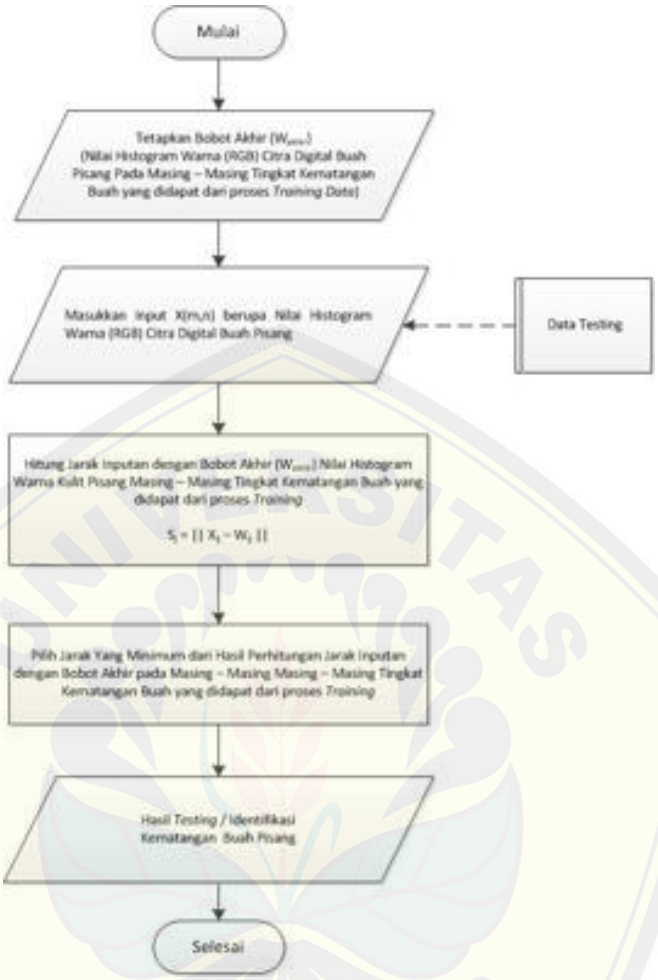
Sumber : (Hasil Analisis, 2014)

Adapun *flowchart* algoritma pelatihan *JST-LVQ* yang telah disesuaikan dengan kebutuhan penelitian dapat dilihat pada Gambar 3.7 dan *testing* pada Gambar 3.8.



Gambar 3.7 *Flowchart* Algoritma Pelatihan JST *Learning Vector Quantization (LVQ)*

Sumber : (Hasil Analisis, 2014)



Gambar 3.8 *Flowchart* Algoritma *Testing* JST *Learning Vector Quantization (LVQ)*

Sumber : (Hasil Analisis, 2014)

BAB 4. DESAIN DAN PERANCANGAN

Bab ini akan menguraikan tentang proses perancangan untuk mengimplementasikan algoritma *Learning Vector Quantization* untuk mengidentifikasi kematangan buah pisang menggunakan android *smartphone*. Proses perancangan sistem dimulai dari analisis kebutuhan fungsional dan non-fungsional sistem, kemudian dilanjutkan dengan pembuatan *usecase diagram*, skenario, *activity diagram, sequence diagram, class diagram* dan *entity relation diagram (ERD).*

4.1. Analisis Kebutuhan Perangkat Lunak

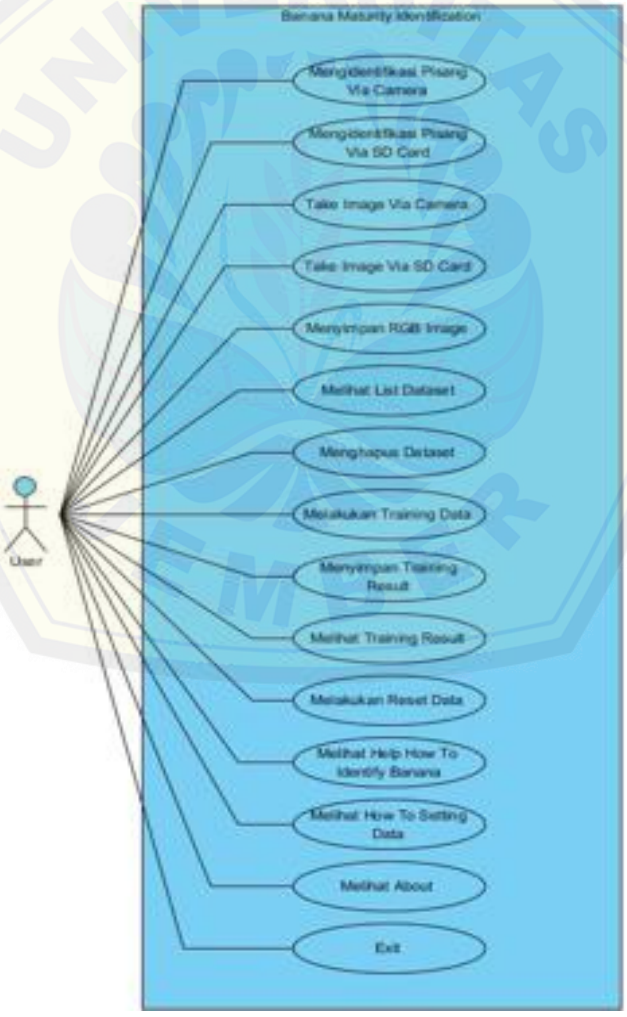
Analisis kebutuhan perangkat lunak dalam penelitian ini yaitu dengan cara mengidentifikasi permasalahan yang ada untuk kemudian dicatat dan dijadikan bahan untuk mulai membangun aplikasi untuk mengidentifikasi kematangan buah pisang berbasis android. Analisis kebutuhan yang dilakukan meliputi proses pengumpulan data kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional sistem pada penelitian ini adalah sebagai berikut :

1. Sistem dapat mengolah citra digital yang diinputkan melalui kamera atau media penyimpanan yang ada pada *smartphone*.
2. Sistem dapat menampilkan nilai histogram *red, green, blue* (RGB) dari citra digital yang telah diolah.
3. Sistem dapat menyimpan dan menghapus *dataset* berupa nilai histogram *red, green, blue* (RGB) dari citra digital.
4. Sistem dapat melakukan *training data* dari *dataset* yang telah dinputkan menggunakan metode *Learning Vector Quantization* dan menyimpan hasilnya.
5. Sistem dapat mengidentifikasi kematangan buah pisang dari citra digital yang diinputkan melalui kamera atau media penyimpanan yang ada pada *smartphone.*
6. Sistem dapat melakukan *reset* data.

Sedangkan kebutuhan non-fungsional sistem pada penelitian ini adalah tampilan aplikasi yang *user friendly*, sehingga pengguna tidak kesulitan dalam mengoperasikannya.

4.2. *Usecase Diagram*

Usecase Diagram berfungsi untuk menggambarkan fitur apa saja yang akan dijalankan pada aplikasi untuk mengidentifikasi kematangan buah pisang dengan mengimplementasikan algoritma *Learning Vector Quantization* di dalamnya. *Usecase* dapat dilihat pada Gambar 4.1.



Gambar 4.1 ***Usecase Banana Maturity Identification***

Definisi *usecase* pada *Usecase Banana Maturity Identification* dapat dilihat pada Tabel 4.1, sedangkan untuk definisi *actor* yang ada pada *usecase Banana Maturity Identification* dapat dilihat pada Tabel 4.2.

Tabel 4.1 Definisi *Usecase Banana Maturity Identification*

No.	*Usecase*	Deskripsi
1	Mengidentifikasi Pisang *Via Camera*	Proses untuk melakukan identifikasi kematangan buah pisang dimana citra digital diperoleh dari *Capture Camera* dan menampilkan hasil dari proses identifikasi tersebut.
2	Mengidentifikasi Pisang *Via SD Card*	Proses untuk melakukan identifikasi kematangan buah pisang dimana citra digital diperoleh dari *SD Card* dan menampilkan hasil dari proses identifikasi tersebut.
3	*Take Image Via Camera*	Proses mengambil citra digital untuk melihat histogram RGB *image* atau menambah *dataset* menggunakan *Camera*.
4	*Take Image Via SD Card*	Proses mengambil citra digital yang diperoleh dari *SD Card* untuk melihat histogram RGB *image* atau menambah *dataset.*
5	Menyimpan RGB *Image*	Menyimpan RGB citra digital sebagai *dataset* ke dalam *database*.
6	Melihat *List Dataset*	Menampilkan *dataset* yang ada pada *database.*
7	Menghapus *Dataset*	Menghapus *dataset* dari *list dataset* yang dipilih.
8	Melakukan *Training Data*	Proses *training data* atau mengolah *dataset* yang ada pada *database.*

9	Menyimpan *Training Result*	Menyimpan hasil *training data* ke dalam *database.*
10	Melihat *Training Result*	Menampilkan hasil *training data* yang telah disimpan ke dalam *database*.
11	Melakukan *Reset Data*	Proses *reset* atau mengatur ulang data ke pengaturan awal aplikasi.
12	Melihat *Help How To Identify Banana*	Menampilkan cara menggunakan fitur *Identify Banana*.
13	Melihat *Help How To Setting Data*	Menampilkan cara menggunakan fitur *Setting Data*.
14	Melihat *About*	Menampilkan informasi tentang aplikasi.
15	*Exit*	Proses untuk mengakhiri aplikasi.

Tabel 4.2 Definisi *Actor Banana Maturity Identification*

No.	*Usecase*	Deskripsi
1	*User*	Petani Pisang atau Penyeleksi Pisang yang mengoperasikan aplikasi *Banana Maturity Identification* untuk melakukan identifikasi kematangan buah pisang berdasarkan warna kulit pisang.

4.3. Skenario

Skenario berfungsi untuk menggambarkan alur sistem beserta alternatif yang akan dijalankan oleh *user* pada aplikasi *Banana Maturity Identification*. Skenario sistem ditunjukkan pada Tabel 4.3 sampai 4.17.

4.3.1. Skenario Mengidentifikasi Pisang *Via Camera*

Tabel 4.3 Skenario Mengidentifikasi Pisang *Via Camera*

Name	Mengidentifikasi Pisang *Via Camera.*
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Home* dan *User* menginputkan citra digital untuk melakukan identifikasi pada buah pisang menggunakan *Camera.*
Exit Condition	Aplikasi menampilkan hasil identifikasi kematangan buah pisang.
Skenario Utama	
Actor	***System***
1. Menekan menu *Identify Banana.*	
	2. Menampilkan *Dialog Box* yang menampilkan pilihan media yaitu *Camera* atau *SD Card* untuk mengambil citra digital beserta tombol *Cancel.*
3. Memilih media *Camera* untuk mengambil citra digital.	
	4. Menampilkan kamera beserta tombol *Capture* dan membuat folder *BMI-Capture.*
5. Menekan tombol *Capture.*	
	6. Mengambil citra dari *camera* dan menyimpan ke *directory folder* yang telah disediakan oleh aplikasi.
	7. Mengolah *scale* citra digital menjadi 200 x 300 pixel.

	8. Menampilkan citra digital hasil *capture* ke halaman *Identify Banana* beserta tombol *Back* dan tombol *Identify*.
9. Menekan tombol *Identify*.	
	10. Mengambil nilai *RGB* citra digital yang telah di *scale* dan melakukan pengidentifikasian data, serta menampilkan *Progress Dialog* "Tunggu Sebentar..." selama aplikasi mengolah data.
	11. Menampilkan hasil identifikasi ke halaman *Identify Banana Result* beserta tombol *Back* dan tombol *Go Home*.
Skenario Alternatif	
Jika *smartphone* memiliki beberapa aplikasi untuk melakukan pengambilan citra.	
	4. Menampilkan *dialog* pilihan beberapa media *camera* yang akan digunakan untuk melakukan pengambilan citra digital.
5. Memilih salah satu pilihan media *camera* yang akan digunakan.	
	6. Kembali ke skenario utama No.4
Jika Menekan tombol *Cancel* pada *Dialog Box* pilihan media.	
3. Menekan tombol *Cancel.*	
	4. Kembali ke halaman *Home.*
Jika Memilih media *Camera* tetapi folder *BMI-Capture* sudah dibuat.	
3. Memilih media *Camera*	

untuk mengambil citra digital.	
	4. Menampilkan kamera beserta tombol *Capture* dan tidak membuat folder *BMI-Capture*.
Jika *Cancel Capture* pada saat kamera di tampilkan	
5. Menekan tombol *Cancel.*	
	6. Menghapus *temp file*.
	7. Kembali ke halaman *Home.*
Jika Menekan tombol *Back* pada halaman *Identify Banana.*	
9. Menekan tombol *Back.*	
	10. Kembali ke halaman *Home.*
Jika Menekan tombol *Back* pada halaman *Identify Banana Result*.	
12. Menekan tombol *Back.*	
	13. Kembali ke halaman *Identify Banana.*
Jika Menekan tombol *Go Home* pada halaman *Identify Banana Result*.	
12. Menekan tombol *Go Home*.	
	13. Menampilkan halaman *Home*.

4.3.2. Skenario Mengidentifikasi Pisang *Via SD Card*

Tabel 4.4 Skenario Mengidentifikasi Pisang *Via SD Card*

Name	Mengidentifikasi Pisang *Via SD Card.*
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Home* dan *User* menginputkan citra digital untuk melakukan identifikasi pada buah pisang dari *SD Card.*
Exit Condition	Aplikasi menampilkan hasil identifikasi kematangan buah

	pisang.
Skenario Utama	
Actor	***System***
1. Menekan menu *Identify Banana*.	
	2. Menampilkan *Dialog Box* yang menampilkan pilihan media yaitu *Camera* atau *SD Card* untuk mengambil citra digital beserta tombol *Cancel.*
3. Memilih media *SD Card* untuk mengambil citra digital.	
	4. Menampilkan *dialog* pilihan beberapa aplikasi *file manager* untuk mengakses media penyimpanan yang akan digunakan untuk melakukan pengambilan citra digital.
5. Memilih salah satu aplikasi *file manager* yang akan digunakan.	
	6. Membuka aplikasi *file manager* yang dipilih untuk mengakses media penyimpanan yang telah dipilih.
7. Memilih salah satu citra digital yang akan diinputkan.	
	8. Menge-*load* citra digital yang telah

	dipilih.
	9. Mengolah *scale* citra digital menjadi 200 x 300 pixel.
	10. Menampilkan citra digital hasil *load* dari media penyimpanan ke halaman *Identify Banana* beserta tombol *Back* dan tombol *Identify*.
11. Menekan tombol *Identify*.	
	12. Mengambil nilai *RGB* citra digital yang telah di *scale* dan melakukan pengidentifikasian data, serta menampilkan *Progress Dialog* "Tunggu Sebentar..." selama aplikasi mengolah data.
	13. Menampilkan hasil identifikasi ke halaman *Identify Banana Result* beserta tombol *Back* dan tombol *Go Home*.
Skenario Alternatif	
Jika Menekan tombol *Cancel* pada *Dialog Box* pilihan media.	
3. Menekan tombol *Cancel.*	
	4. Kembali ke halaman *Home.*
Jika *Cancel* pilih aplikasi *file manager.*	
5. Menekan tombol *Cancel.*	
	6. Kembali ke halaman *Home.*
Jika *Cancel* memilih citra digital yang akan di inputkan.	
7. Menekan tombol *Cancel.*	
	8. Kembali ke halaman *Home.*

Jika Menekan tombol *Back* pada halaman *Identify Banana.*	
11. Menekan tombol *Back.*	
	12. Kembali ke halaman *Home.*
Jika Menekan tombol *Back* pada halaman *Identify Banana Result*.	
14. Menekan tombol *Back.*	
	15. Kembali ke halaman *Identify Banana.*
Jika Menekan tombol *Go Home* pada halaman *Identify Banana Result*.	
14. Menekan tombol *Go Home*.	
	15. Menampilkan halaman *Home.*

4.3.3. Skenario *Take Image Via Camera*

Tabel 4.5 Skenario *Take Image Via Camera*

Name	*Take Image Via Camera.*
Actor	*User.*
Entry Condition	Aplikasi menampilkan halaman *Home* dan *User* melakukan pengambilan citra digital menggunakan *Camera.*
Exit Condition	Menampilkan hasil pengambilan citra yang telah di *scale.*
Skenario Utama	
Actor	***System***
1. Menekan menu *Setting Data.*	
	2. Menampilkan halaman *Setting Data* beserta submenu *Take Image, Setting Data, Training Result* dan *Go Home.*
3. Menekan submenu *Take Image.*	

	4. Menampilkan *Dialog Box* yang menampilkan pilihan media yaitu *Camera* atau *SD Card* untuk mengambil citra digital.
5. Memilih media *Camera* untuk mengambil citra digital.	
	6. Menampilkan kamera dan tombol *Capture* dan membuat folder *BMI-Capture*.
7. Menekan tombol *Capture*.	
	8. Mengambil citra dari kamera dan menyimpan ke *directory folder* yang telah disediakan oleh aplikasi.
	9. Mengolah *scale* citra digital menjadi 200 x 300 pixel.
	10. Menampilkan citra digital hasil *capture* atau *load* dari media penyimpanan ke halaman *Take Image* beserta tombol *Back* dan tombol *Get RGB*.
Skenario Alternatif	
Jika *Smartphone* memiliki beberapa aplikasi untuk melakukan pengambilan citra.	
	6. Menampilkan *dialog* pilihan beberapa media *camera* yang akan digunakan untuk melakukan pengambilan citra digital.

7. Memilih salah satu pilihan media *camera* yang akan digunakan.	
	8. Kembali ke skenario utama No.6
Jika Menekan tombol *Cancel* pada *Dialog Box* pilihan media.	
5. Menekan tombol *Cancel.*	
	6. Kembali ke halaman *Setting Data*.
Jika Memilih media *Camera* tetapi folder *BMI-Capture* sudah dibuat.	
5. Memilih media *Camera* untuk mengambil citra digital	
	6. Menampilkan kamera beserta tombol *Capture* dan tidak membuat folder *BMI-Capture*.
Jika *Cancel Capture* pada saat kamera di tampilkan.	
7. Menekan tombol *Cancel.*	
	8. Menghapus *temp file*.
	9. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *Back* pada halaman *Take Image.*	
11. Menekan tombol *Back*.	
	12. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *onBackPressed* pada halaman *Take Image* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
11. Menekan tombol *onBackPressed.*	
	12. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.

13. Menekan tombol *Yes.*	
	14. Menampilkan halaman *Home.*
Jika Menekan tombol o*nBackPressed* pada halaman *Take Image* dan Memilih tombol *No* pada *Dialog Confirm.*	
11. Menekan tombol *onBackPressed.*	
	12. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
13. Menekan tombol *No.*	
	14. Kembali ke halaman *Take Image.*

4.3.4. Skenario *Take Image Via SD Card*

Tabel 4.6 Skenario *Take Image Via SD Card*

Name	*Take Image Via SD Card.*
Actor	*User.*
Entry Condition	Aplikasi menampilkan halaman *Home* dan *User* melakukan pengambilan citra digital dari *SD Card.*
Exit Condition	Menampilkan hasil pengambilan citra yang telah di *scale.*
Skenario Utama	
Actor	***System***
1. Menekan menu *Setting Data.*	
	2. Menampilkan halaman *Setting Data* beserta submenu *Take Image, Setting Data, Training Result* dan *Go Home.*
3. Menekan submenu *Take*	

Image.	
	4. Menampilkan *Dialog Box* yang menampilkan pilihan media yaitu *Camera* atau *SD Card* untuk mengambil citra digital.
5. Memilih media *SD Card* untuk mengambil citra digital.	
	6. Menampilkan *dialog* pilihan beberapa aplikasi *file manager* untuk mengakses media penyimpanan yang akan digunakan untuk melakukan pengambilan citra digital.
7. Memilih salah satu aplikasi *file manager* yang akan digunakan.	
	8. Membuka aplikasi *file manager* yang dipilih untuk mengakses media penyimpanan yang telah dipilih.
9. Memilih citra digital yang akan diinputkan.	
	10. Mengolah *scale* citra digital menjadi 200 x 300 pixel.
	11. Menampilkan citra digital hasil *load* dari media penyimpanan ke halaman *Take Image* beserta tombol *Back* dan tombol *Get RGB.*
Skenario Alternatif	

Jika Menekan tombol *Cancel* pada *Dialog Box* pilihan media.	
5. Menekan tombol *Cancel.*	
	6. Kembali ke halaman *Setting Data*.
Jika *Cancel* pilih aplikasi *file manager.*	
7. Menekan tombol *Cancel.*	
	8. Kembali ke halaman *Home.*
Jika *Cancel* memilih citra digital yang akan di inputkan.	
9. Menekan tombol *Cancel.*	
	10. Kembali ke halaman *Home.*
Jika Menekan tombol *Back* pada halaman *Take Image.*	
12. Menekan tombol *Back.*	
	13. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *onBackPressed* pada halaman *Take Image* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
12. Menekan tombol *onBackPressed.*	
	13. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
14. Menekan tombol *Yes.*	
	15. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Take Image* dan Memilih tombol *No* pada *Dialog Confirm.*	
12. Menekan tombol *onBackPressed.*	
	13. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.

14. Menekan tombol *No*.	
	15. Kembali ke halaman *Take Image*.

4.3.5. Skenario Menyimpan RGB *Image*

Tabel 4.7 Skenario Menyimpan RGB *Image*

Name	Menyimpan RGB *Image*.
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Take Image* dan *User* sudah melakukan pengambilan citra digital.
Exit Condition	Menampilkan histogram RGB dari citra digital dan menyimpan nilai RGB dari citra digital ke dalam *database* sebagai *dataset* untuk proses *training data*.
Skenario Utama	
Actor	***System***
1. Menekan tombol *Get RGB*.	
	2. Mengambil nilai *RGB* citra digital yang telah di *scale* dan juga menampilkan *Progress Dialog* "Tunggu Sebentar..." selama aplikasi sedang mengolah data.
	3. Menampilkan histogram *RGB* citra digital, normalisasi *RGB* citra digital, dan *spinner* kelas untuk memilih kelas target ke halaman Histogram RGB *Image* beserta tombol *Back* dan tombol *Save*.
4. Memilih target kelas citra	

digital pada *spinner* kelas.	
	5. Menampilkan target kelas yang dipilih pada *spinner* kelas.
6. Menekan tombol *Save*.	
	7. Menyimpan nilai RGB citra digital sebagai *dataset* ke dalam *database.*
	8. Menampilkan *dataset* yang telah disimpan ke pada halaman *Training Data* dan menampilkan Pesan "Data Berhasil Disimpan !"
Skenario Alternatif	
Jika Menekan tombol *Back* pada halaman Histogram RGB *Image.*	
4. Menekan tombol *Back.*	
	5. Kembali ke halaman *Take Image.*
Jika Menekan tombol *Save* pada halaman Histogram RGB *Image* tetapi belum memilih target kelas.	
4. Menekan tombol *Save*.	
	5. Menampilkan Pesan "Pilih Kelas Dulu !".
Jika Menekan tombol *onBackPressed* pada halaman Histogram RGB *Image* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
4. Menekan tombol *onBackPressed.*	
	5. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
6. Menekan tombol *Yes.*	
	7. Menampilkan halaman *Home.*

Jika Menekan tombol *onBackPressed* pada halaman Histogram RGB *Image* dan Memilih tombol *No* pada *Dialog Confirm*.	
4. Menekan tombol *onBackPressed.*	
	5. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
6. Menekan tombol *No.*	
	7. Kembali ke halaman *Histogram* RGB *Image.*

4.3.6. Skenario Melihat *List Dataset*

Tabel 4.8 Skenario Melihat *List Dataset*

Name	Melihat *List Dataset.*
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Home.*
Exit Condition	Aplikasi menampilkan *list dataset* yang ada pada *database*.
Skenario Utama	
Actor	***System***
1. Menekan menu *Setting Data.*	
	2. Menampilkan halaman *Setting Data* beserta submenu *Take Image, Setting Data, Training Result* dan *Go Home.*
3. Menekan submenu *Training Data.*	
	4. Menampilkan *list dataset* yang ada

	di *database* pada halaman *Training Data* beserta tombol *Delete*, tombol *Back* dan tombol *Training*.
Skenario Alternatif	
Jika Menekan submenu *Trainng Data* tetapi *dataset* kosong.	
3. Menekan submenu *Training Data.*	
	4. Menampilkan Pesan "Dataset Kosong, Silahkan Isi Dataset Terlebih Dahulu !".
Jika Menekan tombol *Back* pada halaman *Training Data.*	
5. Menekan tombol *Back.*	
	6. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *onBackPressed* pada halaman *Training Data* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
5. Menekan tombol *onBackPressed.*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *Yes.*	
	8. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Training Data* dan Memilih tombol *No* pada *Dialog Confirm.*	
5. Menekan tombol *onBackPressed.*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home

	?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *No.*	
	8. Kembali ke halaman *Training Data.*

4.3.7. Skenario Menghapus *Dataset*

Tabel 4.9 Skenario Menghapus *Dataset*

Name	Menghapus *Dataset.*
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Training Data* dan *User* memilih salah satu *dataset* yang akan dihapus pada *list dataset*.
Exit Condition	Aplikasi menghapus *dataset* yang dipilih oleh *user* dari *database*.
Skenario Utama	
Actor	***System***
1. Menekan tombol *Delete* berupa *icon trash* pada *dataset* yang akan dihapus.	
	2. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Menghapus Data Ini ?" beserta tombol *Yes* dan *No*.
3. Menekan tombol *Yes.*	
	4. Menghapus *dataset* yang telah dipilih dari *database*.
	5. Menampilkan halaman *Training Data*.
Skenario Alternatif	

Jika Menekan tombol *Delete* pada *list dataset* dan Memilih tombol *Yes* pada *Dialog Confirm* tetapi *Dataset* telah dijadikan bobot kelas matang.	
3. Menekan tombol *Yes.*	
	4. Mengupdate *training result* dengan nilai *default* pada kelas matang.
	5. Kembali ke skenario utama No.4
Jika Menekan tombol *Delete* pada *list dataset* dan Memilih tombol *Yes* pada *Dialog Confirm* tetapi *Dataset* telah dijadikan bobot kelas setengah matang.	
3. Menekan tombol *Yes.*	
	4. Mengupdate *training result* dengan nilai *default* pada kelas setengah matang.
	5. Kembali ke skenario utama No.4
Jika Menekan tombol *Delete* pada *list dataset* dan Memilih tombol *Yes* pada *Dialog Confirm* tetapi *Dataset* telah dijadikan bobot kelas muda.	
3. Menekan tombol *Yes.*	
	4. Mengupdate *training result* dengan nilai *default* pada kelas muda.
	5. Kembali ke skenario utama No.4
Jika Menekan tombol *Delete* pada *list dataset* dan Memilih tombol *No* pada *Dialog Confirm*.	
6. Menekan tombol *No.*	
	7. Kembali ke halaman *Training Data.*
Jika Menekan tombol *Back* pada halaman *Training Data.*	
6. Menekan tombol *Back.*	

	7. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *onBackPressed* pada halaman *Training Data* dan Memilih tombol *Yes* pada *Dialog Confirm*.	
6. Menekan tombol *onBackPressed*.	
	7. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
8. Menekan tombol *Yes*.	
	9. Menampilkan halaman *Home*.
Jika Menekan tombol *onBackPressed* pada halaman *Training Data* dan Memilih tombol *No* pada *Dialog Confirm*.	
6. Menekan tombol *onBackPressed*.	
	7. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
8. Menekan tombol *No*.	
	9. Kembali ke halaman *Training Data*.

4.3.8. Skenario Melakukan *Training Data*

Tabel 4.10 Skenario Melakukan *Training Data*

Name	Melihat Melakukan *Training Data*.
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Training Data*.
Exit Condition	Aplikasi menampilkan *training result* pada halaman *Training Result*.

Skenario Utama	
Actor	***System***
1. Menekan tombol *Training.*	
	2. Menampilkan *list dataset* kelas matang pada halaman *Initial Initial Vartrain CW1* beserta tombol *checkbox*, tombol *Back* dan tombol *Next*.
3. Menekan tombol *Checkbox* pada *dataset* yang dipilih.	
	4. Mengambil *dataset* yang dipilih sebagai bobot awal pada kelas matang, serta menampilkan tombol *Back* dan tombol *Next*.
5. Menekan tombol *Next.*	
	6. Menampilkan *list dataset* kelas setengah matang pada halaman *Initial Vartrain CW2* beserta tombol *checkbox*, tombol *Back* dan tombol *Next*.
7. Menekan tombol *Checkbox* pada *dataset* yang dipilih.	
	8. Mengambil *dataset* yang dipilih sebagai bobot awal pada kelas setengah matang, serta menampilkan tombol *Back* dan tombol *Next*.
9. Menekan tombol *Next.*	
	10. Menampilkan *list dataset* kelas muda

	pada halaman *Initial Vartrain CW3* beserta tombol *checkbox*, tombol Back dan tombol Next.
11. Menekan tombol *Checkbox* pada *dataset* yang dipilih.	
	12. Mengambil *dataset* yang dipilih sebagai bobot awal pada kelas muda, serta menampilkan tombol *Back* dan tombol *Next*.
13. Menekan tombol *Next.*	
	14. Menampilkan *learning rate, spinner max epoch,* dan *eps* pada halaman *Initial Vartrain LREPSME* beserta tombol *Back* dan tombol *Process*.
15. Memilih *max epoch* pada *spinner max epoch*.	
	16. Menampilkan *max epoch* yang dipilih pada *spinner*.
17. Menekan tombol *Process.*	
	18. Mengolah *dataset* untuk proses *training* menggunakan metode *Learning Vector Quantization* dan juga menampilkan *Progress Dialog* "Proses Training..." selama aplikasi sedang mengolah data.
	19. Menampilkan hasil *training data* pada halaman *Training Result* beserta tombol *Back* dan tombol

	Save serta Pesan "Selesai !".
Skenario Alternatif	
Jika Menekan tombol *Training* tetapi *dataset* kosong.	
1. Menekan tombol *Training.*	
	2. Menampilkan Pesan "Dataset Kosong, Silahkan Isi Dataset Terlebih Dahulu !".
Jika Menekan tombol *Training* tetapi jumlah *dataset* pada masing - masing kelas ada tetapi jumlah data hanya 3.	
1. Menekan tombol *Training.*	
	2. Menampilkan Pesan "Pastikan Jumlah Dataset Lebih Dari 3 Data !".
Jika Menekan tombol *Training* tetapi *dataset* pada salah satu kelas tidak ada.	
1. Menekan tombol *Training.*	
	2. Menampilkan Pesan "Dataset Pada Masing – Masing Kelas Harus Ada !".
Jika Menekan *Next* pada halaman *Initial Initial Vartrain CW1* tetapi belum memilih salah satu *dataset* sebagai bobot awal.	
5. Menekan tombol *Next*.	
	6. Menampilkan Pesan "Pilih Bobot Dulu !".
Jika Menekan tombol *Back* pada halaman *Initial Initial Vartrain CW1*.	
3. Menekan tombol *Back.*	
	4. Kembali ke halaman *Training Data.*
Jika Menekan tombol o*nBackPressed* pada halaman *Initial Initial Vartrain CW1* dan Memilih tombol *Yes* pada *Dialog Confirm.*	

3. Menekan tombol *onBackPressed*.	
	4. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
5. Menekan tombol *Yes.*	
	6. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Initial Initial Vartrain CW1* dan Memilih tombol *No* pada *Dialog Confirm.*	
3. Menekan tombol *onBackPressed.*	
	4. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
5. Menekan tombol *No.*	
	6. Kembali ke halaman *Initial Initial Vartrain CW1*.
Jika Menekan *Next* pada halaman *Initial Vartrain CW2* tetapi belum memilih salah satu *dataset* sebagai bobot awal.	
9. Menekan tombol *Next*.	
	10. Menampilkan Pesan "Pilih Bobot Dulu !".
Jika Menekan tombol *Back* pada halaman *Initial Vartrain CW2.*	
7. Menekan tombol *Back.*	
	8. Kembali ke halaman *Initial Initial Vartrain CW1*.
Jika Menekan tombol o*nBackPressed* pada halaman *Initial Vartrain CW2* dan Memilih tombol *Yes* pada *Dialog Confirm.*	

7. Menekan tombol *onBackPressed*.	
	8. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
9. Menekan tombol *Yes.*	
	10. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Initial Vartrain CW2* dan Memilih tombol *No* pada *Dialog Confirm*.	
7. Menekan tombol *onBackPressed.*	
	8. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
9. Menekan tombol *No.*	
	10. Kembali ke halaman *Initial Vartrain CW2.*
Jika Menekan *Next* pada halaman *Initial Vartrain CW3* tetapi belum memilih salah satu *dataset* sebagai bobot awal.	
13. Menekan tombol *Next*.	
	14. Menampilkan Pesan "Pilih Bobot Dulu !".
Jika Menekan tombol *Back* pada halaman *Initial Vartrain CW3.*	
11. Menekan tombol *Back.*	
	12. Kembali ke halaman *Initial Vartrain CW2.*
Jika Menekan tombol o*nBackPressed* pada halaman *Initial Vartrain CW3* dan Memilih tombol *Yes* pada *Dialog Confirm*.	

11. Menekan tombol *onBackPressed*.	
	12. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
13. Menekan tombol *Yes.*	
	14. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Initial Vartrain CW3* dan Memilih tombol *No* pada *Dialog Confirm*.	
11. Menekan tombol *onBackPressed.*	
	12. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
13. Menekan tombol *No.*	
	14. Kembali ke halaman *Initial Vartrain CW3.*
Jika Menekan *Process* pada halaman *Initial Vartrain LREPSME* tetapi belum memilih *max epoch*.	
17. Menekan tombol *Process*.	
	18. Menampilkan Pesan "Pilih Max Epoch Dulu !".
Jika Menekan tombol *Back* pada halaman *Initial Vartrain LREPSME*.	
15. Menekan tombol *Back.*	
	16. Kembali ke halaman *Initial Vartrain CW3.*
Jika Menekan tombol o*nBackPressed* pada halaman *Initial Vartrain LREPSME* dan Memilih tombol *Yes* pada *Dialog Confirm*.	

15. Menekan tombol *onBackPressed*.	
	16. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
17. Menekan tombol *Yes.*	
	18. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Initial Vartrain LREPSME* dan Memilih tombol *No* pada *Dialog Confirm.*	
15. Menekan tombol *onBackPressed.*	
	16. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
17. Menekan tombol *No.*	
	18. Kembali ke halaman *Initial Vartrain LREPSME.*
Jika Menekan tombol *Back* pada halaman *Training Result.*	
20. Menekan tombol *Back.*	
	21. Kembali ke halaman *Initial Vartrain LREPSME.*
Jika Menekan tombol o*nBackPressed* pada halaman *Training Result* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
20. Menekan tombol *onBackPressed*.	
	21. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.

22. Menekan tombol *Yes.*	
	23. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *No* pada *Dialog Confirm*.	
20. Menekan tombol *onBackPressed.*	
	21. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
22. Menekan tombol *No.*	
	23. Kembali ke halaman *Training Result.*

4.3.9. Skenario Menyimpan *Training Result*

Tabel 4.11 Skenario Menyimpan Training Result

Name	Menyimpan *Training Result.*
Actor	*User*.
Entry Condition	Aplikasi telah melakukan proses *training data* dan menampilkan hasilnya pada halaman *Training Result.*
Exit Condition	Aplikasi menyimpan hasil *training* ke dalam *database*.
Skenario Utama	
Actor	***System***
2. Menekan tombol *Save.*	
	3. Menyimpan hasil *training data* pada *database* dengan meng-*update* nilai *training result* sebelumnya dengan hasil *training* yang baru.

	4. Menampilkan Pesan "Data Berhasil Disimpan !" dan mengaktifkan tombol *Reset* dan pada halaman *Training Result.*
Skenario Alternatif	
Jika Menekan tombol *Back* pada halaman *Training Result.*	
5. Menekan tombol *Back.*	
	6. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
5. Menekan tombol *onBackPressed.*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *Yes.*	
	8. Kembali ke halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *No* pada *Dialog Confirm.*	
5. Menekan tombol *onBackPressed*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *No.*	
	8. Kembali ke halaman *Training Result.*

4.3.10. Skenario Melihat *Training Result*

Tabel 4.12 Skenario Melihat Training Result

Name	Melihat *Training Result.*
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Home.*
Exit Condition	Aplikasi menampilkan *list training result* yang ada di *database*.
Skenario Utama	
Actor	***System***
1. Menekan menu *Setting Data.*	
	2. Menampilkan halaman *Setting Data* beserta submenu *Take Image, Setting Data, Training Result* dan *Go Home.*
3. Menekan submenu *Training Result.*	
	4. Menampilkan *list training result* pada halaman *Training Result*, beserta tombol *Back* dan tombol *Reset*.
Skenario Alternatif	
Jika Menekan tombol *Back* pada halaman *Training Result.*	
5. Menekan tombol *Back.*.	
	6. Kembali ke halaman *Setting Data.*
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *Yes* pada *Dialog Confirm*.	
5. Menekan tombol	

onBackPressed.	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *Yes.*	
	8. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *No* pada *Dialog Confirm*.	
5. Menekan tombol *onBackPressed.*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *No.*	
	8. Kembali ke halaman *Training Result.*

4.3.11. Skenario Melakukan *Reset Data*

Tabel 4.13 Skenario Melakukan *Reset Data*

Name	Melakukan *Reset Data.*
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Training Result.*
Exit Condition	Aplikasi melakukan *reset data*.
Skenario Utama	
Actor	***System***
1. Menekan tombol *Reset.*	
	2. Menampilkan *Dialog Confirm*

	"Anda Yakin Ingin Melakukan Reset Data ? ----- Reset Data Akan Mengembalikan Ke Pengaturan Default Aplikasi dan Menghapus Semua Perubahan Data Sebelumnya." beserta tombol *Yes* dan *No*.
3. Menekan tombol *Yes.*	
	4. Menghapus semua *dataset* dan merubah *training result* ke *training result default* aplikasi serta menampilkan hasil *reset* dan Pesan "Reset Selesai !".
Skenario Alternatif	
Jika Menekan tombol *No* pada *Dialog Confirm.*	
1. Menekan tombol *No.*	
	2. Kembali ke halaman *Training Result*.
Jika Menekan tombol *Back* pada halaman *Training Result.*	
5. Menekan tombol *Back.*	
	6. Kembali ke halaman *Setting Data*.
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *Yes* pada *Dialog Confirm.*	
5. Menekan tombol *onBackPressed*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.

7. Menekan tombol *Yes*	
	8. Menampilkan halaman *Home.*
Jika Menekan tombol *onBackPressed* pada halaman *Training Result* dan Memilih tombol *No* pada *Dialog Confirm*.	
5. Menekan tombol *onBackPressed*	
	6. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Ke Menu Home ?" beserta tombol *Yes* dan *No*.
7. Menekan tombol *No.*	
	8. Kembali ke halaman *Training Result.*

4.3.12. Skenario Melihat *Help How To Identify Banana*

Tabel 4.14 Skenario Melihat *Help How To Identify Banana*

Name	Melihat *Help How To Identify Banana*.
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Home.*
Exit Condition	Menampilkan cara atau petunjuk untuk menggunakan fitur *Identify Banana.*
Skenario Utama	
Actor	***System***
1. Menekan menu *Help.*	
	2. Menampilkan submenu dari menu help yaitu, *Submenu How To Identify Banana*, *How To Setting Data*, dan *Go Home.*

3. Menekan submenu *How To Identify Banana.*	
	4. Menampilkan halaman *How To Setting Data* yang berisi cara atau petunjuk menggunakan fitur *Identify Banana* beserta tombol *Back* dan tombol *Go Home.*
Skenario Alternatif	
Jika Menekan tombol *Back* pada halaman *How To Identify Banana*.	
5. Menekan tombol *Back.*	
	6. Kembali ke halaman *Help.*
Jika Menekan tombol *Go Home* pada halaman *How To Identify Banana.*	
5. Menekan tombol *Go Home.*	
	6. Kembali ke halaman *Home.*

4.3.13. Skenario Melihat *Help How To Setting Data*

Tabel 4.15 Skenario Melihat *Help How To Setting Data*

Name	Melihat *Help How To Setting Data.*
Actor	*User.*
Entry Condition	Aplikasi menampilkan halaman *Home.*
Exit Condition	Menampilkan cara atau petunjuk untuk menggunakan fitur *Setting Data.*
Skenario Utama	
Actor	***System***
1. Menekan menu *Help.*	
	2. Menampilkan submenu dari menu help yaitu, *Submenu How To Identify*

	Banana, *How To Setting Data*, dan *Go Home.*
3. Menekan submenu *How To Setting Data.*	
	4. Menampilkan halaman *How To Setting Data* yang berisi cara atau petunjuk menggunakan fitur *Setting Data* beserta tombol *Back* dan tombol *Go Home.*
Skenario Alternatif	
Jika Menekan tombol *Back* pada halaman *How To Setting Data.*	
5. Menekan tombol *Back.*	
	6. Kembali ke halaman *Help.*
Jika Menekan tombol *Go Home* pada halaman *How To Setting Data.*	
5. Menekan tombol *Go Home.*	
	6. Menampilkan halaman *Home.*

4.3.14. Skenario Melihat *About*

Tabel 4.16 Skenario Melihat *About*

Name	Melihat *About.*
Actor	*User.*
Entry Condition	Aplikasi menampilkan halaman *Home.*
Exit Condition	Menampilkan informasi tentang aplikasi *Banana Maturity Identification.*
Skenario Utama	
Actor	***System***
1. Menekan menu *About.*	

	2. Menampilkan halaman *About* yang berisi informasi tentang aplikasi *Banana Maturity Identification* beserta tombol *Back* dan tombol *Go Home*.
Skenario Alternatif	
Jika Menekan tombol *Back* pada halaman *About*.	
3. Menekan tombol *Back*.	
	4. Kembali ke halaman *Home*.
Jika Menekan tombol *Go Home* pada halaman *About*.	
3. Menekan tombol *Go Home*.	
	4. Menampilkan halaman *Home*.

