PENGEMBANGAN SISTEM APLIKASI ANALISIS KONSENTRASI GLUKOSA BERDASARKAN KONTEN WARNA DENGAN MENGGUNAKAN METODE *K-NEAREST NEIGHBOR*

SKRIPSI

Oleh

R. Jefta Shaktika Pamungkas Nathanael
NIM 112410101007

PROGRAM STUDI SISTEM INFORMASI
UNIVERSITAS JEMBER
2016

PENGEMBANGAN SISTEM APLIKASI ANALISIS KONSENTRASI GLUKOSA BERDASARKAN KONTEN WARNA DENGAN MENGGUNAKAN METODE *K-NEAREST NEIGHBOR*

SKRIPSI

digunakan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Sistem Informasi (S1)
dan mencapai gelar Sarjana Komputer

Oleh

R. Jefta Shaktika Pamungkas Nathanael
NIM 112410101007

PROGRAM STUDI SISTEM INFORMASI
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

MOTTO

PERNYATAAN

PEMBIMBINGAN

PENGESAHAN

ABSTRAK

RINGKASAN

PRAKATA

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

Bab ini merupakan bab awal dari laporan tugas akhir. Pada bab ini akan dibahas tentang latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah, dan sistematika penulisan.

1.1 Latar Belakang

Karbohidrat merupakan sumber energi utama bagi tubuh manusia. Pemecahan karbohidrat menghasilkan glukosa. Glukosa adalah bahan bakar universal bagi sel manusia dan merupakan sumber karbon untuk sintesis sebagian besar senyawa lainnya (Marks, Marks, & Smith, 2000). Glukosa terlibat dalam produksi ATP (*Adenosina Trifosfat*), pembawa energi sel. Semua jenis sel manusia menggunakan glukosa untuk memperoleh energi. Gula lain dalam makanan (fruktosa dan galaktosa) dapat diubah menjadi glukosa untuk memenuhi kebutuhan tubuh sehari-hari. Glukosa diserap melalui saluran penceranaan ke dalam peredaran darah. Sebagian glukosa menjadi bahan bakar sel otak, sedangkan lainnya menuju hati dan otot. Glukosa banyak terdapat pada buah-buahan. Glukosa memiliki rasa yang manis tetapi tidak semanis gula tebu.

Glukosa sangat penting dalam kegiatan metabolisme dalam tubuh. Pengukuran konsentrasi glukosa pada buah atau pada bahan makanan lain, dapat diuji melalui laboraturium. Metode yang sering digunakan untuk mengukur konsentrasi larutan glukosa adalah kolorimetri dan spektrofotometri. Kolorimetri sendiri adalah variasi warna suatu sistem dengan berubahnya konsentrasi suatu komponen, membentuk dasar yang lazim (Bassett, Denney, Jeffery, & Mendham, 1994). Dengan metode ini larutan glukosa dicampurkan dengan reagensia yang tepat agar larutan glukosa berubah menjadi berwarna dengan rentang warna inframerah sampai dengan ultraviolet, kemudian diukur dengan larutan standar yang telah diketahui konsentrasinya (kolorimeter). Kekurangan dari metode ini adalah besar sesatan yang disebabkan

karakteristik pribadi tiap pengamat sehingga memberikan presepsi berbeda. Sedangkan dalam analisis spektrofotometri menggunakan suatu sumber radiasi yang menjorok ke dalam daerah ultraviolet spektrum tersebut (Bassett et al., 1994). Dari spektrum tersebut dipilih panjang gelombang tertentu dengan lebar pita kurang dari 1 µm. Dengan metode ini larutan glukosa yang sudah berwarna dimasukkan ke dalam alat yang disebut spektrofotometer yang akan mengukur serapan sinar monokromatis larutan glukosa pada rentang warna yang lebih spesifik. Kekurangan dari metode ini adalah harga alat yang relatif mahal berkisar mulai dari puluhan juta hingga ratusan juta rupiah karena menggunakan instrument yang lebih rumit, belum lagi biaya perawatan alat.

Pada skripsi ini, konsep pengolahan citra digital dapat memberikan solusi dari kekurangan kedua metode tersebut, dengan memanfaatkan larutan berwarna dengan harga yang relatif jauh lebih murah dibandingkan dengan alat spektrofotometer. Pengolahan citra digital dapat menangkap warna dengan bantuan kamera yang akan memberikan persepsi warna yang selalu sama dan lebih akurat. Warna-warna yang terbentuk merupakan hasil kombinasi cahaya dengan panjang gelombang yang berbeda (Kadir & Susanto, 2012). Sistem yang akan dibangun menggunakan metode *K-Nearest Neighbor*.

Penerapan metode *K-Nearest Neighbor* pada pengembangan sistem aplikasi ini bertujuan untuk mendapatkan analisis konsentrasi larutan glukosa. Hasil dari sampel uji yang diklasifikasikan berdasarkan mayoritas dari kategori *K-Nearest Neighbor*. Hanya berdasarkan memori, algoritma ini tidak menggunakan model apapun pada proses klasifikasinya (Krisandi, Prihandono, & Helmi, 2013). Sama halnya dengan metode SVM (*Support Vector Machine*) untuk mengklasifikasikan kelas, metode SVM sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas pada *input space* (Naufal, Wahono, & Syukur, 2015), sehingga untuk menentukan klasifikasi dengan lebih dari dua kelas harus menambahkan dengan pendekatan yang lain.

1.2 Rumusan Masalah

Berdasarkan permasalahan yang telah teruai, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1) Bagaimana mengimplementasikan pengolahan citra digital, pada pengembangan sistem aplikasi analisis konsentrasi glukosa?
2) Bagaimana mengimplementasikan *data mining K-Nearest Neighbor*, pada pengembangan sistem aplikasi analisis konsentrasi glukosa menggunakan pengolahan citra digital?

1.3 Tujuan dan Manfaat

Tujuan pada penelitian diuraikan menjadi dua, yaitu tujuan khusus dan tujuan umum. Sedangkan pada bagian manfaat berisi tentang manfaat yang diperoleh pada penelitian ini, baik bagi akademik, peneliti maupun objek pada penelitian ini.

1.3.1 Tujuan

Tujuan dari penelitian ini adalah:

a. Tujuan Umum

   Membangun "Pengembangan Sistem Aplikasi Analisis Konsentrasi Glukosa Berdasarkan Konten Warna dengan Menggunakan Metode *K-Nearest Neighbor*", agar dapat membantu dalam mengklasifikasikan konsentrasi glukosa dengan memberikan presepsi warna yang stabil dan menghemat biaya.

b. Tujuan Khusus

   1) Merancang dan membangun pengembangan sistem aplikasi analisis konsentrasi larutan glukosa.
   2) Mengimplementasikan pengolahan citra digital pada pengembangan sistem aplikasi analisis konsentrasi glukosa.

3) Mengimplementasikan *data mining K-Nearest Neighbor* pada pengembangan sistem aplikasi analisis konsentrasi glukosa menggunakan pengolahan citra digital.

1.3.2 Manfaat

Manfaat dari penelitian ini adalah:

a. Manfaat Teoritis

Mengembangkan keilmuan dan memperkaya konsep atau teori yang menopang perkembangan ilmu pengetahuan khususnya pada analisis pengklasifikasian konsentrasi larutan glukosa menggunakan pengolahan citra digital dan *data mining classification K-Nearest Neighbor*.

b. Manfaat Praktis

1) Bagi penulis

Untuk penerapan ilmu yang diperoleh selama perkuliahan untuk menciptakan sistem informasi yang berkualitas, sesuai dengan kebutuhan, dan berpedoman standar.

2) Bagi objek penelitian

Menyediakan aplikasi analisis konsentrasi glukosa yang menggunakan pengolahan citra dan *data mining* untuk mendukung pembelajaran dan penelitian, sehingga mampu mengurangi pengaruh perbedaan karakteristik pengamat dalam menafsirkan warna dan menghemat biaya.

1.4 Batasan Masalah

Beberapa hal yang menjadi pembatas dalam penelitian agar penelitian tidak melebar dari konteks utama adalah:

1) Larutan yang digunakan adalah larutan glukosa 1 ml yang telah direaksikan oleh pereagen menggunakan metode Somogyi-Nelson sebagai senyawa kontrol pada aplikasi ini dan menghasilkan larutan glukosa berwarna.
2) Jumlah larutan glukosa sebanyak 11 larutan dengan jarak 10mg/ml tiap larutan.
3) Konsetrasi larutan glukosa yang digunakan sebagai basis data dalam aplikasi berkisar 0mg/ml – 100mg/ml.
4) Menggunakan kamera Logitech C920 sebagai alat masukan untuk menggambil citra larutan glukosa
5) Menggunakan kertas karton yang dilapisi kertas HVS untuk bahan pembuatan *mini studio* berukuran $30cm^3$.
6) Sumber cahaya yang digunakan saat pengambilan gambar menggunakan pencahayaan dari lampu led berdaya 3watt di dalam *mini studio*.
7) Menggunakan 3 ruang warna RGB (*Red*, *Green*, dan *Blue*), HSV (*Hue*, *Sturation*, dan *Value*), dan *grayscale* dalam mengolah citra digital.
8) Menggunakan *data mining clasification K-Nearest Neighbor* dengan k=3 untuk mengklasifikasi tingkat konsentrasi glukosa.

1.5 Sistematika Penulisan

Adapun sistematika penulisan skripsi ini adalah sebagai berikut:

1) Pendahuluan

   Bab pertama berisi tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan.

2) Tinjauan Pustaka

   Bab kedua memaparkan tinjauan terhadap hasil-hasil penelitian terdahulu berkaitan dengan rumusan masalah penelitian, pengolahan citra digital, ruang warna digital, spektrofotmeter, datamining, metode *K-Nearest Neighbor*, dan glukosa yang berkaitan dengan rumusan masalah dalam penelitian.

3) Metodologi Penelitian

Bab ketiga berisi tentang jenis penelitian, tempat dan waktu penelitian, alat penelitian, tahap pengumpulan data, tahap analisis, tahap pengembangan sistem, dan gambaran sistem.

4) Pengembangan Sistem

Bab keempat berisi tentang pengembangan sistem meliputi analisis kebutuhan, deskripsi umum sistem, desain sistem, implementasi sistem, dan pengujian sistem.

5) Hasil dan Pembahasan

Bab kelima berisi tentang pembahasan pengembangan sistem aplikasi analisis konsentrasi glukosa, mulai dari pengumpulan data hingga pengujian analisis data kedalam metode klasifikasi, dan hasil implementasi pengembangan sistem.

6) Penutup

Bab keenam berisi tentang kesimpulan penelitian dan saran penelitian untuk digunakan sebagai acuan penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Bab ini dipaparkan teori - teori serta pustaka yang dipakai pada waktu penelitian. Teori - teori ini diambil dari buku literatur dan jurnal. Berikut merupakan teori - teori yang digunakan dan dibahas dalam penelitian :

2.1 Penelitian Terdahulu

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

1) Jurnal penelitian yang berjudul "Algoritma *K-Nearest Neighbor Classification* Sebagai Sistem Prediksi Predikat Prestasi Mahasiswa" dilakukan oleh Mustakim dan Giantika Oktaviani F, alumnus Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sultan Syarif Kasim Riau. Penetilian ini bertujuan untuk mengklasifikasikan prediksi prestasi mahasiswa. Dengan parameter jenis kelamin, jenis tinggal, umur, jumlah satuan kredit semester, dan jumlah nilai mutu dimana hasil atribut ini didapatkan berdasarkan penelitian terlebih dahulu. Sehingga *output* yang dihasilkan merupakan prediksi prestasi yang akan dicapai pada semester mendatang dan menjadikan *warning* untuk mahasiswa meningkatkan prestasinya dengan lebih baik.
2) Jurnal penelitian yang berjudul "Identifikasi Varietas Berdasarkan Warna dan Tekstur Permukaan Beras Menggunakan Pengolahan Citra Digital dan Jaringan Syaraf Tiruan" dilakukan oleh Adnan, Suhartini dan Bram Kusbiantoro, peneliti di Balai Pengkajian Teknologi Pertanian Papua. Penelitian ini bertujuan untuk mengidentifikasi varietas beras. Kombinasi analisis warna dan tekstur berdasarkan metode pengolahan citra digital. Pengambilan citra digital dilakukan di dalam kotak dengan pencayahan lampu. Penelitian ini sendiri menggunakan kombinasi warna *Red*, *Green* dan *Blue* dan berhasil mengindentifikasi lima varietas beras.

Dari kedua penelitian terdahulu dapat memberikan masukan maupun terapan metode yang dapat digunakan untuk penelitian yang akan datang. Menggunakan metode *data mining K-Nearest Neighbor* yang dapat memberikan klasifikasi yang sebuah objek dengan membandingkannya dengan objek-objek yang sudah jelas dan telah diketahui klasifikasinya. Dengan metode ini membuat rangkaian larutan glukosa dapat dianalisis lebih spesifik mengingat kemampuan spektrofotometer yang dapat memberikan nilai serapan pada konsentrasi larutan hingga tiga angka dibelakang koma. Dipadu dengan ruang warna RGB, HSV atau *grayscale* yang akan menjadikan penelitian ini lebih objektif dalam menganalisis larutan glukosa.

2.2 Pengolahan Citra Digital

Citra adalah sebuah gambar yang terdapat pada bidang dwimatra (dua dimensi). Dilihat dari sudut pandang matematis citra merupakan fungsi terus menerus dari intensitas cahaya pada bidang dwimatra. Sedangkan citra bergerak merupakan sekelompok citra diam yang ditampilkan secara berurutan sehingga memberi efek bergerak pada mata kita. Setiap citra yang terbentuk disebut frame.

Terkadang citra juga bisa mengalami penurunan mutu, misalnya kabur (*blurring*), warna yang terlalu kontras, mengandung cacat atau derau (*noise*), kurang tajam, dan sebagainya. Citra perlu dimanipulasi menjadi citra dengan kualitas lebih baik, agar citra yang mengalami gangguan mudah diinterpretasikan (baik oleh manusia maupun mesin). Bidang studi yang menyangkut hal ini adalah pengolahan citra (Munir, 2004).