4.3.15. Skenario *Exit*

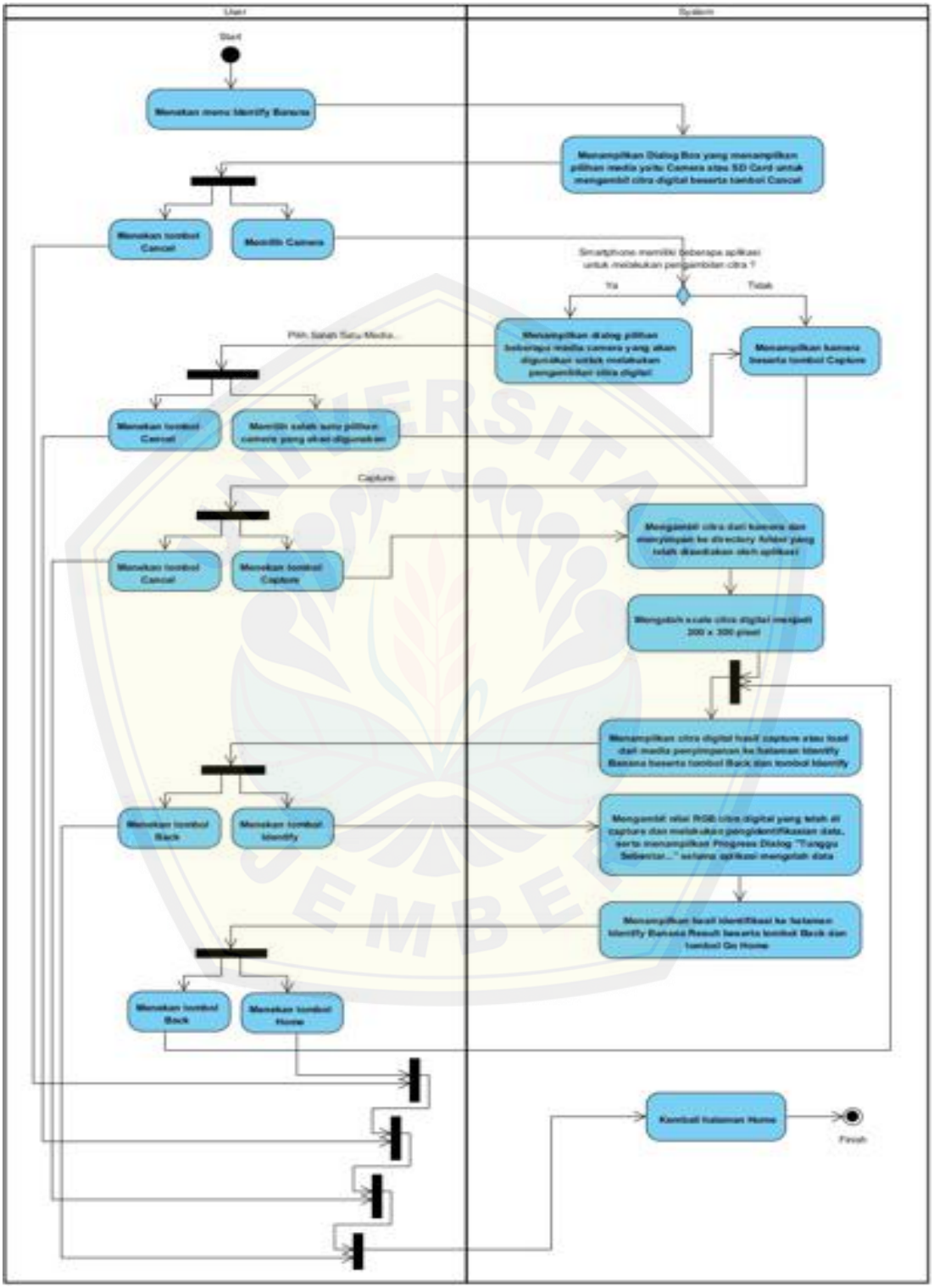
Tabel 4.17 Skenario *Exit*

Name	*Exit*.
Actor	*User*.
Entry Condition	Aplikasi menampilkan halaman *Home*.
Exit Condition	*User* keluar dari aplikasi dan semua aktivitas aplikasi ditutup.
Skenario Utama	
Actor	***System***
1. Menekan tombol *onBackPressed*.	
	2. Menampilkan *Dialog Confirm* "Anda Yakin Ingin Keluar ?" beserta tombol *Yes* dan *No*.

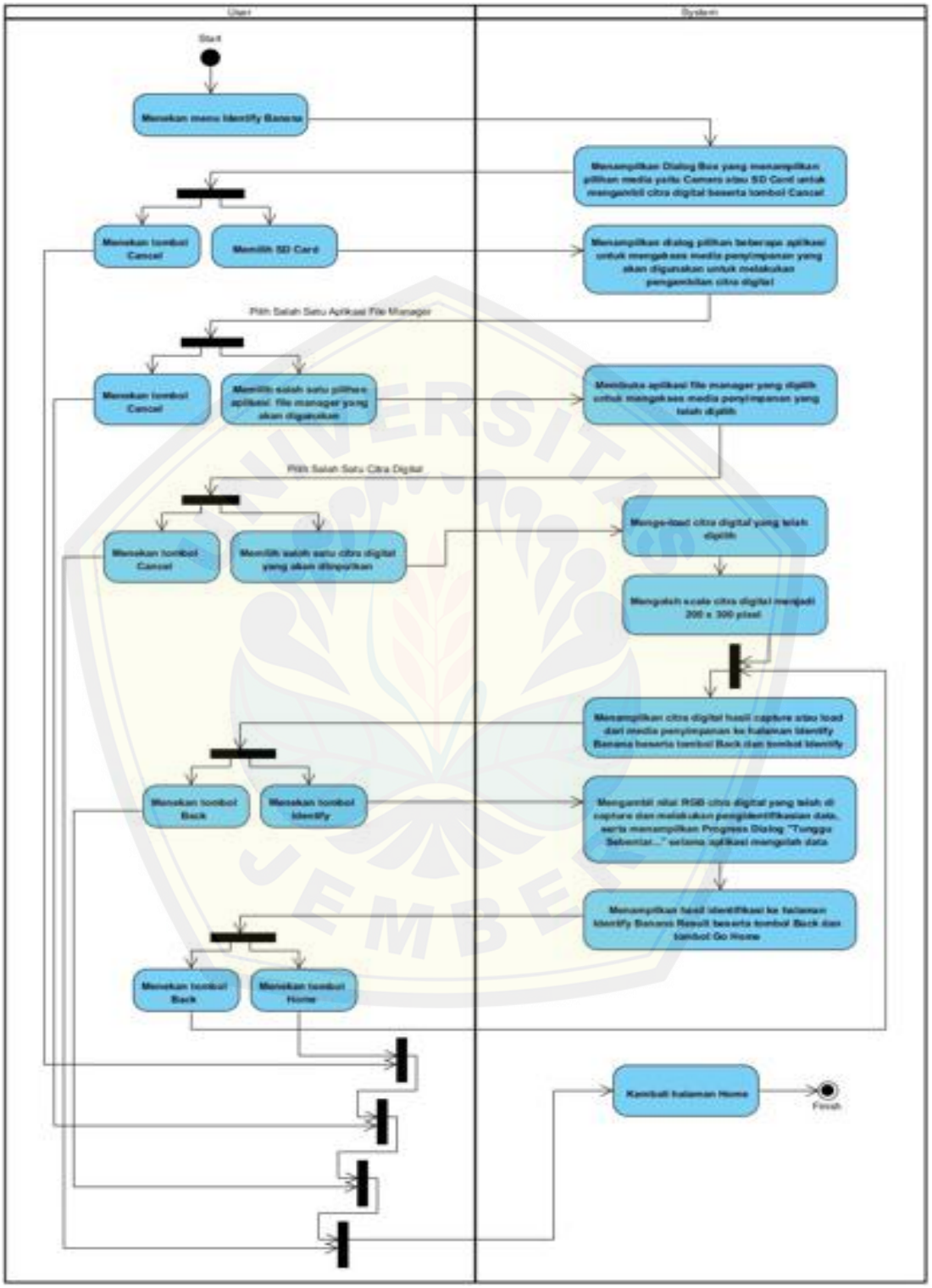
3. Menekan tombol Yes.	
	4. Menutup semua aktivitas aplikasi.
Skenario Alternatif	
Jika Menekan tombol *No* pada *Dialog Confirm*.	
3. Menekan tombol *No.*	
	4. Kembali ke halaman *Home.*

4.4. *Activity Diagram*

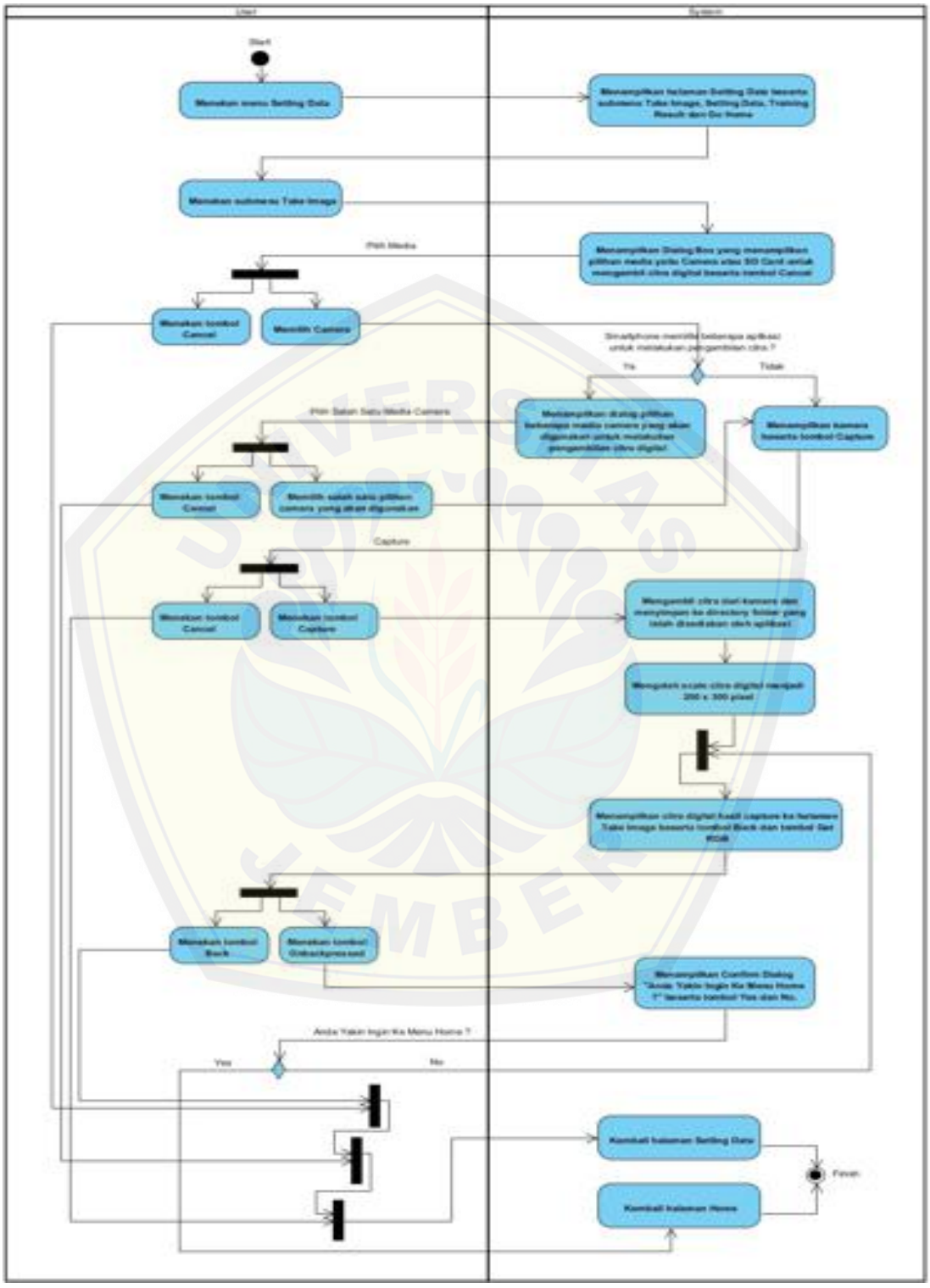
Activity Diagram berfungsi untuk menggambarkan alur aktivitas yang akan dijalankan aplikasi *Banana Maturity Identification* dengan *user* dalam bentuk diagram aktivitas. *Activity Diagram* pada aplikasi ditunjukkan pada Gambar 4.2 sampai Gambar 4.16.



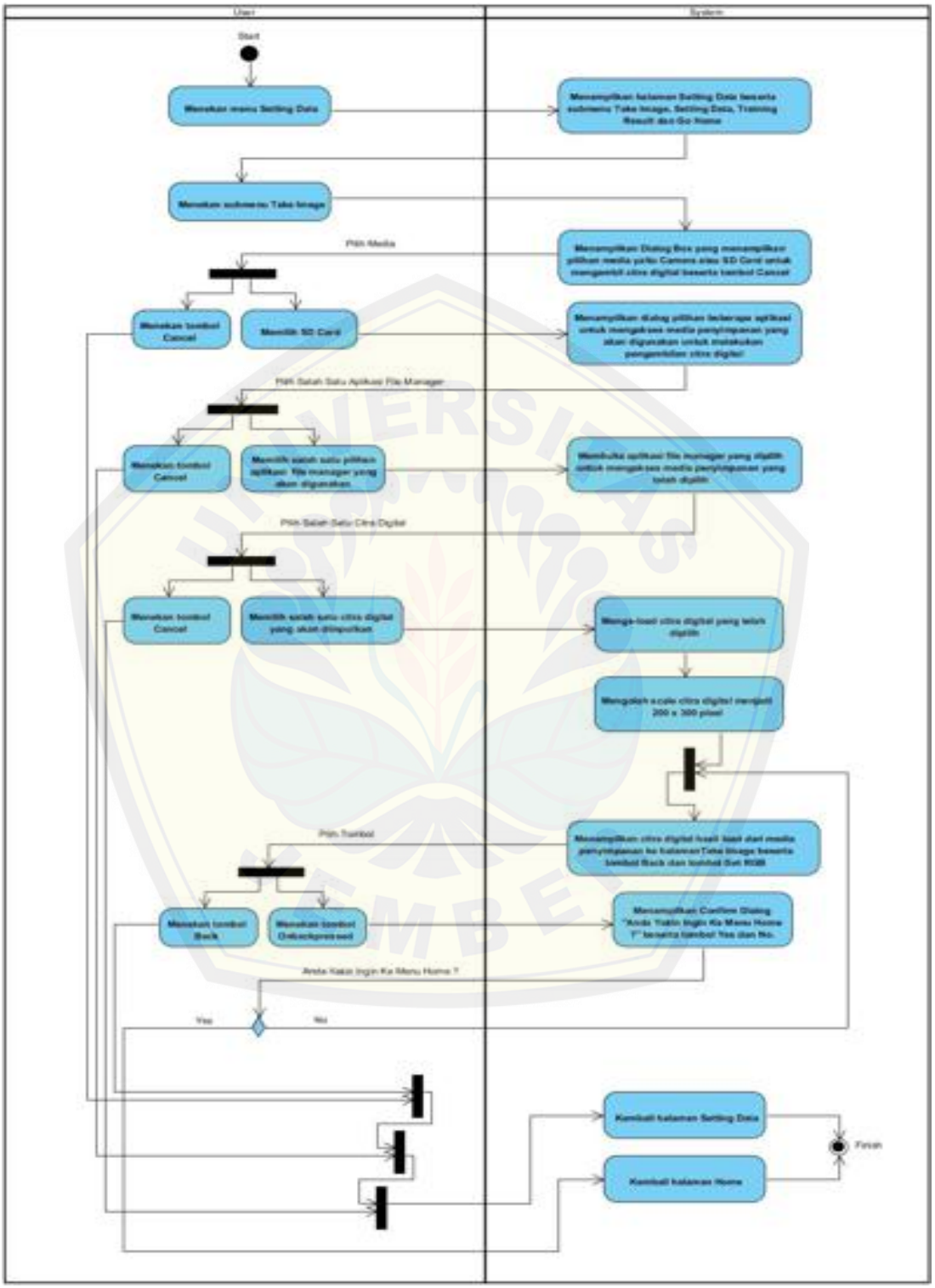
Gambar 4.2 ***Activity Diagram*** **Mengidentifikasi Pisang** ***Via Camera***



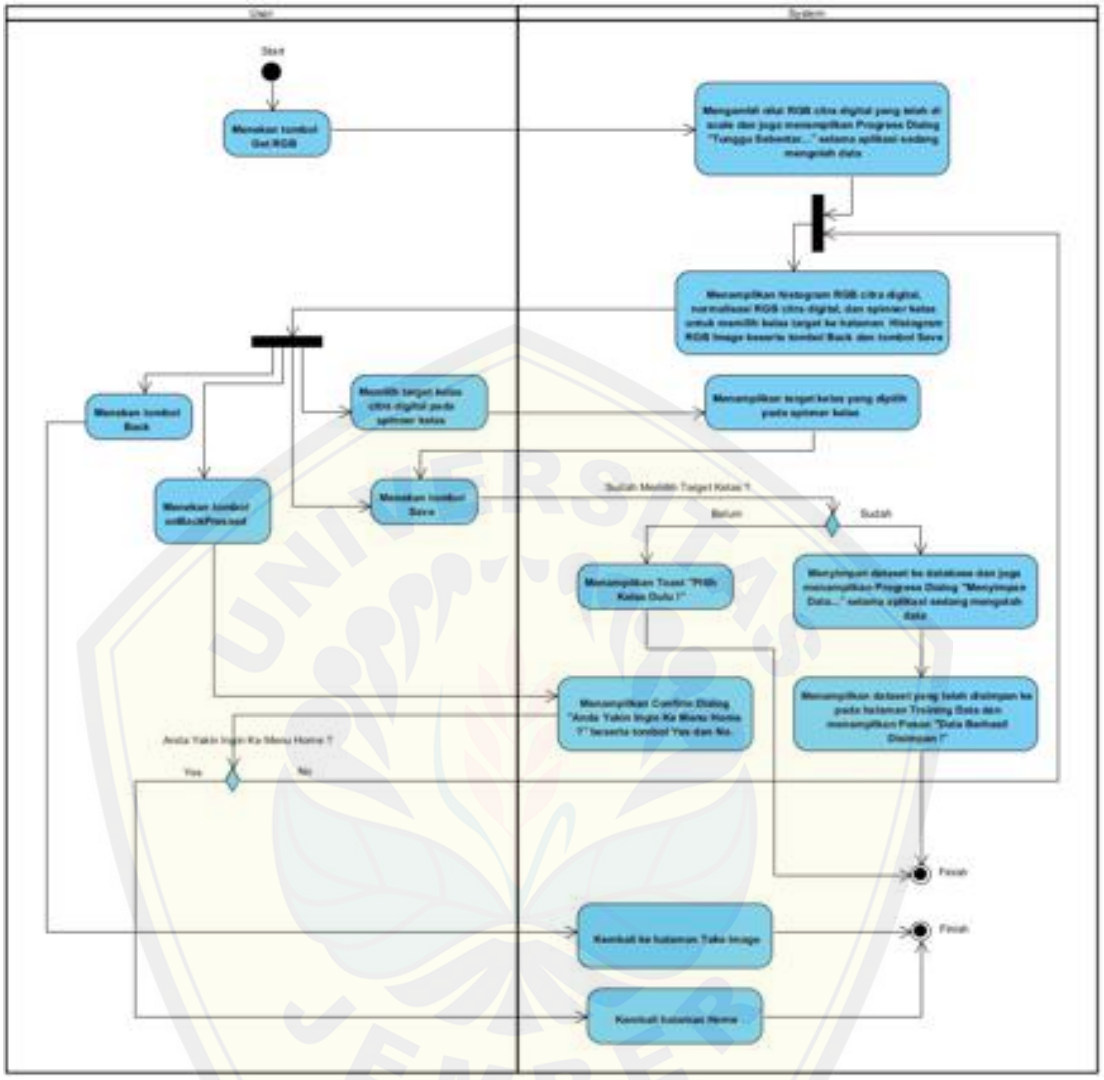
Gambar 4.3 ***Activity Diagram*** **Mengidentifikasi Pisang** ***Via SD Card***



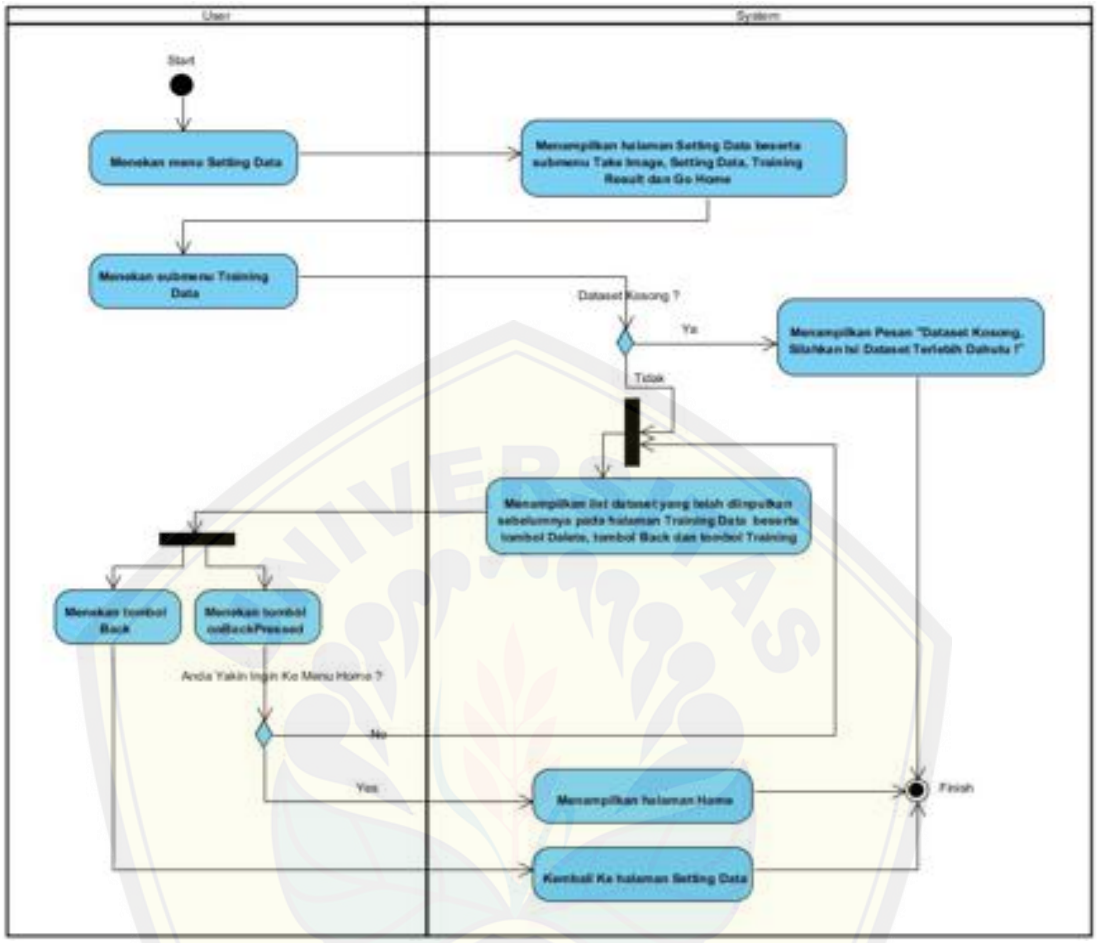
Gambar 4.4 ***Activity Diagram Take Image Via Camera***



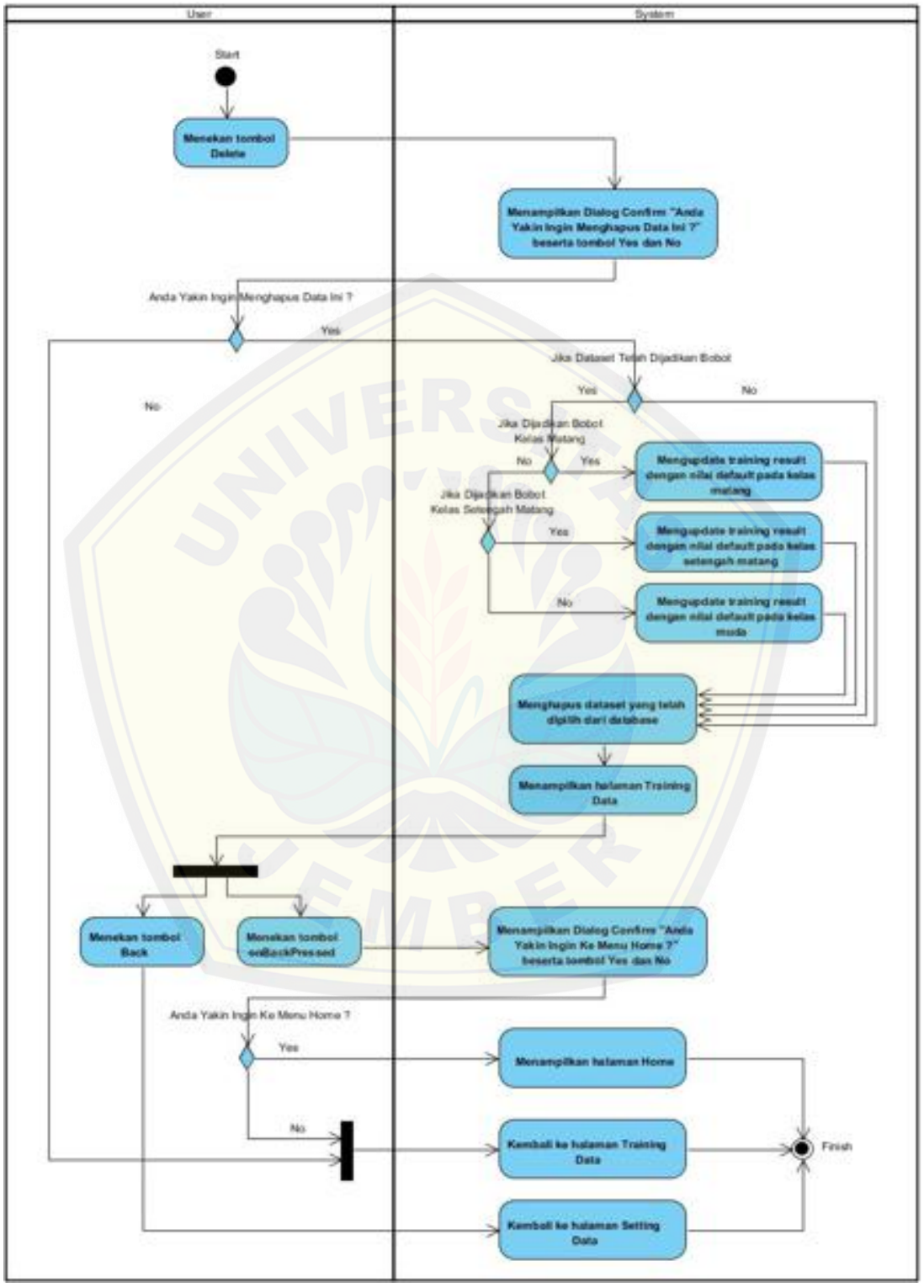
Gambar 4.5 ***Activity Diagram Take Image Via SD Card***



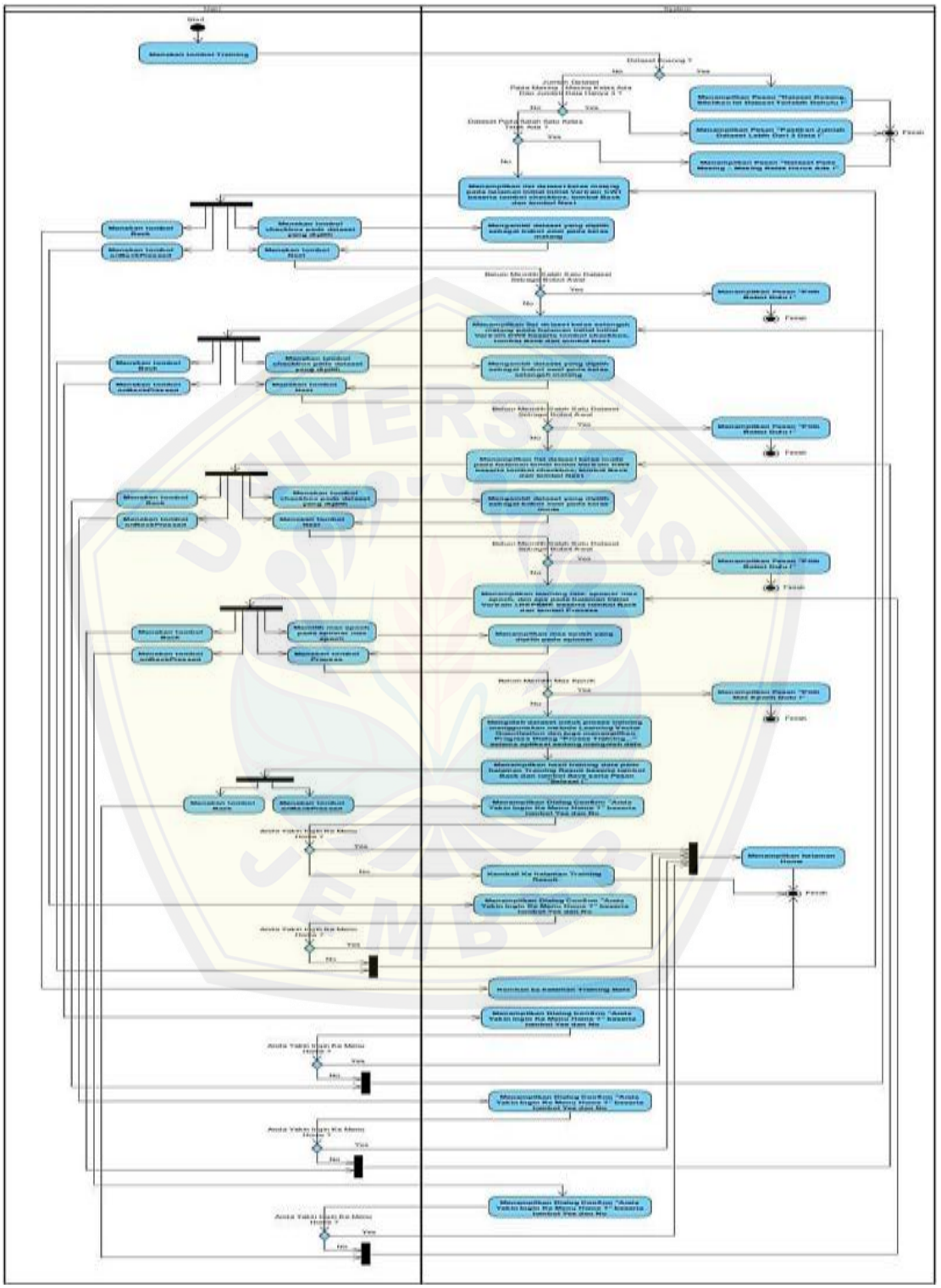
Gambar 4.6 ***Activity Diagram*** **Menyimpan RGB** ***Image***



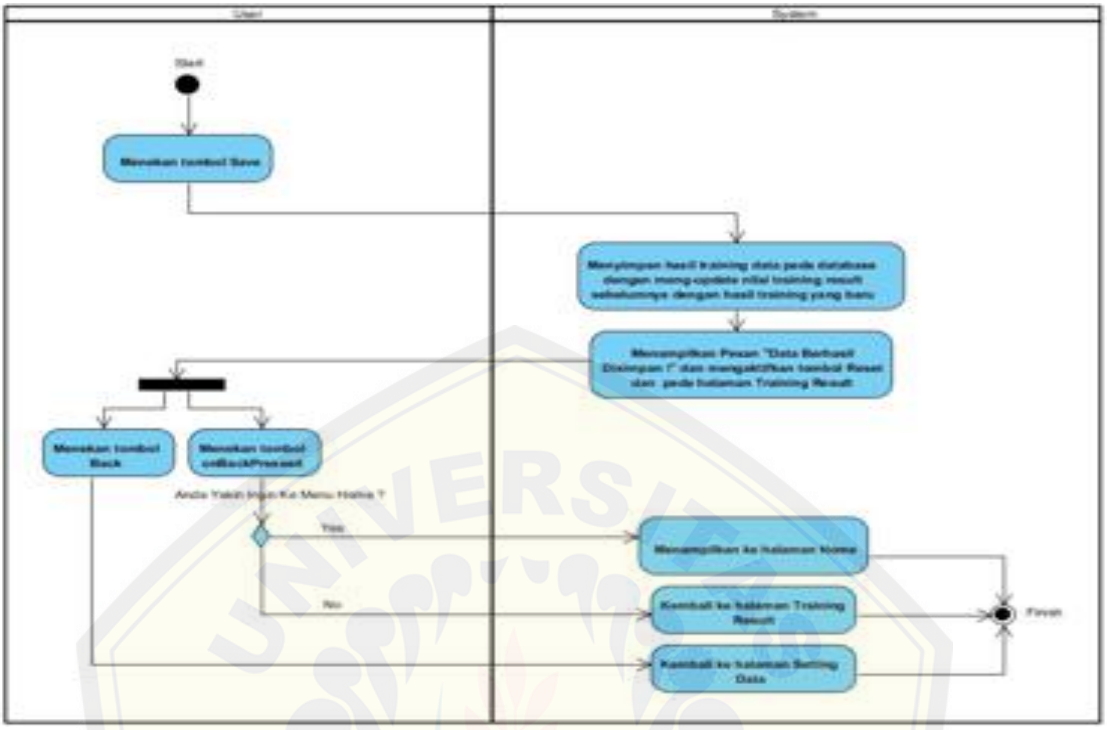
Gambar 4.7 ***Activity Diagram*** **Melihat** ***List Dataset***



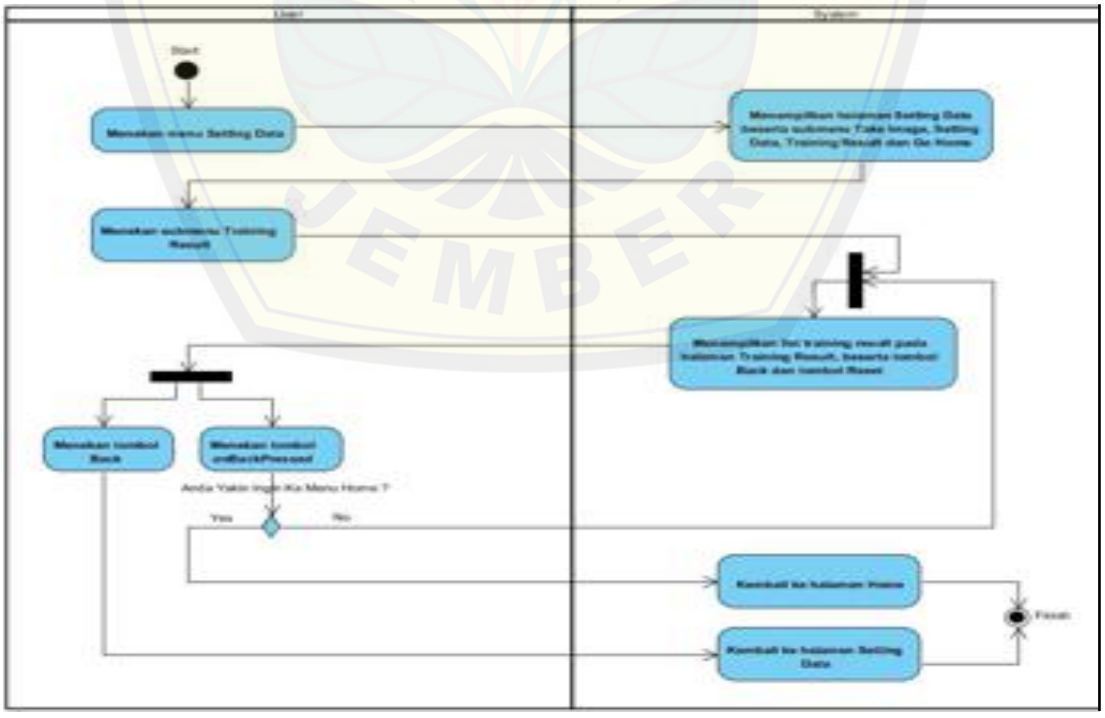
Gambar 4.8 ***Activity Diagram*** **Menghapus** ***Dataset***



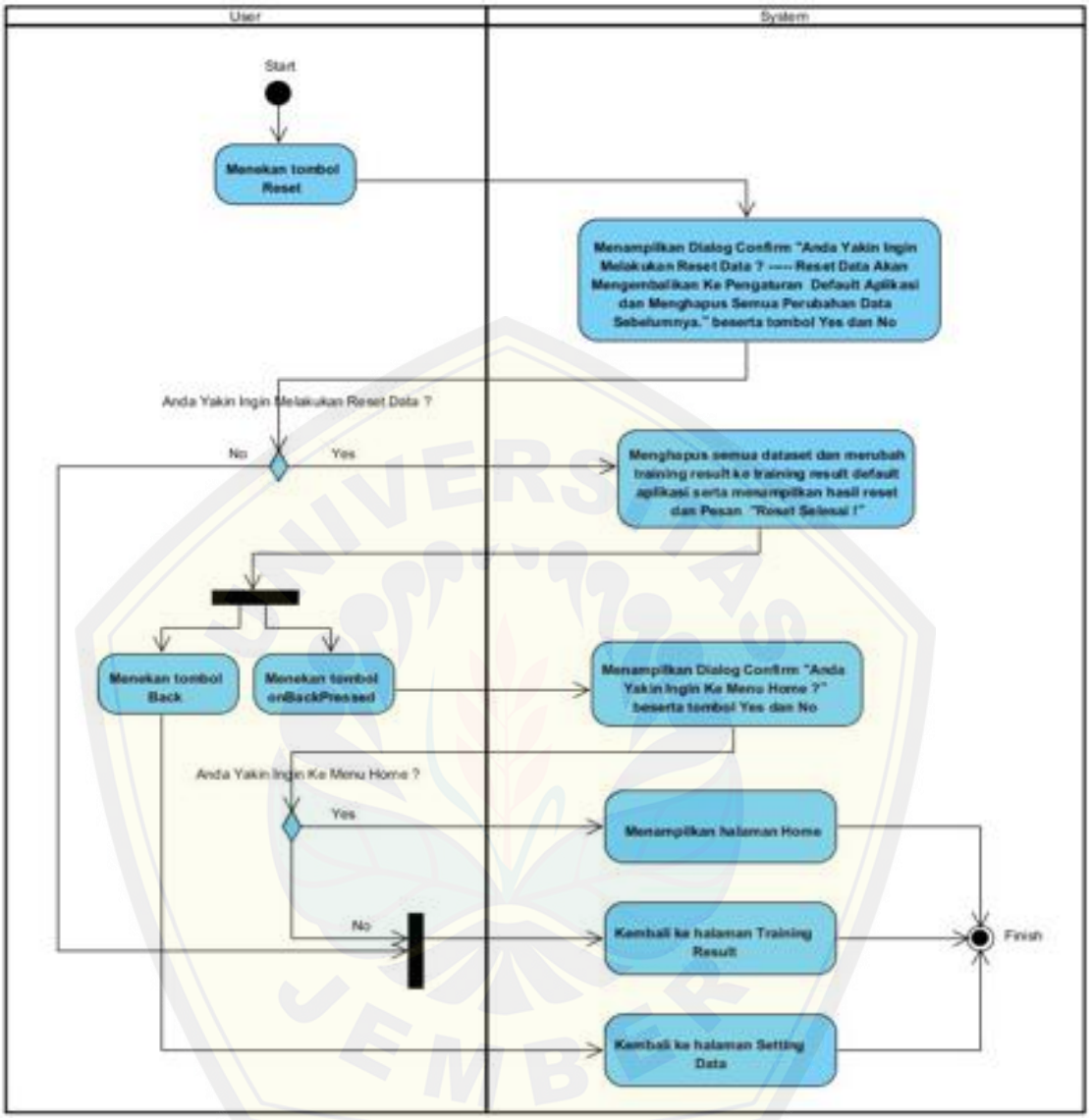
Gambar 4.9 ***Activity Diagram*** **Melakukan** ***Training Data***