Operasi pada pengolahan citra memiliki banyak ragam, namun secara umum dapat diklasifikasikan sebagai berikut :

a. Perbaikan Kualitas Citra (*Image Enhancement*)

   Operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra.

b. Pemugaran Citra (*Image Restoration*)

Operasi ini hampir sama dengan perbaikan citra, bedanya pemugaran citra penyebab degradasi gambar diketahui.

c. Pemampatan Citra (*Image Compression*)

   Operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit.

d. Segmentasi Citra (*Image Segmentation*)

   Operasi ini bertujuan untuk memecah suatu citra kedalam beberapa segmen dengan suatu kriteria tertentu.

e. Pengorakan Citra (*Image Analysis*)

   Operasi ini bertujuan untuk menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya.

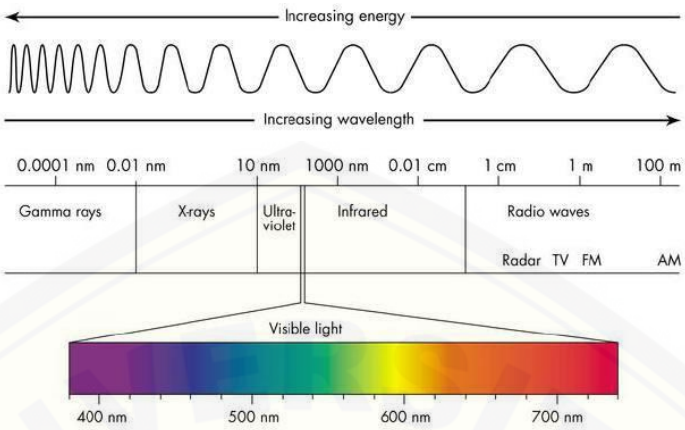
f. Rekonstruksi Citra (*Image Reconstruction*)

   Operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi.

2.3 Ruang Warna Digital

Di dalam suatu citra terdapat banyak informasi pada warna, dan informasi tersebut juga dapat digunakan untuk menyederhanakan analisis citra, seperti identifikasi objek atau ekstraksi citra. Terdapat tiga macam kuantisasi yang dapat digunakan untuk menggambarkan warna, yaitu:

a. *Hue*, ditentukan oleh dominan panjang gelombang sepert pada Gambar 2.1. Warna sinar tampak (visible spectrum) dengan panjang gelombang berkisar dari 400nm (biru) - 700nm (merah) (Munir, 2004).



Gambar 2.1 Kedudukan gelombang cahaya tampak dan panjang gelombangnya

b. *Saturation*, ditentukan oleh tingkat kemurnian, dan tergantung pada jumlah sinar putih yang tercampur dengan *hue*. Suatu warna *hue* murni adalah secara penuh tersaturasi, yaitu tidak ada sinar putih yang tercampur. *Hue* dan *saturation* digabungkan untuk menentukan kromatik suatu warna.
c. Sinar *Achromatic* tidak memiliki warna, tetapi hanya ditentukan oleh atribut intensitas. Tingkat keabuan (*graylevel*) adalah ukuran intensitas yang ditentukan oleh energy, sehingga merupakan suatu kuantitas fisik. Dalam hal ini *brightness* atau *luminance* ditentukan oleh persepsi warna.

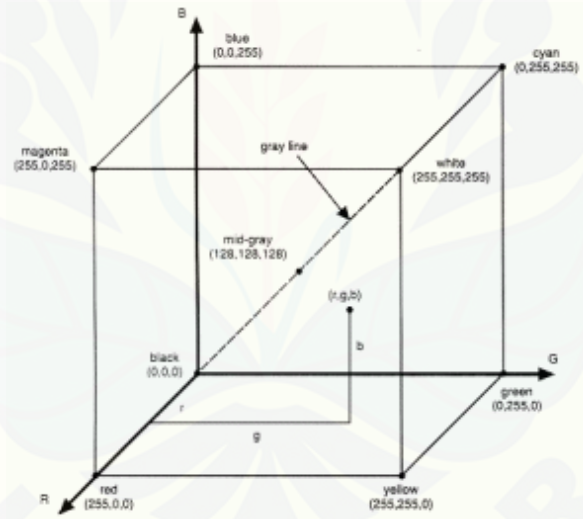
Warna secara utuh bergantung pada sifat pantulan (*reflectance*) suatu objek. Warna yang dilihat merupakan yang dipantulkan, sedangkan yang lainnya diserap. Sehingga sumber sinar perlu diperhitungkan, begitu pula sifat alami sistem visual manusia ketika menangkap suatu warna.

Model warna merupakan cara yang standar untuk menspesifikasikan suatu warna tertentu, dengan mendefinisikannya ke dalam suatu sistem koordinat 3D, dan suatu ruang bagian yang mengandung semua warna yang dapat dibentuk ke dalam suatu model tertentu. Suatu warna yang dapat dispesifikasikan menggunakan suatu model akan berhubungan ke suatu titik tunggal dalam suatu ruang bagian yang

didefinisikannya. Masing-masing warna diarahkan ke salah satu standar hardware tertentu (RGB, CMY, YIQ), atau aplikasi pengolahan citra (HSV).

2.3.1 Ruang Warna RGB

Ruang warna RGB biasa diterapkan pada monitor CRT dan kebanyakan sitem grafika komputer. Ruang warna ini menggunakan tiga komponen dasar yaitu merah (R), hijau (G), biru (B). Setiap piksel dibentuk oleh ketiga komponen tersebut. Model RGB seperti pada Gambar 2.2, biasa disajikan dalam bentuk kubus tiga dimensi, dengan warna merah, hijau, dan biru berada pada pojok sumbu (Kadir & Susanto, 2012).

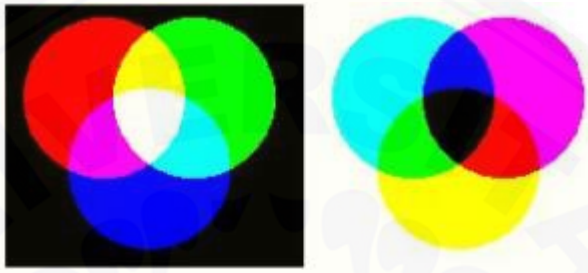


Gambar 2.2 Model ruang warna RGB

Spektrum *grayscale* (tingkat keabuan) merupakan warna yang dibentuk dari gabungan tiga warna utama dengan jumlah yang sama, berada pada garis yang menghubungkan titik hitam dan putih.

Warna direpresentasikan dalam suatu sinar tambahan untuk membentuk warna baru, dan berhubungan untuk membentuk sinar campuran. Gambar 2.3 ini menunjukkan campuran dengan menambahkan warna utama merah, hijau, dan biru

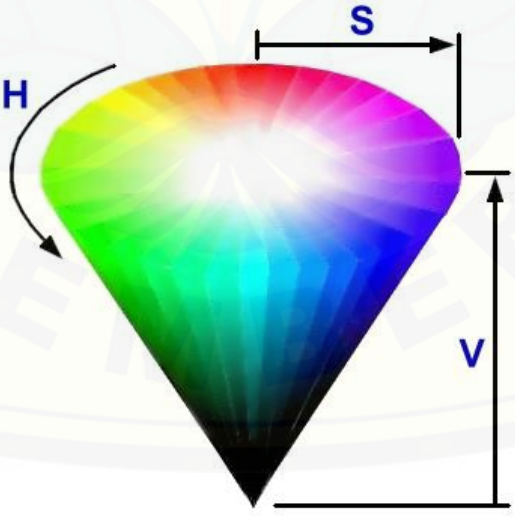
untuk membentuk warna sekunder kuning(merah+hijau), cyan(biru+hijau), magenta(merah+biru) dan putih (merah+hijau+biru). Warna subtraktif sendiri merupakan warna yang berasal dari bahan atau sering disebut pigmen. Warna-warna dasar RGB merupakan warna pokok, dan warna-warna *cyan*, *magenta*, dan *yellow* merupakan model warna CMY.



Gambar 2.3 (a) Model warna aditif, dan (b) Model warna subtraktif

2.3.2 Ruang Warna HSV

Ruang warna HSV merupakan definisi warna dari terminology *Hue*, *Saturation*, dan *Value*. Gambar 2.4 merupakan sistem koordinat warna pada ruang warna HSV.



Gambar 2.4 Sistem koordinat ruang warna HSV

HSV merupakan contoh ruang warna seperti yang dilihat oleh mata manusia (Kadir & Susanto, 2012). HSV memiliki 3 karakteristik pokok yaitu:

a. *Hue*, menyatakan warna sebenarnya, seperti merah, ungu, dan kuning dan digunakan untuk menentukan kemerahan (*redness*), kebiruan (*blueness*), dll
b. *Saturation*, disebut juga chroma, adalah kemurnian warna atau kekuatan warna
c. *Value*, bernilai 0-100% merupakan kecerahan dari warna. 0% maka warna berwarna hitam, dan semakin besar nilainya semakin menunjukkan variasi baru dari warna tersebut

Ruang warna RGB dapat dikonversi ke dalam ruang warna HSV dengan menggunakan rumus sebagai berikut.

$$\boldsymbol{r = \frac{R}{(255)}, g = \frac{G}{(255)}, b = \frac{B}{(255)}} \quad (2.1)$$

R adalah nilai dari *Red*, dan G adalah nilai *Green*, dan B adalah nilai *Blue.*

$$\boldsymbol{Cmax = max(r, g, b)} \quad (2.2)$$

Persamaan 2.2 untuk mendapatkan nilai maksimal dari tiga komponen r, g, dan b.

$$\boldsymbol{Cmin = min(r, g, b)} \quad (2.3)$$

Persamaan 2.3 untuk mendapatkan nilai minimal dari tiga komponen r, g, dan b.

$$\Delta = \boldsymbol{Cmax - Cmin} \quad (2.4)$$

Δ Merupakan hasil dari selisih Cmax dan Cmin

$$\boldsymbol{V = Cmax} \quad (2.5)$$

$$\boldsymbol{S = \begin{cases} 0 & , jika\ V = 0 \\ \frac{\Delta}{V} & , V \neq 0 \end{cases}} \quad (2.6)$$

$$\boldsymbol{H = \begin{cases} 0 & , jika\ \Delta = 0 \\ 60 * \left[\frac{g-b}{\Delta} mod\ 6\right] & , jika\ V = r \\ 60 * \left[2 + \frac{b-r}{\Delta}\right] & , jika\ V = g \\ 60 * \left[4 + \frac{r-g}{\Delta}\right] & , jika\ V = b \end{cases}} \quad (2.7)$$

$$\boldsymbol{H = H + 360\ , jika\ H < 0} \quad (2.8)$$

Persamaan 2.5, 2.6, 2.7, dan 2.8 merupakan persamaan untuk mendapatkan nilai *Hue*, *Saturation*, dan *Value*.

2.3.3 Ruang Warna *Grayscale*

Warna abu-abu pada citra *grayscale* adalah warna R (*red*), G (*green*), B (*blue*) yang memiliki intensitas yang sama. Sehingga dalam *grayscale image* hanya membutuhkan nilai intensitas tunggal dibandingkan dengan citra berwarna membutuhkan tiga intensitas untuk tiap pikselnya. Tingkat keabuan atau Grayscale level dapat dilihat pada gambar 2.5.

Gambar 2.5 Tingkat keabuan *grayscale*

Ruang warna *grayscale* mempunyai nilai minimum dan nilai maksimum. Banyaknya nilai pada *grayscale* tergantung dari jumlah bit yang digunakan, dan biasanya menggunakan 8 bit. Skala keabuan menggunakan 8 bit, maka jumlah kemungkinan nilainya adalah 2^8 = 256, sehingga nilai minimumnya = 0 dan nilai maksimumnya = 255.

Persamaan yang digunakan untuk mengkonversikan citra berwarna menjadi citra skala keabuan adalah sebagai berikut (Basuki, Palandi, & Fatchurrrochman, 2005):

$$\boldsymbol{Gray} = \frac{(\boldsymbol{R+G+B})}{\boldsymbol{3}} \quad \text{..... (2.9)}$$

Dimana R adalah nilai dari *Red*, dan G adalah nilai *Green*, dan B adalah nilai *Blue* dibagi dengan 3 untuk mendapatkan nilai *grayscale* dengan menyetarakan nilai R, G, dan B.

2.4 Spektrofotometer

Spektrofotometer adalah alat yang digunakan untuk mengukur absorbansi (serapan cahaya) dengan cara melewatkan cahaya dengan panjang gelombang tertentu pada suatu objek kaca atau kuarsa yang disebut *kuvet*. Sebagian dari cahaya akan diserap dan sisanya akan dilewatkan. Nilai absorbansi dari cahaya yang dilewatkan akan sebanding dengan konsentrasi larutan di dalam *kuvet*. Alat ini terdiri dari spektrometer dan fotometer. Spektrometer menghasilkan sinar dari spektrum dengan panjang gelombang tertentu dan fotometer adalah alat pengukur intensitas cahaya yang ditransmisikan atau yang diabsorpsi (Day Jr. & Underwood, 1996). Spektrofotometer digunakan untuk mengukur nilai absorbansi larutan dari panjang gelombang.

Secara garis besar spektrofotometer terdiri dari 4 bagian penting yaitu :

a. Sumber Cahaya

   Sebagai sumber cahaya pada spektrofotometer, haruslah memiliki pancaran radiasi yang stabil dan intensitasnya tinggi. Sumber energi cahaya yang biasa untuk daerah tampak, ultraviolet dekat, dan inframerah dekat adalah sebuah lampu pijar dengan kawat rambut terbuat dari wolfram (tungsten). Lampu ini mirip dengan bola lampu pijar biasa, daerah panjang gelombang (l) adalah 350 – 2200 nanometer (nm).

b. Monokromator

   Monokromator adalah alat yang berfungsi untuk menguraikan cahaya polikromatis menjadi beberapa komponen panjang gelombang tertentu (monokromatis) yang bebeda (terdispersi).

c. Kuvet

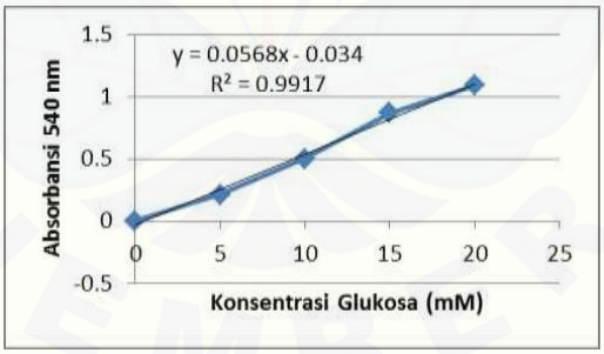
   Kuvet spektrofotometer adalah suatu alat yang digunakan sebagai tempat contoh atau cuplikan yang akan dianalisis. *Kuvet* biasanya terbuat dari kuarsa, *plexigalass*, kaca, plastik dengan bentuk tabung empat persegi panjang 1 x 1 cm dan tinggi 5 cm. Pada pengukuran di daerah UV dipakai *kuvet* kuarsa atau *plexiglass*, sedangkan *kuvet* dari kaca tidak dapat dipakai sebab kaca mengabsorbsi sinar UV. Semua macam *kuvet* dapat dipakai untuk pengukuran di daerah sinar tampak (visible).

d. Detektor

Peranan detektor penerima adalah memberikan respon terhadap cahaya pada berbagai panjang gelombang. Detektor akan mengubah cahaya menjadi sinyal listrik yang selanjutnya akan ditampilkan oleh penampil data dalam bentuk jarum penunjuk atau angka digital.