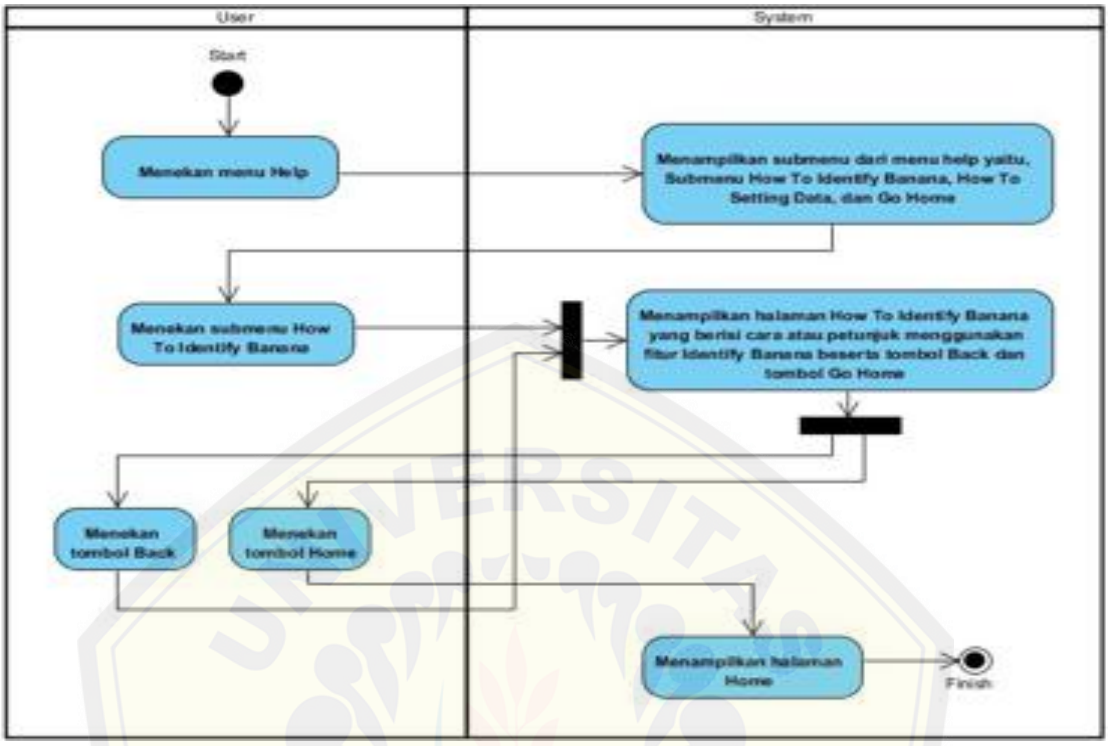
Gambar 4.10 ***Activity Diagram*** **Menyimpan** ***Training Result***



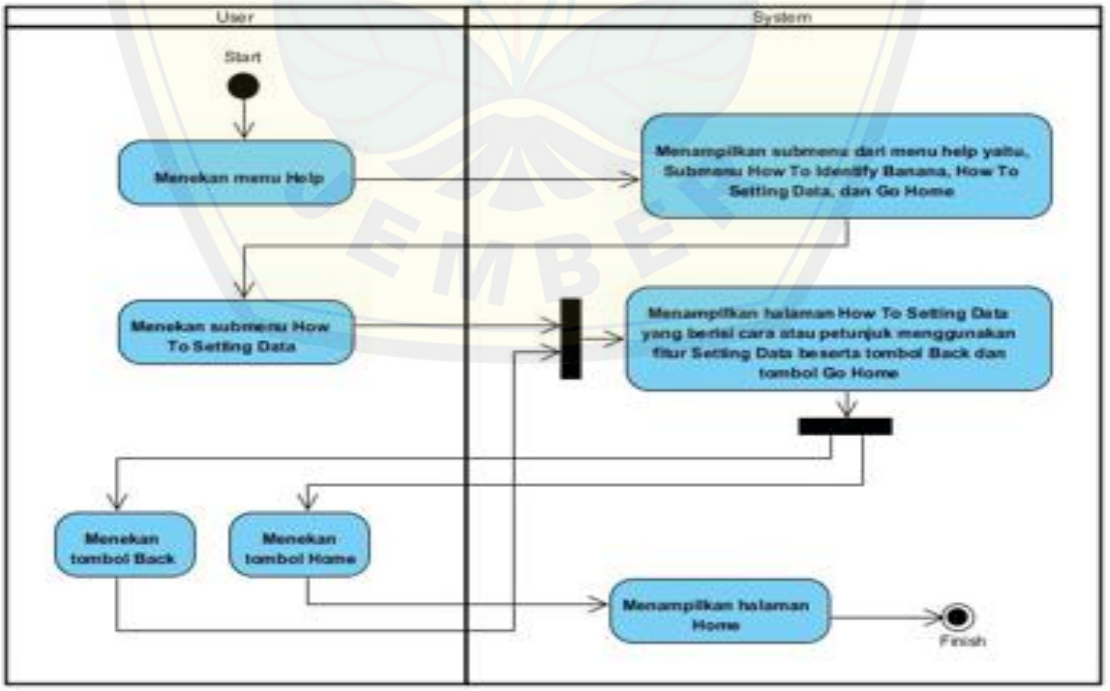
Gambar 4.11 ***Activity Diagram*** **Melihat** ***Training Result***



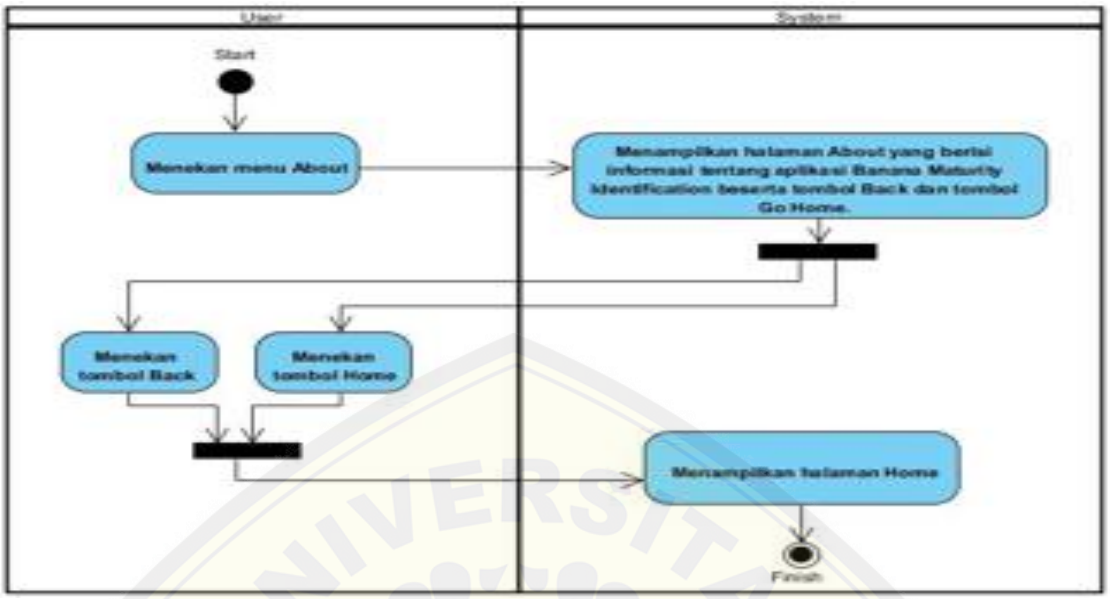
Gambar 4.12 ***Activity Diagram*** **Melakukan** ***Reset Data***



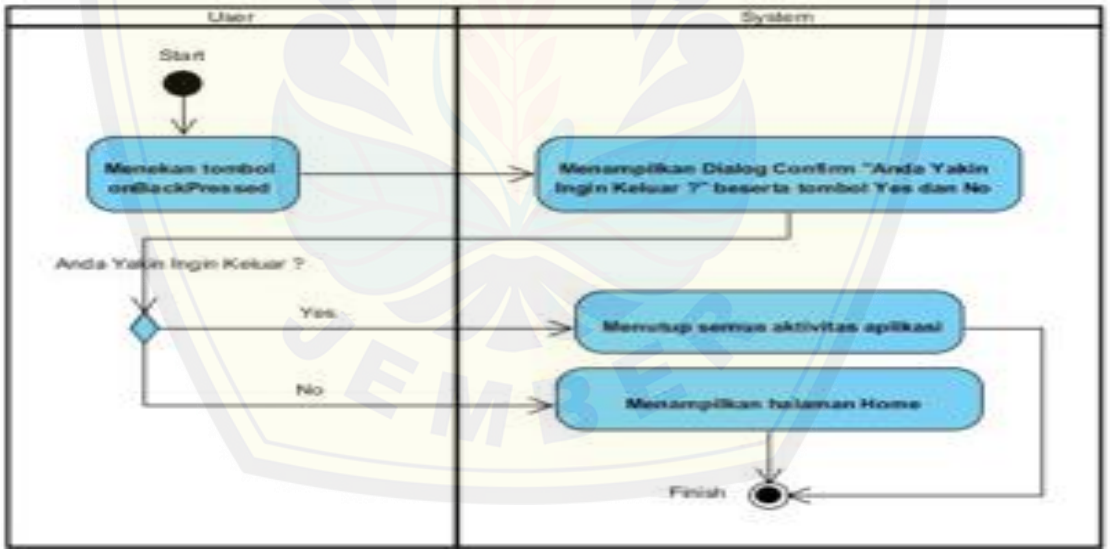
Gambar 4.13 ***Activity Diagram*** **Melihat** ***Help How To Identify Banana***



Gambar 4.14 ***Activity Diagram*** **Melihat** ***Help How To Setting Data***



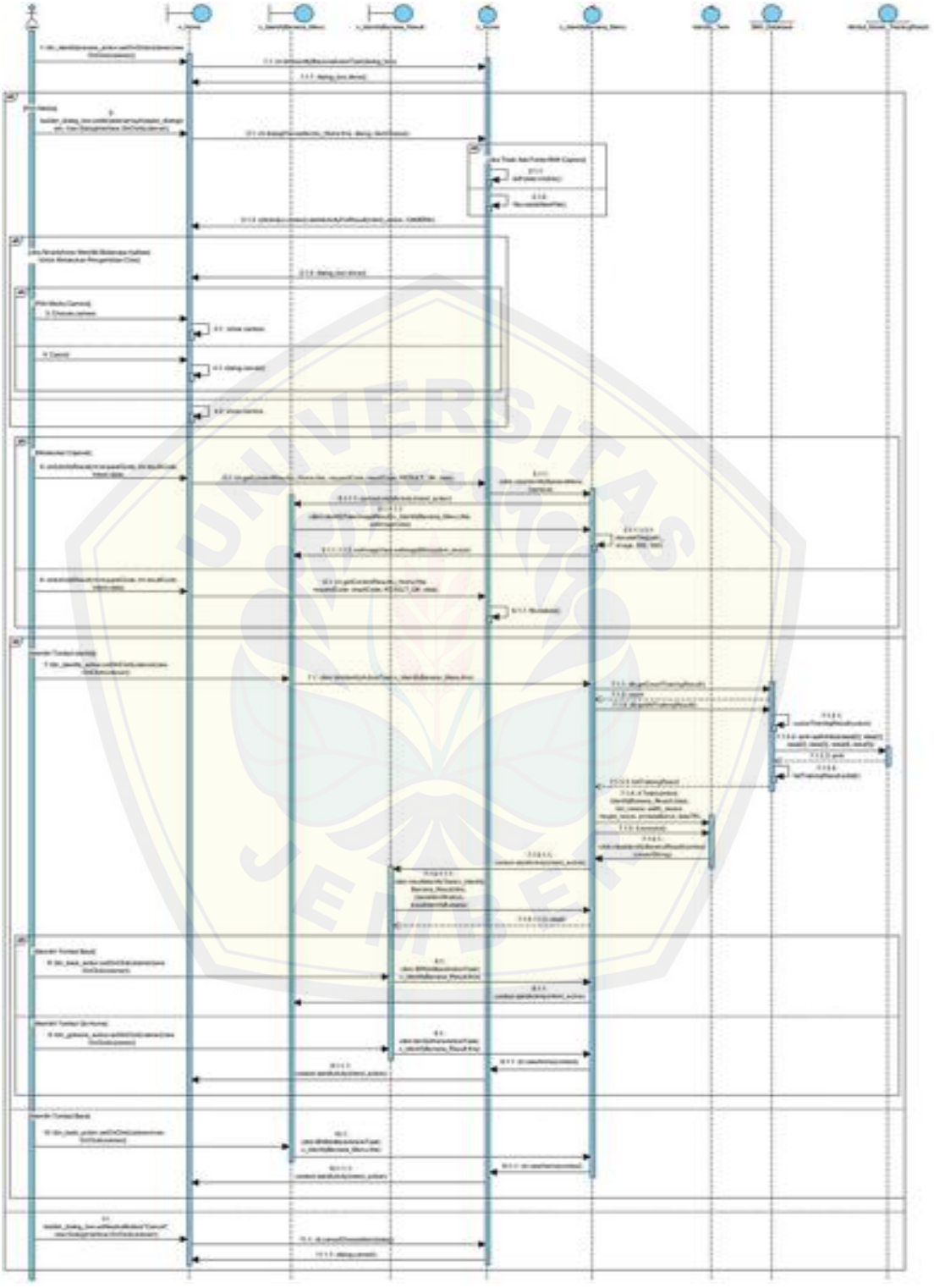
Gambar 4.15 ***Activity Diagram*** **Melihat** ***About***



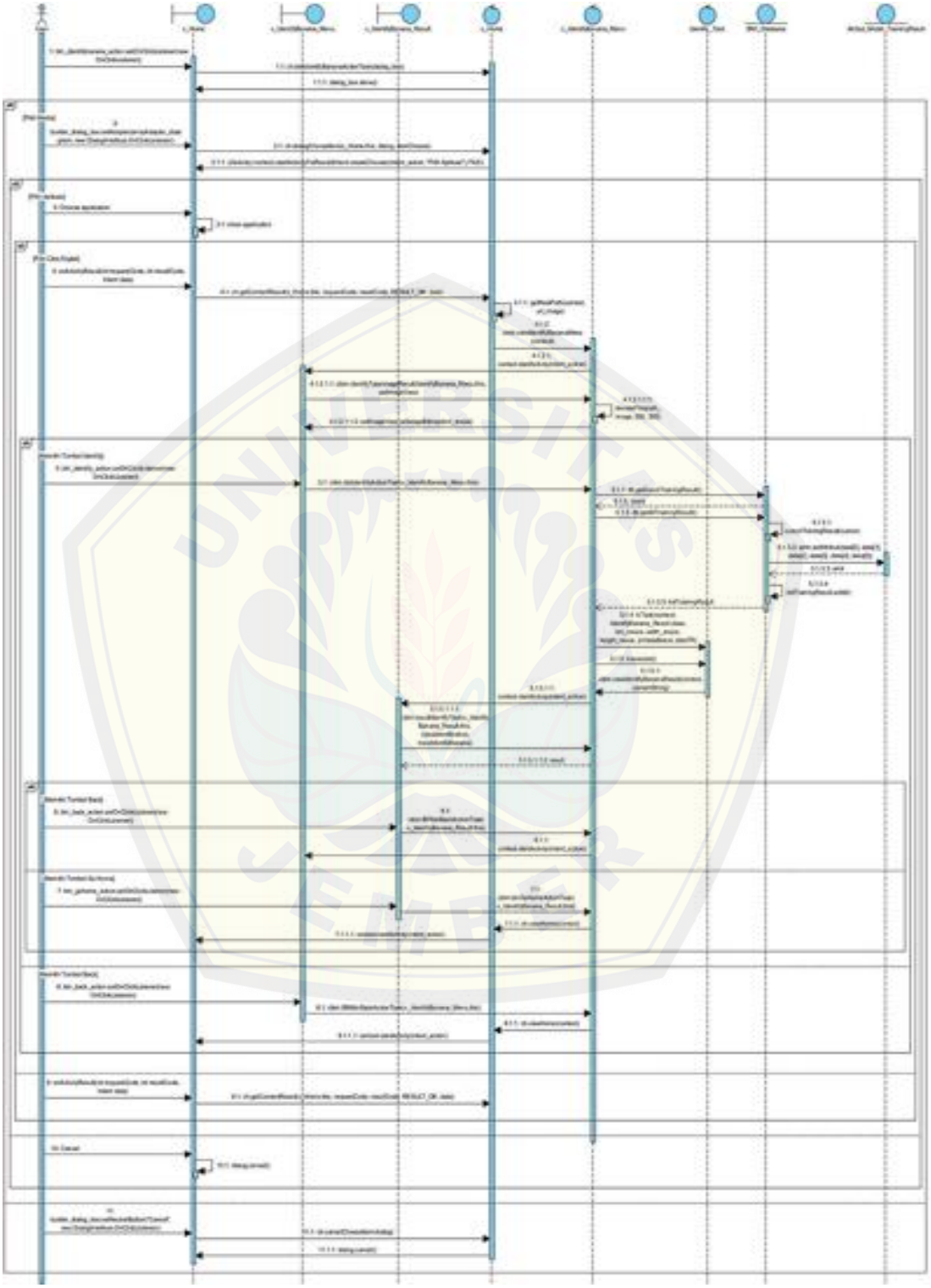
Gambar 4.16 ***Activity Diagram Exit***

4.5. *Sequence Diagram*

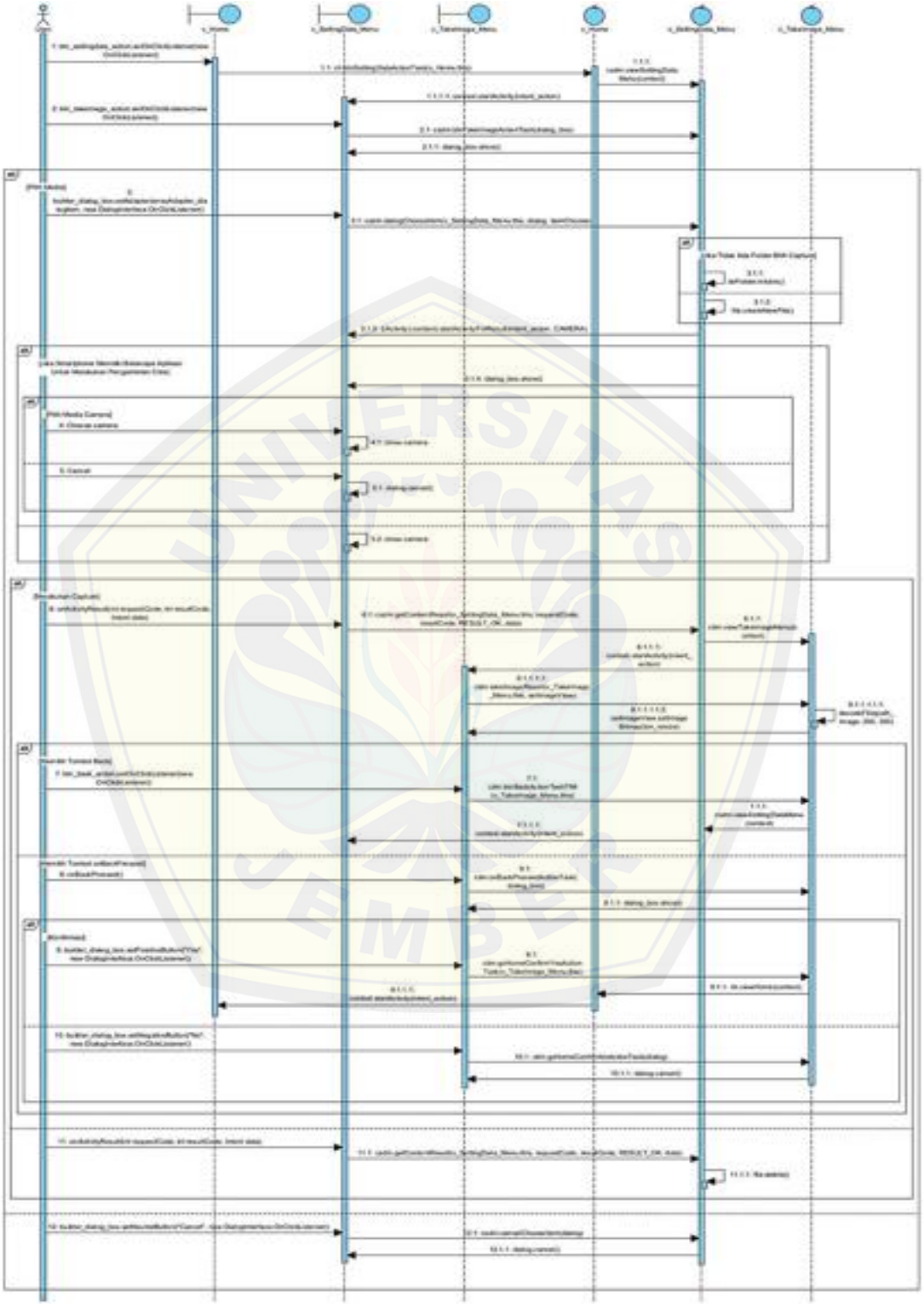
Sequence Diagram berfungsi untuk menggambarkan proses dan perubahan apa saja yang akan terjadi secara internal dan *output* apa yang dihasilkan pada aplikasi *Banana Maturity Identification. Sequence* diagram aplikasi *Banana Maturity Identification* dapat dilihat pada Gambar 4.17 sampai Gambar 4.31.



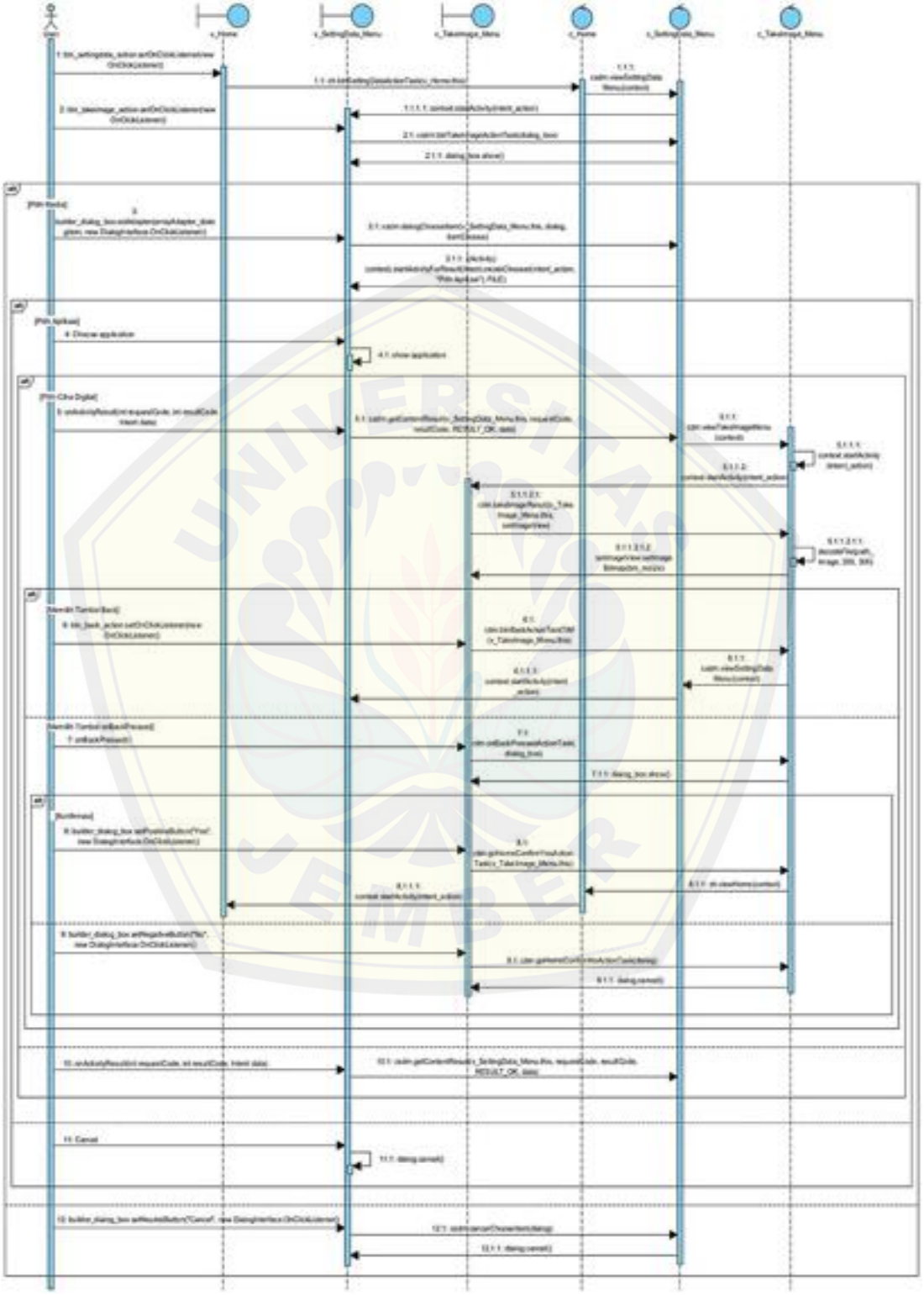
Gambar 4.17 ***Sequence Diagram*** **Mengidentifikasi Pisang** ***Via Camera***



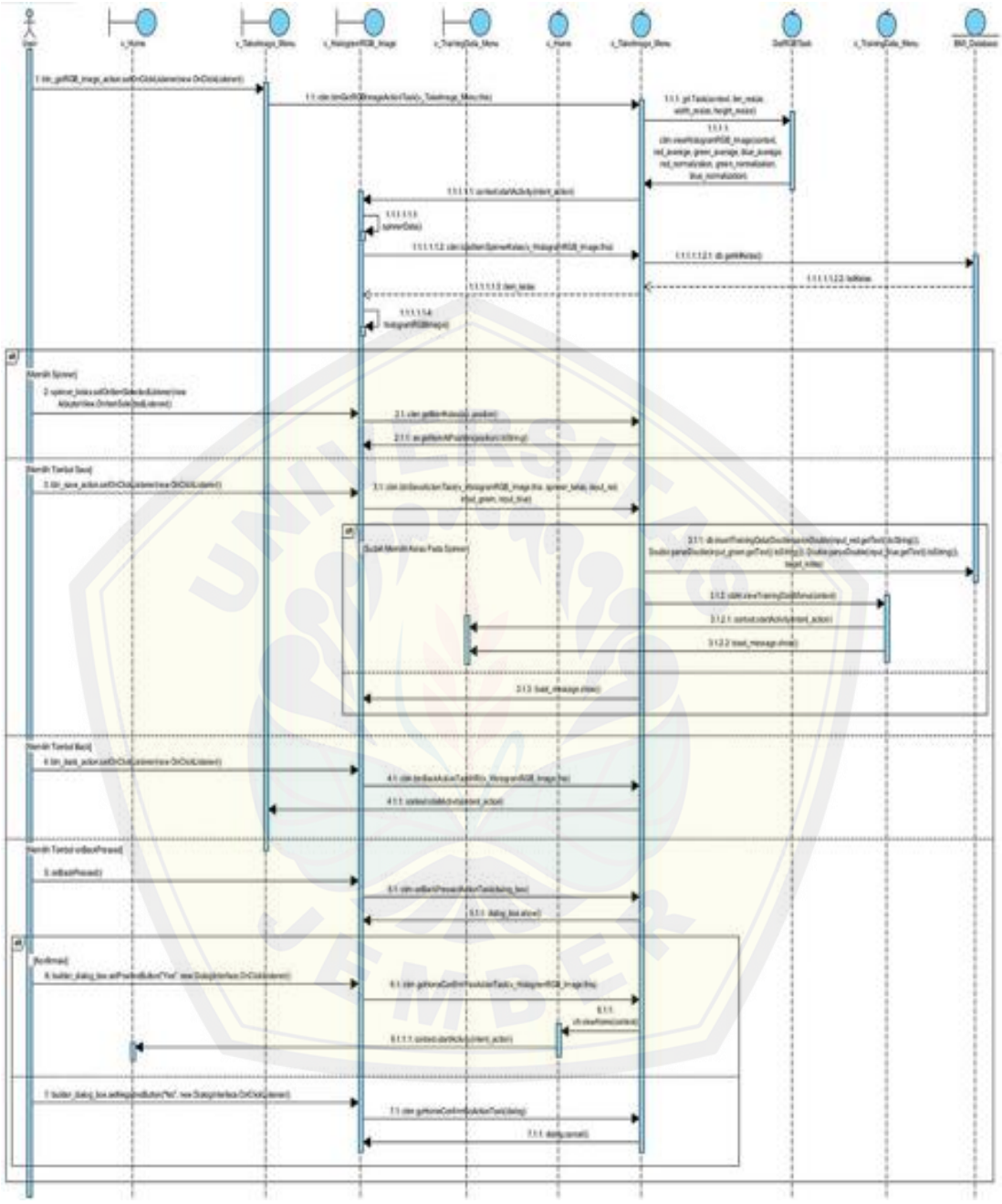
Gambar 4.18 ***Sequence Diagram*** **Mengidentifikasi Pisang** ***Via SD Card***



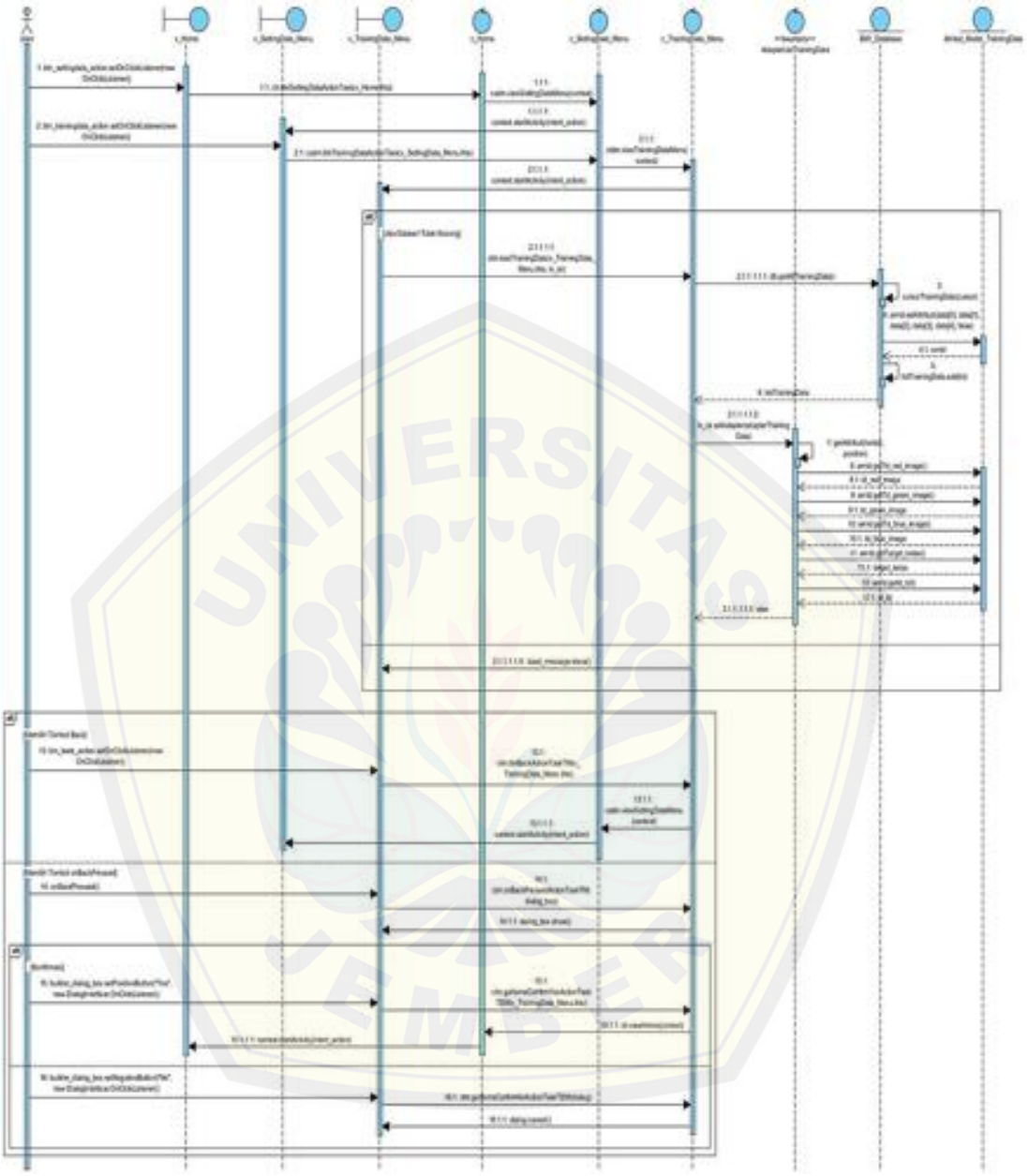
Gambar 4.19 ***Sequence Diagram Take Image Via Camera***



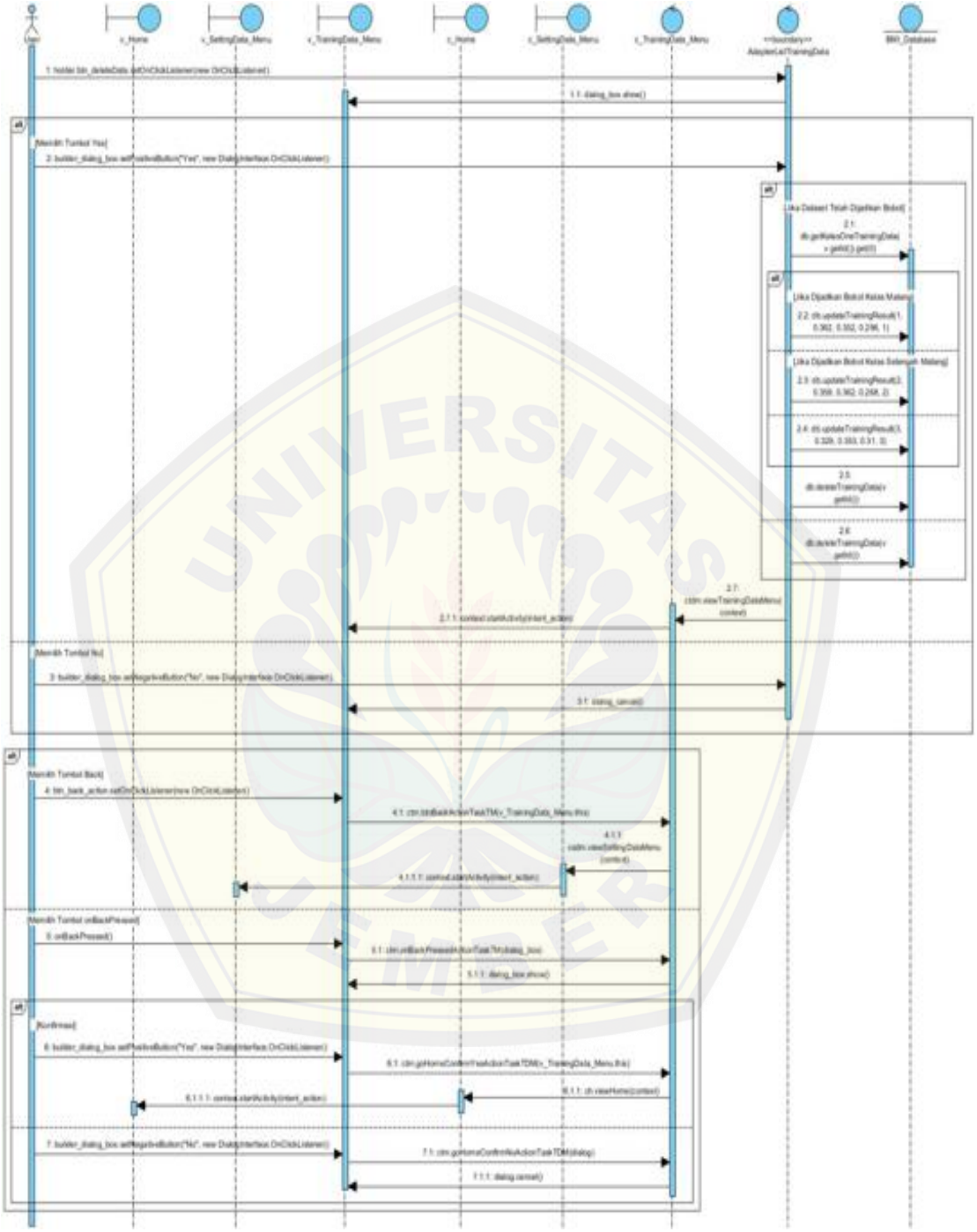
Gambar 4.20 ***Sequence Diagram Take Image Via SD Card***



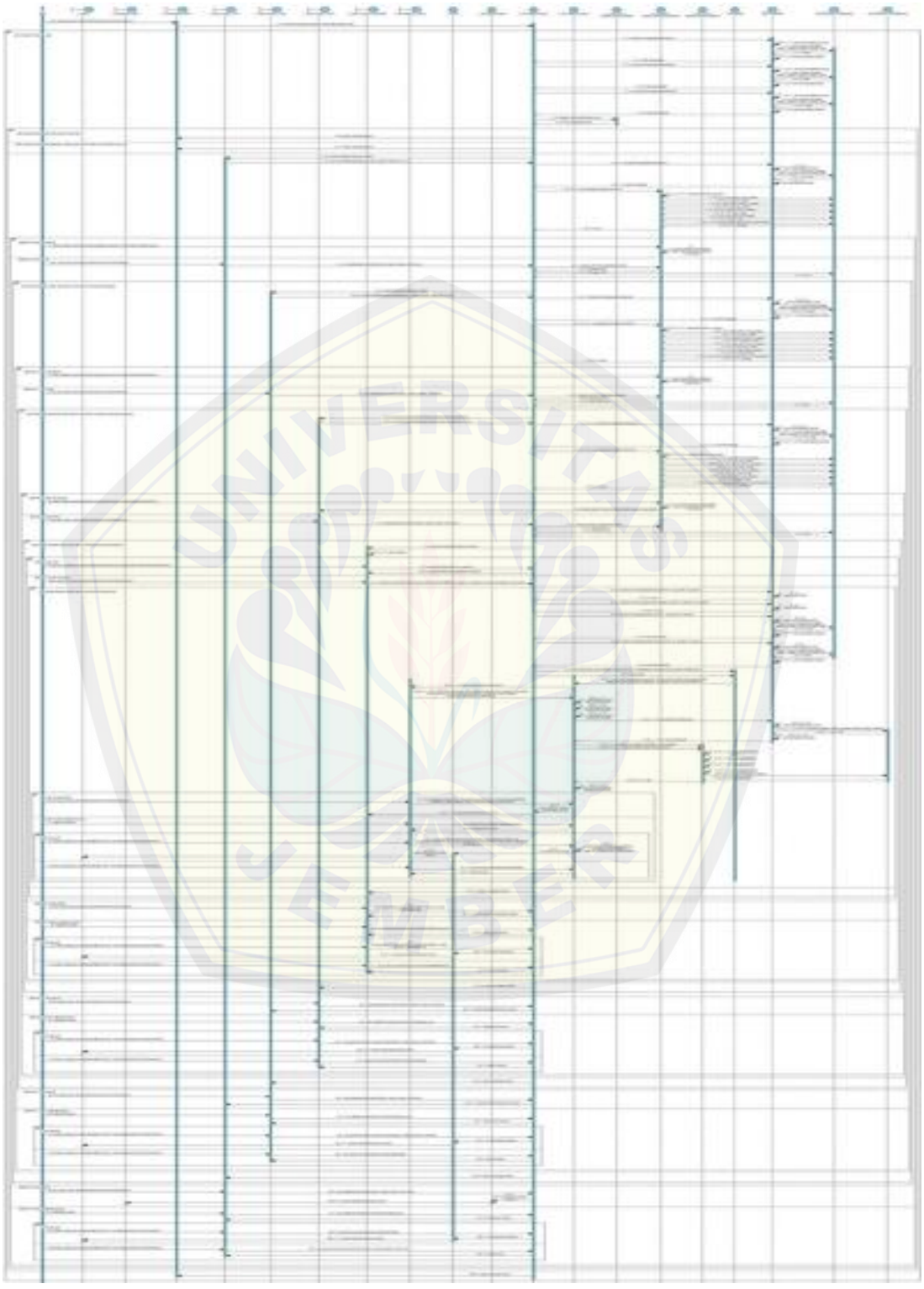
Gambar 4.21 *Sequence Diagram* Menyimpan RGB *Image*