Sinar berasal dari dua lampu yang berbeda, yaitu lampu wolfram untuk sinar *visible* (sinar tampak = 38 – 780nm) dan lampu deuterium untuk sinar ultraviolet (180-380nm) pada video lampu yang besar. Gunakan panjang gelombang yang diinginkan/diperlukan. *Kuvet* ada dua karena alat yang dipakai tipe double beam, tempat menyimpan sampel dan yang satu lagi untuk blanko. Detektor atau pembaca cahaya yang diteruskan oleh sampel, disini terjadi pengubahan data sinar menjadi angka yang akan ditampilkan pada *reader*.

Yang harus dihindari adanya cahaya yang masuk ke dalam alat, biasanya pada saat menutup tempat *kuvet*, karena bila ada cahaya lain otomatis jumlah cahaya yang diukur menjadi bertambah. Hasil dari pengamatan menggunakan spektrofotometer bisa nya menghasilkan grafik hubungan linier seperti pada Gambar 2.6.

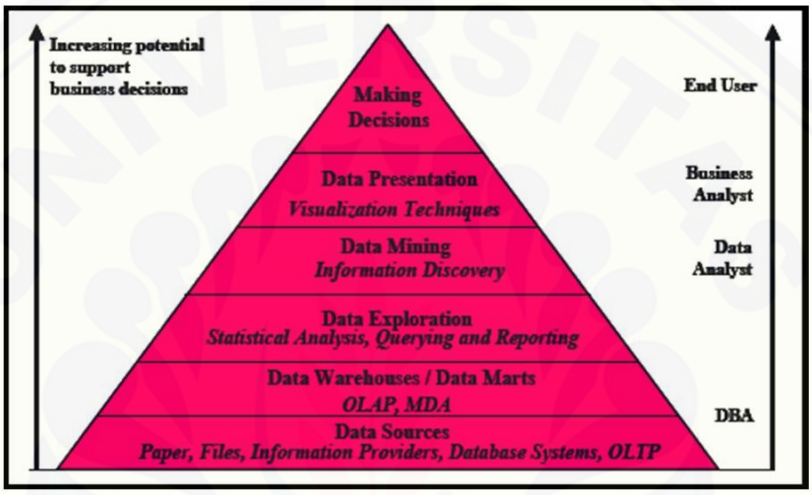


Gambar 2.6 Kurva standar glukosa menggunakan spektrofotometer

2.5 *Data Mining*

Data mining adalah proses mempekerjakan satu atau lebih teknik pembelajaran komputer (*machine learning*) untuk menganalisis dan mengekstrasi pengetahuan

(*knowledge*) secara otomatis (Hermawati, 2013). Data mining merupakan proses iterative dan interaktif untuk menemukan model atau pola baru yang sempurna, bermanfaat dan dapat dimengerti dalam suatu *database*. *Data mining* berisi pencarian pola atau trend di dalam *database* yang akan digunakan untuk membantu pengambilan keputusan diwaktu yang akan datang. Pada Gambar 2.7 merupakan gambaran umum *data mining* dan teknologi *database*.



Gambar 2.7 *Data mining* dan teknologi *database* lainnya

Dari Gambar 2.7 terlihat bahwa *data mining* digunakan untuk melakukan *information discovery* yang informasinya lebih dulu ditunjukkan untuk seorang *Data Analyst* dan *Business Analyst*.

Beberapa teknik dan sifat dalam mengolah data di dalam *data mining* adalah sebagai berikut:

a. *Classification* (Klasifikasi)

   Menentukan sebuah *record* data baru kesalah satu dari beberapa kategori yang telah didefinisikan sebelumnya.

b. *Clustering* (Klustering)

   Membagi data-set menjadi beberapa sub-set atau kelompok sedemikian rupa sehingga elemen-elemen dari suatu kelompok tertentu memiliki *set property* yang

dishare bersama, dengan tingkat similaritas yang tinggi dalam suatu kelompok dan tingkat similaritas antar kelompok yang rendah.

c. *Regression* (Regresi)

   Memprediksi nilai dari suatu variabel kontinyu yang diberikan berdasarkan nilai dari variabel yang lain, dengan mengasumsikan sebuah model ketergantungan linier atau nonlinier

d. *Association Rule Discovery* (Kaidah Asosiasi)

   Mendeteksi kumpulan atribut-atribut yang muncul bersamaan dalam frekuensi yang sering, dan membentuk sejumlah kaidah dari kumpulan-kumpulan tersebut.

e. *Sequential Pattern Discovery* (Pencarian Pola Sekuensial)

   Mencari sejumlah event yang secara umum terjadi bersama-sama. Pola-pola sekuensial pertama, pada dasarnya dibentuk dengan cara mencari semua kemungkinan pola yang ada.

Himpunan data merupakan kumpulan dari objek dan atributnya. Atribut merupakan sifat atau karakteristik dari suatu objek. Data dalam *data mining* mengalami beberapa proses pengolahan. Sebelum diterapkan algoritma *data mining* terhadap sebuah data-set, perlu dilakukan pengolahan awal yang bertujuan untuk mendapatkan data-set yang dapat diolah dengan cepat dan menghasilkan kesimpulan yang tepat.

2.6 Metode *K-Nearest Neighbor*

K-Nearest Neighbor merupakan salah satu metode klasifikasi dalam *data mining*. Didasari pada pembelajaran dengan persamaan sampel yang diujikan dan diuraikan oleh n-dimensi atribut angka. Masing-masing sampel merepresentasikan titik dalam ruang n dimensi. Tujuan dari algoritma *K-Nearest Neighbor* adalah untuk mengklasifikasi objek baru berdasarkan atribut dan *training samples*. Dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada *K-Nearest Neighbor* (Krisandi et al., 2013).

Kedekatan jarak data berdasarkan aturan *Euclidean Distance*. Aturan pada *Euclidean Distance* dihitung antara dua buah titik, dan setiap atribut telah memiliki bobot nilainya. *K-Nearest Neighbor* adalah *instane-based* ataupun *lazy learnes* karena pada metode ini menyimpan semua sampel yang akan dihitung dan tidak membentuk sebuah klasifikasi sampai ada sampel baru yang tidak diketahui klasifikasinya. Ini sangat berbeda dengan metode *eager learning* seperti *decision tree* dan *backpropagation*, yang secara langsung membentuk model klasifikasi sebelum menerima sampel baru untuk diklasifikasikan.

K-Nearest Neighbor merupakan metode klasifikasi yang tangguh terhadap *training data* yang memiliki banyak *noise* dan metode ini memiliki tingkat akurasi yang sangat tinggi serta efektif apabila *training data* yang digunakan. *K-Nearest Neighbor* mencari jarak terdekat antara data yang akan dievaluasi dengan *k*-tetangga (*neighbor*) terdekatnya di dalam *training data*. Adapun algoritma dalam melakukan prediksi dengan menggunakan metode *K-Nearest Neighbor* adalah sebagai berikut (Mustakim & F, 2016):

a. Tentukan nilai *k* dimana nilai tersebut akan digunakan sebagai parameter pembatas jumlah titik sampel yang akan diuji.
b. Hitung jarak nilai atribut testing ke setiap atribut training menggunakan *Euclidean Distance*.

$$\boldsymbol{d_i = \sqrt{\sum_{i=1}^{p}(x_{2i} - x_{1i})^2}} \quad \text{.....(2.10)}$$

Dimana d = jarak, yang didapatkan jumlah dari selisih x_1 = sampel data, dan x_2 = data uji yang dipangkatkan dua bersadarkan iterasi i = variable data sampai dengan p = dimensi data

c. Kemudian urutkan hasil jarak berdasarkan dari nilai terendah.
d. Dimulai dari yang terdekat hingga urutan ke-*k*, mencari nilai mayoritas yang dinyatakan oleh titik sampel
e. Nilai mayoritas tersebut adalah merupakan hasil prediksi

2.7 Glukosa

Glukosa merupakan pusat dari semua metabolisme. Glukosa adalah bahan bakar universal bagi sel mausia dan merupakan sumber karbon untuk sintesis sebagian besar senyawa lainnya (Marks et al., 2000). Sel-sel dalam tubuh manusia membutuhkan glukosa untuk memperoleh energi. Karena adanya proses fotosintesis yang terjadi, maka glukosa tercipta dan inilah yang menjadi alasan mengapa bahan bakar respirasi seluler menggunakan glukosa. Dengan rumus H-(C=O) – (CHOH)5 kita bisa melihat struktur glukosa di mana ada 5 gugus hidroksi dan atom karbonlah yang menyusunnya.

Glukosa bisa juga disebut dengan istilah gula dan memang glukosa di dalam tubuh kita datangnya dari segala sumber makanan yang kita nikmati setiap hari, namun masih banyak orang yang bingung apa bedanya gula alami dan gula olahan. Gula alami selalu dapat diperoleh dari bahan makanan, seperti sayur dan buah dan ini adalah jenis gula yang paling sehat serta aman untuk dikonsumsi. Sementara untuk gula olahan, jenis ini merupakan jenis yang ditambahkan ke dalam minuman atau makanan.

Jenis gula olahan yang paling kerap dijumpai oleh kita adalah sukrosa atau gula pasir, beberapa orang juga senang menggunakan sirup jagung yang tinggi fruktosa. Gula olahan lebih berisiko meningkatkan berbagai penyakit pada tubuh sehingga sangat penting untuk menjauhi atau membatasi makanan dengan kandungan gula olahan.

Glukosa dapat dikatakan bahwa asal glukosa adalah dari luar tubuh kita. Glikogen adalah bentuk setelah glukosa disimpan di dalam tubuh dan glikogen ini berada di otot rangka tubuh serta organ hati. Somastostasin, *glucagon* dan insulin adalah sejumlah faktor utama yang memengaruhi jumlah glukosa pada tubuh dan hormon-hormon tersebut adalah yang kelenjar pankreas produksi selama ini.

Glukosa diserap ke dalam peredaran darah melalui saluran pencernaan. Sebagian glukosa ini kemudian langsung menjadi bahan bakar sel otak, sedangkan yang lainnya menuju hati dan otot, yang menyimpannya sebagai glikogen ("pati hewan") dan sel lemak, yang menyimpannya sebagai lemak. Glikogen merupakan sumber energi cadangan yang akan dikonversi kembali menjadi glukosa pada

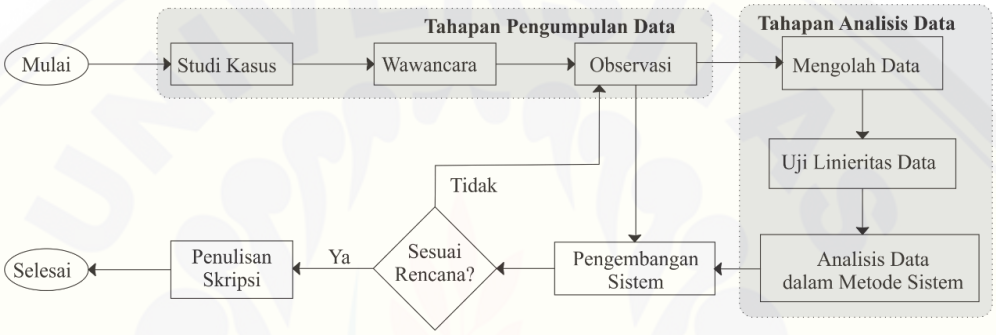
saat dibutuhkan lebih banyak energi. Lemak tidak akan pernah secara langsung dikonversi menjadi glukosa, meskipun lemak juga merupakan sumber energi. Fruktosa dan galaktosa, gula lain yang dihasilkan dari pemecahan karbohidrat, langsung diangkut ke hati, yang mengkonversinya menjadi glukosa.

Penentuan glukosa dengan salah satu cara sebagai berikut (Sudarmadji, 1996):

a. Cara Luff Schrool, yang ditentukan bukannya koprooksida yang mengendaptetapi dengan menentukan kuprioksida dalam larutan sebelum direaksikandengan gula reduksi (titrasi blanko) dan sesudah direaksikan dengan sampelgula reduksi (titrasi sampel). Penentuannya dengan titrasi denganmenggunakan Na-Tiosulfat. Selisih titrasi blanko dengan titrasi sampelekuivalen dengan gula reduksi.
b. Cara Munson Walker, penentuan gula cara ini adalah dengan menentukan banyaknya koprooksida yang terbentuk dengan cara penimbangan atau denganmelarutkan kembali dengan asam nitrat kemudian menitrasi dengan tiosulfat.Jumlah kuprooksida yang terbentuk ekuivalen dengan banyaknya gula reduksiyang ada dalam larutan.
c. Cara Lane-Eynon, penentuan gula cara ini adalah dengan cara menitrasiresgen Soxhlet (larutan CuSO4, K-Na-tartrat) dengan larutan gula yang akan diselidiki. Banyaknya larutan contoh yang dibutuhkan untuk menitrasi reagen Soxhlet dapat diketahui banyaknya gula yang ada dengan melihat tabel Lane-Eynon. Agar diperoleh penentuan yang tepat maka reagen Soxhlet perlu distandarisasi dengan larutan standar. Standarisasi ini dikerjakan untuk menentukan besarnya faktor koreksi dengan menggunakan tabel Lane-Eynon.
d. Cara Nelson Somogyi, penentuan kadar gula reduksi dengan menggunakan pereaksi tembaga-arsenol-molibdat. Prinsip kerja Nelson Somogyi yaitu tereduksinya jumlah endapan kuprookdisa yang bereaksi dengan arsenomolibdat yang tereduksi menjadi *molybdine blue* dan warna biru yang diukur absorbansinya. Reagen Nelson Somogyi berfungsi sebagai oksidator antara kuprooksida yang bereaksi dengan gula reduksi membentuk endapan merah bata. Dengan membandingkannya terhadap larutan standar, konsentrasi gula dalam sampel dapat ditentukan.

BAB 3. METODOLOGI PENELITIAN

Bab ini merupakan langkah dan prosedur yang dilakukan dalam mengumpulkan data atau informasi empiris guna memecahkan permasalahan. Alur yang digunakan dalam mengumpulkan data yang diperlukan untuk menyusun dan membangun aplikasi pada penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram alir tahapan penelitian

3.1 Jenis Penelitian

Pada penelitian ini digunakan dua jenis penelitian, yaitu penelitian kualitatif dan kuantitatif. Jenis penelitian kualitatif digunakan karena penelitian ini menganalisis studi pustaka dan melakukan wawancara untuk mengumpulkan sampel data 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 Laboraturium Mikrobiologi Fakultas MIPA Universitas Jember. Waktu dilaksanakannya penelitian adalah selama 6 bulan dimulai bulan November 2016 hingga April 2017.

3.3 Alat Penelitian

Alat yang digunakan dalam penelitian ini :

a. Laptop dengan aplikasi :
   1) *Ms. Office*
   2) *Google Chrome*
   3) *Neatbeans*
   4) *Xampp*
   5) *MySql*
   6) *Adobe Photoshop*
   7) *CorelDRAW*
   8) *yEd Graph Editor*
b. Kamera *webcam* Logitech C920
c. Spektrofotometer UV-5200PC UV/VIS
d. *Mini studio*
e. *Kuvet*

3.4 Tahap Pengumpulan Data

Tahap pengumpulan data yang akan dilakukan pada penelitian ini adalah sebagai berikut:

a. Studi Pustaka

   Studi pustaka merupakan teknik pengumpulan data dengan mengadakan studi penelaahan terhadap buku-buku, literatur-literatur, catatan-catatan, karya ilmiah, dan situs web yang ada hubungannya dengan masalah yang akan dipecahkan. Bertujuan untuk menyusun dasar teori yang akan digunakan dalam penelitian.

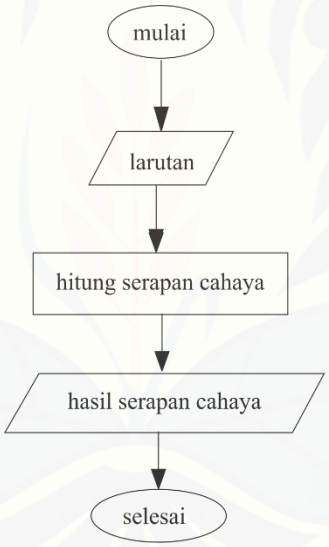
b. Wawancara

   Wawancara merupakan cara pengumpulan data yang behubungan langsung dengan narasumber dengan menyajukan pertanyaan kepada narasumber. Wawancara

terhadap narasumber bertujuan untuk memperoleh data yang dibutuhkan dalam penyelesaian penelitian.

c. Observasi

Observasi merupakan cara pengumpulan data dengan mengadakan pengamatan langsung terhadap objek yang diteliti dan mengadakan pencatatan secara sistematis dalam suatu periode tertentu. Observarsi ini bertujuan untuk mendapatkan data serapan cahaya berdasarkan kondisi objek dilapangan secara akurat dan variabel-variabel yang berpengaruh menggunakan spektrofotometri. Objek yang sama akan diambil gambarnya untuk mendapatkan nilai RGB dan proses analisis selanjutnya. Diagram alir pengumpulan data secara observasi dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram alir pengumpulan data nilai serapan cahaya larutan glukosa dengan spektrofotometer

3.5 Tahap Analisis

Tahap analisis yang akan dilakukan pada penelitian ini adalah sebagai berikut:

a. Mengolah Data

Mengolah data merupakan tahapan yang dilakukan setelah pengumpulan data atau observasi. Data yang telah terkumpul diolah dengan mengambil gambar dari kamera

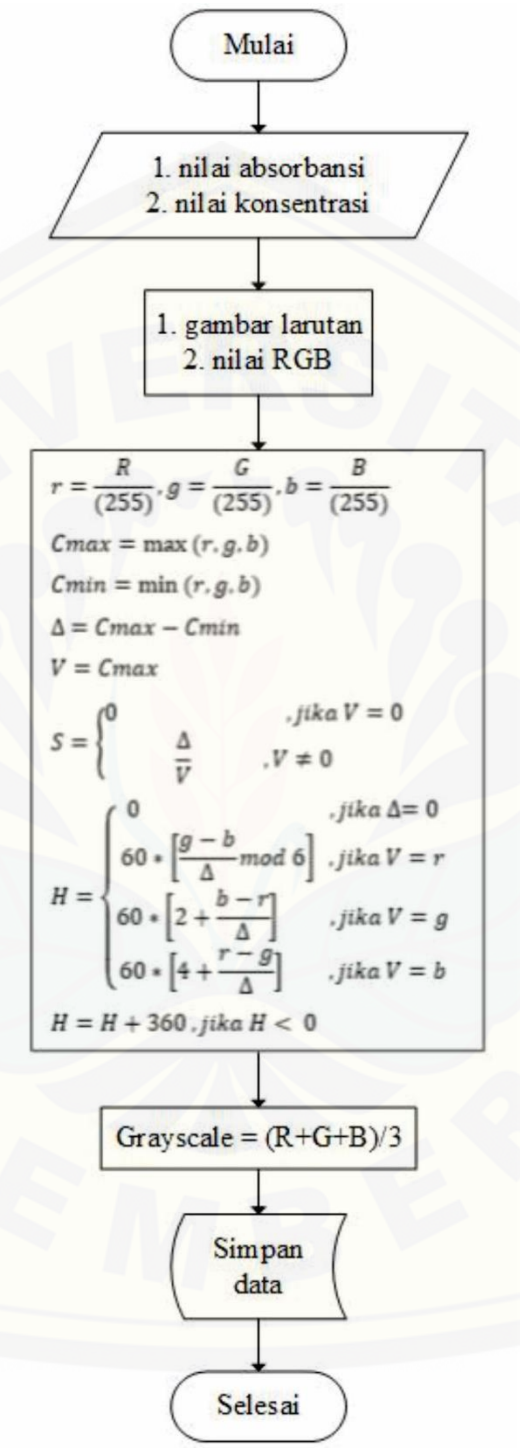
dan gambar dipotong sesuai ukuran yang telah disiapkan dan hitung agar mendapatkan nilai RGB objek gambar tersebut. Nilai RGB dikonfersikan kedalam penghitungan HSV dan *grayscale*. Nilai HSV dan *grayscale* yang telah didapatkan diolah untuk dicari data ruang warna yang dapat menghasilkan grafik linier yang merupakan representasi dari alat spektrofotometer dan mendapatkan rumus linier untuk direpresentasikan ke dalam bentuk data. Diagram alir pengolahan data ruang warna dapat dilihat pada Gambar 3.3.

b. Uji Linieritas Data

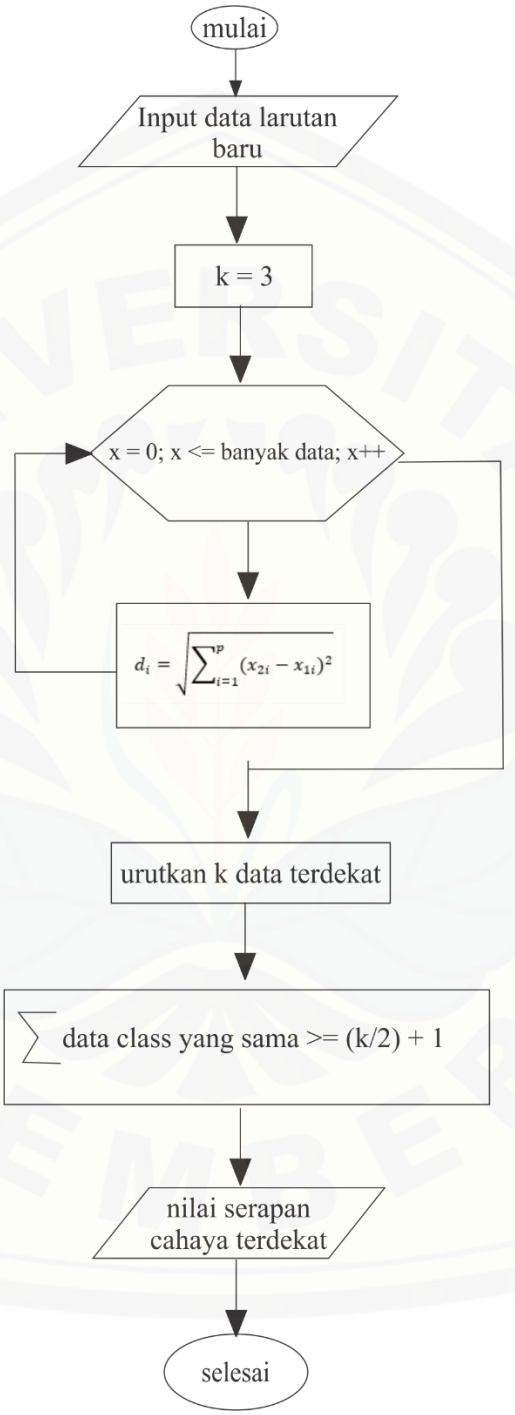
Menganalisis data dari ruang warna yang telah didapatkan menggunakan metode statistik linier. Baik korelasi maupun regresi linier dibangun berdasarkan asumsi bahwa variabel-variabel yang dianalisis memiliki hubungan linier. Strategi untuk memverifikasi hubungan linier dapat dilakukan dalam beberapa cara. Prosedur uji linieritas terbagi menjadi dua jenis, yaitu prosedur analisis melalui grafik dan melalui uji statistika (Widhiarso, 2010). Dalam penelitian ini uji linieritas menggunakan analisis melalui grafik dan uji statistik untuk mendapatkan perbandingan *R-squared* yang dapat merepresentasikan nilai data dari spektrofotometer. *R-squared* merupakan hubungan antar variabel yang dianalisis dibandingkan dengan model linier (Widhiarso, 2010).

c. Analisis Data dalam Metode Sistem

Data ruang warna setelah uji linieritas diambil untuk membentuk nilai klasifikasi. Menganalisis data menggunakan metode *K-Nearest Neighbor* untuk mendapatkan nilai klasifikasi yang sesuai dengan kebutuhan. Diagram alir analisis data menggunakan metode *K-Nearest Neighbor* dapat dilihat pada Gambar 3.4. *K-Nearest Neighbor* sendiri menghitung kedekatan data menggunakan *Euclidien Distance* pada tiap variabel. Dengan jumlah jarak terdekat kelas yang sama minimal nilai (K/2)+1.



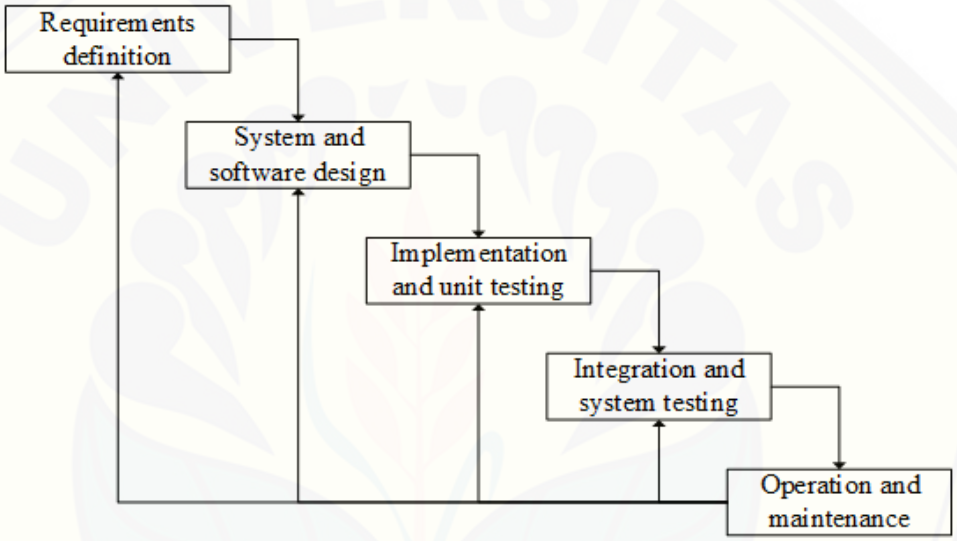
Gambar 3.3 Diagram alir pengolahan data ruang warna



Gambar 3.4 Diagram alir analisis data ke dalam metode sistem

3.6 Tahap Pengembangan Sistem

Pengembangan sistem aplikasi analisis konsentrasi glukosa ini menggunakan pengembangan perangkat lunak model *waterfall*. Terdapat 5 tahapan pada model *waterfall*, yaitu *requirement definition*, *system and software design*, *implementation and unit testing*, *integration and system testing*, dan *operation and maintenance* (Sommerville, 2003). Metode *waterfall* adalah proses pengembangan yang prosesnya mengalir kebawah (seperti air terjun) seperti pada Gambar 3.5.



Gambar 3.5 Metode *waterfall*

3.4.1 Analisis Kebutuhan

Tahap pertama pada proses pengembangan perangkat lunak ini adalah *requirement definition* (mendefinisikan kebutuhan). Pada tahap ini analisis kebutuhan perangkat lunak dibagi menjadi 2 jenis. Jenis pertama adalah kebutuhan fungsional. Dalam kebutuhan fungsional berisi proses-proses apa saja yang nantinya dilakukan oleh sistem, dan juga berisis informasi apa saja yang harus ada dan dihasilkan oleh sistem. Jenis kedua adalah kebutuhan non-fungsional. Dalam kebutuhan non-

fungsional berisi tentang properti perilaku yang dimiliki oleh sistem meliputi operasional, performa, keamanan, politik dan budaya.