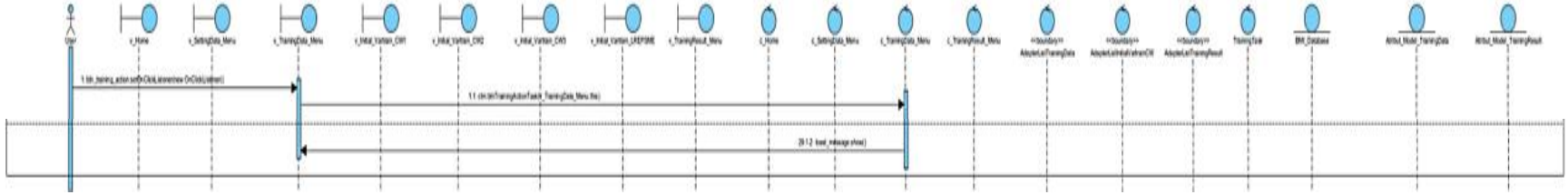
Gambar 4.22 *Sequence Diagram* Melihat *List Dataset*



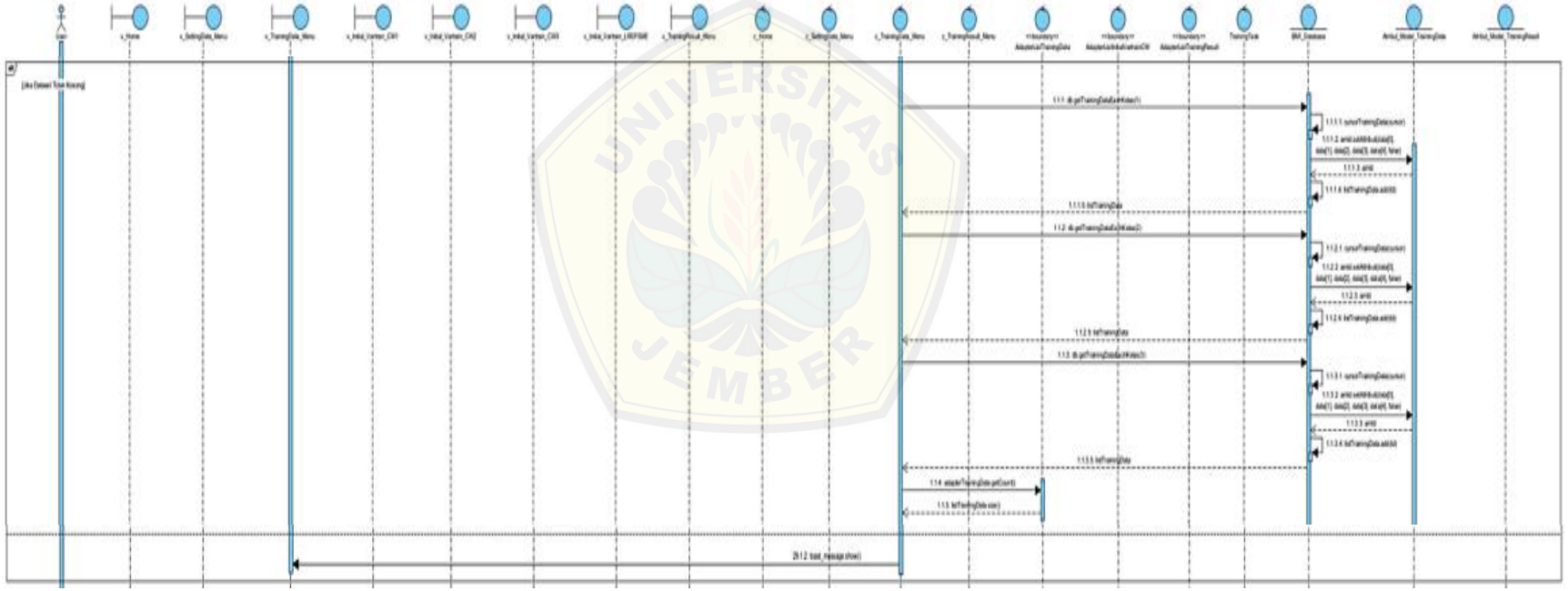
Gambar 4.23 *Sequence Diagram* Mengahapus *Dataset*



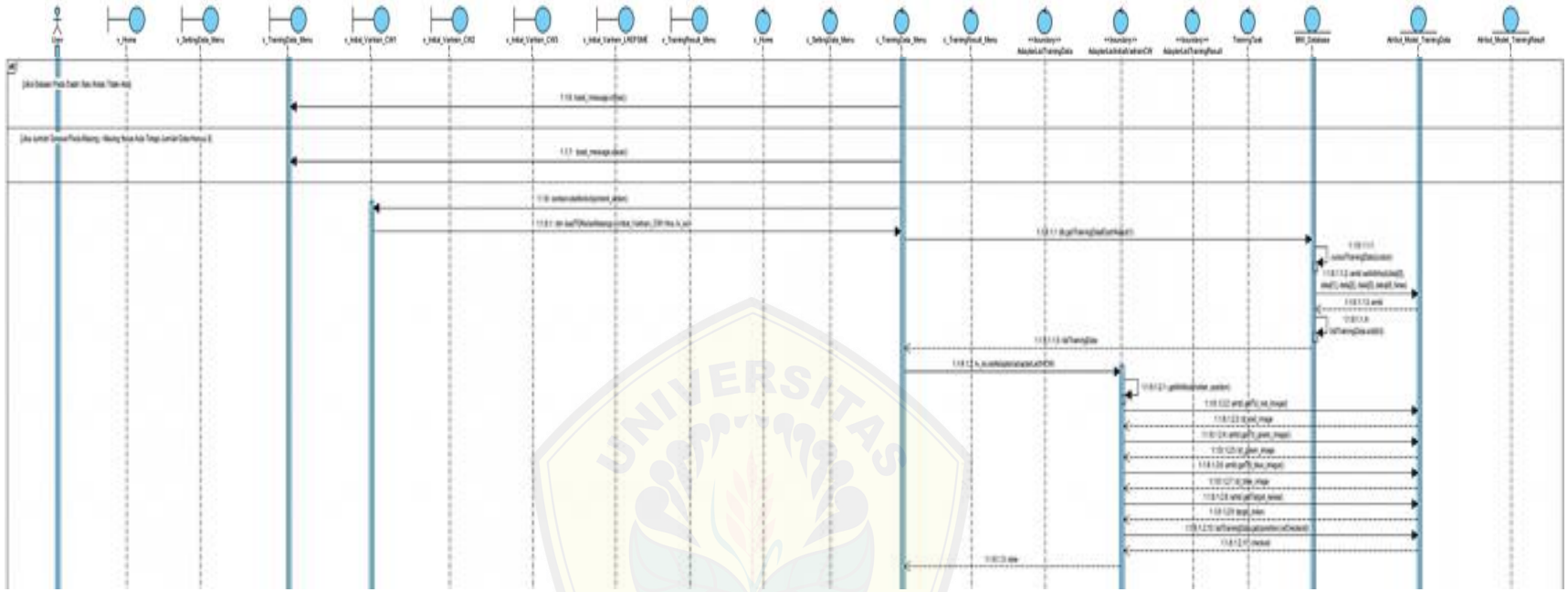
Gambar 4.24 ***Sequence Diagram*** **Melakukan** ***Training Data***



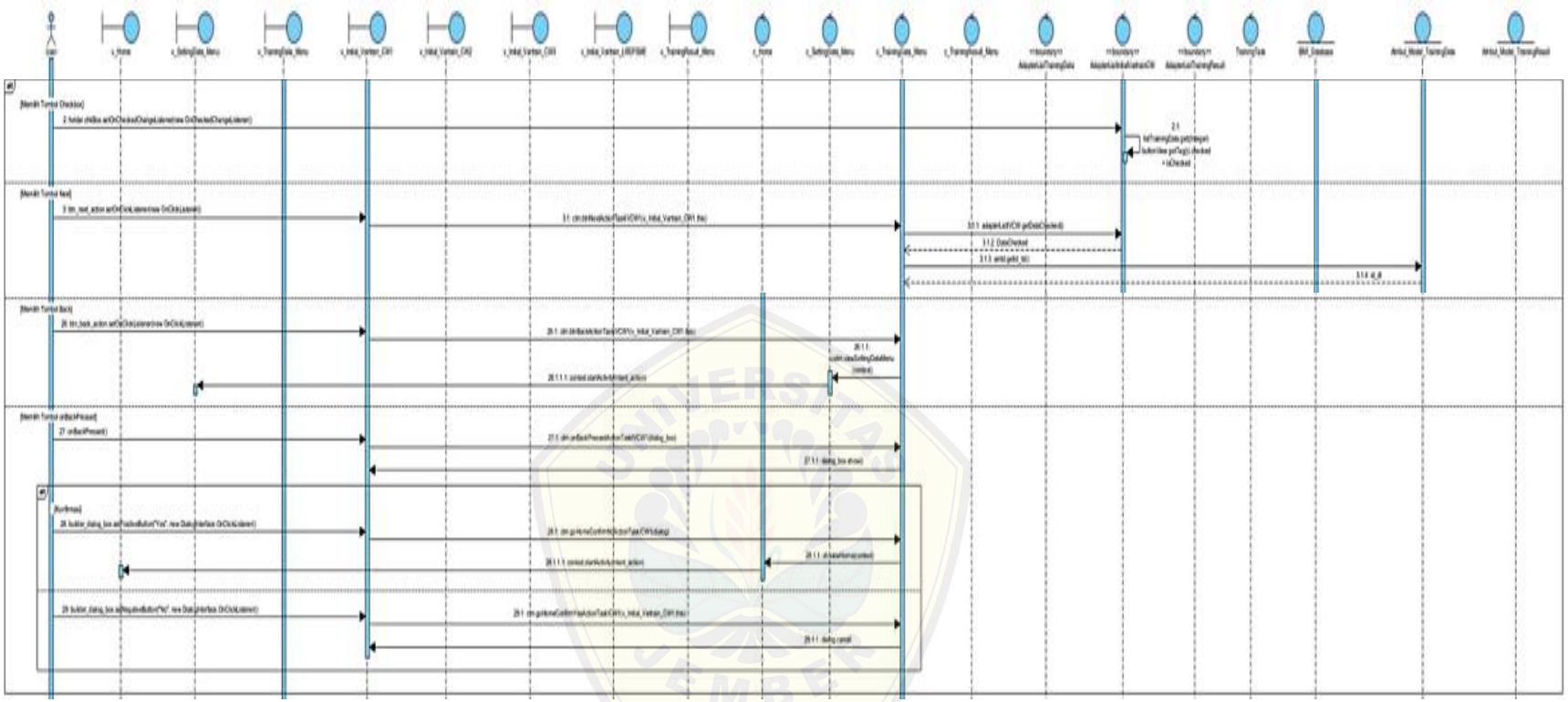
Gambar 4.24.1 Snapshoot Sequence Diagram Alternatif Training Data Jika Dataset Kosong



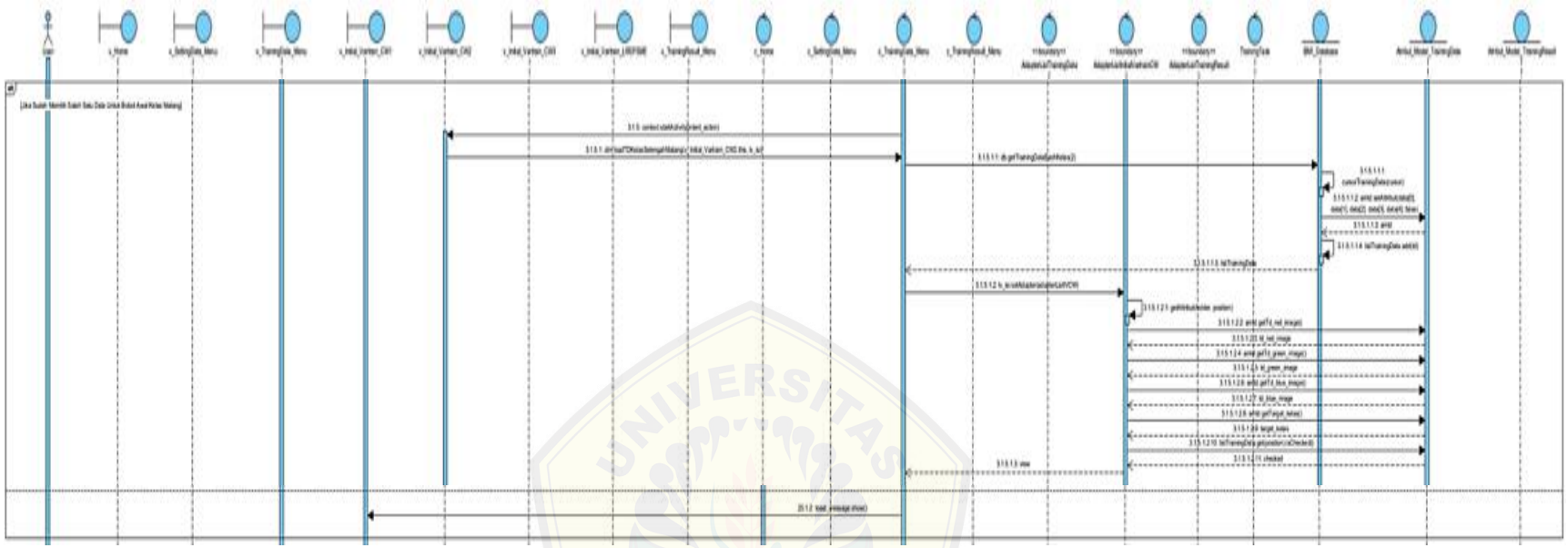
Gambar 4.24.2 *Snapshoot Sequence* Diagram Alternatif *Training Data* Jika *Dataset* Tidak Kosong



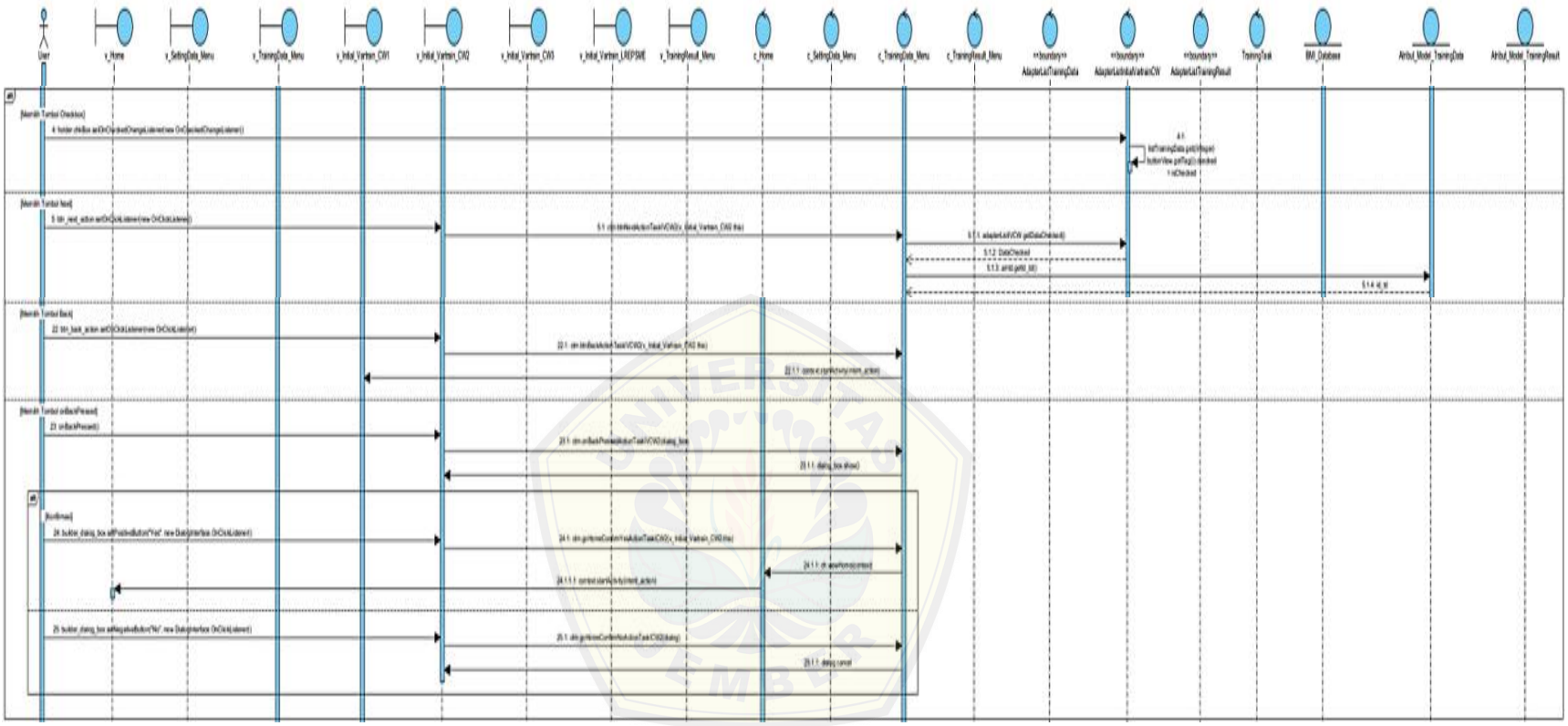
Gambar 4.24.3 *Snapshoot Sequence Diagram* Alternatif *Training Data* Jika *Dataset* Pada Salah Satu Kelas Tidak Ada dan Jika Jumlah *Dataset* Pada Masing – Masing Kelas Ada Tetapi Jumlah Data Hanya 3



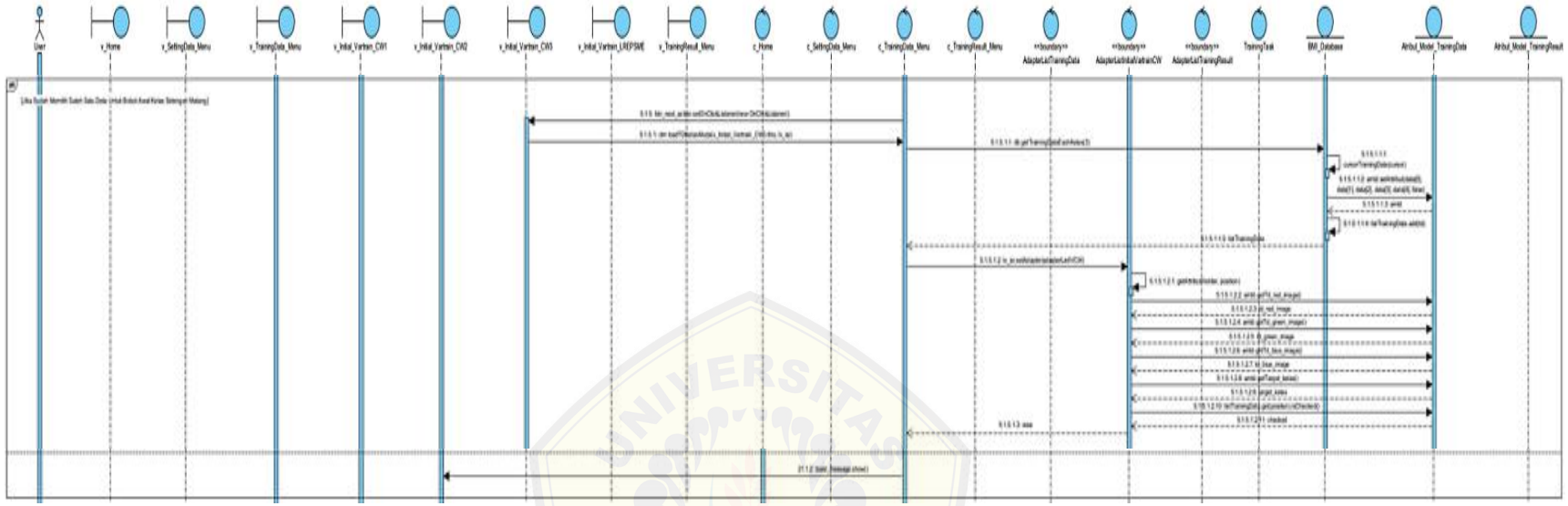
Gambar 4.24.4 *Snapshoot Sequence Diagram* Alternatif Memilih Tombol Pada Halaman *Initial Vartrain CW1*



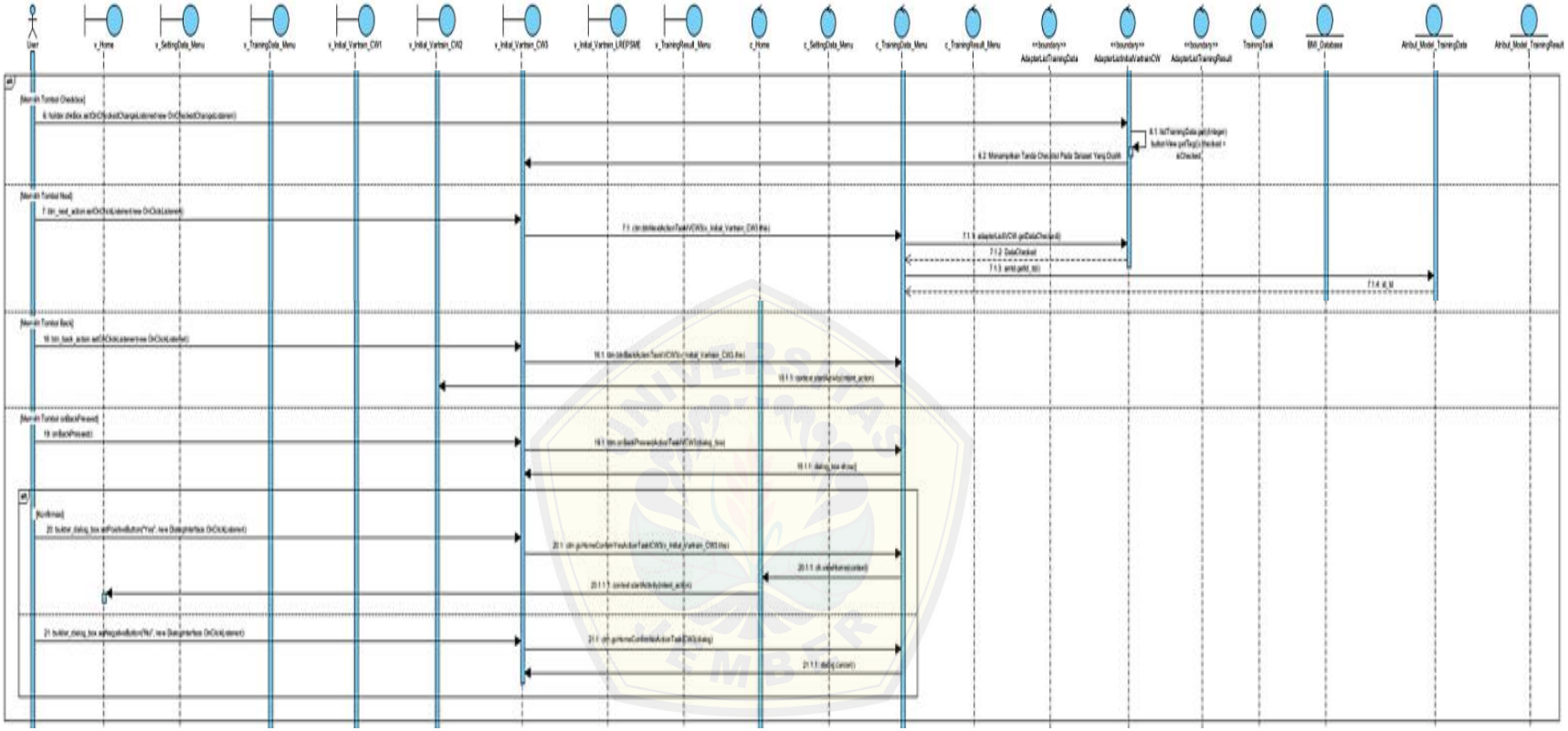
Gambar 4.24.5 *Snapshoot Sequence Diagram* Alternatif Jika Menekan Tombol *Next* Pada Halaman *Initial Vartrain CW1*



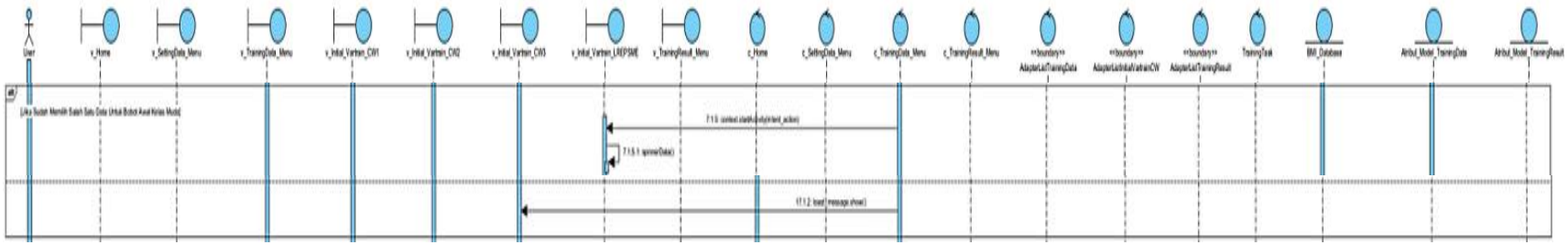
Gambar 4.24.6 *Snapshoot Sequence Diagram* Alternatif Memilih Tombol Pada Halaman *Initial Vartrain CW2*



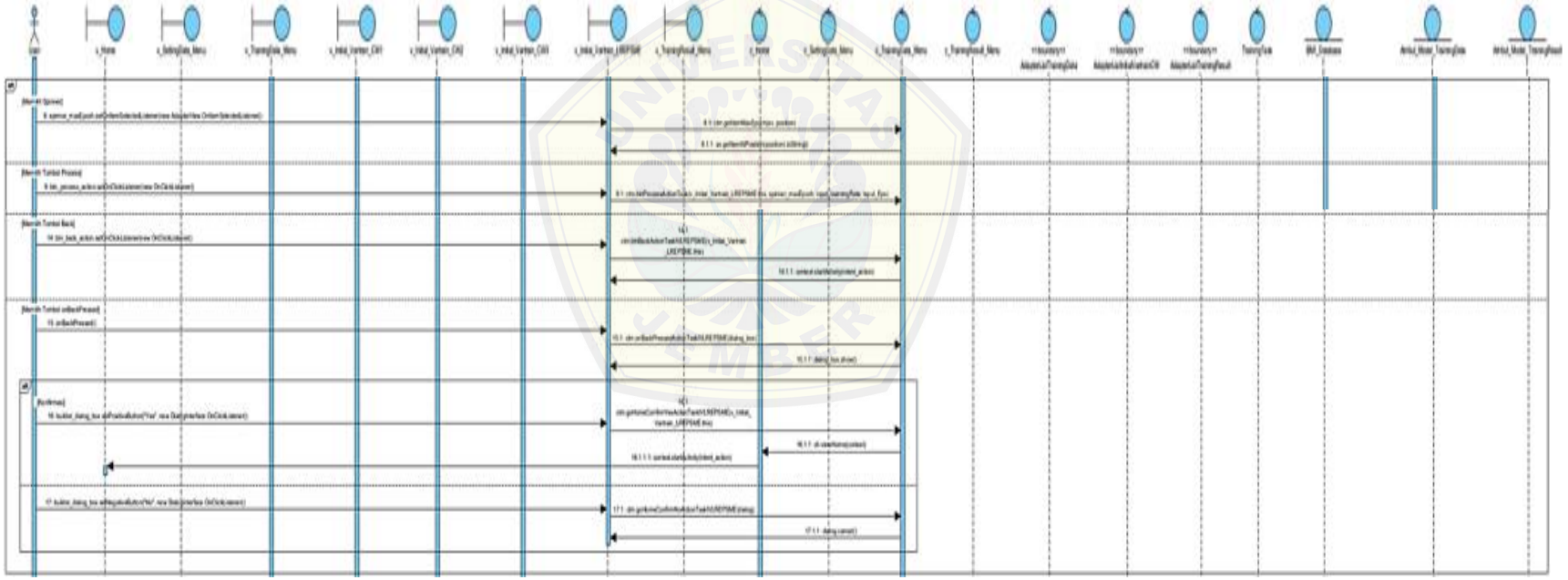
Gambar 4.24.7 *Snapshoot Sequence Diagram* Alternatif Jika Menekan Tombol *Next* Pada Halaman *Initial Vartrain CW2*



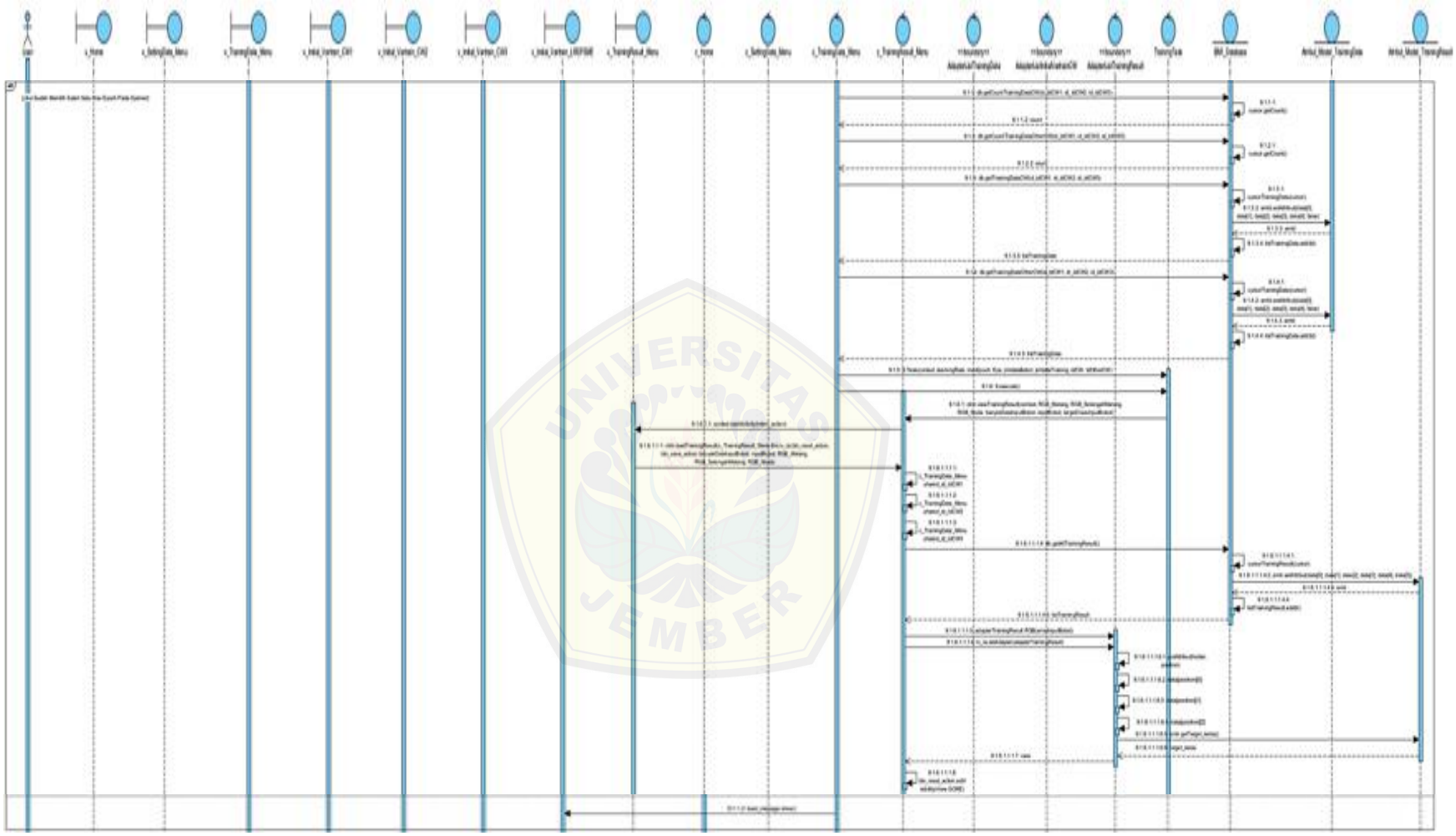
Gambar 4.24.8 *Snapshoot Sequence Diagram* Alternatif Memilih Tombol Pada Halaman *Initial Vartrain CW3*



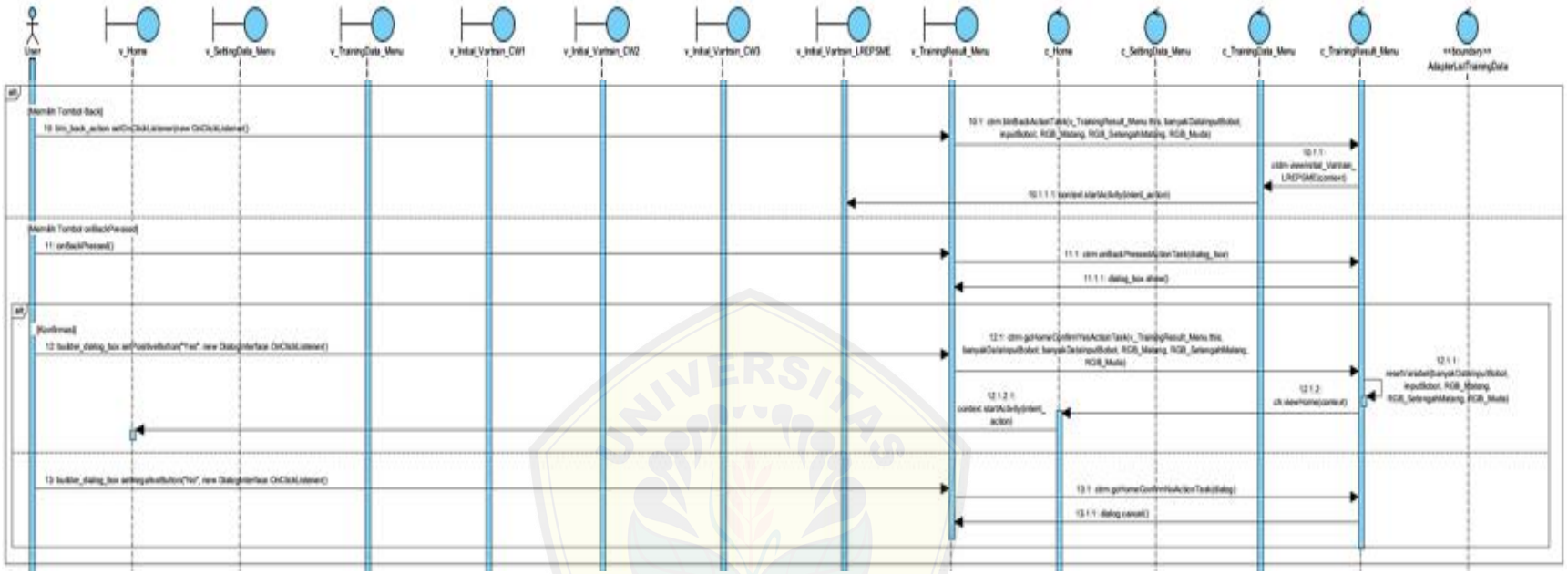
Gambar 4.24.9 *Snapshoot Sequence Diagram* Alternatif Jika Menekan Tombol *Next* Pada Halaman *Initial Vartrain CW3*



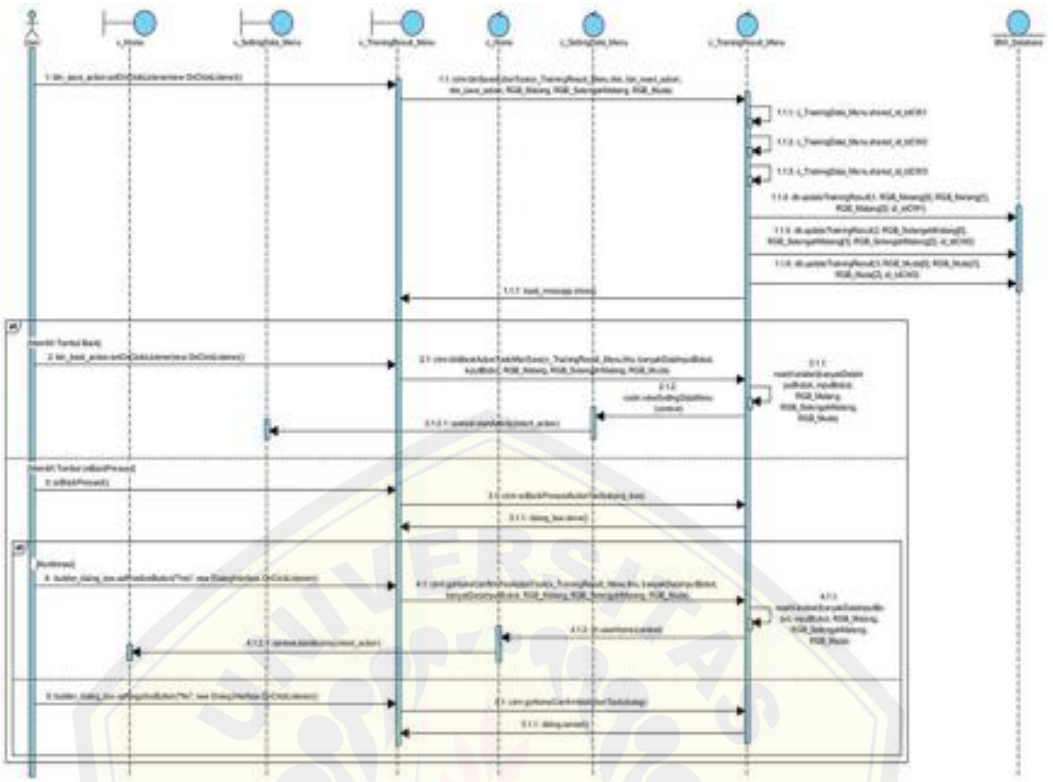
Gambar 4.24.10 *Snapshoot Sequence Diagram* Alternatif Memilih Tombol Pada Halaman *Initial Vartrain LREPSME*



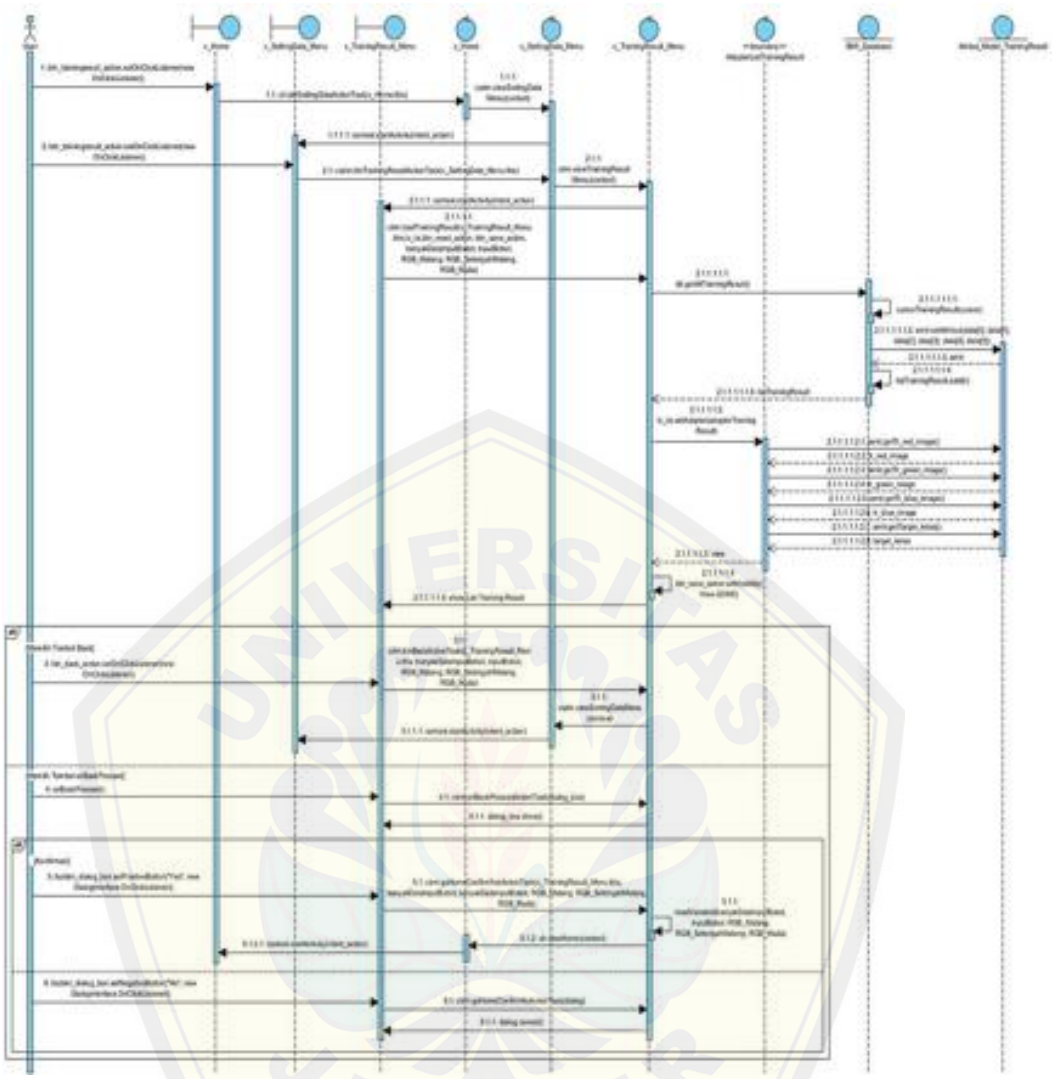
Gambar 4.24.11 *Snapshoot Sequence Diagram* Alternatif Jika Menekan Tombol *Process* Pada Halaman *Initial Vartrain LREPSME*