3.4.2 Desain Sistem

Pembuatan desain perangkat lunak menggunakan *Unified Modeling Languange* (*UML*) yang dirancang menggunakan konsep *Object-Oriented Programming* (*OOP*). Berikut Pemodelan UML yang digunakan antara lain :

a. *Business Process*

   Business process merupakan diagram yang menggambarkan sebuah proses lengkap dengan *resources* dan informasi yang dibutuhkan, event yang mendorong terjadinya proses dan *goal* yang dituju.

b. *Use Case Diagram*

   Use case diagram merupakan diagram yang menggambarkan fungsi dan tugas yang dilakukan oleh aktor. *Use case diagram* juga dapat menggambarkan hak akses setiap actor dalam sebuah system.

c. Skenario

   Skenario merupakan penggambaran fitur atau isi yang ada di dalam *use case diagram*. Skenario juga menjelasan alur sistem yang menggambarkan aksi aktor dan reaksi sistem.

d. *Sequence Diagram*

   Sequence diagram adalah diagram yang menggambarkan interaksi antara objek satu dengan yang lain di dalam sistem yang dibangun pada urutan waktu. Diagram juga menggambarkan interaksi antara aktor, fitur, serta data yang berjalan.

e. *Activity Diagram*

   Activity diagram menggambarkan aliran system yang akan dibangun, dari awal sistem hingga sistem ditutup. *Activity diagram* mempunyai fungsi yang sama dengan *scenario* namun diimplementasikan menggunkan diagram alir.

f. *Class Diagram*

Class diagram digunakan untuk menggambarkan keterkaitan setiap *class* dalam pengembangan sistem yang dapat memudahkan dalam proses implementasi atau pengkodean program.

g. *Entity Relationship Diagram*

Entity Relation Diagram adalah diagram yang menggambarkan relasi data dalam sebuah basis data.

3.4.3 Implementasi Sistem

Pada tahap ini desain yang telah dibuat akan diimplemantasikan ke dalam kode program. Beberapa hal yang dilakukan dalam tahap implementasi antara lain:

a. Penulisan kode program (*coding*) menggunakan bahasa pemograman *Java* dengan arsitektur MVC (*Model View Controller*)
b. Manajemen basisdata menggunakan *MySQL*

3.4.4 Pengujian Sistem

Tahap testing harus dilakukan sebelum sistem diserahkan kapada *user*. Tahap ini dilakukan agar dapat mengetahui apakah sistem yang dibangun sesuai dengan kebutuhan yang telah dianalisis diawal. Serta agar mengetahui apakah terdapat kesalahan pada sistem yang dibangun. Tahap testing dilakukan guna menyempurnakan sistem sebelum diserahkan kepada *user*. Pada tahap testing ini dilakukan pengujian dengan dengan metode *white-box* dan metode *black-box*.

a. *White Box Testing*

White-box testing merupakan cara pengujian dengan melihat modul yang telah dibuat dengan program-program yang ada dan menganalisis apakah terjadi kesalahan atau tidak pada penulisan kode program. Pengujian *white-box* merupakan

teknik pengujian jalur dasar yang digunakan untuk menentukan kompleksitas logis dengan menentukan rangkaian dasar jalur eksekusinya.

Tahapan teknik pengujian jalur dasar meliputi dari *listing programe*, grafik alir, kompleksitas siklomatik, jalur program independen, dan pengujian basis set.

1) *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 *listing program* yang mengunakan bahasa pemrograman *Hypertext Preprocessor* (PHP) dapat dilihat pada Gambar 3.6.

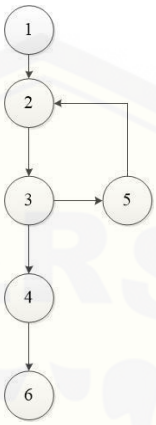
Gambar 3.6 Contoh listing program

2) Grafik Alir

Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah grafik alir (atau grafik program) yang menggunakan notasi lingkaran (simpul atau *node*) dan anak panah (*link* atau *edge*). Notasi ini menggambarkan aliran kontrol logika yang digunakan dalam suatu bahasa pemrograman.

Grafik alir merupakan sebuah notasi sederhana yang digunakan untuk mempresentasikan aliran kontrol (Pressman, 2009). Aliran kontrol yang digambarkan merupakan hasil penomoran dari *listing programe*. 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 3.7.



Gambar 3.7 Contoh grafik alir

3) *Cyclomatic Complexity*

Cyclomatic complexity adalah alat pengukuran untuk mengidentifikasi kompleksitas dari suatu program dengan cara menelusuri nomor dari jalur independen melalui *source code*-nya. Kompleksitas siklomatik merupakan metrik perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program (Pressman, 2009). Rumus yang digunakan untuk menghitung kompleksitas siklomatik, yaitu:

$$\boldsymbol{V(G) = E - N + 2} \quad \text{(3.1)}$$

V(G) merupakan kompleksitas siklomatik, E merupakan jumlah *edge*, dan N merupakan jumlah *node*.

Berdasarkan grafik alir pada Gambar 3.7 diketahui jumlah *edge* adalah 6 dan jumlah *node* adalah 6, sehingga dapat dihitung kompleksitas siklomatik V(G) = E – N + 2 = 6 – 6 + 2 = 2. Jadi jumlah jalur independen adalah 2 jalur.

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, 2009). Bila dinyatakan dalam grafik

alir, jalur independen harus bergerak setidaknya sepanjang satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisi (Pressman, 2009). Dari perhitungan kompleksitas siklomatik *Basis Set* pada Gambar 3.7 yang dihasilkan dari jalur independen secara linier adalah 2 jalur, yaitu:

Jalur / *Path* 1 : 1-2-3-4-6

Jalur / *Path* 2 : 1-2-3-5-2-3-4-6

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*, aplikasi ini telah memenuhi syarat.

b. *Black Box Testing*

Black-box testing adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi apakah sesuai dengan spesifikasi yang dibutuhkan (Rosa A.S, 2011). Contoh tabel pengujian *black-box* seperti pada Tabel 3.1 berikut:

Tabel 3.1 Tabel uji *black box*

Kelas Uji	Skenario uji	Hal yang diharapkan	Kesimpulan

3.4.5 Pemeliharaan

Pemeliharaan aplikasi diperlukan ketika aplikasi telah digunakan oleh *user*. Ketika aplikasi dijalankan mungkin saja masih terjadi kesalahan atau *error* yang tidak ditemukan sebelumnya. Sehingga diperlukan perbaikan pada aplikasi tersebut.

3.7 Gambaran Sistem

Sistem yang akan dibangun adalah "Pengembangan Sistem Aplikasi Analisis Konsentrasi Glukosa Berdasarkan Konten Warna dengan Menggunakan Metode *K-Nearest Neighbor*". Sistem ini akan dibantu dengan alat *input* berupa kamera untuk menangkap citra dalam sebuah *mini studio*.

Tahap pertama menganalisis konsentrasi glukosa yang akan dijadikan *datatrain* dalam menentukan klasifikasi konsentrasi glukosa. Tahap kedua sistem yang sudah memiliki data berupa citra glukosa dengan klasifikasi konsentrasi berbeda yang telah diketahui klasifikasi konsentrasinya, akan menangkap citra yang berada di *mini studio* menggunakan kamera. Tahap ketiga mencari jarak citra input dengan citra *database* untuk mendapatkan nilai menggunakan *Euclidean Distance*. Jarak yang terkecil diurutkan hingga urutan ke-K tetangga. Nilai klasifikasi mayoritas merupakan hasil terdekat dengan citra input.

Aplikasi ini dibangun menggunakan bahasa pemrograman *java* dan pengolahan basis data *MySQL*. Pada aplikasi ini terdapat tata cara penggunaan aplikasi, inputan data baru, test klasifikasi konsentrasi, dan hasil pengamatan sebelumnya. Hak akses untuk login yang berguna untuk mengetahui siapa yang melakukan pengamatan menggunakan aplikasi ini. *Userinterface* yang mudah dipahami dan aplikasi yang berukuran *full screen* yang dapat di *minimize* untuk mempermudah penggunaan.

BAB 4. PENGEMBANGAN SISTEM

Bab ini menguraikan mengenai analisis kebutuhan dan perancangan hingga tahap pengkodean dan pengujian aplikasi yang digunakan dalam proses pengembangan sistem aplikasi analisis konsentrasi glukosa berdasarkan konten warna dengan menggunakan metode *K-Nearest Neighbor*.

4.1 Analisis Kebutuhan Sistem

Berdasarkan metode pengembangan sistem model *waterfall*, tahapan awal yang dilakukan adalah tahapan analisis kebutuhan sistem. Tahapan ini dilakukan terhadap objek penelitian untuk memperoleh kebutuhan fungsional dan non-fungsional dari sistem yang dibangun. Dimana hasil analisis tersebut sangat mempengaruhi fungsionalitas sistem yang dibangun untuk dapat digunakan sesuai dengan fungsi dan kebutuhan pengguna.

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional aplikasi berisi fitur-fitur inti yang harus dipenuhi dalam sistem agar sistem mampu difungsikan sesuai dengan tujuan dan kebutuhan pengguna terhadap sistem itu sendiri.

Kebutuhan fungsional dari sistem prediksi bahaya tanah longsor, yaitu:

1. Sistem mampu mengelola data *user* meliputi *insert*, *update*, dan *delete*.
2. Sistem mampu menginputkan datatrain.
3. Sistem mampu menguji klasifikasi larutan menggunakan metode *K-Nearest Neighbor*.
4. Sistem mampu melihat hasil pengujian klasifikasi sebelumnya.
5. Sistem mampu melihat datatrain.

6. Sistem mampu mengubah password user untuk asisten lab
7. Sistem *login* dengan *user password*.

4.1.2 Kebutuhan Non-Fungsional

Kebutuhan *non-fungsional* merupakan fitur-fitur yang dimiliki untuk mendukung sistem dalam memenuhi fungsionalitasnya untuk dapat memenuhi kebutuhan dari pengguna. Kebutuhan *non-fungsional* dari sistem ini, yaitu :

1. Sistem berbasis *desktop*.
2. Tampilan sistem *user friendly*.

4.2 Deskripsi Umum Sistem

Sistem ini membantu mengklasifikasikan konsentrasi glukosa untuk meminimalisir kesalahan dalam mengukur konsentrasi glukosa dan menghemat biaya. Aplikasi ini mempunyai fitur mengklasifikasikan konsentrasi glukosa, input datatrain, dan melihat hasil klasifikasi sebelumnya.

Deskripsi umum dari Sistem yang dibangun dalam penelitian ini akan dijelaskan lebih detail pada SOP (*statement of perpose*) sistem dan fungsi sistem.

4.2.1 Statement Of Perpose (SOP)

Sistem klasifikasi konsentrasi glukosa diperlukan untuk meminimalisir kesalahan dalam mengukur konsentrasi glukosa dan meghemat biaya menggunakan pengolahan citra digital dan metode *K-Nearest Neighbor*. Klasifikasi yang dilakukan ditujukan pada konsentrasi glukosa berdasarkan *datatrain* yang telah diinputkan. *Datatrain* yang diinputkan dari hasil uji linieritas, dan menggunakan metode *K-Nearest Neighbor* untuk mengklasifikasikan konsentrasi glukosa dengan memberikan *output*

hasil klasifikasi konsentrasi glukosa. Keakuratan hasil klasifikasi dilakukan agar dapat mengetahui keakuratan klasifikasi dari sistem dan layak untuk digunakan sebagai mengklasifikasikan tingkat konsentrasi glukosa.

4.2.2 Fungsi Sistem

Sistem yang akan dibuat adalah aplikasi analisis konsentrasi glukosa berdasarkan konten warna dengan menggunakan metode *K-Nearest Neighbor*. Sistem ini merupakan sistem yang dapat mengklasifikasikan konsentrasi glukosa. Digunakan untuk meminimalisir kesalahan dalam menentukan konsentrasi glukosa dan dengan biaya yang lebih murah. Sistem ini juga sebagai upaya untuk memberikan hasil yang lebih akurat dan mengurangi kecenderungan tiap individu dalam merepresentasikan warna larutan menggunakan metode *K-Nearest Neighbor*. Model klasifikasi dilakukan berdasarkan *datatrain*.

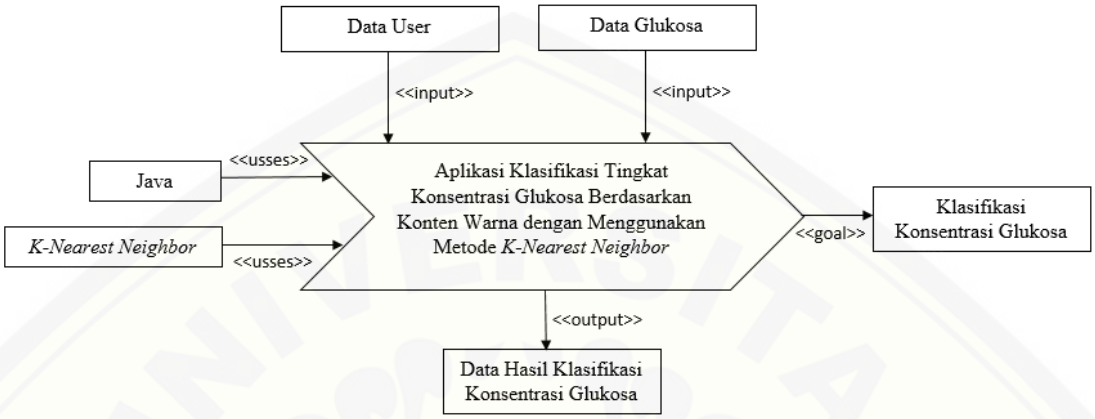
4.3 Desain Sistem

Tahapan yang dilakukan setelah melakukan analisis kebutuhan sistem yaitu tahap perencanaan pembangunan sistem yang dapat digambarkan dengan desain sistem. Desain pengembangan sistem aplikasi analisis konsentrasi glukosa, meliputi *bussiness process*, *use case diagram*, *skenario*, *activity diagram*, *sequence diagram*, *class diagram*, dan *entity relationship diagram*.

4.3.1 *Business Process*

Bussiness process digunakan untuk menggambarkan serangkaian aktifitas suatu aplikasi secara keseluruhan, lengkap dengan *resources* dan informasi yang dibutuhkan

seperti *input* dan *output* aplikasi, untuk melaksanakan bagian dari aplikasi sehingga mencapai tujuan yang telah ditentukan, seperti pada Gambar 4.1.