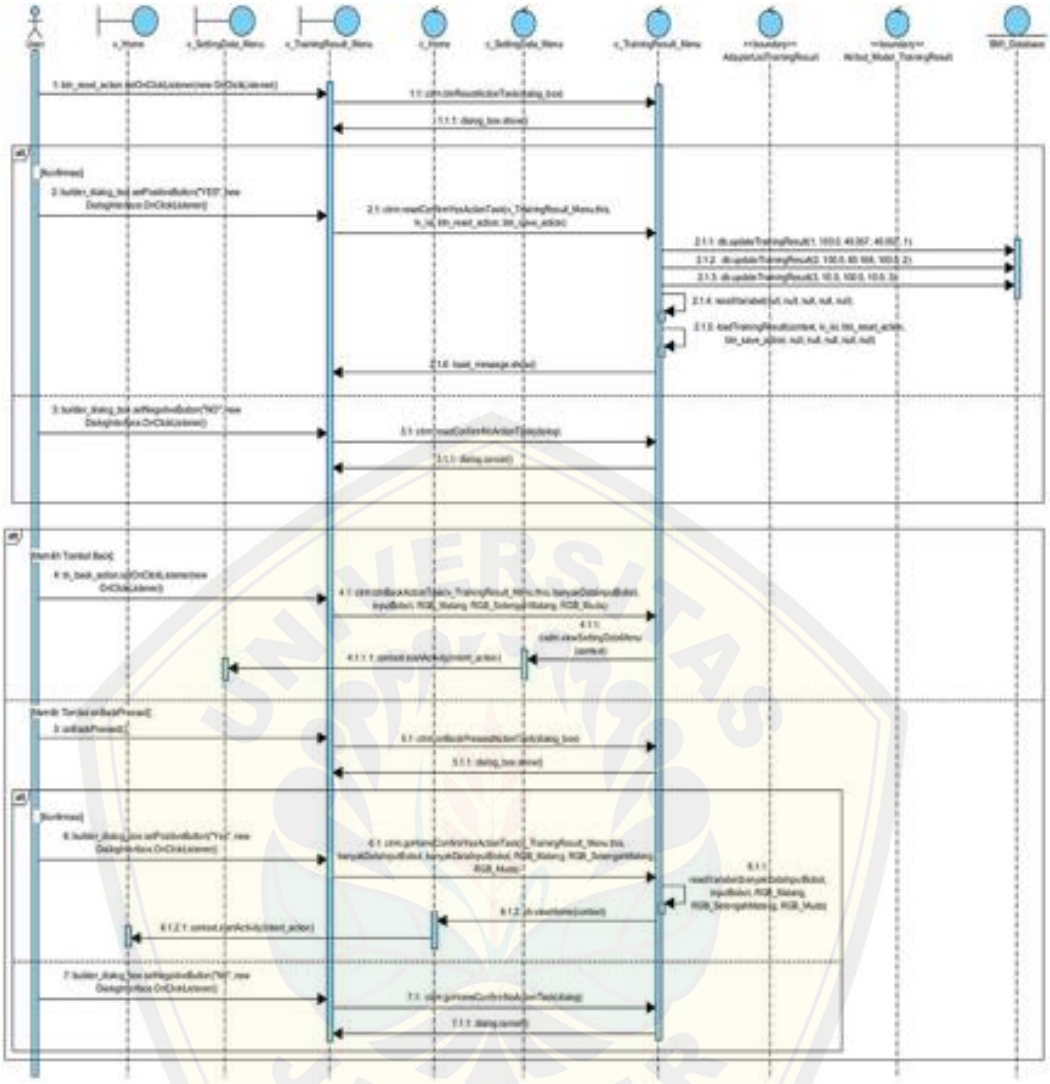
Gambar 4.24.12 ***Snapshoot Sequence Diagram*** **Alternatif Memilih Tombol Pada Halaman** ***Training Result***



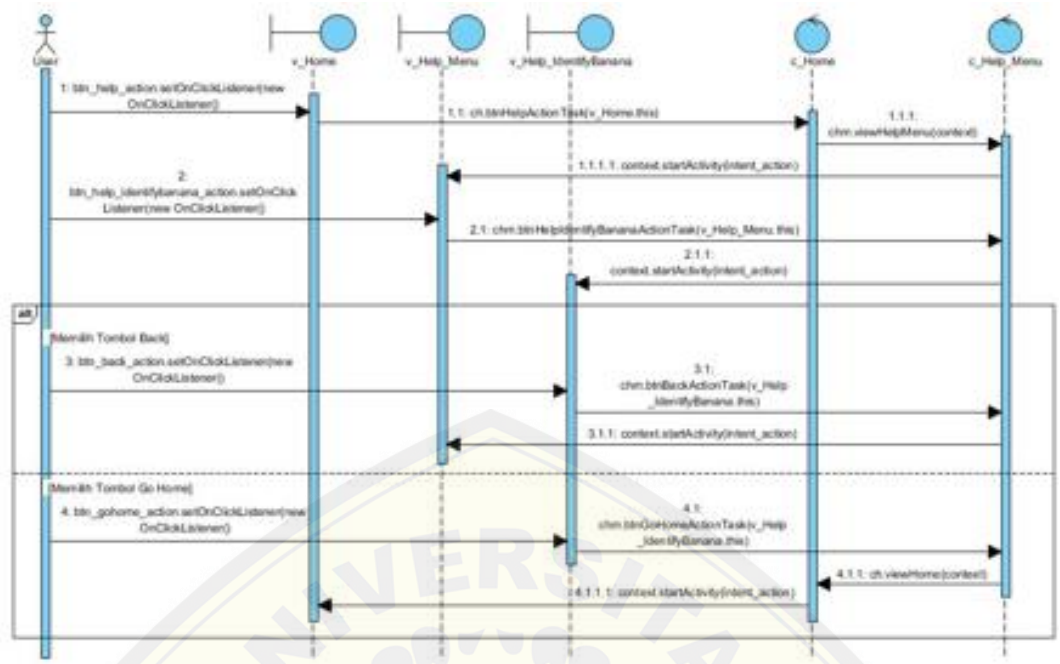
Gambar 4.25 *Sequence Diagram* Menyimpan *Training Result*



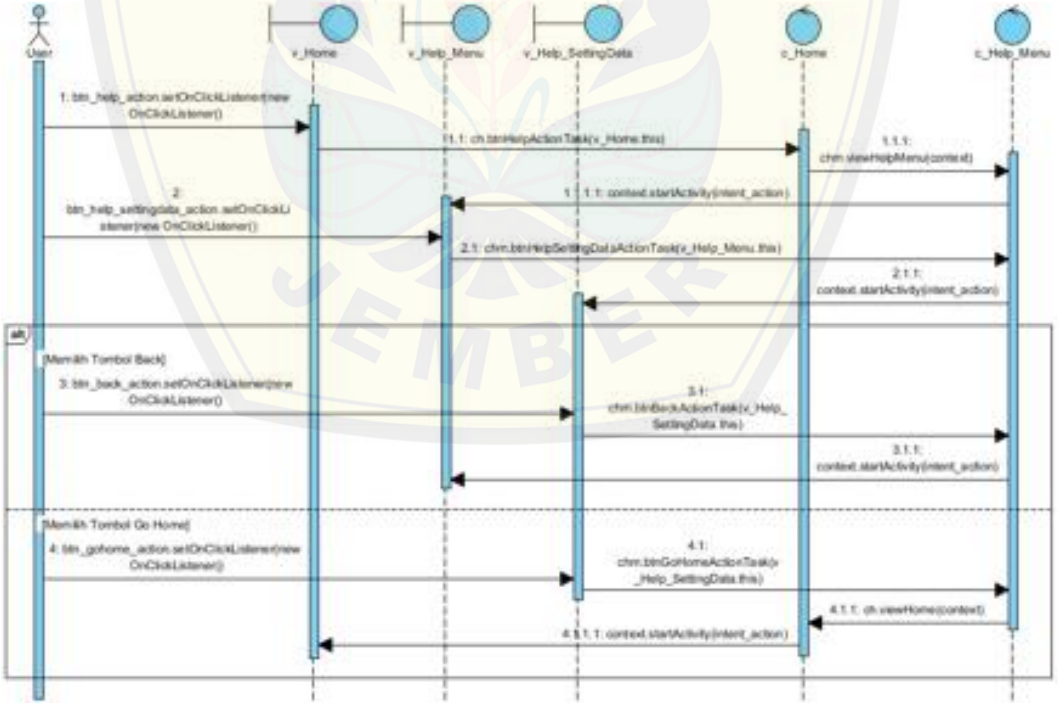
Gambar 4.26 ***Sequence Diagram*** **Melihat** ***Training Result***



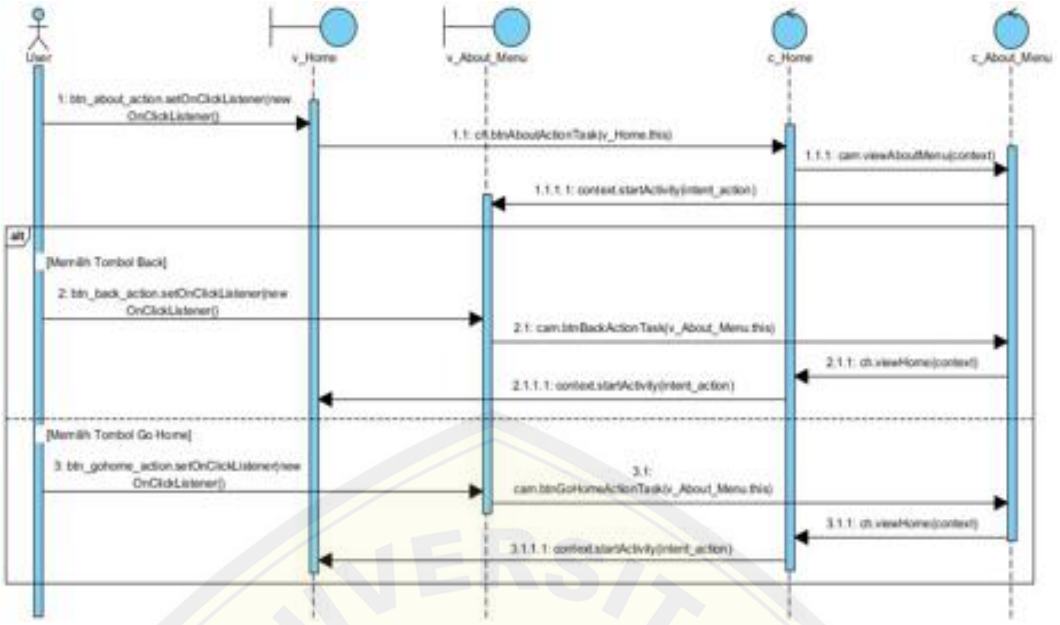
Gambar 4.27 *Sequence Diagram* Melakukan *Reset Data*



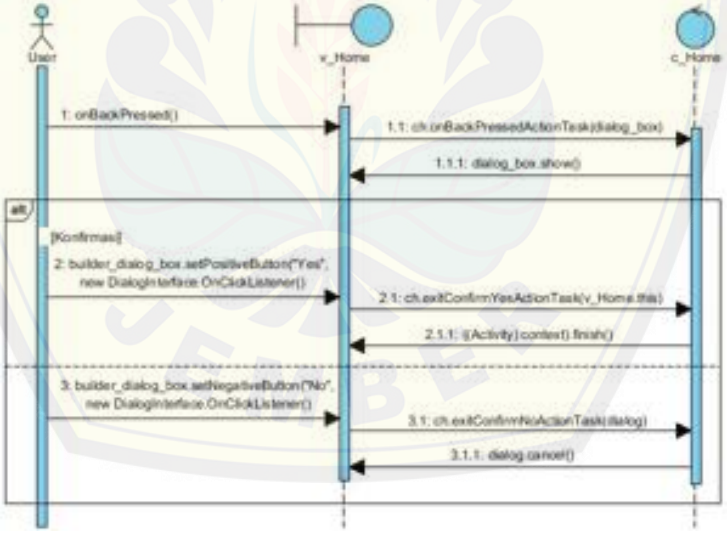
Gambar 4.28 *Sequence Diagram* Melihat *Help How To Identify Banana*



Gambar 4.29 *Sequence Diagram* Melihat *Help How To Setting Data*



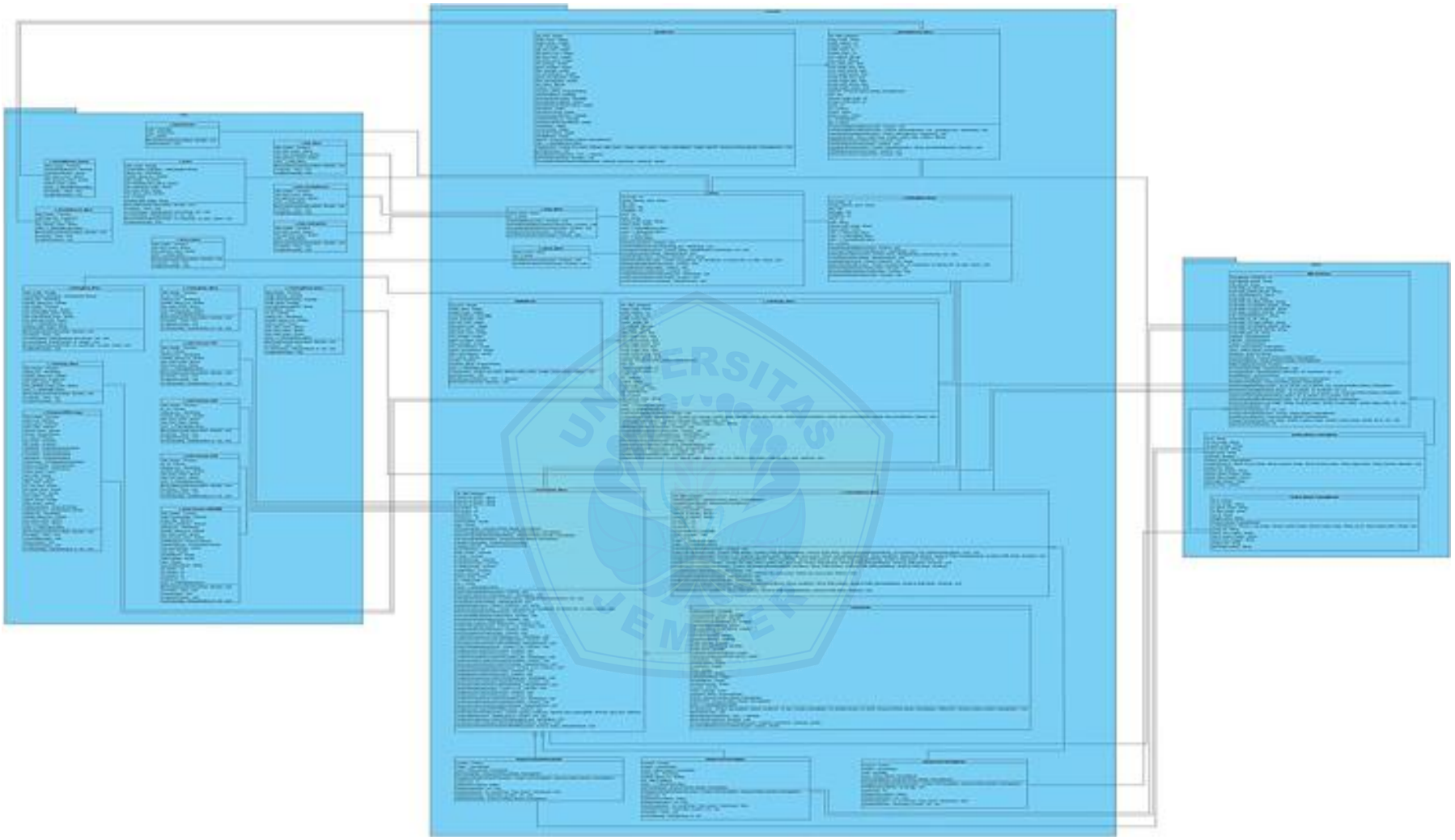
Gambar 4.30 *Sequence Diagram* Melihat *About*



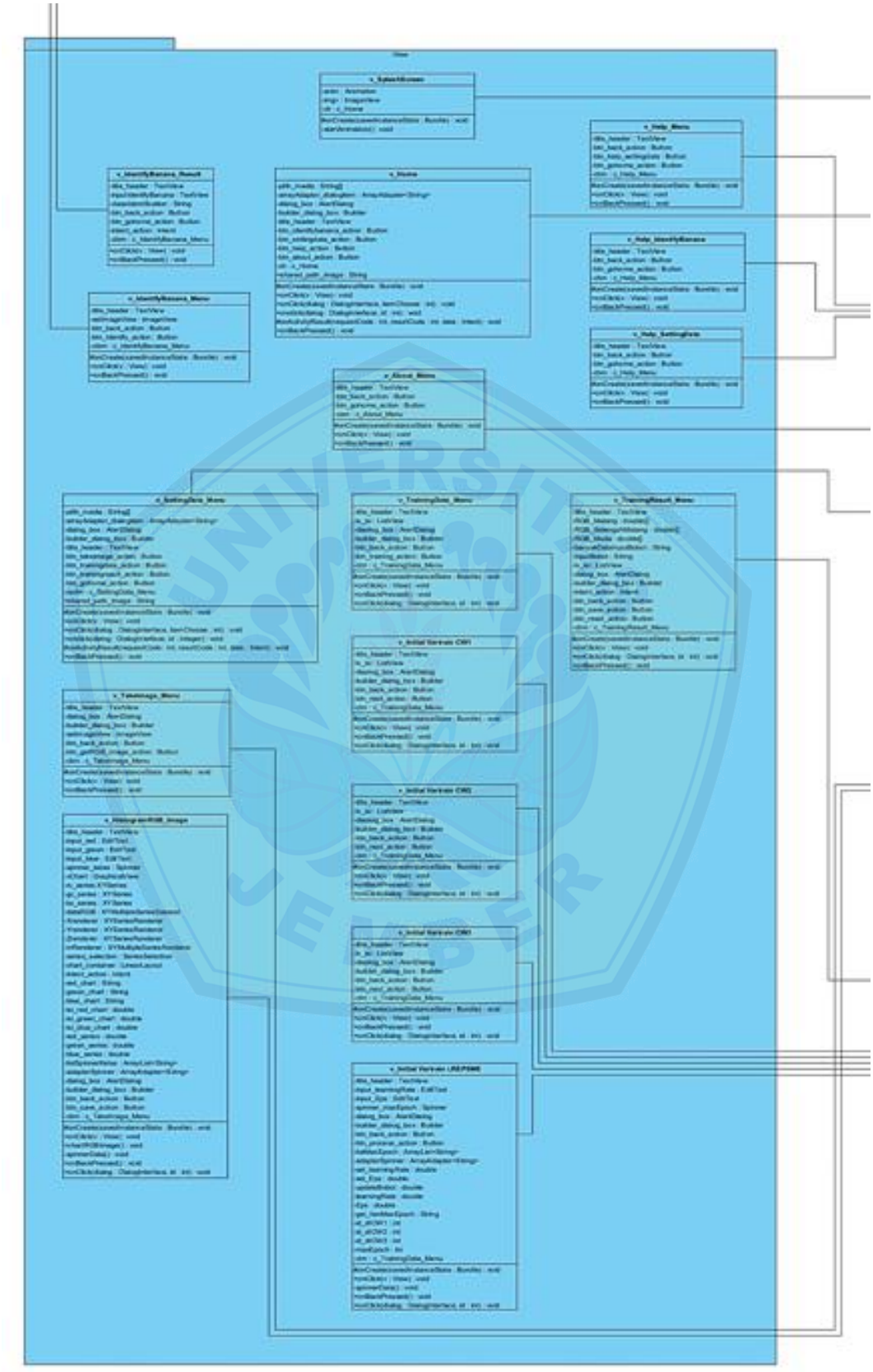
Gambar 4.31 *Sequence Diagram Exit*

4.6. *Class Diagram*

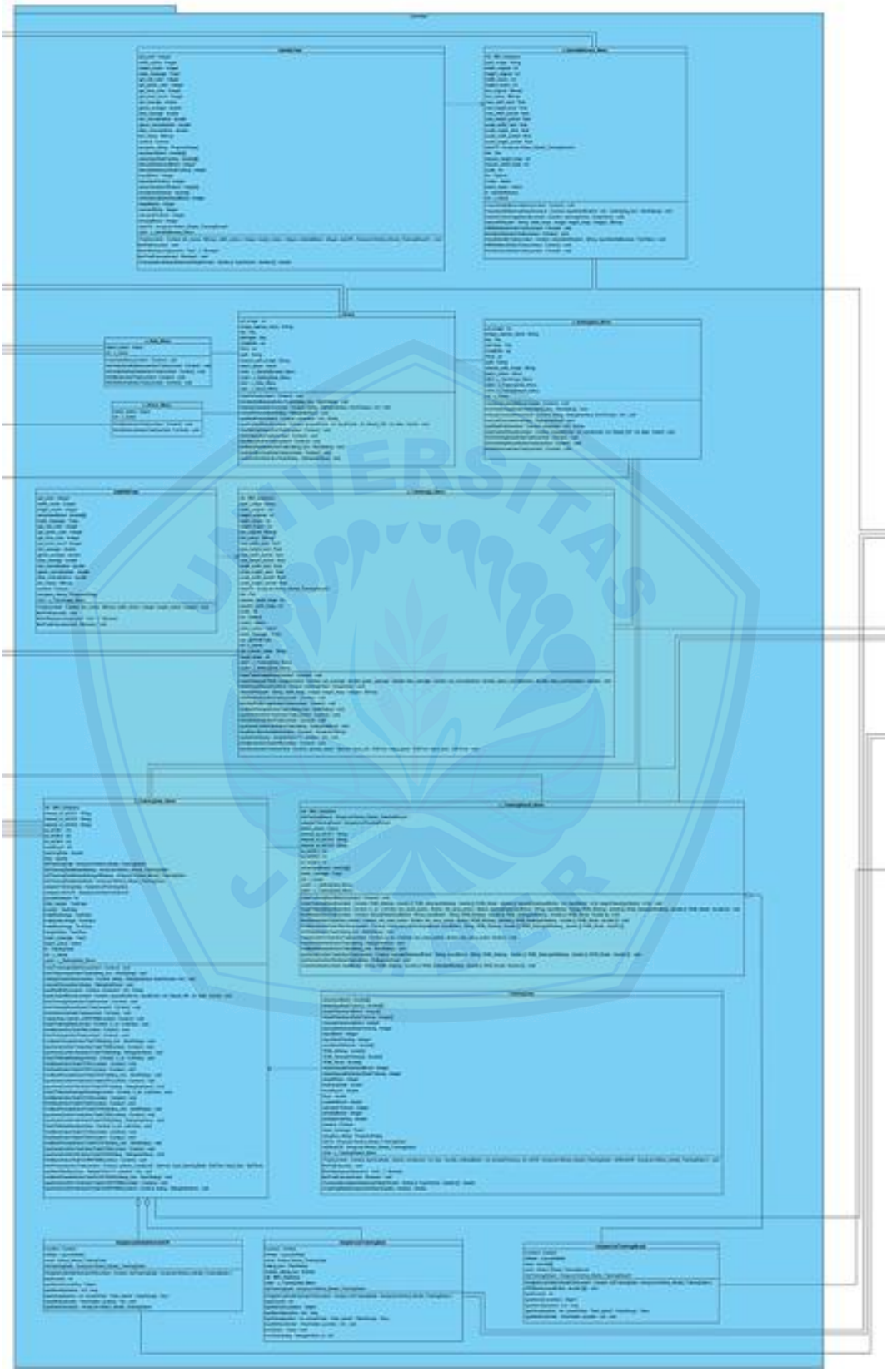
Class Diagram berfungsi untuk menggambarkan *class* - *class* atau objek apa saja yang akan digunakan untuk membuat aplikasi *Banana Maturity Identification* serta relasi atau hubungan yang terjadi pada *class* – *class* atau objek tersebut. *Class diagram* aplikasi *Banana Maturity Identification* dapat dilihat pada Gambar 4.32.



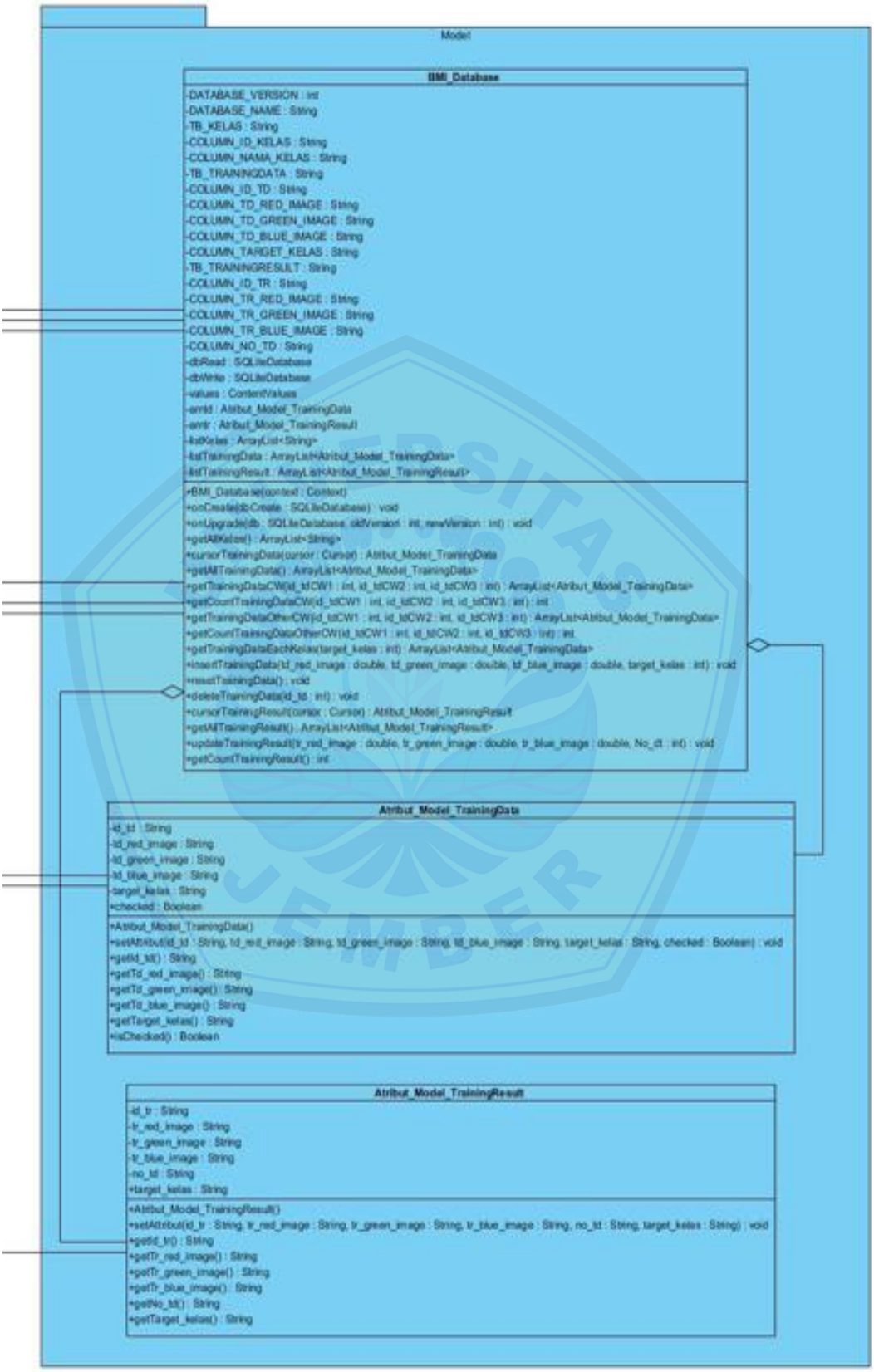
Gambar 4.32 ***Class Diagram Banana Maturity Identification***



Gambar 4.32.1 ***Class Diagram (Package View) Banana Maturity Identification***



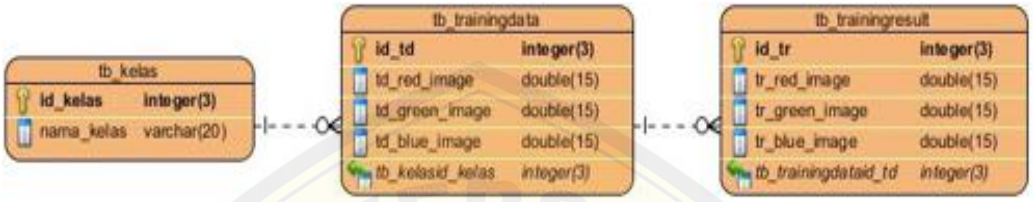
Gambar 4.32.2 ***Class Diagram (Package Controller) Banana Maturity Identification***



Gambar 4.32.3. ***Class Diagram (Package Model) Banana Maturity Identification***

4.7. *Entity Relation Diagram*

ERD (Entity Relation Diagram) berfungsi untuk menggambarkan entitas - entitas apa saja yang akan digunakan pada aplikasi *Banana Maturity Identification* serta relasi atau hubungan yang terjadi pada entitas – entitas tersebut. *ERD (Entity Relation Diagram)* aplikasi *Banana Maturity Identification* dapat dilihat pada Gambar 4.33.



Gambar 4.33 *ERD Banana Maturity Identification*

4.8. Implementasi Perancangan

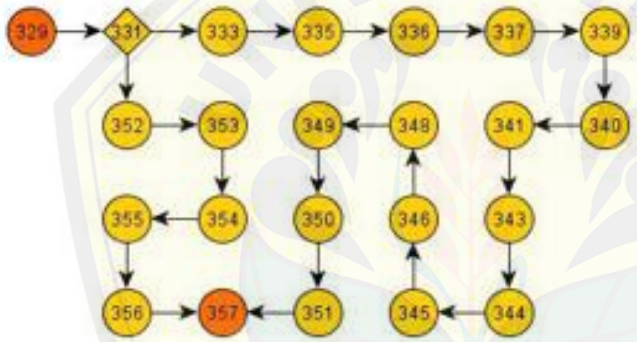
Setelah tahap desain perancangan selesai, tahap selanjutnya dalam penelitian ini yaitu tahap pengimplementasian. Tahap ini merupakan proses pengimplementasian desain perancangan yang telah dilakukan ke dalam bahasa pemograman. Bahasa pemrograman yang dipakai untuk proses pengaplikasian pada perangkat *mobile* android adalah bahasa pemrograman *Java* dan *database* yang digunakan adalah *SQLite*.

4.9. Pengujian

Pengujian meruapakan tahap untuk mengevaluasi aplikasi yang telah dibangun atau dibuat. Proses pengujian dilakukan dengan melakukan pengujian *whitebox* terlebih dahulu, kemudian akan dilanjutkan dengan melakukan pengujian *blackbox.* Pengujian *whitebox* yang dilakukan diawali dengan pembuatan diagram alir dari *listing program* yang diujikan. *Listing Program 1* yang diujikan dapat dilihat pada Gambar 4.34 dan *Listing Program 2* pada Gambar 4.35. Sedangkan untuk diagram alir pengujian 1 dapat dilihat pada Gambar 4.36 dan diagram alir pengujian 2 pada Gambar 4.37.

```
3290 public void btnProcessActionTask(Context context, Spinner spinner_maxEpoch, EditText input_learningRate, EditText input_fps){
330
331     if (!spinner_maxEpoch.getSelectedItem().equals("--Pilih MaxEpoch--")) {
332
333         db = new BMI_Database(context);
334
335         id_tdCW1 = Integer.parseInt(shared_id_tdCW1);
336         id_tdCW2 = Integer.parseInt(shared_id_tdCW2);
337         id_tdCW3 = Integer.parseInt(shared_id_tdCW3);
338
339         learningRate = Double.parseDouble(input_learningRate.getText().toString()); // nilai alpha
340         maxEpoch     = Integer.parseInt(spinner_maxEpoch.getSelectedItem().toString()); // nilai iterasi
341         fps          = Double.parseDouble(input_fps.getText().toString()); // nilai min alpha
342
343         int jmldataBobot     = db.getCountTrainingDataCW(id_tdCW1, id_tdCW2, id_tdCW3);
344         int jmldataTraining  = db.getCountTrainingDataOtherCW(id_tdCW1, id_tdCW2, id_tdCW3);
345         ArrayList<Atribut_Model_TrainingData> tdCW      = db.getTrainingDataCW(id_tdCW1, id_tdCW2, id_tdCW3);
346         ArrayList<Atribut_Model_TrainingData> tdOtherCW = db.getTrainingDataOtherCW(id_tdCW1, id_tdCW2, id_tdCW3);
347
348         tt = new TrainingTask();
349         tt.Task(context, learningRate, maxEpoch, fps, jmldataBobot, jmldataTraining, tdCW, tdOtherCW);
350         tt.execute();
351     }
352     else {
353         toast_message = Toast.makeText(context,"Pilih Max Epoch Dulu !", Toast.LENGTH_LONG);
354         toast_message.setGravity(Gravity.CENTER_VERTICAL, 0, 50);
355         toast_message.show();
356     }
357 }
```

Gambar 4.34 *Listing Program 1*



CC = Edge – Node + 2
CC = 23 – 23 + 2
CC = 2

Gambar 4.35 Diagram Alir Pengujian 1

Jalur *basis set* pada pengujian *listing program 1* adalah 329 – 331 – 333 – 335 – 336 – 337 – 339 – 340 – 341 – 343 – 344 – 345 – 346 – 348 – 349 – 350 – 351 – 357 dan 329 – 331 – 352 – 353 – 354 – 355 – 356 - 357 . Pengujian kebenaran kedua jalur tersebut dapat dilihat pada Tabel 4.26 dan Tabel 4.27.

Tabel 4.26 *Test Case* Pengujian Jalur

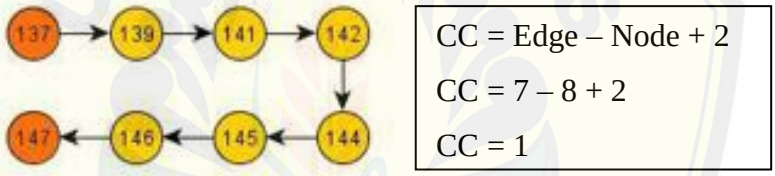
Test Case	Jika *user* sudah memilih *max epoch*.
Target yang diharapkan	Mengambil *dataset* dan melakukan proses *training data*.
Hasil Pengujian	Benar
Path / Jalur	329 – 331 – 333 – 335 – 336 – 337 – 339 – 340 – 341 – 343 – 344 – 345 – 346 – 348 – 349 – 350 – 351 – 357

Tabel 4.27 *Test Case* Pengujian Jalur 2

Test Case	Jika *user* belum memilih *max epoch.*
Target yang diharapkan	Menampilkan Pesan “Pilih Max Epoch Dulu !”.
Hasil Pengujian	Benar
Path / Jalur	329 – 331 – 352 – 353 – 354 – 355 – 356 - 357

```
137@    public void btnIdentifyActionTask(Context context){
138
139         db = new BMI_Database(context);
140
141         jmldataBobot = db.getCountTrainingResult();
142         dataTR = db.getAllTrainingResult();
143
144         it = new IdentifyTask();
145         it.Task(context, bm_resize, width_resize, height_resize, jmldataBobot, dataTR);
146         it.execute();
147     }
```

Gambar 4.36 *Listing Program 2*



Gambar 4.37 Diagram Alir Pengujian 2

Jalur *basis set* pada pengujian *listing program 2* adalah 137 – 139 – 141 – 142 – 144 – 145 – 146 - 147. Pengujian kebenaran jalur tersebut dapat dilihat pada Tabel 4.28.

Tabel 4.18 *Test Case* Pengujian Jalur

Test Case	Jika *user* menekan tombol *Identify*.
Target yang diharapkan	Melakukan proses identifikasi pada citra digital.
Hasil Pengujian	Benar
Path / Jalur	137 – 139 – 141 – 142 – 144 – 145 – 146 – 147.

Adapun pengujian *black box* yang dilakukan pada aplikasi *Banana Maturity Identification*. Tabel Pengujian *black box* dapat dilihat pada Tabel 4.29.

Tabel 4.19 Pengujian *black box* aplikasi

No	Menu	Fungsi	Kasus	Hasil	Ket.
1.	*Identify Banana*.	Untuk mengambil citra digital.	Ketika *user* menekan menu *identify banana*.	Menampilkan dialog pilihan media *camera* atau *sd card.*	OK.
			Ketika *user* memilih media *camera.*	Menampilkan *camera.*	OK.
			Ketika *user* memilih media *camera* tetapi pada *smartphone* terdapat beberapa aplikasi pengambilan citra digital.	Menampilkan dialog pilihan beberapa aplikasi media *camera.*	OK.
			Ketika *user* menakan tombol *capture* pada *camera*	Mengambil citra digital kemudian men-*scale* 200 x 300 px dan menampilkannya ke layar.	OK.
			Ketika *user* memilih media *sd card*	Menampilkan dialog pilihan aplikasi *file manager* untuk mengakses citra digital.	OK.
			Ketika *user* telah memilih citra digital pada salah satu *file manager*.	Mengambil citra digital kemudian men-*scale* 200	OK.

				x 300 px dan menampilkannya ke layar.	
		Untuk melakukan identifikasi buah pisang.	Ketika *user* menekan tombol *Identify*.	Mengambil nilai RGB citra digital dan melakukan identifikasi berdasarkan bobot hasil *training data* menggunakan *Learning Vector Quantization* dan menampilkan hasilnya.	OK.
2.	*Take Image*	Untuk mengambil citra digital.	Ketika *user* menekan menu *identify banana*.	Menampilkan dialog pilihan media *camera* atau *sd card.*	OK.
			Ketika *user* memilih media *camera.*	Menampilkan *camera.*	OK.
			Ketika *user* memilih media *camera* tetapi pada *smartphone* terdapat beberapa aplikasi pengambilan citra digital.	Menampilkan dialog pilihan beberapa aplikasi media *camera.*	OK.
			Ketika *user* menakan tombol *capture* pada *camera*	Mengambil citra digital kemudian men-*scale* 200 x 300 px dan menampilkann	OK.

				ya ke layar.	
			Ketika *user* memilih media *sd card*	Menampilkan dialog pilihan aplikasi *file manager* untuk mengakses citra digital.	OK.
			Ketika *user* telah memilih citra digital pada salah satu *file manager*.	Mengambil citra digital kemudian men-*scale* 200 x 300 px dan menampilkann ya ke layar.	OK.
		Untuk melihat histogram RGB *image*.	Ketika *user* menekan menu tombol *Get RGB.*	Menampilkan histogram RGB *image.*	OK.
		Untuk menyimpan nilai RGB *image* sebagai *dataset*.	Ketika *user* menekan tombol *Save* dan sudah memilih target kelas pada *spinner*.	Menyimpan nilai RGB *image* kedalam *database* dan menampilkan hasilnya pada halaman *training data*.	OK.
			Ketika *user* menekan tombol *Save* dan belum memilih target kelas pada *spinner*.	Menampilkan pesan “Pilih Kelas Dulu !”	OK.
3.	*Training Data*	Untuk melihat *list dataset*.	Ketika *dataset* masih kosong.	Menampilkan pesan “Dataset Kosong, Silahkan Isi Dataset Terlebih	OK.