Gambar 4.1 *Business process*

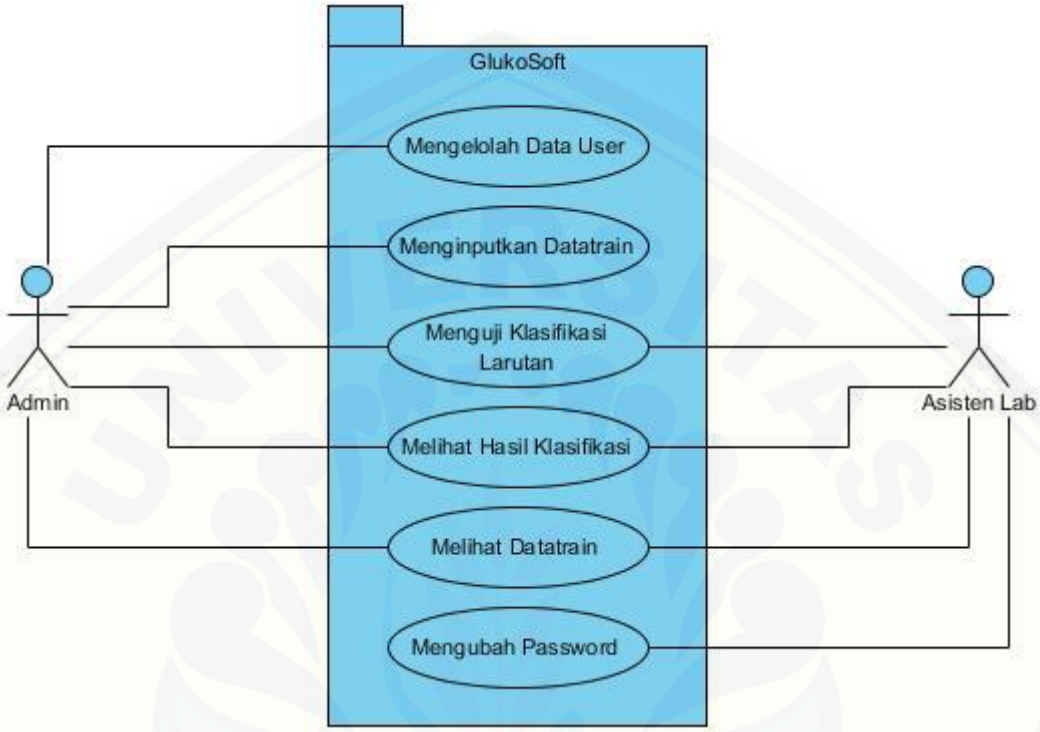
4.3.2 *Use Case Diagram*

Use case diagram merupakan diagram yang digunakan untuk menggambarkan secara ringkas siapa yang menggunakan sistem dan apa saja yang bisa dilakukan oleh sistem. *Use case* digambarkan dari beberapa aktor, *use-case*, dan interaksi diantara komponen tersebut yang dapat memberikan informasi dari sistem yang akan dibangun. Fitur-fitur pada sistem ini terdapat 6 fitur yang digambarkan dengan *elips* dan terdapat 2 *user*.

User yang pertama, yaitu Admin. Admin memiliki beberapa fitur, yaitu mengolah data *user*, menginputkan *datatrain*, menguji klasifikasi larutan, melihat hasil klasifikasi, dan melihat *datatrain*. *User* selanjutnya, yaitu asisten lab. Asisten lab memiliki beberapa fitur, yaitu menguji klasifikasi larutan, melihat hasil klasifikasi, melihat *datatrain*, dan mengubah password. Pada Gambar 4.2 digambarkan *use case diagram* aplikasi analisis konsentrasi glukosa yang akan dibangun.

Definisi *use case* merupakan penjelasan dari setiap *use case* yang merupakan fitur dari sistem. Penjelasan definisi dari *use case* aplikasi analisis konsentrasi glukosa

berdasarkan konten warna dengan menggunakan metode *K-Nearest Neighbor* dilihat pada Tabel 4.1



Gambar 4.2 *Use case diagram*

Tabel 4.1 Tabel *Use case diagram*

No	Use Case	Deskripsi
1	**Mengelola Data *User***	
	Tambah Data *User*	*Use case* yang menggambarkan proses menginputkan data *user* yang disimpan dalam sistem
	Ubah Data *User*	*Use case* yang menggambarkan proses mengubah data *user* yang ada di dalam sistem
	Hapus Data *User*	*Use case* yang menggambarkan proses menghapus data *user* yang ada di dalam sistem
2	**Menginputkan *Datatrain***	

No	Use Case	Deskripsi
	Tambah *Datatrain*	*Use case* yang menggambarkan proses menginputkan data *datatrain* yang disimpan dalam sistem
3	**Menguji Klasifikasi Larutan**	
	Uji Klasifikasi Larutan	*Use case* yang menggambarkan proses menguji klasifikasi konsentrasi larutan dengan metode *K-Nearest Neighbor*.
4	**Melihat Hasil Klasifikasi**	
	Melihat Hasil Klasifikasi	*Use case* yang menggambarkan proses melihat hasil klasifikasi larutan terdahulu.
5	**Melihat *Datatrain***	
	Melihat *Datatrain*	*Use case* yang menggambarkan proses melihat *datatrain* yang ada di dalam sistem.
6	**Mengubah Password**	
	Mengubah Password	*Use case* yang menggambarkan proses mengubah password user yang bersangkutan.

4.3.3 Skenario

Skenario digunakan untuk menceritakan isi fitur yang ada di *use case* diagram. Skenario menjelaskan alur sistem dan keadaan yang akan terjadi ketika terjadi suatu *event* tertentu.

Skenario pada pengembangan sistem aplikasi analisis konsentrasi glukosa berdasarkan konten warna dengan menggunakan metode *K-Nearest Neighbor* dapat dilihat pada Tabel 4.2 dibawah ini.

Tabel 4.2 Tabel skenario menguji klasifikasi larutan

Nomor *Use Case*	UC-03
Nama	Menguji Klasifikasi Larutan
Aktor	Admin dan Asisten Lab
Pre Condition	Aktor harus sudah *login* kedalam sistem.

Post Condition	Aktor berhasil mengklasifikasikan larutan glukosa
SKENARIO NORMAL MENGUJI KLASIFIKASI LARUTAN	
MENGUJI KLASIFIKASI LARUTAN GLUKOSA	
Aktor	**Sistem**
1. Klik menu uji	
	2. Menampilkan halaman uji larutan glukosa
3. Memasukkan larutan ke dalam "mini studio"	
4. Klik tombol test data	
	5. Mengambil citra gambar
	6. Mendapatkan nilai grayscale
	7. Menguji data dengan metode K-Nearest Neighbor
	8. Menampilkan hasil klasifikasi
	9. Menyimpan ke dalam database

Tabel 4.2 merupakan skenario dari *use case* menguji klasifikasi larutan. Skenario menguji klasifikasi larutan menjelaskan alur menguji klasifikasi larutan hingga menampilakan hasil klasifikasi. Skenario normal merupakan alur utama, yaitu menguji klasifikasi larutan glukosa. Sedangkan skenario alternatif merupakan bagian yang menangani *exception* atau alur alternatif dari proses sebelumnya. Kondisi setelah skenario ini dijalankan adalah aktor berhasil mengklasifikasikan larutan glukosa. Skenario dengan nomer *use case* lain dapat dilihat pada lampiran A.

4.3.4 *Activity Diagram*

Activity diagram menggambarkan aktivitas aktor dan sistem yang saling berhubungan dalam suatu aktivitas atau *event*. *Activity diagram* harus sesuai dengan skenario sistem yang telah dirancang. Sistem memberikan respon pada aktivitas yang dilakukan aktor.

Activity diagram menguji klasifikasi larutan menggambarkan alur aktivitas pada fitur menguji klasifikasi larutan. Fitur menguji klasifikasi larutan memiliki aktivitas

utama, yaitu menguji klasifikasi larutan. Detail urutan aktifitas dalam memenguji klasifikasi larutan ditunjukan pada Gambar 4.3. *Activity diagram* yang menggambarkan aktifitas sesuai skenario yang lain dapat dilihat pada lampiran B.

4.3.4 *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan prilaku pada sebuah skenario. *Sequence diagram* menguji klasifikasi larutan digunakan untuk menunjukkan interaksi antar objek pada fitur memenguji klasifikasi larutan. Objek yang terlibat pada fitur menguji klasifikasi larutan dengan aktor admin, antara lain *view* (*back*), *controller* (*control_access, control_page, control_formula*), *model* (*model_research*). Detail *sequence diagram* menguji klasifikasi larutan ditunjukan pada Gambar 4.4. *Sequence diagram* dengan perilaku skenario yang lain dapat dilihat pada lampiran C.

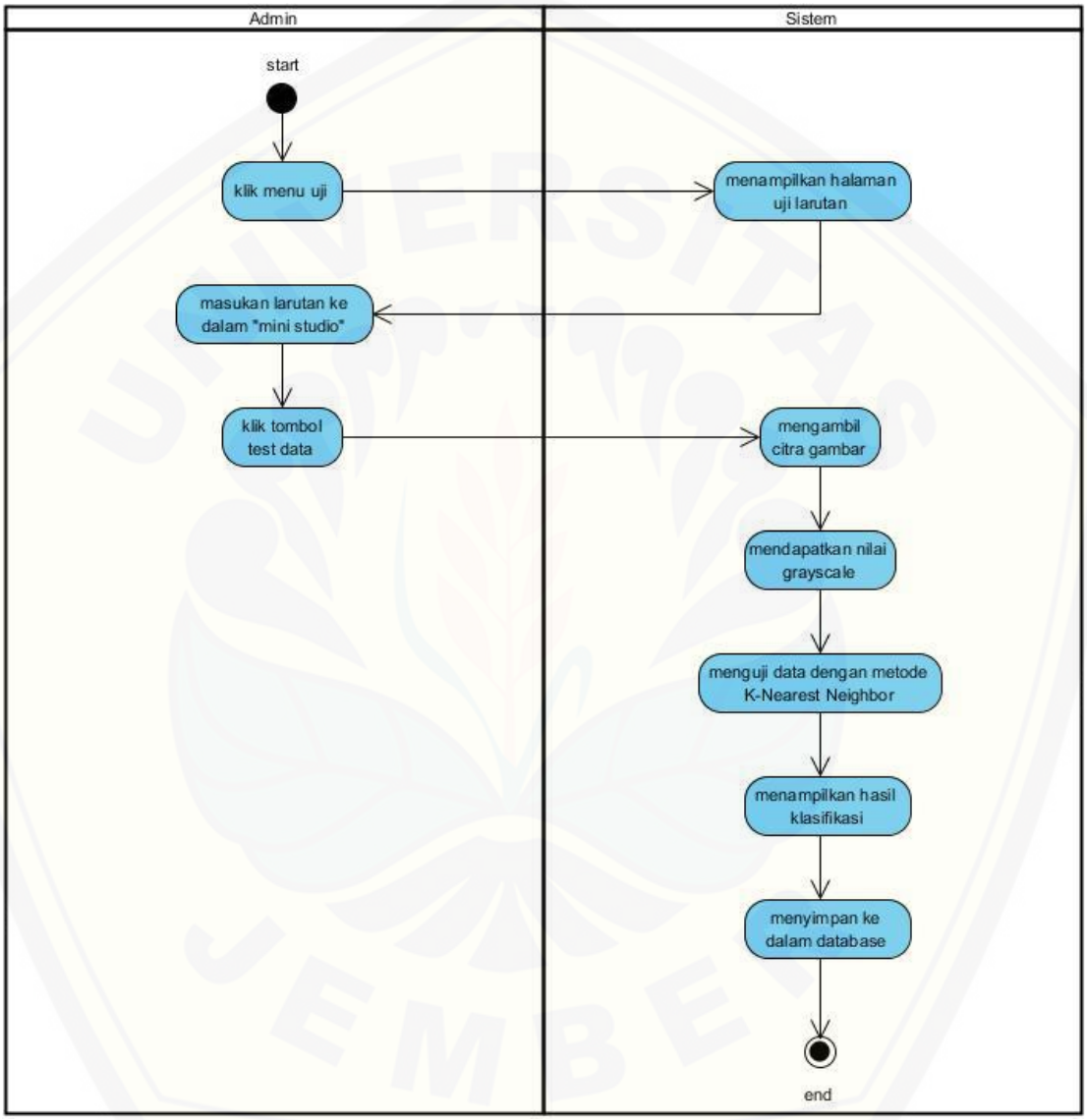
4.3.5 *Class Diagram*

Setelah melalui tahap pembuatan desain dengan *sequence diagram*, tahap selanjutnya membuat desain perancangan *class diagram*. *Class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membuat sistem. *Class diagram* memiliki 3 bagian utama yaitu attribute, operation, dan relasi. *Class diagram* lebih lengkapnya dapat dilihat pada Gambar 4.5.