				Dahulu !”	
		Untuk melakukan *training data*.	Ketika *user* menekan tombol *Training* tetapi *dataset* masih kosong.	Menampilkan pesan “Dataset Kosong, Silahkan Isi Dataset Terlebih Dahulu !”.	OK.
			Ketika *user* menekan tombol *Training* tetapi jumlah *dataset* pada masing - masing kelas ada tetapi jumlah data hanya 3.	Menampilkan pesan “Pastikan Jumlah Dataset Lebih Dari 3 Data !”.	OK.
			Ketika *user* menekan tombol *Training* tetapi *dataset* pada salah satu kelas tidak ada.	Menampilkan pesan “Dataset Pada Masing - Masing Kelas Harus Ada !”.	OK.
			Ketika menekan tombol *Training*.	Menampilkan *dataset* kelas matang pada halaman *Initial Vartrain CW1*.	OK.
			Ketika menekan tombol *Next* tetapi belum memilih salah satu *dataset* sebagai bobot awal kelas matang.	Menampilkan pesan “Pilih Bobot Dulu !”.	OK.
			Ketika menekan tombol *Next* tetapi sudah memilih salah satu *dataset*	Menampilkan *dataset* kelas setengah matang pada	OK.

			sebagai bobot awal kelas matang.	halaman *Initial Vartrain CW2*.	
			Ketika menekan tombol *Next* tetapi belum memilih salah satu *dataset* sebagai bobot awal kelas setengah matang.	Menampilkan pesan "Pilih Bobot Dulu !".	OK.
			Ketika menekan tombol *Next* tetapi sudah memilih salah satu *dataset* sebagai bobot awal kelas setengah matang.	Menampilkan *dataset* kelas muda pada halaman *Initial Vartrain CW3*.	OK.
			Ketika menekan tombol *Next* tetapi belum memilih salah satu *dataset* sebagai bobot awal kelas muda.	Menampilkan pesan "Pilih Bobot Dulu !".	OK.
			Ketika menekan tombol *Next* tetapi sudah memilih salah satu *dataset* sebagai bobot awal kelas muda.	Menampilkan halaman *Initial Vartrain LREPSME*.	OK.
			Ketika menekan tombol *Process* tetapi belum memilih salah satu *max epoch*.	Menampilkan pesan "Pilih Max Epoch Dulu !".	OK.
			Ketika menekan tombol *Process* tetapi sudah memilih salah satu *max epoch*.	Mengambil dataset dan melakukan proses *training data*	OK.

				menggunakan *Learning Vector Quantization* dan menampilkan hasilnya pada halaman *training result.*	
		Untuk menyimpan *training result.*	Ketika menekan tombol *Save*.	Menyimpan *training result* ke dalam *database* dan mengaktifkan tombol *reset*.	OK.
4.	*Training Result*	Untuk melihat *training result.*	Ketika *user* menekan submenu *Training Result.*	Menampilkan *list training result* pada halaman *training result.*	OK.
		Untuk melakukan *reset data*.	Ketika *user* menekan tombol *Reset* dan menekan tombol *Yes.*	Menghapus semua *dataset* dan meng-*update training result* ke *default* aplikasi.	OK.
			Ketika menekan tombol *Reset* dan menekan tombol *No.*	Kembali menampilkan halaman *training result.*	OK.
5.	*Help How To Identify Banana*	Untuk melihat *help how to identify banana*.	Ketika *user* menekan submenu *Help How To Identify Banana.*	Menampilkan halaman *Help How To Identify Banana.*	OK

6.	*Help How To Setting Data*	Untuk melihat *help how to identify banana*.	Ketika *user* menekan submenu *Help How To Setting Data*.	Menampilkan halaman *Help How To Setting Data.*	OK.
7	*About*	Untuk melihat *About.*	*User* menekan menu *About*	Menampilkan halaman *About*	OK.
8	*Exit*	Untuk keluar dari aplikasi.	Ketika menekan tombol *onBackPressed* dan menekan tombol *Yes*.	Menutup semua aktivitas aplikasi ditutup.	OK.
			Ketika menekan tombol *onBackPressed* dan menekan tombol *No*.	Menampilkan halaman *Home*.	OK.

BAB 5. HASIL DAN PEMBAHASAN

Bab ini memaparkan tentang pisang mas kirana, aplikasi *Banana Maturity Identification* yang mengimplementasikan *Learning Vector Quantization (LVQ)* dan pengujian untuk mengetahui akurasi aplikasi dalam mengidentifikasi kematangan buah pisang. Pembahasan yang dipaparkan diutamakan dalam hal implementasi algoritma *Learning Vector Quantization (LVQ)* dan pengujian identifikasi.

5.1. Pisang Mas Kirana

Berdasarkan SK Menteri Pertanian No.516/KPTS/SR/.120/12/2005, pisang Mas Kirana merupakan salah satu varietas pisang dengan kualitas yang baik diantara varietas pisang yang lain di Indonesia. Pisang Mas Kirana ini menjadi salah satu komoditas unggulan di Kabupaten Lumajang berdasarkan surat keputusan Bupati Lumajang No.188.45/408/427.12/2006. Sejak itulah masyarakat Lumajang mulai percaya diri menanam pisang Mas Kirana sebagai penunjang kebutuhan hidup. Salah satunya Kelompok Tani Raja Mas Desa Kandang Tepus, Kecamatan Senduro yang bergerak dibidang ini. Berdasarkan hasil wawancara yang dilakukan di tempat penelitian ini, tingkat kematangan pisang Mas Kirana dibagi menjadi tiga kategori atau kelas yaitu matang, setengah matang (kemuning), dan muda. Adapun deskripsi dari tiap kategori atau kelas, yaitu :

a. Kategori Matang

   Pisang Mas Kirana dengan keadaan warna kulit pisang kuning agak pekat diseluruh bagian buah pisang.

b. Kategori Setengah Matang (Kemuning)

   Pisang Mas Kirana dengan keadaan warna kulit pisang kekuningan disebagian atau seluruh bagian buah pisang dan warna hijau pada kulit pisang masih terlihat.

c. Kategori Muda

   Pisang Mas Kirana dengan keadaan warna kulit pisang masih hijau.

5.2. Aplikasi *Banana Maturity Identification*

Aplikasi *Banana Maturity Identification* yang dibangun pada penelitian ini terdiri atas beberapa fitur yang mengimplementasikan *Learning Vector Quantization (LVQ)*. Beberapa fitur yang tersedia pada aplikasi dapat digunakan *user* untuk melakukan proses identifikasi maupun *training data*.

5.2.1. Tampilan *Splash Screen*

Tampilan *Splash Screen* merupakan tampilan awal dari aplikasi *Banana Maturity Identification* sebelum aplikasi menampilkan halaman *Home*. Tampilan *Splash Screen* pada aplikasi *Banana Maturity Identification* dapat dilihat pada Gambar 5.1.

Gambar 5.1 Tampilan *Splash Screen*

5.2.2. Tampilan *Home*

Tampilan *Home* merupakan tampilan utama dari aplikasi, dimana pada tampilan *Home* terdapat beberapa menu utama yaitu *Identify Banana, Setting Data, Help, About, dan Exit*. Tampilan *Home* pada aplikasi *Banana Maturity Identification* dapat dilihat pada Gambar 5.2.

Gambar 5.2 Tampilan *Home*

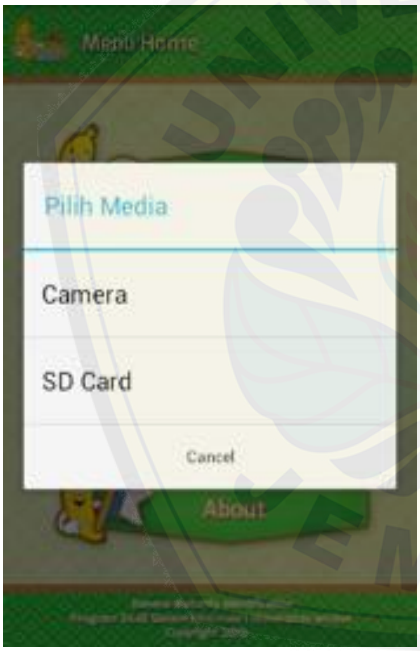
5.2.3. Tampilan *Identify Banana*

Tampilan *Identify Banana* merupakan tampilan untuk menampilkan citra digital yang telah diinputkan untuk proses identifikasi. Tampilan *Identify Banana* dapat dilihat pada Gambar 5.3.

Gambar 5.3 Tampilan Identify Banana

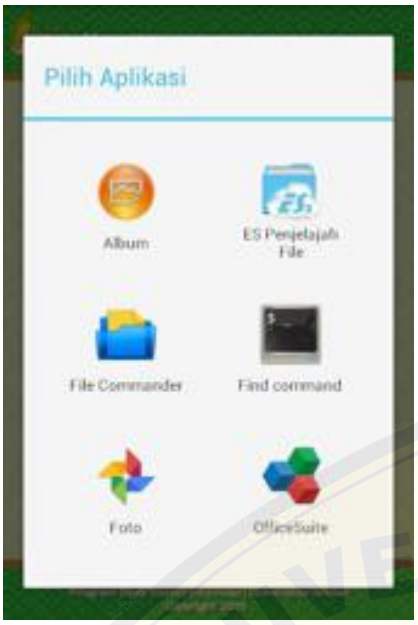
5.2.3.1. Tampilan Pilihan Media *Input* Citra Digital untuk Identifikasi

Sebelum tampilan *Identify Banana* ditampilkan, *user* diarahkan untuk memilih media untuk mengambil atau menginputkan citra digital melalui *camera* ataupun media penyimpanan *(sd card),* untuk tampilannya dapat dilihat pada Gambar 5.4. Tampilan pilihan media *input* citra digital melalui *Camera* dapat dilihat pada Gambar 5.5. Sedangkan, Tampilan pilihan media *input* citra digital melalui *SD Card* dapat dilihat pada Gambar 5.6. Setelah memilih aplikasi *file manager* untuk mengakses citra digital pada media penyimpanan, kemudian *user* dapat memilih citra digital yang akan diinputkan. Tampilannya dapat dilihat pada Gambar 5.7.



Gambar 5.4 Tampilan Pilihan Media *Input* Citra Digital

Gambar 5.5 Tampilan Pilihan Media *Input* Citra Digital Melalui *Camera*



Gambar 5.7 Tampilan Pilihan Media *Input* Citra Digital Melalui *SD Card*

Gambar 5.6 Tampilan Memilih Citra Digital Yang Akan Diinputkan

5.2.3.2. Tampilan *Identify Banana Result*

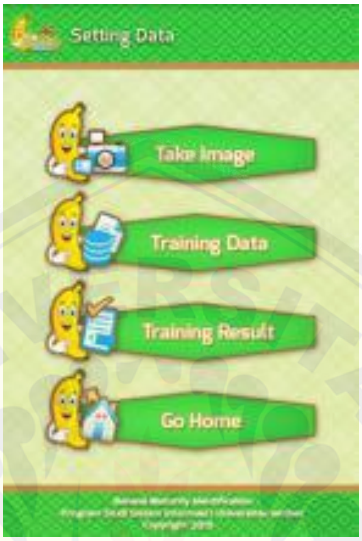
Tampilan *Identify Banana Result* merupakan tampilan untuk menampilkan hasil identifikasi pada buah pisang. Tampilan *Identify Banana Result* dapat dilihat pada Gambar 5.8

.

Gambar 5.8 Tampilan *Identify Banana Result*

5.2.4. Tampilan *Setting Data*

Tampilan *Setting Data* merupakan tampilan untuk menampilkan submenu dalam melakukan pengaturan data. Submenu yang terdapat pada tampilan *Setting Data* adalah submenu *Take Image*, *Training Data*, *Training Result*, dan *Go Home*. Tampilan *Setting Data* dapat dilihat pada Gambar 5.9.



Gambar 5.9 Tampilan *Identify Banana Result*

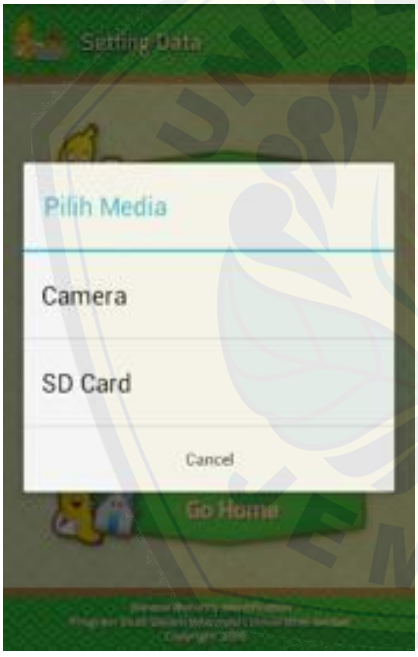
5.2.4.1. Tampilan *Take Image*

Tampilan *Take Image* merupakan tampilan untuk menampilkan citra digital yang telah diinputkan untuk proses menyimpan RGB citra digital sebagai *dataset* ke dalam *database*. Tampilan *Take Image* dapat dilihat pada Gambar 5.10.

Gambar 5.10 Tampilan *Take Image*

5.2.4.1.1. Tampilan Pilihan Media *Input* Citra Digital untuk Pembentukan *Dataset*

Sebelum tampilan *Take Image* ditampilkan, *user* diarahkan untuk memilih media untuk mengambil atau menginputkan citra digital melalui *camera* ataupun media penyimpanan *(sd card),* untuk tampilannya dapat dilihat pada Gambar 5.11. Tampilan pilihan media *input* citra digital melalui *Camera* dapat dilihat pada Gambar 5.12. Sedangkan, Tampilan pilihan media *input* citra digital melalui *SD Card* dapat dilihat pada Gambar 5.13. Setelah memilih aplikasi *file manager* untuk mengakses citra digital pada media penyimpanan, kemudian *user* dapat memilih citra digital yang akan diinputkan. Tampilannya dapat dilihat pada Gambar 5.14.



Gambar 5.12 Tampilan Pilihan Media *Input* Citra Digital

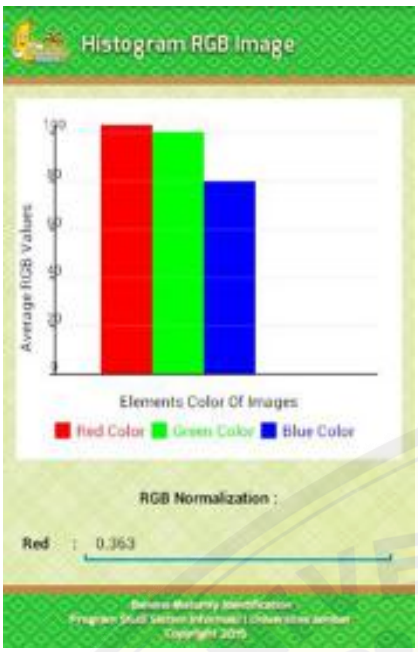
Gambar 5.11 Tampilan Pilihan Media *Input* Citra Digital Melalui *Camera*

Gambar 5.14 Tampilan Pilihan Media *Input* Citra Digital Melalui *SD Card*

Gambar 5.13 Tampilan Memilih Citra Digital Yang Akan Diinputkan

5.2.4.1.2. Tampilan *Histogram RGB Image*

Tampilan *Histogram RGB Image* merupakan tampilan untuk menampilkan histogram RGB dan normalisasi RGB dari citra digital yang telah diinputkan sebelumnya. Selain itu, *user* dapat memilih target kelas tingkat kematangan pada citra digital yang telah diinputkan sebelumnya dan kemudian *user* dapat menyimpan data sebagai *dataset* untuk proses *training data.* Tampilan *Histogram RGB Image* dapat dilihat pada Gambar 5.15. Sedangkan, Tampilan memilih target kelas dapat dilihat pada Gambar 5.16.



Gambar 5.16 Tampilan Histogram RGB *Image*

Gambar 5.15 Tampilan Memilih Target Kelas

5.2.4.2. Tampilan *Training Data*

Tampilan *Training Data* merupakan tampilan untuk menampilkan *dataset*, tombol untuk menghapus *dataset* serta tombol untuk melakukan *training data*. Tampilan *Training Data* dapat dilihat pada Gambar 5.17. Untuk menghapus *dataset*, *user* tinggal memilih dan menekan *icon trash dataset* yang akan dihapus pada *list dataset* dan kemudian akan menampilkan dialog konfirmasi seperti pada Gambar 5.18. Untuk melakukan *training data*, *user* tinggal menekan tombol *training* dan kemudian tinggal mengikuti langkah – langkah dalam melakukan *training data*.

Gambar 5.17 Tampilan Konfirmasi Menghapus *Dataset*

Gambar 5.18 Tampilan *Training Data*

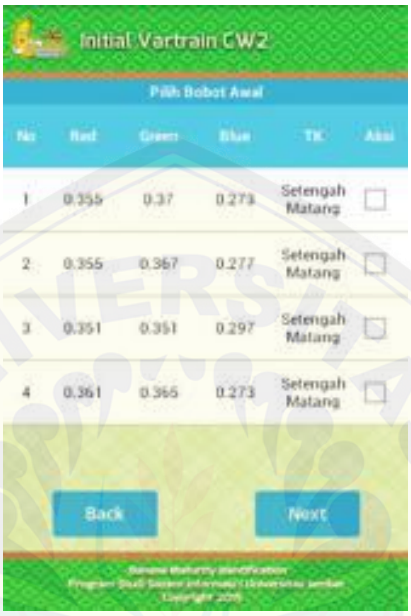
5.2.4.2.1. Tampilan *Initial Vartrain CW 1*

Tampilan *Initial Vartrain CW 1* merupakan tampilan yang menampilkan *list dataset* pada kelas matang. Tampilan ini dimaksudkan agar *user* dapat memilih bobot awal pada kelas matang sebagai kebutuhan proses *training*. Tampilan *Initial Vartrain CW 1* dapat dilihat pada Gambar 5.19.

Gambar 5.19 Tampilan Konfirmasi Menghapus *Dataset*

5.2.4.2.2. Tampilan *Initial Vartrain CW 2*

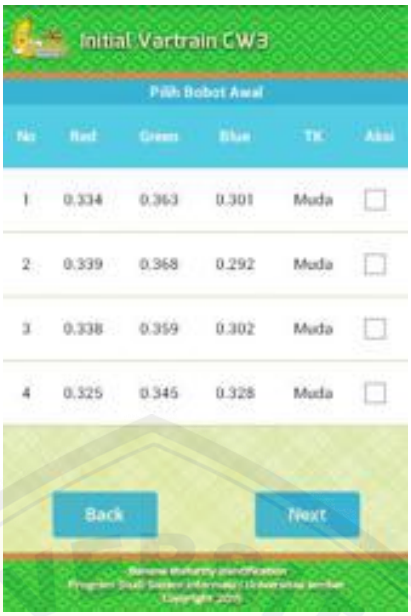
Tampilan *Initial Vartrain CW 2* merupakan tampilan yang menampilkan *list dataset* pada kelas setengah matang. Tampilan ini dimaksudkan agar *user* dapat memilih bobot awal pada kelas setengah matang sebagai kebutuhan proses *training*. Tampilan *Initial Vartrain CW 2* dapat dilihat pada Gambar 5.20.



Gambar 5.20 Tampilan *Initial Vartrain CW 2*

5.2.4.2.3. Tampilan *Initial Vartrain CW 3*

Tampilan *Initial Vartrain CW 3* merupakan tampilan yang menampilkan *list dataset* pada kelas muda. Tampilan ini dimaksudkan agar *user* dapat memilih bobot awal pada kelas muda sebagai kebutuhan proses *training*. Tampilan *Initial Vartrain CW 3* dapat dilihat pada Gambar 5.21.



Gambar 5.21 Tampilan *Initial Vartrain CW 3*

5.2.4.2.4. Tampilan *Initial Vartrain LREPSME*

Tampilan *Initial Vartrain LREPSME* merupakan tampilan yang menampilkan nilai *learning rate*, *eps* dan *max epoch*. Tampilan ini dimaksudkan agar *user* dapat memilih *max epoch* (iterasi maksimal) proses *training*. Tampilan *Initial Vartrain LREPSME* dapat dilihat pada Gambar 5.22. Sedangkan Tampilan memilih *max epoch* dapat dilihat pada Gambar 5.23. Setelah memilih *max epoch*, *user* dapat melakukan proses *training* dengan menekan tombol *Process* pada tampilan, kemudian setelah proses selesai maka hasilnya akan ditampilkan seperti pada Gambar 5.24 dan *user* dapat menyimpan hasilnya ke dalam *database* dengan menekan tombol *Save*.

Gambar 5.22 Tampilan *Initial Vartrain LREPSME*

Gambar 5.23 *Tampilan Memilih Max Epoch*

Gambar 5.24 Tampilan *Training Result* Setelah Proses *Training*

5.2.4.3. Tampilan *Training Result*

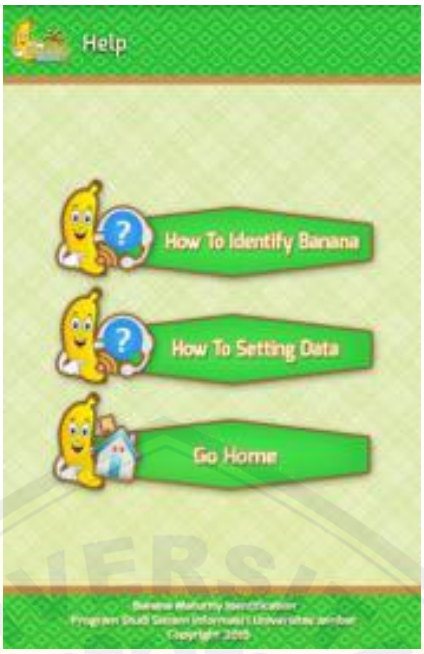
Tampilan *Training Result* merupakan tampilan yang menampilkan hasil *training result* yang ada di *database* serta tombol untuk melakukan *reset data*. Tampilan *Training Result* dapat dilihat pada Gambar 5.25. Untuk melakukan *reset data*, *user* tinggal menekan tombol *Reset* pada tampilan kemudian aplikasi akan menampilkan dialog konfirmasi seperti apada Gambar 5.26.

Gambar 5.26 Tampilan *Training Result* Menampilkan Hasil *Training* Yang Ada di *Database*

Gambar 5.25 Tampilan Konfirmasi *Reset Data*

5.2.5. Tampilan *Help*

Tampilan *Help* merupakan tampilan untuk menampilkan submenu untuk melihat *help* penggunaan fitur utama pada aplikasi. Submenu yang terdapat pada tampilan *Help* adalah submenu *How To Identify Banana*, *How To Setting Data*, dan *Go Home.* Tampilan *Help* dapat dilihat pada Gambar 5.27.



Gambar 5.27 Tampilan *Help*

5.2.5.1. Tampilan *How To Identify Banana*

Tampilan *How To Identify Banana* merupakan tampilan untuk menampilkan *help* penggunaan fitur *Identify Banana*. Tampilan *How To Identify Banana* dapat dilihat pada Gambar 5.28.

Gambar 5.28 Tampilan *How To Identify Banana*

5.2.5.2. Tampilan *How To Setting Data*

Tampilan *How To Setting Data* merupakan tampilan untuk menampilkan *help* penggunaan fitur *Setting Data*. Tampilan *How To Setting Data* dapat dilihat pada Gambar 5.29.

Gambar 5.29 Tampilan *How To Identify Banana*

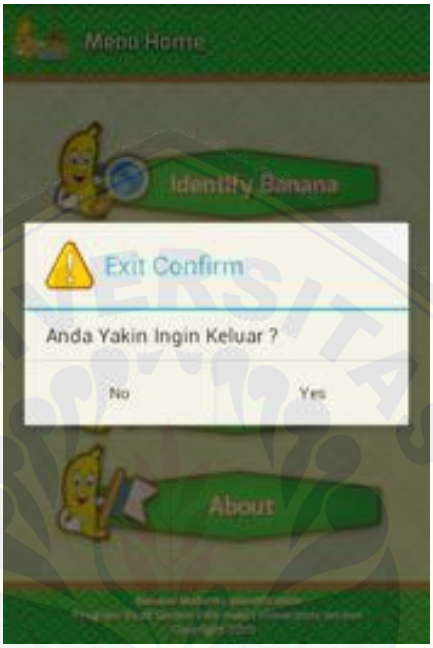
5.2.6. Tampilan *About*

Tampilan *About* merupakan tampilan untuk menampilkan *about* aplikasi. Tampilan *About* dapat dilihat pada Gambar 5.30.

Gambar 5.30 Tampilan *About*

5.2.7. Tampilan *Exit*

Tampilan *Exit* merupakan tampilan untuk menampilkan dialog konfirmasi untuk keluar aplikasi. Untuk keluar *(exit)* dari aplikasi, *user* tinggal menekan tombol *onBackpressed* pada tampilan *Home.* Tampilan *Exit* dapat dilihat pada Gambar 5.31.



Gambar 5.31 Tampilan *Exit*

5.3. Implementasi *Learning Vector Quantization (LVQ)* Pada Aplikasi *Banana Maturity Identification*

Pada aplikasi *Banana Maturity Identification*, algoritma *Learning Vector Quantization (LVQ)* digunakan untuk melakukan proses pembelajaran *(training)* data. Sebelum proses pembelajaran *(training)* dilakukan, tentunya *dataset* atau data yang akan dilatih sudah tersedia. Proses pembentukan *dataset* dilakukan sesuai dengan kebutuhan proses *training* menggunakan algoritma *LVQ*. Pada penelitian ini *dataset* dibentuk dari nilai histogram warna (RGB) yang telah di normalisasi dari citra digital buah pisang dengan berbagai tingkat kematangan yaitu matang, setengah matang dan muda yang kemudian dimasukkan ke dalam *database*. Kode program untuk mengambil nilai histogram warna (RGB) citra digital dapat dilihat pada Gambar 5.32.

```
public class GetRGBTask extends AsyncTask<Void, Void, Boolean> {

    private int get_pixel, width_resize, height_resize, get_red_color = 0, get_green_color = 0, get_blue_color = 0, get_pixel_count = 0 ;
    private double red_average, green_average, blue_average, red_normalization, green_normalization, blue_normalization;
    private Bitmap bm_resize;
    private Context context;
    private Toast toast_message;
    private ProgressDialog progress_dialog;
    private c_TakeImage_Menu ctim;

    public void Task (Context context, Bitmap bm_resize, int width_resize, int height_resize ) {
        this.context = context;
        this.bm_resize = bm_resize;
        this.width_resize = width_resize;
        this.height_resize = height_resize;
        this.progress_dialog = new ProgressDialog(context);
    }

    @Override
    protected void onPreExecute() {
        this.progress_dialog.setMessage("Tunggu Sebentar...");
        this.progress_dialog.show();
    }

    @Override
    protected Boolean doInBackground(Void... params) {

        try {

            for (int x=0; x<width_resize; x++) {
                for(int y=0; y<height_resize; y++){

                    get_pixel = bm_resize.getPixel(x,y);
                    get_red_color   += Color.red(get_pixel);
                    get_green_color += Color.green(get_pixel);
                    get_blue_color  += Color.blue(get_pixel);
                    get_pixel_count++;

                }};

                // Calculate average of bitmap r,g,b values
                red_average   = (get_red_color / get_pixel_count);
                green_average = (get_green_color / get_pixel_count);
                blue_average  = (get_blue_color / get_pixel_count);

                // Calculate normalisasi of bitmap r,g,b values
                red_normalization   = Math.floor((red_average / (red_average + green_average + blue_average))*1000)/1000;
                green_normalization = Math.floor((green_average / (red_average + green_average + blue_average))*1000)/1000;
                blue_normalization  = Math.floor((blue_average / (red_average + green_average + blue_average))*1000)/1000;

                return true;

        } catch (Exception e) {
            Log.i("Pesan Error !", ">>>>>>>>>>>>>>>>" + e.getMessage());
            return false;
        }
    }

    protected void onPostExecute(Boolean result) {
        if (!result) {
            toast_message = Toast.makeText(context, "Terjadi Kesalahan !", Toast.LENGTH_LONG);
            toast_message.setGravity(Gravity.CENTER_VERTICAL, 0, 50);
            toast_message.show();
        }
        else {
            ctim = new c_TakeImage_Menu();
            ctim.viewHistogramRGB_Image(context, red_average, green_average, blue_average, red_normalization, green_normalization, blue_normalization);
        }
        this.progress_dialog.cancel();
    }

}
```

Gambar 5.32 Kode Program Untuk Mengambil Nilai Histogram Warna (RGB) Citra Digital

Pengimplementasian kode program untuk mengambil nilai histogram warna (RGB) citra digital harus sesuai dengan perhitungan atau rumus *manual* yang ada, agar *output* yang dihasilkan sama dan sesuai dengan kebutuhan yang diharapkan. Adapun pencocokan hasil perhitungan aplikasi dengan perhitungan atau rumus *manual* menggunakan citra digital buah pisang berukuran 200 x 300 px, yaitu :

Gambar 5.33 Citra Digital Yang Diuji

a. Perhitungan *Manual*

Untuk mengambil atau mengekstraksi warna RGB dari citra digital 200 x 300 px maka ada 60.000 piksel yang harus diambil nilai RGBnya. Proses ekstraksi RGB tiap piksel untuk perhitungan ini dibantu menggunakan aplikasi *eclipse*. Sehingga, diperoleh :

Pada piksel (x = 0, y = 0) didapat nilai R = 42, G = 43, B = 48. Setelah seluruh piksel diambil nilai RGBnya, diperoleh jumlah RGB seluruh piksel yaitu : R = 6.198.417, G = 6.043.083, B = 4.838.820. Kemudian diambil nilai rata – ratanya tiap *channel* warna menggunakan rumus :

1. *Average (R)* = $\frac{\text{Jumlah Nilai } Red \text{ Semua Piksel}}{\text{Jumlah Piksel}} = \frac{6.198.417}{60.000} = 103$
2. *Average (G)* = $\frac{\text{Jumlah Nilai } Green \text{ Semua Piksel}}{\text{Jumlah Piksel}} = \frac{6.043.083}{60.000} = 100$
3. *Average (B)* = $\frac{\text{Jumlah Nilai } Blue \text{ Semua Piksel}}{\text{Jumlah Piksel}} = \frac{4.838.820}{60.000} = 80$

Selanjutnya nilai rata – rata tiap *channel* warna dinormaliasasi untuk mengurangi perbedaan pencahayaan menggunakan rumus :

1. *Norm (R)* = $\frac{R}{R+G+B} = \frac{6.198.417}{6.198.417+ 6.043.083 + 4.838.820} = 0{,}363$
2. *Norm (G)* = $\frac{G}{R+G+B} = \frac{6.043.083}{6.198.417+ 6.043.083 + 4.838.820} = 0{,}353$
3. *Norm (B)* = $\frac{B}{R+G+B} = \frac{4.838.820}{6.198.417+ 6.043.083 + 4.838.820} = 0{,}282$

Nilai ini merupakan nilai RGB dari citra digital yang siap untuk dijadikan *dataset*. Jadi nilai RGB citra digital pada Gambar 5.33 adalah ***Red* = 0,363 *Green* = 0,353 *Blue* = 0,282**.

b. Perhitungan Aplikasi

Perhitungan pada aplikasi dilakukan sama dengan perhitungan *manual*, perbedaannya perhitungan pada aplikasi dilakukan dengan mengubah perhitungan atau rumus *manual* ke dalam kode program, dimana hasil perhitungannya sama dengan perhitungan *manual*. Gambar 5.34 menunjukkan citra digital yang akan diekstraksi RGBnya. Hasil perhitungan pada aplikasi dapat dilihat pada Gambar 5.35.

Gambar 5.34 Citra Digital Yang Akan Diekstraksi RGBnya

Gambar 5.35 Hasil Perhitungan RGB Pada Aplikasi

Berdasarkan pencocokan perhitungan yang dilakukan, dapat diketahui bahwa perhitungan aplikasi dengan perhitungan *manual* menghasilkan *output* yang sama, sehingga kode program yang di implementasikan sudah benar.

Pada proses pembelajaran *(training)*, diawali dengan memilih salah satu *dataset* sebagai bobot awal pada masing – masing kelas dan kemudian dilanjutkan dengan menetapkan nilai laju pembelajaran *(learning rate)*, jumlah maksimum iterasi *(max epoch)*, dan *error* mininum *(eps)*. Setelah hal tersebut dilakukan maka proses pengklasifikasian dapat dilakukan, proses ini merupakan proses klasifikasi algoritma *LVQ*. Setelah proses pembelajaran *(training)* selesai dilakukan maka hasil dari *training* tersebut adalah bobot akhir dari masing – masing tingkat

kematangan yang telah dipilih sebagai bobot awal tadi, pada penelitian ini akan diperoleh 3 bobot akhir yaitu bobot akhir untuk kelas matang, setengah matang dan muda. Kemudian bobot – bobot tersebut disimpan kedalam *database* sebagai acuan pada proses identifikasi kematangan buah pisang. Kode program, proses pembelajaran *(training)* dapat dilihat pada Gambar 5.36.

```
public class TrainingTask extends AsyncTask<Void, Void, Boolean> {

    // Mendeklarasikan Variabel
    private double[][] arrayInputBobot, arrayInputDataTraining;
    private int[] targetClassInputBobot, targetClassInputDataTraining;
    private int banyakDataInputBobot, banyakDataInputDataTraining, inputBobot = 3, inputDataTraining = 3;
    private double [] euclideanDistance, NOB_Matang, NOB_SetengahMatang, NOB_Muda;
    public int indexbanyakDataInputBobot = 0, target_kelas = 0, indexbanyakDataInputDataTraining = 0, cekJarakTerkecil;
    public double learningRate, maxEpoch, Eps, updateBobot = 0;
    private Context context;
    private Toast toast_message;
    private ProgressDialog progress_dialog;
    private int jmlDataBobot, jmlDataTraining;
    private ArrayList<Atribut_Model_TrainingData> tdCk, tdOtherCk;
    private c_TrainingResult_Menu ctrm;

    public void Task(Context context, double learningRate, int maxEpoch, double Eps, int jmlDataBobot, int jmlDataTraining,
                     ArrayList<Atribut_Model_TrainingData> tdCk, ArrayList<Atribut_Model_TrainingData> tdOtherCk ) {

        this.context = context;
        this.learningRate    = learningRate;
        this.maxEpoch        = maxEpoch;
        this.Eps             = Eps;
        this.jmlDataBobot    = jmlDataBobot;
        this.jmlDataTraining = jmlDataTraining;
        this.tdCk            = tdCk;
        this.tdOtherCk       = tdOtherCk;
        this.progress_dialog = new ProgressDialog(context);
    }

    @Override
    protected void onPreExecute() {
        this.progress_dialog.setMessage("Proses Training...");
        this.progress_dialog.show();
    }

    @Override
    protected Boolean doInBackground(Void... params) {

        try {

            banyakDataInputBobot = jmlDataBobot;
            arrayInputBobot      = new double[banyakDataInputBobot][inputBobot]; // banyak data | banyak inputan vektor
            banyakDataInputDataTraining   = jmlDataTraining;
            arrayInputDataTraining        = new double[banyakDataInputDataTraining][inputDataTraining]; // banyak data | banyak inputan vektor
            targetClassInputBobot         = new int [banyakDataInputBobot];
            targetClassInputDataTraining  = new int [banyakDataInputDataTraining];

            //Ambil Training Data From Database
            for (Atribut_Model_TrainingData amtd : tdCk) {
                // Data Bobot
                arrayInputBobot[indexbanyakDataInputBobot][0] = Double.parseDouble(amtd.getTd_red_image());
                arrayInputBobot[indexbanyakDataInputBobot][1] = Double.parseDouble(amtd.getTd_green_image());
                arrayInputBobot[indexbanyakDataInputBobot][2] = Double.parseDouble(amtd.getTd_blue_image());

                    if (amtd.getTarget_kelas().equals("Matang")){
                        target_kelas = 1;
                    } else if (amtd.getTarget_kelas().equals("Setengah Matang")) {
                        target_kelas = 2;
                    } else {
                        target_kelas = 3;
                    }

                targetClassInputBobot[indexbanyakDataInputBobot] = target_kelas;
                indexbanyakDataInputBobot++;
            }

            for (Atribut_Model_TrainingData amtd : tdOtherCk) {
                // Data Training
                arrayInputDataTraining[indexbanyakDataInputDataTraining][0] = Double.parseDouble(amtd.getTd_red_image());
                arrayInputDataTraining[indexbanyakDataInputDataTraining][1] = Double.parseDouble(amtd.getTd_green_image());
                arrayInputDataTraining[indexbanyakDataInputDataTraining][2] = Double.parseDouble(amtd.getTd_blue_image());

                    if (amtd.getTarget_kelas().equals("Matang")){
                        target_kelas = 1;
                    } else if (amtd.getTarget_kelas().equals("Setengah Matang")) {
                        target_kelas = 2;
                    } else {
                        target_kelas = 3;
                    }

                targetClassInputDataTraining[indexbanyakDataInputDataTraining] = target_kelas;
                indexbanyakDataInputDataTraining++;
            }

        //-------------------------------------------------------------------------------------------------------

        euclideanDistance = new double[banyakDataInputBobot];
        System.out.print("Training CVQ \n\n");

        for (int iter = 1; iter < maxEpoch + 1; iter++) {
            System.out.println("[+]Iterasi ke ----------------------------------------------------------------> " + iter);

            if (learningRate > Eps) {
                System.out.println("[+]learningRate = " + learningRate + " > Eps = " + Eps);

                for (int k = 0; k < banyakDataInputDataTraining; k++) {
                    System.out.println("[+]banyakDataInputDataTraining = "+ k);

                    // Dapatkan Jarak Terkecil
                    cekJarakTerkecil = 0;
                    for (int i = 0; i < banyakDataInputBobot; i++) {
                        euclideanDistance[i] = ComputeEuclideanDistance(arrayInputBobot[i], arrayInputDataTraining[k]);
```

Gambar 5.36 Kode Program Proses Pembelajaran (*Training)*

Pengimplementasian kode program untuk proses pembelajaran *(training)* harus sesuai juga dengan perhitungan atau rumus *manual* yang ada, agar *output* yang dihasilkan juga sama dan sesuai dengan kebutuhan yang diharapkan. Berikut hasil pencocokan perhitungan aplikasi dengan perhitungan atau rumus *manual* menggunakan *dataset training* berjumlah 6 data, dimana perhitungan ini dilakukan hanya 1 iterasi *(max epoch)* dengan *learning rate* = 0,5 dan *eps* = 0,001 yaitu :

a. Perhitungan *Manual*

Dataset citra digital buah pisang yang digunakan adalah sebagai berikut :

Tabel 5.1 *Dataset* Citra Digital Buah Pisang

No	Nilai Red (R)	Nilai Green (G)	Nilai Blue (B)	Kelas
1.	0,367	0,352	0,279	Matang
2.	0,363	0,353	0,282	Matang
3.	0,355	0,37	0,273	Setengah Matang
4.	0,355	0,367	0,277	Setengah Matang
5.	0,334	0,363	0,301	Muda
6.	0,339	0,368	0,292	Muda

Kemudian bobot awal yang dipilih dari *dataset* diatas dapat dilihat pada Tabel 5.2 dan data yang lain sebagai data *training*.

Tabel 5.2 Dataset Yamg Dipilih Bobot Awal

No	Nilai Red (R)	Nilai Green (G)	Nilai Blue (B)	Kelas
1.	0,367	0,352	0,279	Matang
2.	0,355	0,37	0,273	Setengah Matang
3.	0,334	0,363	0,301	Muda

Setelah pemilihan bobot awal dan penetapan *learning rate* dan *eps* serta iterasi *(max epoch)* selanjutnya proses *training* dapat dilakukan sesuai dengan algoritma *LVQ*. Berikut ini perhitungan *training*-nya :

a. Data ke-1 (R = 0,363 G = 0,353 B = 0,282)

1. Target data = 1 (Matang)
2. Menghitung Jarak pada bobot dengan aturan *Euclidean Distance* :

   a. Bobot ke-1 (R = 0,367 G = 0,352 B = 0,279)

   $= \sqrt{(0{,}363 - 0{,}367)^2 + (0{,}353 - 0{,}352)^2 + (0{,}282 - 0{,}279)^2} = 0{,}005$

   b. Bobot ke-2 (R = 0,355 G = 0,37 B = 0,273)

   $= \sqrt{(0{,}363 - 0{,}355)^2 + (0{,}353 - 0{,}37)^2 + (0{,}282 - 0{,}273)^2} = 0{,}02$

   c. Bobot ke- 3 (R = 0,334 G = 0,363 B = 0,301)

   $= \sqrt{(0{,}363 - 0{,}334)^2 + (0{,}353 - 0{,}363)^2 + (0{,}282 - 0{,}301)^2} = 0{,}036$
3. Jarak terdekat pada bobot ke = 1
4. Target data = Kelas jarak terdekat, maka :

   Bobot baru pada bobot ke = 1, yaitu :

   $W_{baru} = W_{lama} + (alpha(learning\ rate) * (x - W_{lama}))$

   $W_{11} = 0{,}367 + (0{,}5 * (0{,}363 - 0{,}367)) = 0{,}365$

   $W_{12} = 0{,}352 + (0{,}5 * (0{,}353 - 0{,}352)) = 0{,}352$

   $W_{13} = 0{,}279 + (0{,}5 * (0{,}282 - 0{,}279)) = 0{,}28$
5. Jadi, bobot sekarang adalah :

   a. Bobot ke-1 **(R = 0,365 G = 0,352 B = 0,28)**

   b. Bobot ke-2 (R = 0,355 G = 0,37 B = 0,273)

   c. Bobot ke-3 (R = 0,334 G = 0,363 B = 0,301)

b. Data ke-2 (R = 0,355 G = 0,367 B = 0,277)

1. Target data = 2 (Setengah Matang)
2. Menghitung Jarak pada bobot dengan aturan *Euclidean Distance* :

   a. Bobot ke-1 (R = 0,365 G = 0,352 B = 0,28)

   $= \sqrt{(0{,}355 - 0{,}365)^2 + (0{,}367 - 0{,}352)^2 + (0{,}277 - 0{,}28)^2} = 0{,}018$

   b. Bobot ke-2 (R = 0,355 G = 0,37 B = 0,273)

   $= \sqrt{(0{,}355 - 0{,}355)^2 + (0{,}367 - 0{,}37)^2 + (0{,}277 - 0{,}273)^2} = 0{,}005$

   c. Bobot ke- 3 (R = 0,334 G = 0,363 B = 0,301)

   $= \sqrt{(0{,}355 - 0{,}334)^2 + (0{,}367 - 0{,}363)^2 + (0{,}277 - 0{,}301)^2} = 0{,}032$
3. Jarak terdekat pada bobot ke = 2

4. Target data = Kelas jarak terdekat, maka :

   Bobot baru pada bobot ke = 2, yaitu :

   $W_{baru} = W_{lama} + (alpha(learning\ rate) * (x - W_{lama}))$

   $W_{21} = 0,355 + (0,5 * (0,355 – 0,355)) = 0,355$

   $W_{22} = 0,37 + (0,5 * (0,367 – 0,37)) = 0,368$

   $W_{23} = 0,273 + (0,5 * (0,277 – 0,273)) = 0,275$

5. Jadi, bobot sekarang adalah :

   a. Bobot ke-1 (R = 0,365 G = 0,352 B = 0,28)

   b. Bobot ke-2 **(R = 0,355 G = 0,368 B = 0,275)**

   c. Bobot ke-3 (R = 0,334 G = 0,363 B = 0,301)

c. Data ke-3 (R = 0,339 G = 0,368 B = 0,292)

1. Target data = 3 (Muda)
2. Menghitung Jarak pada bobot dengan aturan *Euclidean Distance* :

   a. Bobot ke-1 (R = 0,365 G = 0,352 B = 0,28)

   $= \sqrt{(0,339 - 0,365)^2 + (0,368 - 0,352)^2 + (0,292 - 0,28)^2} = 0,018$

   b. Bobot ke-2 (R = 0,355 G = 0,368 B = 0,275)

   $= \sqrt{(0,339 - 0,355)^2 + (0,368 - 0,368)^2 + (0,292 - 0,275)^2} = 0,005$

   c. Bobot ke- 3 (R = 0,334 G = 0,363 B = 0,301)

   $= \sqrt{(0,339 - 0,334)^2 + (0,368 - 0,363)^2 + (0,292 - 0,301)^2} = 0,032$

3. Jarak terdekat pada bobot ke = 3
4. Target data = Kelas jarak terdekat, maka :

   Bobot baru pada bobot ke = 3, yaitu :

   $W_{baru} = W_{lama} + (alpha(learning\ rate) * (x - W_{lama}))$

   $W_{21} = 0,344 + (0,5 * (0,339 – 0,344)) = 0,336$

   $W_{22} = 0,363 + (0,5 * (0,368 – 0,363)) = 0,365$

   $W_{23} = 0,301 + (0,5 * (0,292 – 0,301)) = 0,296$

5. Jadi, bobot sekarang adalah :

   a. Bobot ke-1 (R = 0,365 G = 0,352 B = 0,28)

   b. Bobot ke-2 (R = 0,355 G = 0,368 B = 0,275)

   c. Bobot ke-3 **(R = 0,336 G = 0,365 B = 0,296)**

Bobot akhir yang diperoleh dari proses *training* diatas adalah sebagai berikut :

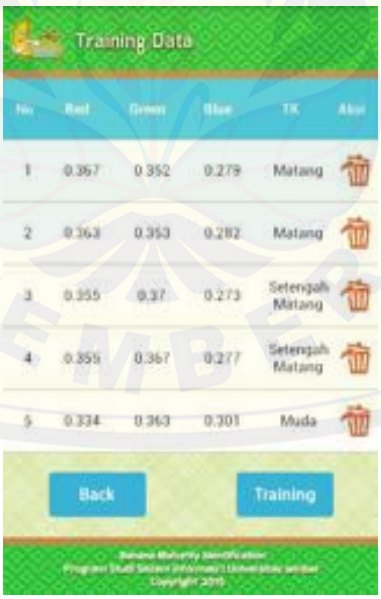
$W_{1(akhir)}$: R = 0,365 G = 0,352 B = 0,28

$W_{2(akhir)}$: R = 0,355 G = 0,368 B = 0,275

$W_{3(akhir)}$: R = 0,336 G = 0,365 B = 0,296

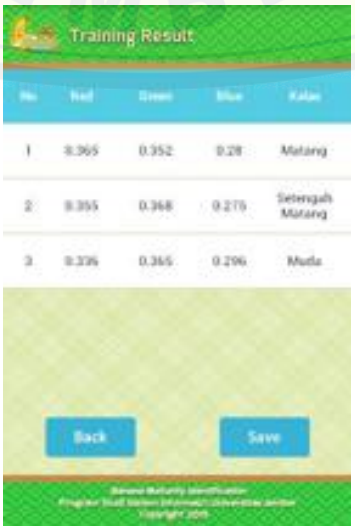
b. Perhitungan Aplikasi

Perhitungan pada aplikasi dilakukan sama dengan perhitungan *manual*, perbedaannya perhitungan pada aplikasi dilakukan dengan mengubah perhitungan atau rumus *manual* ke dalam kode program, dimana hasil perhitungannya sama dengan perhitungan *manual*. Gambar 5.37 menunjukkan *dataset training* yang digunakan. Gambar 5.38 menunjukkan pemilihan bobot dan penetapan *learning rate*, *eps* dan iterasi *(max epoch)*. Hasil proses *training* pada aplikasi dapat dilihat pada Gambar 5.39.



Gambar 5.37 *Dataset* Yang Digunakan

Gambar 5.38 Pemilihan Bobot dan Penetapan *Learning Rate, Eps* dan Iterasi *(Max Epoch)*



Gambar 5.39 Hasil Proses *Training*

Berdasarkan pencocokan perhitungan yang dilakukan dapat diketahui bahwa perhitungan aplikasi dengan perhitungan *manual* menghasilkan *output* yang sama, sehingga kode program yang di implementasikan sudah benar.

Pada proses pengujian *(testing)* atau identifikasi, aplikasi membandingkan jarak terdekat dari nilai histogram warna (RGB) citra digital yang telah dinormalisasi dan tidak diketahui kelasnya dengan nilai bobot akhir dari hasil *training data*. Sehingga, hasil identifikasi diperoleh dari jarak yang paling dekat antara nilai histogram warna (RGB) dengan bobot akhir hasil *training data*. Kode program, proses pengujian *(testing)* atau identifikasi dapat dilihat pada Gambar 5.39.

Gambar 5.40 Kode Program Proses Pengujian (*Testing)* atau Identifikasi

Pengimplementasian kode program untuk proses pengujian *(testing)* harus sesuai juga dengan perhitungan atau rumus *manual* yang ada, agar *output* yang dihasilkan juga sama dan sesuai dengan kebutuhan yang diharapkan. Berikut hasil pencocokan perhitungan aplikasi dengan perhitungan atau rumus *manual* menggunakan *dataset testing* berjumlah 1 data, dimana pada pencocokan ini menggunakan citra digital buah pisang berukuran 200 x 300 px, yaitu :

Gambar 5.41 Citra Digital Yang Di Identifikasi

a. Perhitungan *Manual*

Citra digital yang akan di identifikasi harus melalui proses ekstraksi RBG terlebih dahulu. Proses ekstraksi RGB tiap piksel dibantu menggunakan aplikasi *eclipse*. Sehingga, diperoleh :

Pada piksel (x = 0, y = 0) diperoleh nilai R = 42, G = 43, B = 48. Setelah seluruh piksel didapat nilai RGBnya, diperoleh jumlah RGB seluruh piksel yaitu : R = 6.198.417, G = 6.043.083, B = 4.838.820. Kemudian diambil nilai rata – ratanya tiap *channel* warna menggunakan rumus :

1. *Average (R)* = $\frac{\text{Jumlah Nilai } Red \text{ Semua Piksel}}{\text{Jumlah Piksel}} = \frac{6.198.417}{60.000} = 103$
2. *Average (G)* = $\frac{\text{Jumlah Nilai } Green \text{ Semua Piksel}}{\text{Jumlah Piksel}} = \frac{6.043.083}{60.000} = 100$
3. *Average (B)* = $\frac{\text{Jumlah Nilai } Blue \text{ Semua Piksel}}{\text{Jumlah Piksel}} = \frac{4.838.820}{60.000} = 80$

Selanjutnya nilai rata – rata tiap *channel* warna dinormaliasasi untuk mengurangi perbedaan pencahayaan menggunakan rumus :

1. *Norm (R)* = $\frac{R}{R+G+B} = \frac{6.198.417}{6.198.417+ 6.043.083 + 4.838.820} = 0,363$
2. *Norm (G)* = $\frac{G}{R+G+B} = \frac{6.043.083}{6.198.417+ 6.043.083 + 4.838.820} = 0,353$
3. *Norm (B)* = $\frac{B}{R+G+B} = \frac{4.838.820}{6.198.417+ 6.043.083 + 4.838.820} = 0,282$

Jadi, nilai RGB citra digital pada Gambar 5.41 adalah ***Red* = 0,363 *Green* = 0,353 *Blue* = 0,282**. Nilai RGB ini nantinya dibandingkan dengan bobot akhir hasil proses *training* menggunakan rumus *Euclidean Distance*. Bobot akhir yang digunakan pada perhitungan ini adalah sebagai berikut :

$W_{1(akhir)}$: R = 0,365 G = 0,352 B = 0,28 (Kelas Matang)

$W_{2(akhir)}$: R = 0,355 G = 0,368 B = 0,275 (Kelas Setengah Matang)

$W_{3(akhir)}$: R = 0,336 G = 0,365 B = 0,296 (Kelas Muda)

Perhitungan yang dilakukan untuk melakukan proses identifikasi adalah sebagai berikut :

a. Bobot Akhir ke-1 (R = 0,365 G = 0,352 B = 0,28)

$$= \sqrt{(0{,}363 - 0{,}365)^2 + (0{,}353 - 0{,}352)^2 + (0{,}282 - 0{,}28)^2} = 0{,}002$$

b. Bobot Akhir ke-2 (R = 0,355 G = 0,368 B = 0,275)

$$= \sqrt{(0{,}339 - 0{,}355)^2 + (0{,}368 - 0{,}368)^2 + (0{,}292 - 0{,}275)^2} = 0{,}018$$

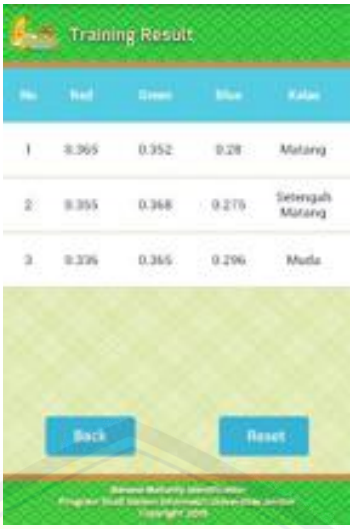
c. Bobot Akhir ke- 3 (R = 0,336 G = 0,365 B = 0,296)

$$= \sqrt{(0{,}339 - 0{,}334)^2 + (0{,}368 - 0{,}363)^2 + (0{,}292 - 0{,}301)^2} = 0{,}032$$

Jarak terdekat pada bobot ke = 1, maka citra digital pada Gambar 5.41 yang di inputkan teridentifikasi **Matang**.

b. Perhitungan Aplikasi

Perhitungan pada aplikasi dilakukan sama dengan perhitungan *manual*, perbedaannya perhitungan pada aplikasi dilakukan dengan mengubah perhitungan atau rumus *manual* ke dalam kode program, dimana hasil perhitungannya sama dengan perhitungan *manual*. Gambar 5.42 menunjukkan hasil *training* yang digunakan sebagai acuan proses identifikasi. Gambar 5.43 menunjukkan citra digital yang akan di identifikasi. Hasil proses *testing* pada aplikasi dapat dilihat pada Gambar 5.44.



Gambar 5.42 Hasil *Training* Sebagai Acuan Proses Identifikasi

Gambar 5.43 Citra Digital Yang Akan Di Identifikasi

Gambar 5.44 Hasil Identifikasi

5.4. Pengujian Aplikasi *Banana Maturity Identification*

Pengujian pada aplikasi *Banana Maturity Identification* dilakukan untuk menilai kinerja aplikasi dalam mengidentifikasi kematangan buah pisang. Sebelum melakukan identifikasi tentunya dibutuhkan *dataset* untuk proses pembelajaran *(training)* data nantinya. Tabel 5.3 menunjukkan *dataset* yang akan digunakan dalam penelitian ini. Data ini nantinya akan diinputkan ke dalam aplikasi ini.

Tabel 5.3 *Dataset* Citra Digital Buah Pisang

No	Nilai Red (R)	Nilai Green (G)	Nilai Blue (B)	Kelas
1.	0,367	0,352	0,279	Matang
2.	0,363	0,353	0,282	Matang
3.	0,362	0,358	0,279	Matang
4.	0,361	0,357	0,281	Matang
5.	0,355	0,37	0,273	Setengah Matang
6.	0,355	0,367	0,277	Setengah Matang
7.	0,351	0,351	0,297	Setengah Matang
8.	0,361	0,365	0,273	Setengah Matang
9.	0,334	0,363	0,301	Muda
10.	0,339	0,368	0,292	Muda
11.	0,338	0,359	0,302	Muda
12.	0,325	0,345	0,328	Muda

Dataset pada Tabel 5.3 diperoleh dari nilai histogram warna (RGB) citra digital yang telah dinormalisasi pada 3 kelas tingkat kematangan buah pisang yaitu matang, setengah matang, dan muda. Citra digital buah pisang diambil menggunakan *smartphone Sony Xperia E* dengan jarak pengambilan 14 cm.

Pada proses pembelajaran *(training)*, *dataset* diatas akan dipilih sebanyak 3 data untuk dijadikan bobot awal dan yang lainnya akan digunakan untuk *training data*. Pemilihan bobot ini dapat dipilih secara acak, namun penetapan bobot awal pada pengujian ini dimaksudkan agar hasil pengujian dapat optimal Tabel 5.4 menunjukkan *dataset* yang dipilih menjadi bobot awal pada penelitian ini.

Tabel 5.4 *Dataset* Yang Dijadikan Bobot Awal

No	Nilai Red (R)	Nilai Green (G)	Nilai Blue (B)	Kelas
1.	0,367	0,352	0,279	Matang
2.	0,355	0,37	0,273	Setengah Matang
3.	0,334	0,363	0,301	Muda

Kemudian nilai *learning rate* yang digunakan pada proses *training* ini adalah 0,5 dan nilai *eps* atau *error* minimun yang digunakan adalah 0,001 serta iterasi maksimal *(max epoch)* yang digunakan adalah 10 kali.

Proses pengujian *(testing)* pada penelitian ini dilakukan dengan beberapa pengujian. Pengujian ini menggunakan data *testing* yang berupa nilai histogram warna (RGB) citra digital yang disembunyikan kelas tingkat kematangannya. Berikut pemaparan pengujian yang dilakukan, yaitu :

a. Pengujian 1

Pengujian 1 ini dilakukan untuk mengetahui pengaruh perubahan jumlah *dataset* yang digunakan pada proses *training data* terhadap hasil identifikasi. Jumlah *dataset* yang digunakan untuk melakukan pengujian 1 ini adalah 5 data, 9 data dan 12 data. Hasil pengujian 1 ini dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil Pengujian 1

No.	Kelas Pengujian	Hasil Pengujian		
		Dataset = 5	Dataset = 9	Dataset = 12
1.	Matang	Benar	Benar	Benar
2.	Matang	Benar	Benar	Benar
3.	Matang	Benar	Benar	Benar
4.	Setengah Matang	Salah	Benar	Benar
5.	Setengah Matang	Salah	Benar	Benar
6.	Setangah Matang	Benar	Salah	Salah
7.	Setangah Matang	Salah	Benar	Benar
8.	Muda	Benar	Benar	Benar
9.	Muda	Benar	Benar	Benar
10.	Muda	Benar	Benar	Benar
Persentase Berhasil		**70 %**	**90 %**	**90 %**
Persentase Gagal		**30 %**	**10 %**	**10 %**

b. Pengujian 2

Pengujian 2 ini dilakukan untuk mengetahui pengaruh perubahan nilai jumlah maksimal iterasi *(max epoch)* pada saat *training data* terhadap hasil identifikasi. Jumlah maksimal iterasi *(max epoch)* yang digunakan untuk melakukan pengujian ini adalah 5 kali, 10 kali, dan 25 kali. Hasil pengujian 2 ini dapat ditunjukkan pada Tabel 5.6.

Tabel 5.6 Hasil Pengujian 2

No.	Kelas Pengujian	Hasil Pengujian		
		Max Epoch = 5	Max Epoch = 10	Max Epoch = 25
1.	Matang	Benar	Benar	Benar
2.	Matang	Benar	Benar	Benar
3.	Matang	Benar	Benar	Benar
4.	Setengah Matang	Benar	Benar	Benar
5.	Setengah Matang	Benar	Benar	Benar
6.	Setangah Matang	Salah	Salah	Salah
7.	Setangah Matang	Benar	Benar	Benar
8.	Muda	Benar	Benar	Benar
9.	Muda	Benar	Benar	Salah
10.	Muda	Benar	Benar	Benar
Persentase Berhasil		**90 %**	**90 %**	**80 %**
Persentase Gagal		**10 %**	**10 %**	**20 %**

c. Pengujian 3

Pengujian 3 ini dilakukan untuk mengetahui pengaruh perubahan jarak pengambilan objek buah pisang pada saat akusisi citra digital terhadap hasil identifikasi. Jarak pengambilan objek yang digunakan untuk melakukan pengujian 3 ini adalah 14 cm, 20 cm, dan 30 cm. Hasil pengujian 3 ini dapat dilihat pada Tabel 5.7.

Tabel 5.7 Hasil Pengujian 3

No.	Kelas Pengujian	Hasil Pengujian		
		Jarak = 14	Jarak = 20	Jarak = 30
1.	Matang	Benar	Salah	Salah
2.	Matang	Benar	Salah	Salah
3.	Matang	Benar	Salah	Salah
4.	Setengah Matang	Benar	Salah	Salah
5.	Setengah Matang	Benar	Salah	Salah
6.	Setangah Matang	Salah	Salah	Salah
7.	Setangah Matang	Benar	Salah	Salah
8.	Muda	Benar	Benar	Benar
9.	Muda	Benar	Benar	Benar
10.	Muda	Benar	Benar	Benar
Persentase Berhasil		**90 %**	**30 %**	**30 %**
Persentase Gagal		**10 %**	**70 %**	**70 %**

d. Pengujian 4

Pengujian 4 ini dilakukan untuk mengetahui pengaruh *smartphone* yang digunakan, terhadap akurasi identifikasi. Pengujian ini dilakukan pada beberapa *smartphone* yang memiliki spesifikasi yang berbeda – beda. Hasil pengujian 4 ini dapat ditunjukkan pada Tabel 5.8.

Tabel 5.8 Hasil Pengujian 4

No.	Tipe *Smartphone*	*OS* dan *Hardware*	*Camera*	Pengujian		
				Jarak *Capture*	Lama Proses	Akurasi
1.	Sony Xperia E	OS : 4.1.0 Jelly Bean CPU Cores : Single-core Total RAM : 512 MB	3.15 MP	14 cm	1,03 *second*	90 %
2.	Evercoss A28T	OS : 4.2.0 Jelly Bean CPU Cores : Dual-core Total RAM : 256 MB	3 MP	14 cm	0,88 *second*	90 %

3.	Samsung Galaxy Chat	OS : 4.0.0 Ice Cream Sandwich CPU Cores : Single-core Total RAM : 512 MB	2 MP	14 cm	1,00 second	80 %
4.	Samsung Galaxy Core	OS : 4.1.2 Jelly Bean CPU Cores : Dual-core Total RAM : 1 GB	5 MP	14 cm	0,82 *second*	90 %
5.	Asus Zenfone 5	OS : 4.3.0 Jelly Bean CPU Cores : Dual-core Total RAM : 2 GB	8 MP	14 cm	0,81 *second*	90 %

BAB 6. PENUTUP

Bab ini merupakan bagian akhir di dalam penulisan skripsi, berisi tentang kesimpulan dan saran. Kesimpulan yang ditulis merupakan kesimpulan dari hasil penelitian yang telah dilakukan dan saran lanjutan untuk dilakukan pada penelitian selanjutnya.

6.1. Kesimpulan

Adapun kesimpulan dari penelitian yang telah dilakukan adalah sebagai berikut :

1. Kemampuan aplikasi *Banana Maturity Identification* dalam mengidentifikasi buah pisang dengan citra digital yang diinputkan melalui *camera* atau media penyimpanan *smartphone* menggunakan algoritma *Learning Vector Quantization (LVQ)* berbeda – berbeda, hal ini dipengaruhi beberapa kondisi seperti berikut ini :
   a. Berdasarkan jumlah *dataset* yang digunakan pada proses *training* dengan *learning rate* = 0,5 , *eps* = 0,001 dan *max epoch* 10 kali, aplikasi ini mampu mengidentifikasi citra digital buah pisang dengan tingkat akurasi kebenaran mencapai 90 % dengan jumlah *dataset* = 9 data dan 12 data serta tingkat akurasi kebenaran hanya 70 % dengan jumlah *dataset* = 5 data menggunakan perangkat *smartphone Sony Xperia E* pada jarak pengambilan citra digital 14 cm.
   b. Berdasarkan jumlah *max epoch* yang digunakan pada proses *training* dengan *learning rate* = 0,5 , *eps* = 0,001 dan jumlah *dataset* = 12 data, aplikasi ini mampu mengidentifikasi citra digital buah pisang dengan tingkat akurasi kebenaran mencapai 90 % dengan jumlah *max epoch* = 5 kali dan 10 kali serta tingkat akurasi kebenaran hanya 80 % dengan jumlah *max epoch* = 25 kali menggunakan perangkat *smartphone Sony Xperia E* pada jarak pengambilan citra digital 14 cm.
   c. Berdasarkan jarak pengambilan citra digital dengan acuan bobot akhir hasil proses *training* dengan *learning rate* = 0,5 , *eps* = 0,001 dan

jumlah *dataset* = 12 data, aplikasi ini mampu mengidentifikasi citra digital buah pisang dengan tingkat akurasi kebenaran mencapai 90 % dengan jarak pengambilan = 14 cm serta tingkat akurasi kebenaran hanya 30 % dengan jarak pengambilan = 20 cm dan 30 cm menggunakan perangkat *smartphone Sony Xperia E*.

d. Berdasarkan *smartphone* yang digunakan dengan acuan bobot akhir hasil proses *training* dengan *learning rate* = 0,5 , *eps* = 0,001 dan jumlah *dataset* = 12 data menggunakan perangkat *smartphone Sony Xperia E* pada jarak pengambilan citra digital 14 cm, aplikasi ini mampu mengidentifikasi buah pisang dengan tingkat akurasi kebenaran mencapai 90 % menggunakan perangkat *smartphone Sony Xperia E* dengan lama proses 1,03 *second*, *Evercoss A28T* dengan lama proses 0,88 *second*, *Samsung Galaxy Core* dengan lama proses 0,82 *second*, dan *Asus Zenfone 5* dengan lama proses 0,81 *second* serta tingkat akurasi kebenaran hanya 80 % ketika menggunakan perangkat *Samsung Galaxy Chat* dengan lama proses 1,00 *second*. Lama proses yang dihasilkan masing – masing *smartphone* dipengaruhi oleh *OS* dan *hardware* yang dimiliki.

2. Tingkat akurasi kebenaran pada aplikasi juga dipengaruhi dari kualitas citra digital yang diinputkan. Kualitas citra digital sendiri dipengaruhi oleh tingkat resolusi *camera* yang dimiliki oleh *smartphone* yang digunakan. Semakin baik resolusi *camera* yang dimiliki *smartphone* maka semakin baik kualitas citra digital yang didapat dan akurasi pada aplikasi juga semakin baik.
3. Berdasarkan hasil pengujian, adapun kondisi optimal aplikasi ini mampu mengidentifikasi citra digital mencapai 90 % menggunakan *smartphone Sony Xperia E* yaitu jumlah *dataset* yang digunakan 9 data dan 12 data, jumlah *max epoch* 5 kali dan 10 kali, dan jarak pengambilan citra digital 14 cm.
4. Aplikasi *Banana Maturity Identification* ini hanya mampu melakukan klasifikasi benar dan klasifikasi salah.

6.2. Saran

Pengembangan lebih lanjut untuk penelitian ini dapat dilakukan dengan membangun aplikasi *Banana Maturity Identification* platform *mobile* lainnya seperti *iOS*, *windows phone*, dan lain sebagainya serta disarankan menggunakan metode yang lainnya untuk menciptakan perbandingan antar metode yang satu dengan yang lain.

DAFTAR PUSTAKA

Ahmad,Usman.2005.*Pengolahan Citra Digital dan Teknik Pemrogramannya*.Yogyakarta : Graha Ilmu

Agissa,Wildan. 2013. *White Box and Black Box Testing. http://bangwildan.web.id/*

Ayuliana.2009.Testing dan Implementasi. *http://rifiana.staff.gunadarma.ac.id/*

Budhi, Gregorius Satia dkk.2009. *Algoritma Generalized Sequential Pattern Untuk Menggali Data Sekuensial Sirkulasi Buku Pada Perpustakaan UK Petra*. Seminar Nasional Aplikasi Teknologi Informasi (SNATI) 2009. ISBN : 979-756-061-6

Badan Pusat Statistik.2013.*Produksi Buah-buahan dan Sayuran Tahunan di Indonesia 1995 - 2013.http://www.bps.go.id/*

Deswari, Dila dkk.2013.*Identifikasi Kematangan Buah Tomat Menggunakan Metoda Backpropagation.http://repository.unand.ac.id/*

Fayyad, Usama dkk.1996. *From Datamining to Knowledge Discovery in Databases.* AI Magazine : 37-54

Han, Jiawei dkk.2000. *Data Mining: Concepts and Techniques*. San Francisco : Morgan Kaufmann Publishers

Hendrayudi.2009.*VB 2008 untuk Berbagai Keperluan Programming*.Jakarta : PT Elex Media Komputindo

Hidayati, Nurul dkk.2010.*Prediksi Terjangkitnya Penyakit Jantung Dengan Metode Learning Vector Quantization*.Jurnal Media Statistika, Vol. 3, No 1, Juni 2010: 21 - 30

Jogiyanto, H.M. 2004. *Pengenalan Komputer: Dasar Ilmu Komputer, Pemrograman, Sistem Informasi dan Intelegensi Buatan*. Yogyakarta: Andi Publisher.

Kusumadewi, Sri. (2003). *Artificial Intellegence (Teknik dan Aplikasinya).* Yogyakarta : Graha Ilmu.

Kusrini, dkk.2009.*Algoritma Data Mining*.Yogyakarta : CV.Andi Offset

Munir, Rinaldi.2004.*Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Bandung : Informatika

Munir, Rinaldi.2006.*Aplikasi Image Thresholding Untuk Segmentasi Objek*.Seminar Nasional Aplikasi Teknologi Informasi (SNATI) 2006. ISSN : 1907 – 5022

Prabawati, Sulusi dkk.2008.Teknologi Pascapanen dan Teknik Pengolahan Buah Pisang.Jakarta.Badan Penelitian dan Pengembangan Pertanian

Pressman, R. S. 2002. *Rekayasa Perangkat Lunak Pendekatan Praktisi*. Yogyakarta : Andi

Pressman, R. S. 2012. *Rekayasa Perangkat Lunak Pendekatan Praktisi Edisi 7 Buku 1*. Yogyakarta : Andi

Putra, Darma.2010.*Pengolahan Citra Digital*.Yogyakarta : CV.Andi Offset

Qur'ani, Difla Yustisia dkk.2010.*Jaringan Syaraf Tiruan Learning Vector Quatization Untuk Aplikasi Pengenalan Tanda Tangan*.Seminar Nasional Aplikasi Teknologi Informasi (SNATI) 2010.ISSN : 1907-5022

Ranadhi, Djalu dkk.2006.*Implementasi Learning Vector Quantization (LVQ) Untuk Pengenal Pola Sisik Jari Pada Sistem Informasi Narapidana LP Wirogunan*. Jurnal Media Informatika, Volume 4 Nomor 1, 51 – 56, ISSN : 0854-4743

Santoso,Bambang B.2011.*Kematangan Produk dan Indek Panen*. *http://fp.unram.ac.id/*

Sommerville, Ian.2000.*Software Engineering 6th Edition*.Boston : Addison Wesley. ISBN-10 : 020139815X, ISBN-13 : 978-0201398151

Supriyanto, Dodit, dkk.2012.*Pemrograman Aplikasi Android : Step by Step Membuat Aplikasi Android untuk Smartphone dan Tablet*.Yogyakarta : Mediakom

Sunarjono.2000. Prospek Tanaman Buah. Jakarta: Penebar Swadaya

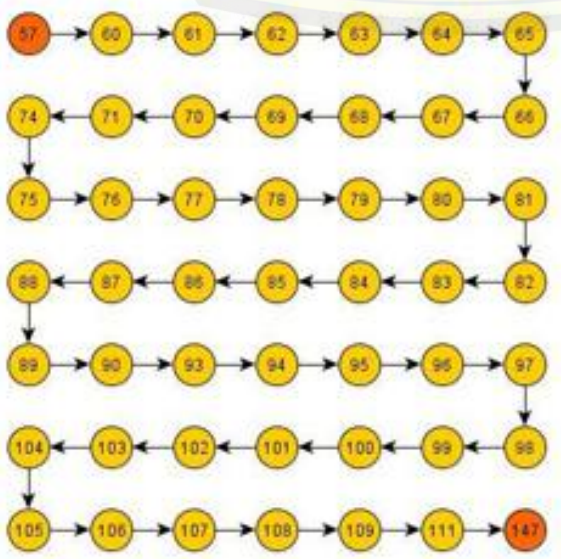
Wuryandri, Maharani Dessy dkk.2012. *Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation Dan Learning Vector Quantization Pada Pengenalan Wajah*. Jurnal Komputer dan Informatika (KOMPUTA) 2012. Edisi I Volume I

LAMPIRAN

A. Pengujian *White Box*

A1. Membuat *Database* Aplikasi

```
57    public void onCreate(SQLiteDatabase dbCreate) {
58
59        // Query Membuat Tabel Kelas
60        String Tabel_Kelas = "CREATE TABLE " + TB_KELAS + " (" + COLUMN_ID_KELAS+ " INTEGER PRIMARY KEY, " +
61                             COLUMN_NAMA_KELAS + " VARCHAR(20))";
62        dbCreate.execSQL(Tabel_Kelas);
63        String isi_tbKelas1 = "INSERT INTO " + TB_KELAS + " (" + COLUMN_ID_KELAS + "," + COLUMN_NAMA_KELAS +
64                              ") VALUES (1, 'Matang')";
65        dbCreate.execSQL(isi_tbKelas1);
66        String isi_tbKelas2 = "INSERT INTO " + TB_KELAS + " (" + COLUMN_ID_KELAS + "," + COLUMN_NAMA_KELAS +
67                              ") VALUES (2, 'Setengah Matang')";
68        dbCreate.execSQL(isi_tbKelas2);
69        String isi_tbKelas3 = "INSERT INTO " + TB_KELAS + " (" + COLUMN_ID_KELAS + "," + COLUMN_NAMA_KELAS +
70                              ") VALUES (3, 'Muda')";
71        dbCreate.execSQL(isi_tbKelas3);
72
73        // Query Membuat Tabel Data Training
74        String Tabel_TrainingData = "CREATE TABLE " + TB_TRAININGDATA + " (" + COLUMN_ID_TD +
75                                    " INTEGER PRIMARY KEY AUTOINCREMENT, " + COLUMN_TD_RED_IMAGE + " DOUBLE(15), " +
76                                    COLUMN_TD_GREEN_IMAGE + " DOUBLE(15), " + COLUMN_TD_BLUE_IMAGE + " DOUBLE(15), " +
77                                    COLUMN_TARGET_KELAS + " INTEGER(3))";
78        dbCreate.execSQL(Tabel_TrainingData);
79        String isi_TBTrainingData1 = "INSERT INTO " + TB_TRAININGDATA + " (" + COLUMN_ID_TD + "," + COLUMN_TD_RED_IMAGE +
80                                     "," + COLUMN_TD_GREEN_IMAGE + "," + COLUMN_TD_BLUE_IMAGE + "," + COLUMN_TARGET_KELAS +
81                                     ") VALUES (1, 0.367, 0.352, 0.279, 1)";
82        dbCreate.execSQL(isi_TBTrainingData1);
83        String isi_TBTrainingData2 = "INSERT INTO " + TB_TRAININGDATA + " (" + COLUMN_ID_TD + "," + COLUMN_TD_RED_IMAGE +
84                                     "," + COLUMN_TD_GREEN_IMAGE + "," + COLUMN_TD_BLUE_IMAGE + "," + COLUMN_TARGET_KELAS +
85                                     ") VALUES (2, 0.355, 0.37, 0.273, 2)";
86        dbCreate.execSQL(isi_TBTrainingData2);
87        String isi_TBTrainingData3 = "INSERT INTO " + TB_TRAININGDATA + " (" + COLUMN_ID_TD + "," + COLUMN_TD_RED_IMAGE +
88                                     "," + COLUMN_TD_GREEN_IMAGE + "," + COLUMN_TD_BLUE_IMAGE + "," + COLUMN_TARGET_KELAS +
89                                     ") VALUES (3, 0.334, 0.363, 0.301, 3)";
90        dbCreate.execSQL(isi_TBTrainingData3);
91
92
93    String Tabel_TrainingResult = "CREATE TABLE " + TB_TRAININGRESULT + " (" + COLUMN_ID_TR +
94                                  " INTEGER PRIMARY KEY AUTOINCREMENT, " + COLUMN_TR_RED_IMAGE + " DOUBLE(15), " +
95                                  COLUMN_TR_GREEN_IMAGE + " DOUBLE(15), " + COLUMN_TR_BLUE_IMAGE + " DOUBLE(15), " +
96                                  COLUMN_NO_TD + " INTEGER(3))";
97    dbCreate.execSQL(Tabel_TrainingResult);
98    String isi_TBTrainingResult1 = "INSERT INTO " + TB_TRAININGRESULT + " (" + COLUMN_ID_TR + "," + COLUMN_TR_RED_IMAGE +
99                                   "," + COLUMN_TR_GREEN_IMAGE + "," + COLUMN_TR_BLUE_IMAGE + "," + COLUMN_NO_TD +
100                                  ") VALUES (1, 100.0, 40.057, 40.057, 1)";
101   dbCreate.execSQL(isi_TBTrainingResult1);
102   String isi_TBTrainingResult2 = "INSERT INTO " + TB_TRAININGRESULT + " (" + COLUMN_ID_TR + "," + COLUMN_TR_RED_IMAGE +
103                                  "," + COLUMN_TR_GREEN_IMAGE + "," + COLUMN_TR_BLUE_IMAGE + "," + COLUMN_NO_TD +
104                                  ") VALUES (2, 100.0, 80.166, 100.0, 2)";
105   dbCreate.execSQL(isi_TBTrainingResult2);
106   String isi_TBTrainingResult3 = "INSERT INTO " + TB_TRAININGRESULT + " (" + COLUMN_ID_TR + "," + COLUMN_TR_RED_IMAGE +
107                                  "," + COLUMN_TR_GREEN_IMAGE + "," + COLUMN_TR_BLUE_IMAGE + "," + COLUMN_NO_TD +
108                                  ") VALUES (3, 10.0, 100.0, 10.0, 3)";
109   dbCreate.execSQL(isi_TBTrainingResult3);
110
111 }
```



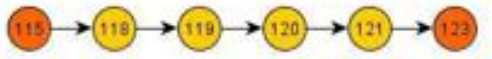
CC = Edge – Node + 2

CC = 48 – 49 + 2

CC = 1

A2. *Upgrade Database*

```
115  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
116
117      // Drop older table if existed
118      db.execSQL("DROP TABLE IF EXISTS" + TB_KELAS);
119      db.execSQL("DROP TABLE IF EXISTS" + TB_TRAININGDATA);
120      db.execSQL("DROP TABLE IF EXISTS" + TB_TRAININGRESULT);
121      onCreate(db); // Create tables again
122
123  }
```



CC = Edge – Node + 2
CC = 5 – 6 + 2
CC = 1

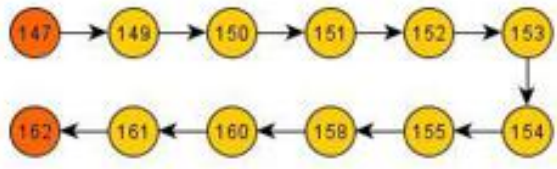
A3. Membaca Data Kelas Pada *Database*

```
127⊖ public ArrayList<String> getAllKelas() {
128
129      listKelas = new ArrayList<String>();
130      dbRead = this.getReadableDatabase();
131      String query = "SELECT * FROM " + TB_KELAS;
132      Cursor cursor = dbRead.rawQuery(query, null);
133      // pindah ke data paling pertama
134      cursor.moveToFirst();
135      // jika masih ada data, masukkan data ke list data kelas
136      while (!cursor.isAfterLast()) {
137          listKelas.add(cursor.getString(1));
138          cursor.moveToNext();
139      }
140      cursor.close();
141      dbRead.close();
142      return listKelas;
143  }
```

CC = Edge – Node + 2
CC = 17 – 17 + 2
CC = 2

A4. *Cursor Dataset Database*

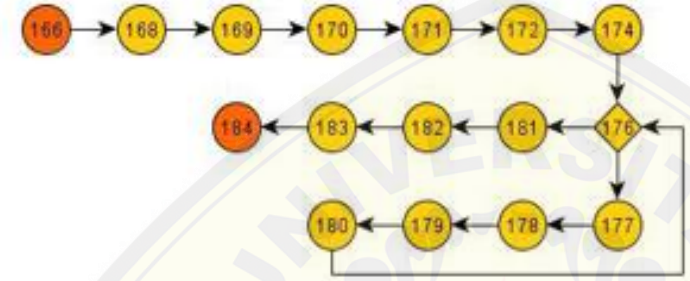
```
147⊖ private Atribut_Model_TrainingData cursorTrainingData(Cursor cursor) {
148
149      String [] data = new String [] {
150              cursor.getString(cursor.getColumnIndex(COLUMN_ID_TD)),
151              cursor.getString(cursor.getColumnIndex(COLUMN_TD_RED_IMAGE)),
152              cursor.getString(cursor.getColumnIndex(COLUMN_TD_GREEN_IMAGE)),
153              cursor.getString(cursor.getColumnIndex(COLUMN_TD_BLUE_IMAGE)),
154              cursor.getString(cursor.getColumnIndex(COLUMN_NAMA_KELAS))
155      };
156
157      // buat objek training data baru
158      amtd = new Atribut_Model_TrainingData();
159      /* set atribut pada objek data training dengan data kursor yang diambil dari database*/
160      amtd.setAttribut(data[0], data[1], data[2], data[3], data[4], false);
161      return amtd;
162  }
```



CC = Edge – Node + 2
CC = 11 – 12 + 2
CC = 1

A5. Membaca Semua *Dataset* Pada *Database*

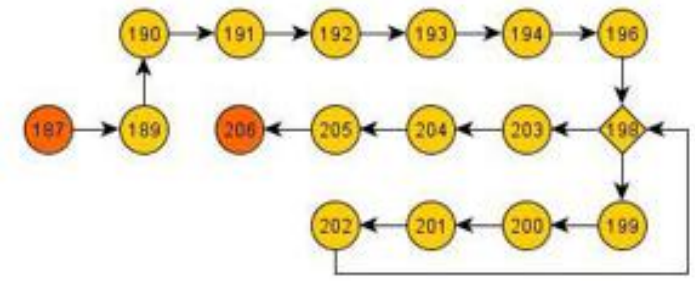
```
public ArrayList<Atribut_Model_TrainingData> getAllTrainingData() {

    listTrainingData = new ArrayList<Atribut_Model_TrainingData>();
    dbRead = this.getReadableDatabase();
    String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_KELAS + " tbk ON tbdt." +
                    COLUMN_TARGET_KELAS + " = tbk."+ COLUMN_ID_KELAS +" WHERE "+ COLUMN_ID_TD + " NOT IN (1,2,3) ";
    Cursor cursor = dbRead.rawQuery(query, null);
    // pindah ke data paling pertama
    cursor.moveToFirst();
    // jika masih ada data, masukkan data ke list training data
    while (!cursor.isAfterLast()) {
        Atribut_Model_TrainingData td = cursorTrainingData(cursor);
        listTrainingData.add(td);
        cursor.moveToNext();
    }
    cursor.close();
    dbRead.close();
    return listTrainingData;
}
```



CC = Edge – Node + 2
CC = 16 – 16 + 2
CC = 2

A6. Membaca *Dataset* Yang Dipilih Sebagai Bobot Awal Pada Masing – Masing Kelas Pada *Database*