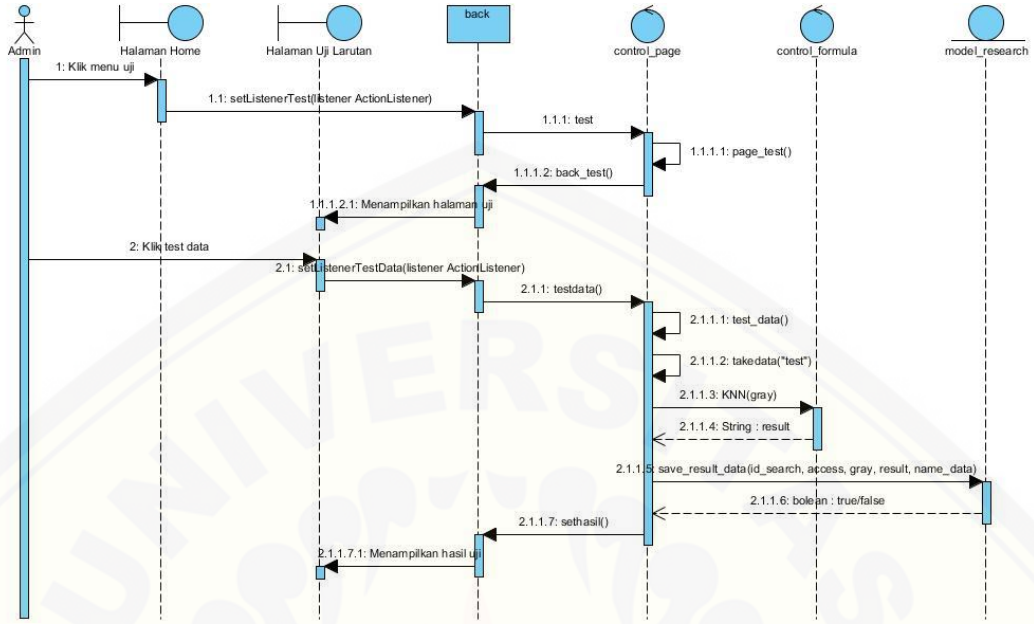
4.3.6 *Entity Relationship Diagram* (ERD)

Setelah pembuatan *class diagram*, tahap perancangan selanjutnya adalah membuat desain *database*. Desain ini berisi basis data yang akan digunakan oleh sistem yang akan dibangun. *Entity Relationship Diagram* menjelaskan hubungan antar data

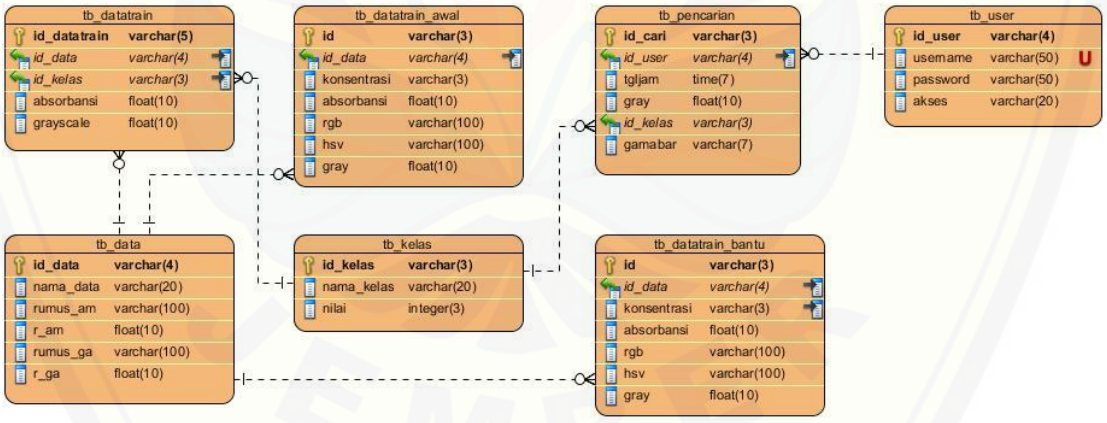
dalam basis data berdasarkan objek-objek dasar yang memiliki relasi. ERD lebih lengkapnya dapat dilihat pada Gambar 4.6.



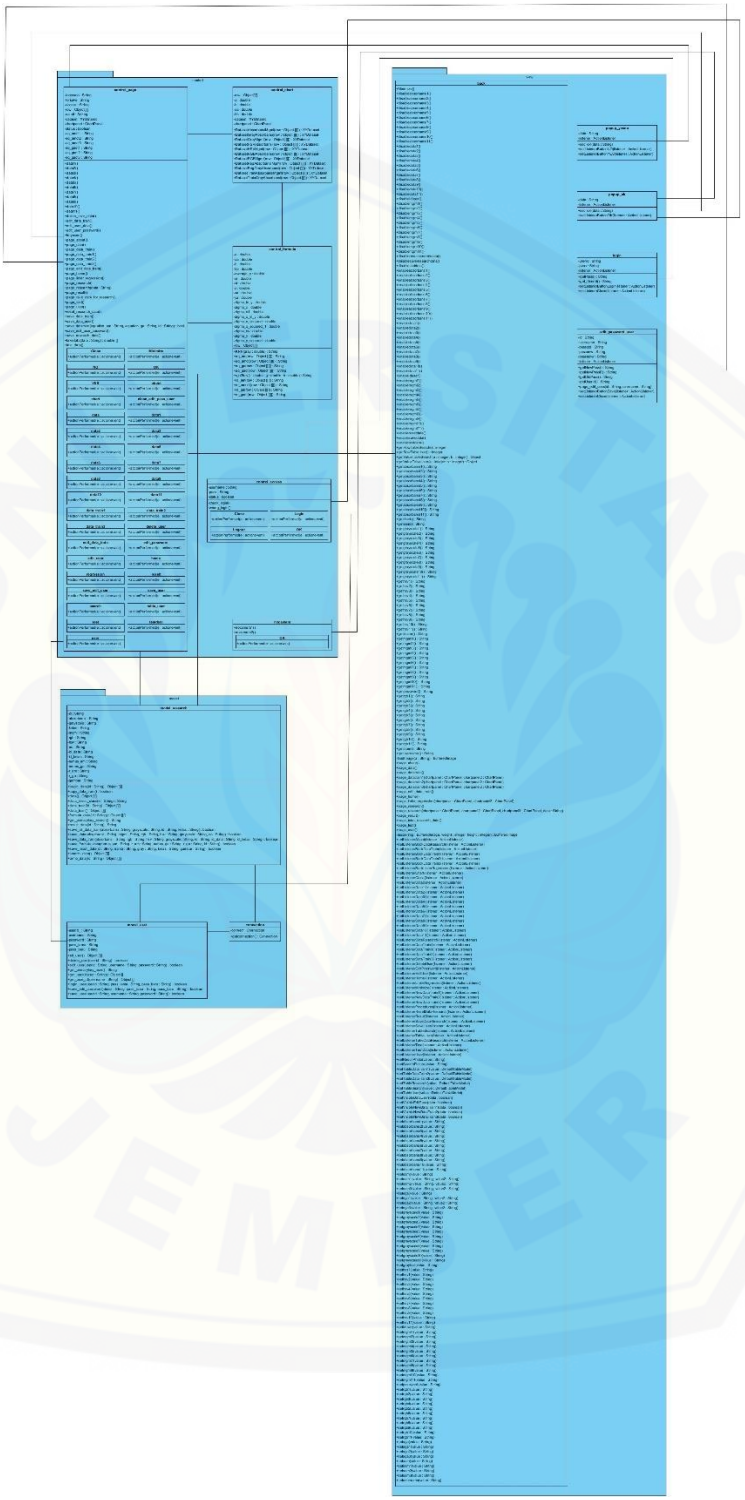
Gambar 4.3 *Activity diagram* menguji klasifikasi larutan



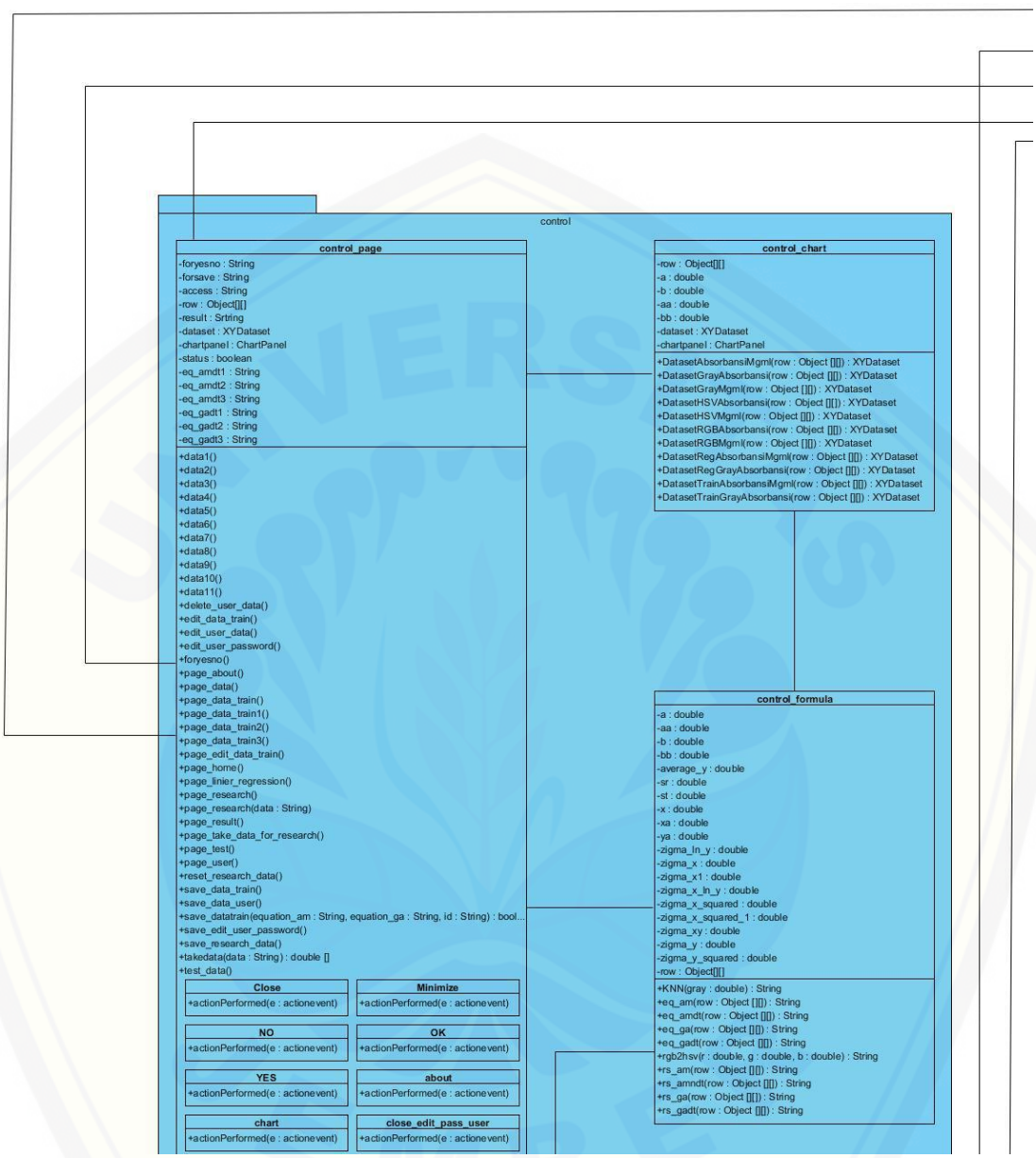
Gambar 4.4 *Sequence diagram* menguji klasifikasi larutan



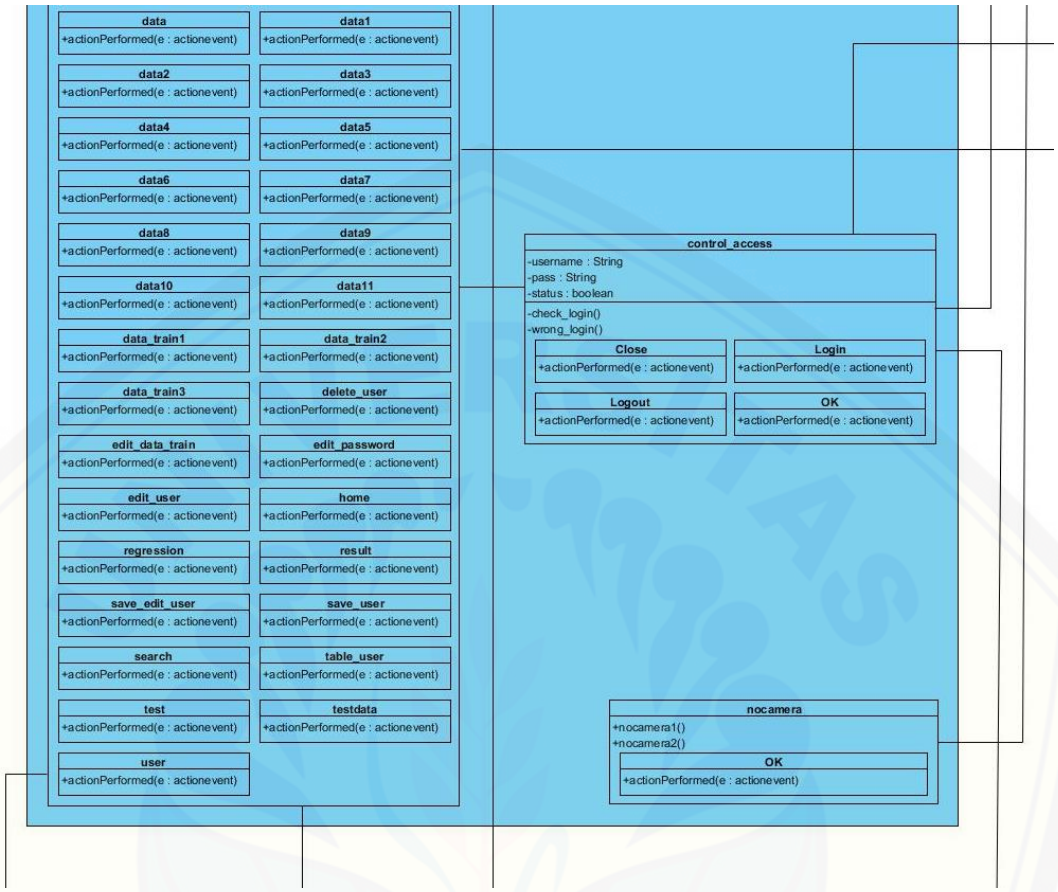
Gambar 4.5 *Entity relationship diagram*