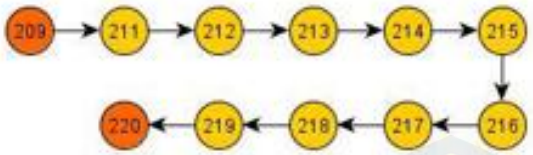
```
public ArrayList<Atribut_Model_TrainingData> getTrainingDataCW (int id_tdCW1, int id_tdCW2, int id_tdCW3) {

    listTrainingData = new ArrayList<Atribut_Model_TrainingData>();
    dbRead = this.getReadableDatabase();
    String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_KELAS + " tbk ON tbdt." +
                    COLUMN_TARGET_KELAS + " = tbk."+ COLUMN_ID_KELAS +" WHERE " + COLUMN_ID_TD +
                    " IN ( "+ id_tdCW1 +", "+ id_tdCW2 +", "+ id_tdCW3 +" ) ";
    Cursor cursor = dbRead.rawQuery(query, null);
    // pindah ke data paling pertama
    cursor.moveToFirst();
    // jika masih ada data, masukkan data ke list training data
    while (!cursor.isAfterLast()) {
        Atribut_Model_TrainingData td = cursorTrainingData(cursor);
        listTrainingData.add(td);
        cursor.moveToNext();
    }
    cursor.close();
    dbRead.close();
    return listTrainingData;
}
```



CC = Edge – Node + 2
CC = 16 – 16 + 2
CC = 2

A7. *Count Dataset* Yang Dipilih Sebagai Bobot Awal Pada *Database*

```
209  public int getCountTrainingDataCW (int id_tdCW1, int id_tdCW2, int id_tdCW3) {
210
211      dbRead = this.getReadableDatabase();
212      String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_KELAS + " tbk ON tbdt." +
213                      COLUMN_TARGET_KELAS + " = tbk."+ COLUMN_ID_KELAS +" WHERE " + COLUMN_ID_TD +
214                      " IN ( "+ id_tdCW1 +", "+ id_tdCW2 +", "+ id_tdCW3 +" ) ";
215      Cursor cursor = dbRead.rawQuery(query, null);
216      int count = cursor.getCount();
217      cursor.close();
218      dbRead.close();
219      return count;
220  }
```



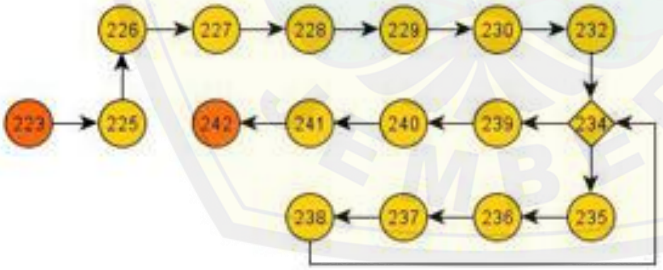
CC = Edge – Node + 2

CC = 10 – 11 + 2

CC = 1

A8. Membaca Dataset Yang Tidak Dipilih Sebagai Bobot Awal Pada Masing – Masing Kelas Pada *Database*

```
2239  public ArrayList<Atribut_Model_TrainingData> getTrainingDataOtherCW (int id_tdCW1, int id_tdCW2, int id_tdCW3) {
2240
2251      listTrainingData = new ArrayList<Atribut_Model_TrainingData>();
2262      dbRead = this.getReadableDatabase();
2273      String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_KELAS + " tbk ON tbdt." +
2284                      COLUMN_TARGET_KELAS + " = tbk."+ COLUMN_ID_KELAS +" WHERE " + COLUMN_ID_TD +
2295                      " NOT IN ( 1, 2, 3, "+ id_tdCW1 +", "+ id_tdCW2 +", "+ id_tdCW3 +" ) ";
2306      Cursor cursor = dbRead.rawQuery(query, null);
2317      // pindah ke data paling pertama
2328      cursor.moveToFirst();
2339      // jika masih ada data, masukkan data ke list training data
2340      while (!cursor.isAfterLast()) {
2351          amtd = cursorTrainingData(cursor);
2362          listTrainingData.add(amtd);
2373          cursor.moveToNext();
2384      }
2395      cursor.close();
2406      dbRead.close();
2417      return listTrainingData;
2428  }
```



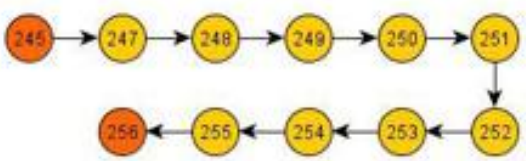
CC = Edge – Node + 2

CC = 17 – 17 + 2

CC = 2

A9. *Count Dataset* Yang Tidak Dipilih Sebagai Bobot Awal Pada *Database*

```
2459  public int getCountTrainingDataOtherCW (int id_tdCW1, int id_tdCW2, int id_tdCW3) {
2460
2471      dbRead = this.getReadableDatabase();
2482      String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_KELAS + " tbk ON tbdt." +
2493                      COLUMN_TARGET_KELAS + " = tbk."+ COLUMN_ID_KELAS +" WHERE " + COLUMN_ID_TD +
2504                      " NOT IN ( 1, 2, 3, "+ id_tdCW1 +", "+ id_tdCW2 +", "+ id_tdCW3 +" ) ";
2515      Cursor cursor = dbRead.rawQuery(query, null);
2526      int count = cursor.getCount();
2537      cursor.close();
2548      dbRead.close();
2559      return count;
2560  }
```



CC = Edge – Node + 2

CC = 10 – 11 + 2

CC = 1

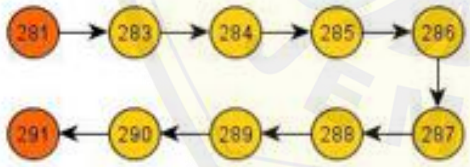
A10. Membaca *Dataset* Berdasarkan Kelas Pada *Database*

```
public ArrayList<Atribut_Model_TrainingData> getTrainingDataEachKelas(int target_kelas) {

    listTrainingData = new ArrayList<Atribut_Model_TrainingData>();
    dbRead = this.getReadableDatabase();
    String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_KELAS + " tbk ON tbdt." +
                    COLUMN_TARGET_KELAS + " = tbk."+ COLUMN_ID_KELAS + " WHERE " + COLUMN_TARGET_KELAS +
                    " = "+target_kelas+ " AND "+ COLUMN_ID_TD + " NOT IN (1,2,3)";
    Cursor cursor = dbRead.rawQuery(query, null);
    // pindah ke data paling pertama
    cursor.moveToFirst();
    // jika masih ada data, masukkan data ke list training data
    while (!cursor.isAfterLast()) {
        amtd = cursorTrainingData(cursor);
        listTrainingData.add(amtd);
        cursor.moveToNext();
    }
    cursor.close();
    dbRead.close();
    return listTrainingData;
}
```

CC = Edge – Node + 2
CC = 17 – 17 + 2
CC = 2

A11. *Insert Dataset* Pada *Database*

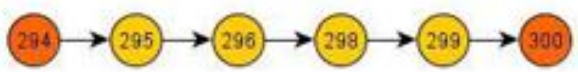
```
public void insertTrainingData(double td_red_image, double td_green_image, double td_blue_image, int target_kelas) {

    dbWrite = this.getWritableDatabase();
    values = new ContentValues();
    values.put(COLUMN_TD_RED_IMAGE,  td_red_image);
    values.put(COLUMN_TD_GREEN_IMAGE, td_green_image);
    values.put(COLUMN_TD_BLUE_IMAGE, td_blue_image);
    values.put(COLUMN_TARGET_KELAS, target_kelas);
    dbWrite.insert(TB_TRAININGDATA, null, values);
    dbWrite.close();
}
```



CC = Edge – Node + 2
CC = 9 – 10 + 2
CC = 1

A12. *Reset Dataset* Pada *Database*

```
public void resetTrainingData() {
    dbWrite = this.getWritableDatabase();
    dbWrite.delete(TB_TRAININGDATA, COLUMN_ID_TD + " NOT IN (1,2,3) ", null);
    // reset auto-increment
    dbWrite.delete("SQLITE_SEQUENCE", "NAME = '" + TB_TRAININGDATA + "' AND '" + COLUMN_ID_TD + "' NOT IN (1,2,3) ", null);
    dbWrite.close();
}
```



CC = Edge – Node + 2
CC = 5 – 6 + 2
CC = 1

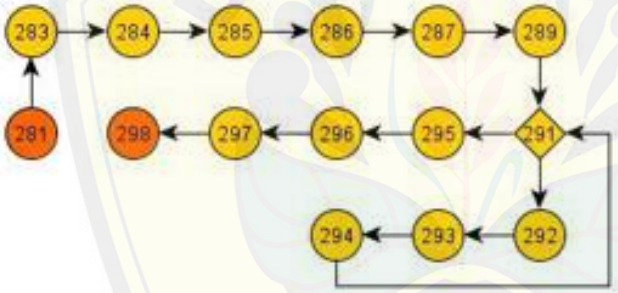
A13. Menghapus *Dataset* Pada *Database*

```
public void deleteTrainingData(int id_td) {
    dbWrite = this.getWritableDatabase();
    dbWrite.delete(TB_TRAININGDATA, COLUMN_ID_TD + " = " +id_td+ " ", null);
    dbWrite.close();
}
```

CC = Edge – Node + 2
CC = 5 – 6 + 2
CC = 1

A14. Membaca Salah Satu *Dataset* Pada *Database*

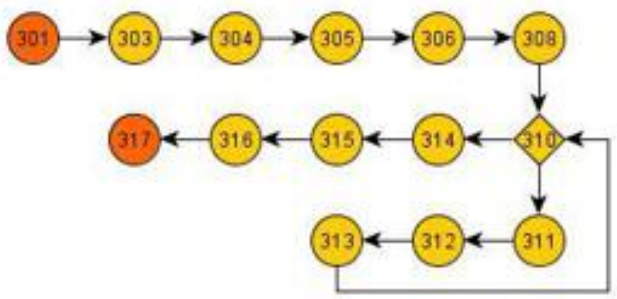
```
public ArrayList<String> getOneTrainingData(int id_td) {

    ArrayList<String> listOneTrainingData = new ArrayList<String>();
    dbRead = this.getReadableDatabase();
    String query  = "SELECT * FROM " + TB_TRAININGDATA + " tbdt JOIN " + TB_TRAININGRESULT + " tbtr ON tbdt." +
                    COLUMN_ID_TD + " = tbtr."+ COLUMN_NO_TD +" WHERE "+ COLUMN_NO_TD + " = "+id_td+"";
    Cursor cursor = dbRead.rawQuery(query, null);
    // pindah ke data paling pertama
    cursor.moveToFirst();
    // jika masih ada data, masukkan data ke list training data
    while (!cursor.isAfterLast()) {
        listOneTrainingData.add(cursor.getString(cursor.getColumnIndex(COLUMN_NO_TD)));
        cursor.moveToNext();
    }
    cursor.close();
    dbRead.close();
    return listOneTrainingData;
}
```



CC = Edge – Node + 2
CC = 15 – 15 + 2
CC = 2

A15. Membaca Salah Satu Kelas *Dataset* Pada *Database*

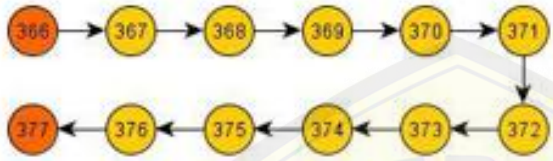
```
public ArrayList<String> getKelasOneTrainingData(int id_td) {

    ArrayList<String> listKelasOneTrainingData = new ArrayList<String>();
    dbRead = this.getReadableDatabase();
    String query  = "SELECT "+ COLUMN_TARGET_KELAS +" FROM "+ TB_TRAININGDATA +" WHERE "+ COLUMN_ID_TD + " = "+id_td+"";
    Cursor cursor = dbRead.rawQuery(query, null);
    // pindah ke data paling pertama
    cursor.moveToFirst();
    // jika masih ada data, masukkan data ke list training data
    while (!cursor.isAfterLast()) {
        listKelasOneTrainingData.add(cursor.getString(cursor.getColumnIndex(COLUMN_TARGET_KELAS)));
        cursor.moveToNext();
    }
    cursor.close();
    dbRead.close();
    return listKelasOneTrainingData;
}
```



CC = Edge – Node + 2
CC = 14 – 14 + 2
CC = 2

A16. *Cursor Training Result* Pada *Database*

```
3120    private Atribut_Model_TrainingResult cursorTrainingResult(Cursor cursor)  {
313
314         String [] data = new String [] {
315                 cursor.getString(cursor.getColumnIndex(COLUMN_ID_TR)),
316                 cursor.getString(cursor.getColumnIndex(COLUMN_TR_RED_IMAGE)),
317                 cursor.getString(cursor.getColumnIndex(COLUMN_TR_GREEN_IMAGE)),
318                 cursor.getString(cursor.getColumnIndex(COLUMN_TR_BLUE_IMAGE)),
319                 cursor.getString(cursor.getColumnIndex(COLUMN_NO_TD)),
320                 cursor.getString(cursor.getColumnIndex(COLUMN_NAMA_KELAS))
321         };
322
323         // buat objek training data baru
324         amtr = new Atribut_Model_TrainingResult();
325         /* set atribut pada objek data training dengan data kursor yang diambil dari database*/
326         amtr.setAttribut(data[0], data[1], data[2], data[3], data[4], data[5]);
327         return amtr;
328     }
```



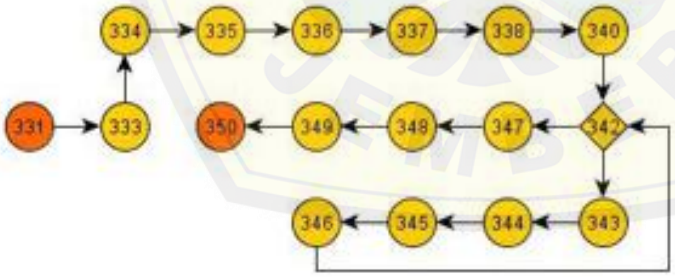
CC = Edge – Node + 2

CC = 11 – 12 + 2

CC = 1

A17. Membaca *Training Result* Pada *Database*

```
3310    public ArrayList<Atribut_Model_TrainingResult> getAllTrainingResult() {
332
333         listTrainingResult = new ArrayList<Atribut_Model_TrainingResult>();
334         dbRead = this.getReadableDatabase();
335         String query  = "SELECT * FROM " + TB_TRAININGRESULT + " tbht JOIN " + TB_TRAININGDATA + " tbdt ON tbht." +
336                         COLUMN_NO_TD + " = tbdt." + COLUMN_ID_TD + " JOIN " + TB_KELAS + " tbk ON tbdt." +
337                         COLUMN_TARGET_KELAS + " = tbk." + COLUMN_ID_KELAS;
338         Cursor cursor = dbRead.rawQuery(query, null);
339         // pindah ke data paling pertama
340         cursor.moveToFirst();
341         // jika masih ada data, masukkan data ke list training result
342         while (!cursor.isAfterLast()) {
343             Atribut_Model_TrainingResult tr = cursorTrainingResult(cursor);
344             listTrainingResult.add(tr);
345             cursor.moveToNext();
346         }
347         cursor.close();
348         dbRead.close();
349         return listTrainingResult;
350     }
```



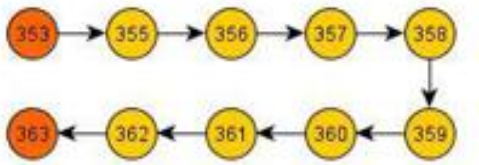
CC = Edge – Node + 2

CC = 17 – 17 + 2

CC = 2

A18. *Update Training Result* Pada *Database*

```
3539    public void updateTrainingResult (int id_tr, double tr_red_image, double tr_green_image, double tr_blue_image, int no_dt)
354
355         dbWrite = this.getWritableDatabase();
356         values = new ContentValues();
357         values.put(COLUMN_TR_RED_IMAGE,  tr_red_image);
358         values.put(COLUMN_TR_GREEN_IMAGE, tr_green_image);
359         values.put(COLUMN_TR_BLUE_IMAGE, tr_blue_image);
360         values.put(COLUMN_NO_TD, no_dt);
361         dbWrite.update(TB_TRAININGRESULT, values, COLUMN_ID_TR + " = " + id_tr, null);
362         dbWrite.close();
363     }
```



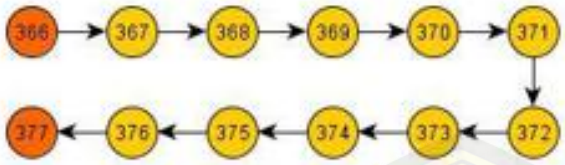
CC = Edge – Node + 2

CC = 9 – 10 + 2

CC = 1

A19. *Count Training Result* Pada *Database*

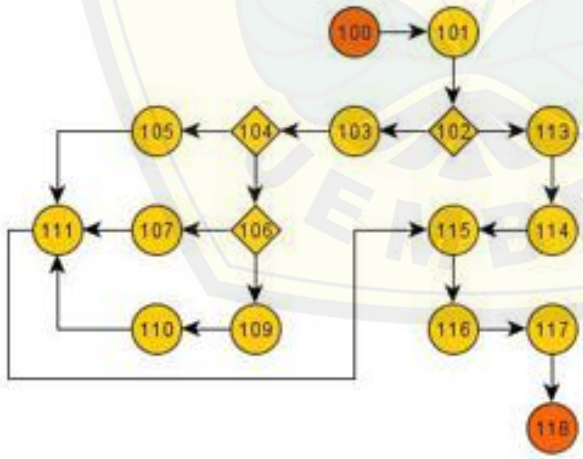
```
public int getCountTrainingResult () {

    dbRead = this.getReadableDatabase();
    String query  = "SELECT * FROM " + TB_TRAININGRESULT + " tbht JOIN " + TB_TRAININGDATA + " tbdt ON tbht." +
                    COLUMN_NO_TD + " = tbdt." + COLUMN_ID_TD + " JOIN " + TB_KELAS + " tbk ON tbdt." +
                    COLUMN_TARGET_KELAS + " = tbk." + COLUMN_ID_KELAS;
    Cursor cursor = dbRead.rawQuery(query, null);
    int count = cursor.getCount();
    cursor.close();
    dbRead.close();
    return count;
}
```



CC = Edge – Node + 2
CC = 11 – 12 + 2
CC = 1

A20. Menghapus *Dataset* Pada *Controller*

```
builder.dialog_box.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        if(!db.getOneTrainingData(v.getId()).isEmpty()) {
            ck = db.getKelasOneTrainingData(v.getId()).get(0);
            if(ck.equals("1")){
                db.updateTrainingResult(1, 0.362, 0.352, 0.296, 1);
            } else if (ck.equals("2")){
                db.updateTrainingResult(2, 0.359, 0.362, 0.268, 2);
            } else {
                db.updateTrainingResult(3, 0.329, 0.353, 0.31, 3);
            }
            db.deleteTrainingData(v.getId());
        } else {
            db.deleteTrainingData(v.getId());
        }
        ctdm = new c_TrainingData_Menu();
        ctdm.viewTrainingDataMenu(context);
    }
});
```



CC = Edge – Node + 2
CC = 19 – 17 + 2
CC = 4

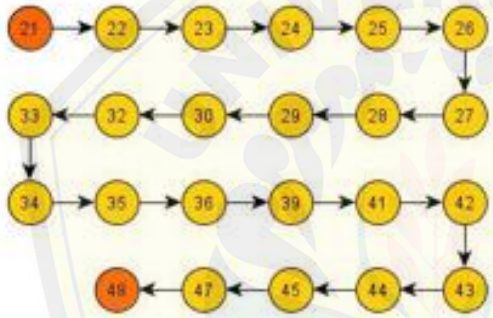
A21. Memilih *Dataset* Sebagai Bobot Pada *Controller*

```
holder.chkBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        listTrainingData.get((Integer) buttonView.getTag()).checked = isChecked;
    }
});
```

CC = Edge – Node + 2
CC = 4 – 5 + 2
CC = 1

A22. Tampilan *Splash Screen*

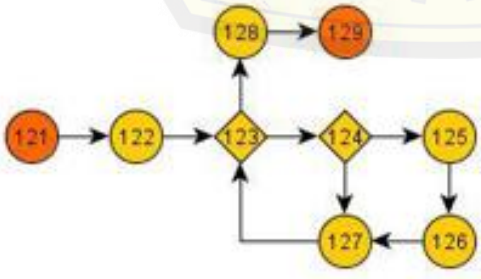
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE); // menghilangkan title app
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
    setContentView(R.layout.splash_screen);
    anim = AnimationUtils.loadAnimation(this, R.anim.alpha_effect);
    anim.reset();
    LinearLayout layout = (LinearLayout) findViewById(R.id.ll_splashscreen);
    layout.clearAnimation();
    layout.startAnimation(anim);

    anim = AnimationUtils.loadAnimation(this, R.anim.translate_effect);
    anim.reset();
    imgv = (ImageView) findViewById(R.id.splash_img);
    imgv.clearAnimation();
    imgv.startAnimation(anim);

    //Digunakan untuk menghandler activity baru dari splashscreen
    new Handler().postDelayed(new Thread () {
        @Override
        public void run(){
            ch = new c_Home();
            ch.viewHome(v_Splash_Screen.this);
            overridePendingTransition(R.anim.fade_in_translation, R.anim.fade_out_translation);
        }

    },4000); // 4000 (4 detik) adalah waktu splash screen muncul sebelum masuk ke activity baru
}
```



CC = Edge – Node + 2
CC = 22 – 23 + 2
CC = 1

A23. Memasukkan *Dataset* Yang Dipilih Pada *ArrayList*

```
public ArrayList<Atribut_Model_TrainingData> getDataChecked() {
    ArrayList<Atribut_Model_TrainingData> DataChecked = new ArrayList<Atribut_Model_TrainingData>();
    for (Atribut_Model_TrainingData td : listTrainingData) {
        if (td.isChecked()) {
            DataChecked.add(td);
        }
    }
    return DataChecked;
}
```



CC = Edge – Node + 2
CC = 10 – 9 + 2
CC = 3

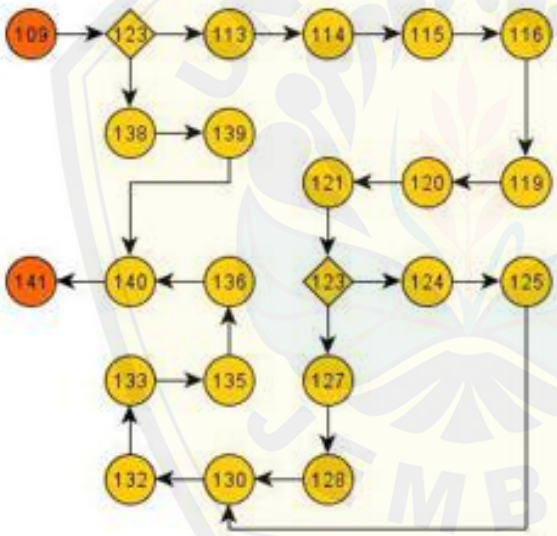
A24. Mengambil RGB *Image*

```
public void btnGetRGBImageActionTask(Context context){
    grt = new GetRGBTask();
    grt.Task(context, bm_resize, width_resize, height_resize);
    grt.execute();
}
```

CC = Edge – Node + 2
CC = 4 – 5 + 2
CC = 1

A25. *Decode File Bitmap*

```
110     private static Bitmap decodeFile(String path, int width_heap, int height_heap) {
111
112         try {
113
114             file = new File(path);
115             bo = new BitmapFactory.Options();
116             bo.inJustDecodeBounds = true;
117             BitmapFactory.decodeStream(new FileInputStream(file),null,bo);
118
119             //The scale manipulation image on memory heap
120             require_width_heap  = width_heap;
121             require_height_heap = height_heap;
122             scale = 0;
123
124             if (bo.outWidth > require_width_heap || bo.outHeight > require_height_heap ) {
125                 scale = Math.min(bo.outWidth / require_width_heap, bo.outHeight / require_height_heap);
126             }
127             else {
128                 scale = 1; //Original heap size image on memory
129             }
130
131             System.out.println("cek scale = "+scale);
132
133             bo = new BitmapFactory.Options();
134             bo.inSampleSize = scale;
135
136             return BitmapFactory.decodeStream(new FileInputStream(file), null, bo);
137         }
138         catch (FileNotFoundException e) {
139             e.printStackTrace();
140         }
141             return null;
142         }
```



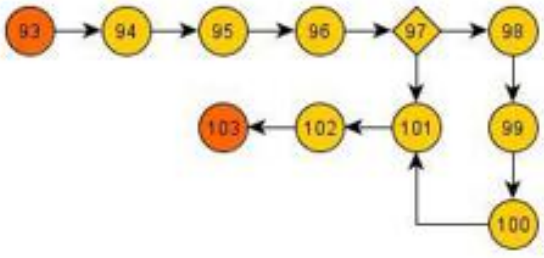
CC = Edge – Node + 2

CC = 24 – 23 + 2

CC = 3

A26. Mengambil *Path / Url Image*

```
93      public String getRealPath(Context context, Uri contentUri) {
94          path = null;
95          String [] images_data = { MediaStore.Images.Media.DATA };
96          Cursor cursor = context.getContentResolver().query(contentUri, images_data, null, null, null);
97          if(cursor.moveToFirst()) {
98            int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
99            path = cursor.getString(column_index);
100         }
101         cursor.close();
102         return path;
103     }
```



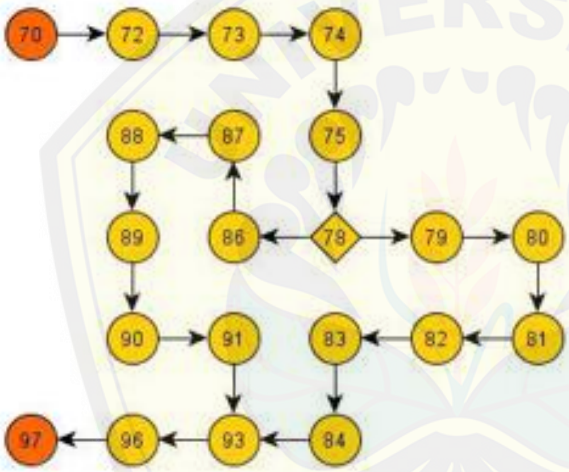
CC = Edge – Node + 2

CC = 11 – 11 + 2

CC = 2

A27. *Scale Image*

```
70@    public void takeImageResult(Context context, ImageView setImageView) {
71
72         path_image    = c_SettingData_Menu.shared_path_image;
73         bm_original   = decodeFile(path_image, 200, 300);
74         width_original  = bm_original.getWidth();
75         height_original = bm_original.getHeight();
76
77             // Scale image
78             if (width_original > height_original) {
79                 scale_width_land  = ((float) new_width_land) / width_original;
80                 scale_height_land = ((float) new_height_land) / height_original;
81                 matrix = new Matrix();
82                 matrix.postScale(scale_width_land, scale_height_land);
83                 matrix.postRotate(90);
84             }
85             else {
86                 scale_width_potrait = ((float) new_width_potrait) / width_original;
87                 scale_height_potrait = ((float) new_height_potrait) / height_original;
88                 matrix = new Matrix();
89                 matrix.postScale(scale_width_potrait, scale_height_potrait);
90                 matrix.postRotate(0);
91             }
92
93          bm_resize = Bitmap.createBitmap(bm_original, 0, 0, width_original, height_original, matrix, false);
94
95          // Result after scale
96          setImageView.setImageBitmap(bm_resize);
97     }
```



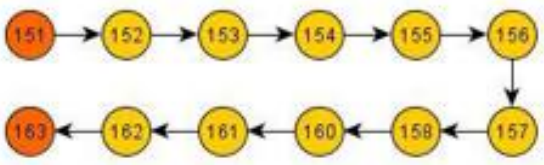
CC = Edge – Node + 2

CC = 21 – 21 + 2

CC = 2

A28. *Reset Data* Pada *Controller*

```
151@   public void resetConfirmYesActionTask(Context context, ListView lv_isi, Button btn_reset_action, Button btn_save_action){
152        db = new BMI_Database(context);
153        db.updateTrainingResult(1, 0.367, 0.352, 0.279, 1);
154        db.updateTrainingResult(2, 0.355, 0.37, 0.273, 2);
155        db.updateTrainingResult(3, 0.334, 0.363, 0.301, 3);
156        db.resetTrainingData();
157        resetVariabel(null, null, null, null, null);
158        loadTrainingResult(context, lv_isi, btn_reset_action, btn_save_action, null, null, null, null, null);
159
160        toast_message = Toast.makeText(context, "Reset Data Selesai !", Toast.LENGTH_SHORT);
161        toast_message.setGravity(Gravity.CENTER_VERTICAL, 0, 50);
162        toast_message.show();
163    }
```



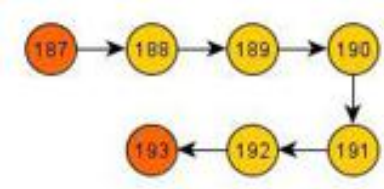
CC = Edge – Node + 2

CC = 11 – 12 + 2

CC = 1

A29. Mengurangi *Learning Rate*

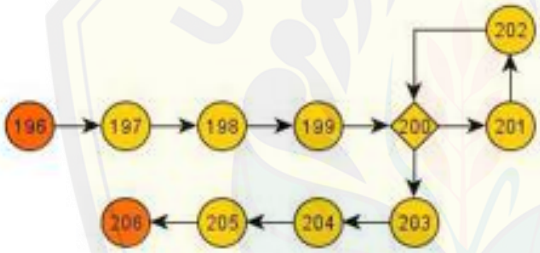
```
187  private double LearningRateDecay(double currentlearningrate) {
188      double result;
189      double reduceLearningRate = 0;
190      reduceLearningRate = currentlearningrate - (0.1 * currentlearningrate);
191      result = Math.floor(reduceLearningRate * 1000) / 1000;
192      return result;
193  }
```



CC = Edge – Node + 2
CC = 6 – 7 + 2
CC = 1

A30. Menghitung *Euclidean Distance* Pada Proses *Training*

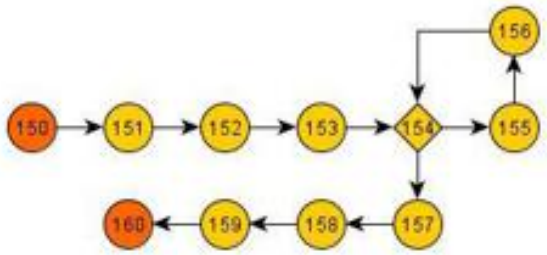
```
196  private double ComputeEuclideanDistance(double[] WeightVector,  double[] InputVector) {
197      double result;
198      double distance = 0;
199      double akar;
200      for (int j = 0; j < inputDataTraining; j++) {
201          distance += Math.pow((InputVector[j] - WeightVector[j]), 2);
202      }
203      akar = Math.sqrt(distance);
204      result = Math.floor(akar * 1000) / 1000;
205      return result;
206  }
```



CC = Edge – Node + 2
CC = 11 – 11 + 2
CC = 2

A31. Menghitung *Euclidean Distance* Pada Proses *Testing*

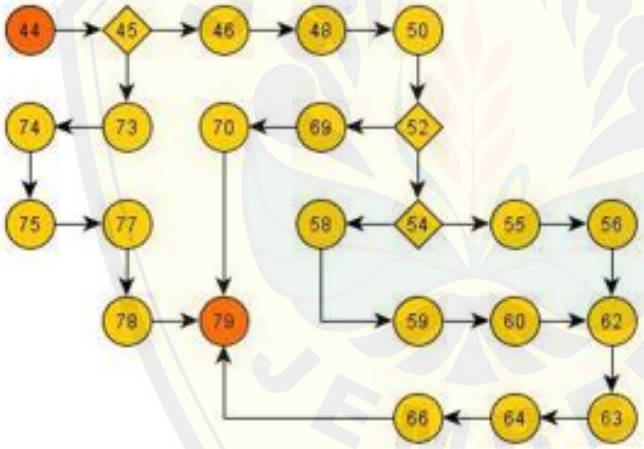
```
150  private double ComputeEuclideanDistance(double[] WeightVector,  double[] InputVector) {
151      double result;
152      double distance = 0;
153      double akar;
154      for (int j = 0; j < inputDataTesting; j++) {
155          distance += Math.pow((InputVector[j] - WeightVector[j]), 2);
156      }
157      akar = Math.sqrt(distance);
158      result = Math.floor(akar * 1000) / 1000;
159      return result;
160  }
```



CC = Edge – Node + 2
CC = 11 – 11 + 2
CC = 2

A32. Memilih Media *Camera* atau *SD Card*

```
public void dialogChooseItem(Context context, DialogInterface dialog, int itemChoose) {
    if (itemChoose == 0) {
        intent_action = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Membuat direktori folder
        dirFolder = new File(Environment.getExternalStorageDirectory() + "/BMI-Capture/");
        // Membuat nama file
        image_capture_name = "Image_" + String.valueOf(System.currentTimeMillis() + ".jpg");

        try {

            if (!dirFolder.exists()) {
                dirFolder.mkdirs();
            }
            else {
                file = new File(dirFolder, image_capture_name);
                file.createNewFile();
            }

            url_image = Uri.fromFile(file);
            intent_action.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, url_image);
            intent_action.putExtra("return-data", true);
            //tampilkan result
            ((Activity) context).startActivityForResult(intent_action, CAMERA);
        }
        catch (Exception e) {
            e.printStackTrace();
        }

    } else if (itemChoose == 1) {
        intent_action = new Intent();
        intent_action.setType("image/*");
        intent_action.setAction(Intent.ACTION_GET_CONTENT);
        //tampilkan result
        ((Activity) context).startActivityForResult(Intent.createChooser(intent_action, "Pilih Aplikasi"), FILE);
    }
}
```



CC = Edge – Node + 2

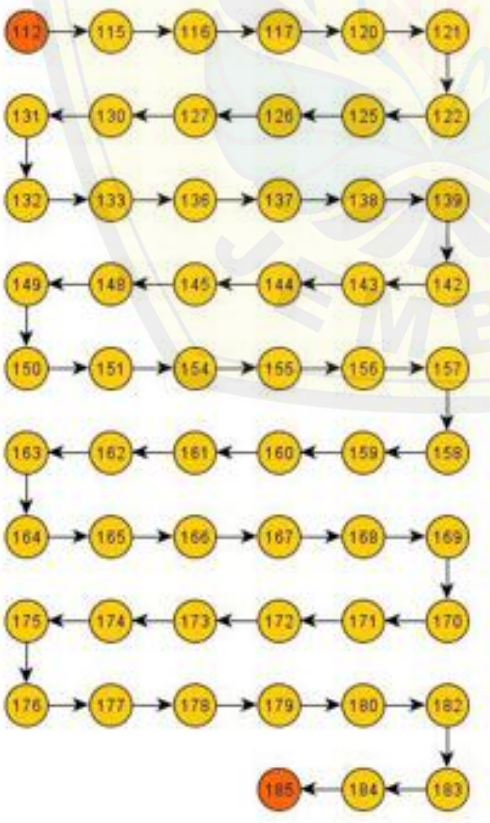
CC = 26 – 24 + 2

CC = 4

A33. Menampilkan Histogram RGB *Image*

```
private void histogramRGBImage() {

    // Input yang divisualisasikan ke dalam chart
    red_series   = isi_red_chart;
    green_series = isi_green_chart;
    blue_series  = isi_blue_chart;

    // Create XYZ.
    rc_series = new XYSeries("Red Color");
    gc_series = new XYSeries("Green Color");
    bc_series = new XYSeries("Blue Color");

    // Adding data to Chart
    rc_series.add(0, red_series);
    gc_series.add(0, green_series);
    bc_series.add(0, blue_series);

    // Create a Dataset
    dataRGB = new XYMultipleSeriesDataset();
    dataRGB.addSeries(rc_series);
    dataRGB.addSeries(gc_series);
    dataRGB.addSeries(bc_series);

    // Create XYSeriesRenderer to customize XSeries
    Xrenderer = new XYSeriesRenderer();
    Xrenderer.setColor(Color.RED);
    Xrenderer.setPointStyle(PointStyle.DIAMOND);
    Xrenderer.setLineWidth(2);
```

```
140
141         // Create XYSeriesRenderer to customize YSeries
142         Yrenderer = new XYSeriesRenderer();
143         Yrenderer.setColor(Color.GREEN);
144         Yrenderer.setPointStyle(PointStyle.DIAMOND);
145         Yrenderer.setLineWidth(2);
146
147         // Create XYSeriesRenderer to customize ZSeries
148         Zrenderer = new XYSeriesRenderer();
149         Zrenderer.setColor(Color.BLUE);
150         Zrenderer.setPointStyle(PointStyle.DIAMOND);
151         Zrenderer.setLineWidth(2);
152
153         // Create XYMultipleSeriesRenderer to customize the whole chart
154         mRenderer = new XYMultipleSeriesRenderer();
155         mRenderer.setOrientation(XYMultipleSeriesRenderer.Orientation.HORIZONTAL);
156         mRenderer.setXTitle("Elements Color Of Images");
157         mRenderer.setYTitle("Average RGB Values");
158         mRenderer.setZoomButtonsVisible(false);
159         mRenderer.setZoomEnabled(false);
160         mRenderer.setPanEnabled(true);
161         mRenderer.setPanLimits(new double[] { 0, 10, 0, 256 });
162         mRenderer.setShowGrid(true);
163         mRenderer.setXLabels(0);
164         mRenderer.setYLabels(10);
165         mRenderer.setXAxisMin(-0.5);
166         mRenderer.setXAxisMax(0.8);
167         mRenderer.setYAxisMin(0);
168         mRenderer.setBarWidth(40);
169         mRenderer.setApplyBackgroundColor(true);
170         mRenderer.setBackgroundColor(Color.WHITE);
171         mRenderer.setMarginsColor(Color.WHITE);
172         mRenderer.setAxesColor(Color.BLACK);
173         mRenderer.setLabelsColor(Color.BLACK);
174         mRenderer.setXLabelsColor(Color.BLACK);
175         mRenderer.setYLabelsColor(0, Color.BLACK);
176         mRenderer.setXLabelsAlign(Align.CENTER);
177         mRenderer.setYLabelsAlign(Align.CENTER);
178         mRenderer.addSeriesRenderer(Xrenderer);
179         mRenderer.addSeriesRenderer(Yrenderer);
180         mRenderer.addSeriesRenderer(Zrenderer);
181
182         vChart = (GraphicalView) ChartFactory.getBarChartView(v_HistogramRGB_Image.this, dataRGB, mRenderer, null);
183         vChart.setBackgroundColor(Color.WHITE);
184         chart_container.addView(vChart);
185     }
```



CC = Edge – Node + 2

CC = 56 – 57 + 2

CC = 1

A34. *Training Data* Citra Digital Buah Pisang

```
329  public void btnProcessActionTask(Context context, Spinner spinner_maxEpoch, EditText input_learningRate, EditText input_fps){
330
331      if (!spinner_maxEpoch.getSelectedItem().equals("--Pilih MaxEpoch--")) {
332
333          db = new BMI_Database(context);
334
335          id_tdCW1 = Integer.parseInt(shared_id_tdCW1);
336          id_tdCW2 = Integer.parseInt(shared_id_tdCW2);
337          id_tdCW3 = Integer.parseInt(shared_id_tdCW3);
338
339          learningRate = Double.parseDouble(input_learningRate.getText().toString()); // nilai alpha
340          maxEpoch     = Integer.parseInt(spinner_maxEpoch.getSelectedItem().toString()); // nilai iterasi
341          fps          = Double.parseDouble(input_fps.getText().toString()); // nilai min alpha
342
343          int jmldataBobot     = db.getCountTrainingDataCW(id_tdCW1, id_tdCW2, id_tdCW3);
344          int jmldataTraining  = db.getCountTrainingDataOtherCW(id_tdCW1, id_tdCW2, id_tdCW3);
345          ArrayList<Atribut_Model_TrainingData> tdCW      = db.getTrainingDataCW(id_tdCW1, id_tdCW2, id_tdCW3);
346          ArrayList<Atribut_Model_TrainingData> tdOtherCW = db.getTrainingDataOtherCW(id_tdCW1, id_tdCW2, id_tdCW3);
347
348          tt = new TrainingTask();
349          tt.Task(context, learningRate, maxEpoch, fps, jmldataBobot, jmldataTraining, tdCW, tdOtherCW);
350          tt.execute();
351      }
352      else {
353          toast_message = Toast.makeText(context,"Pilih Max Epoch Dulu !", Toast.LENGTH_LONG);
354          toast_message.setGravity(Gravity.CENTER_VERTICAL, 0, 50);
355          toast_message.show();
356      }
357  }
```

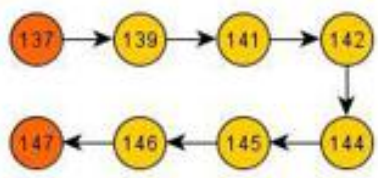
CC = Edge – Node + 2

CC = 23 – 23 + 2

CC = 2

A35. Mengidentifikasi Citra Digital Buah Pisang

```
137@    public void btnIdentifyActionTask(Context context){
138
139         db = new BMI_Database(context);
140
141         jmldataBobot = db.getCountTrainingResult();
142         dataTR = db.getAllTraimingResult();
143
144         it = new IdentifyTask();
145         it.Task(context, bm_resize, width_resize, height_resize, jmldataBobot, dataTR);
146         it.execute();
147     }
```



CC = Edge – Node + 2

CC = 7 – 8 + 2

CC = 1