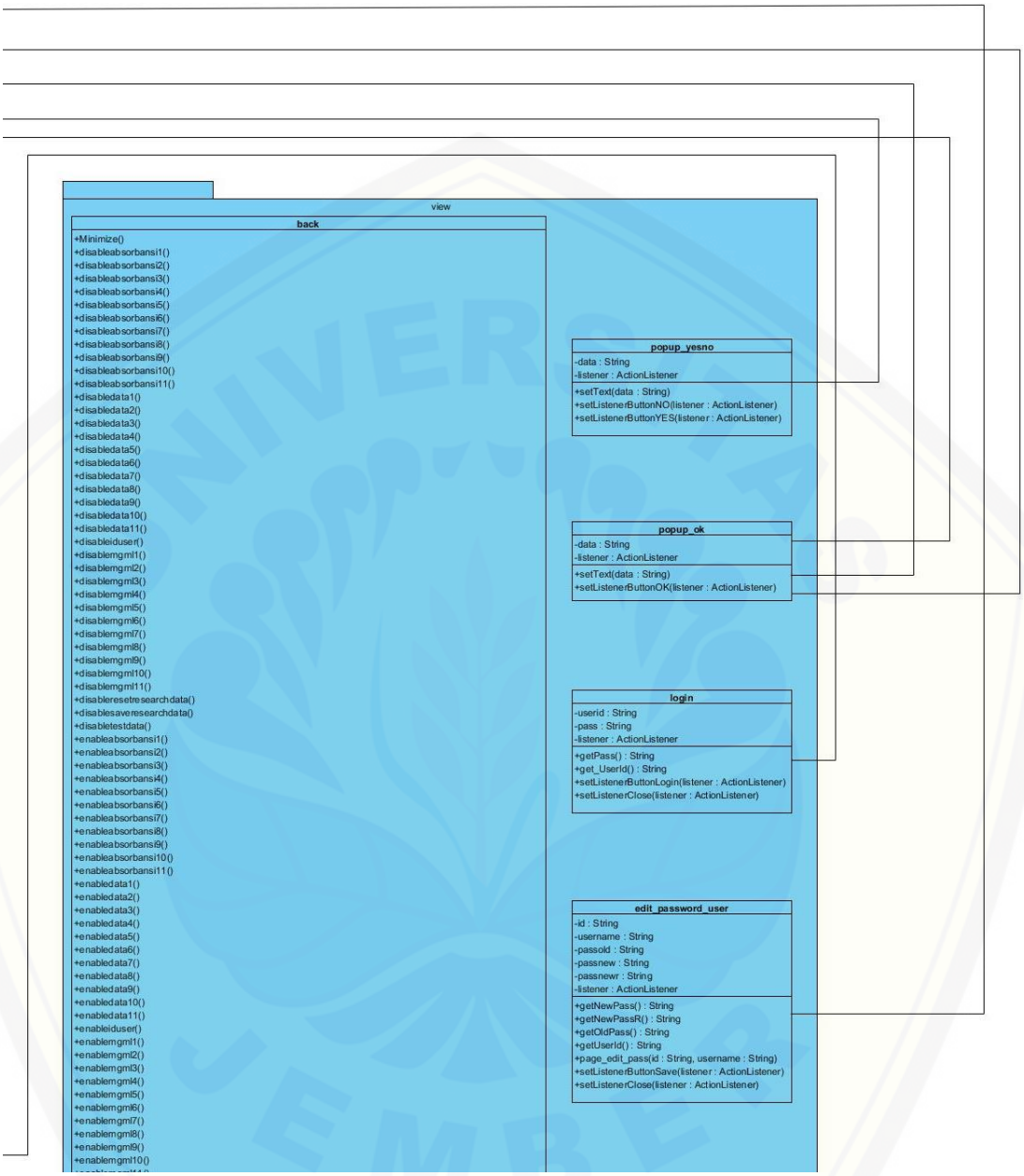
Gambar 4.6 *Class diagram* aplikasi



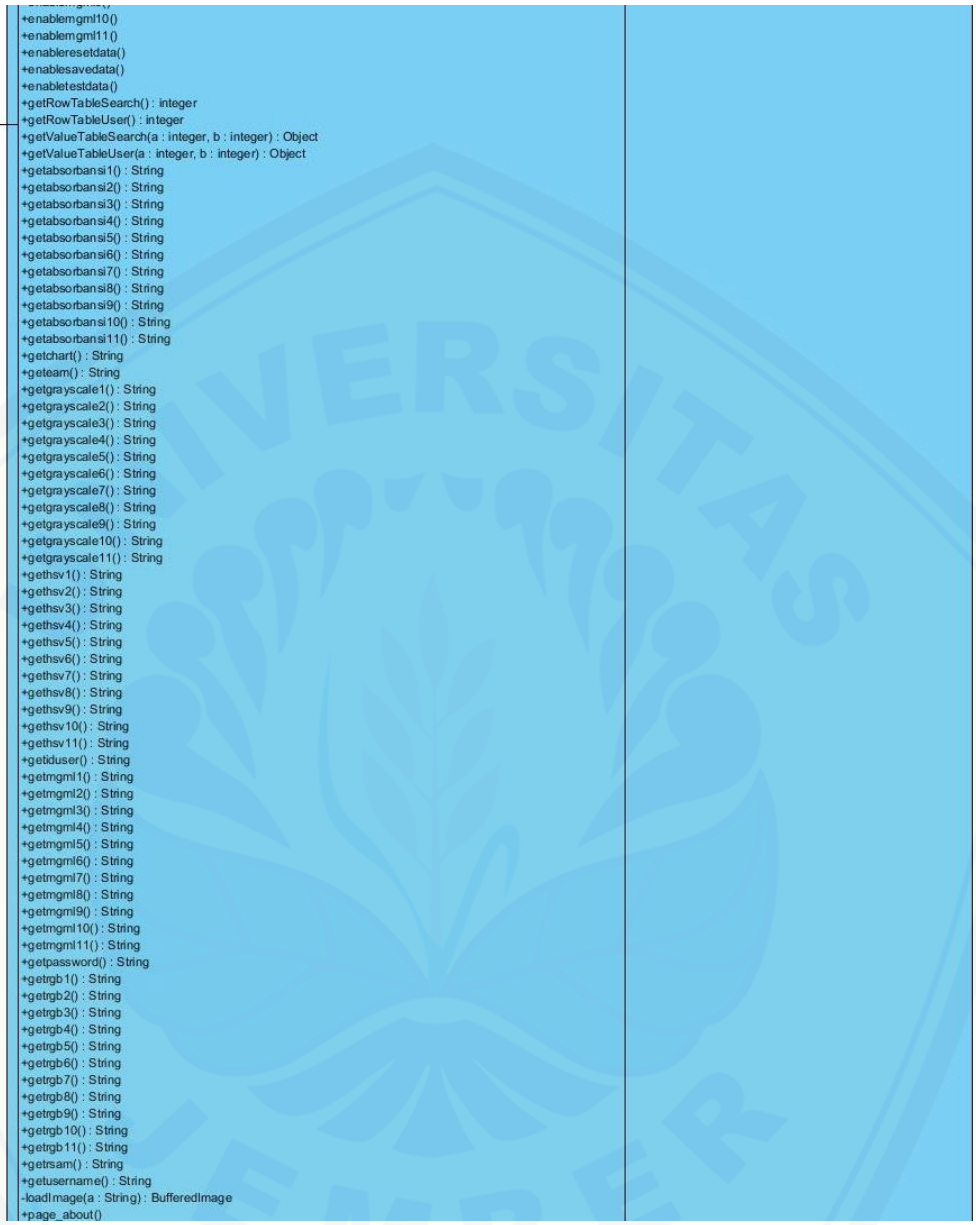
Gambar 4.7 Potongan A *class diagram* pada *package control*



Gambar 4.8 Potongan B *class diagram* pada *package control*

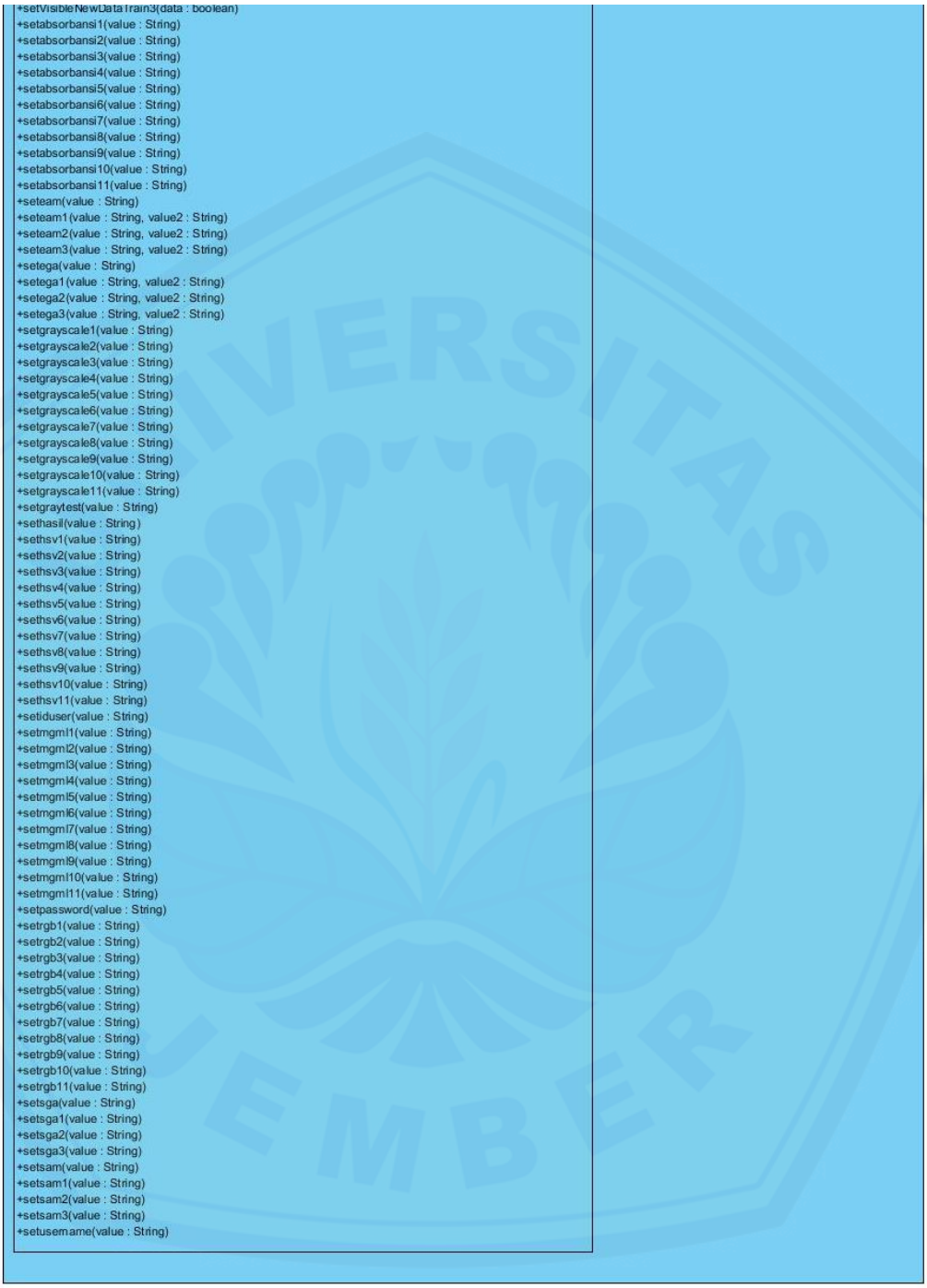


Gambar 4.9 Potongan A *class diagram* pada *package view*

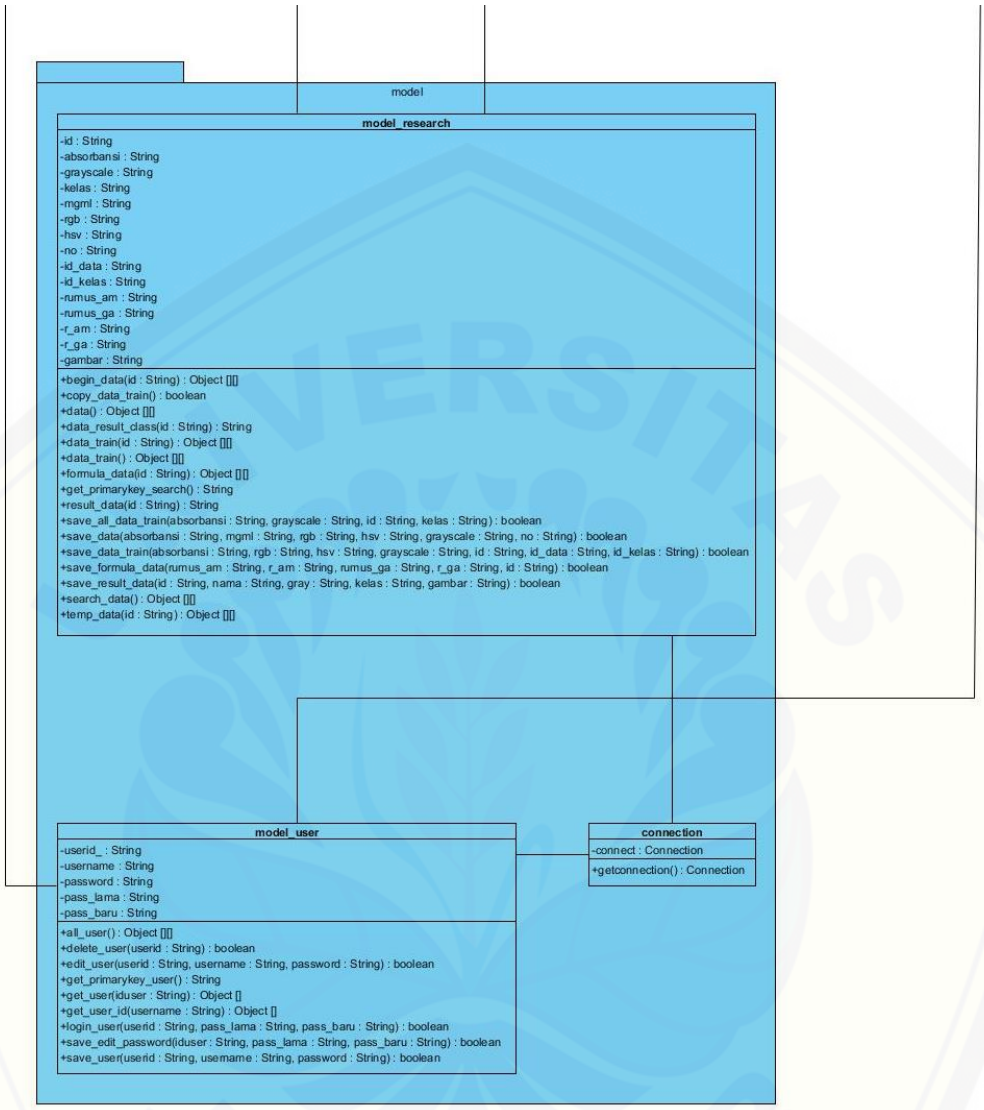


Gambar 4.10 Potongan B *class diagram* pada *package view*

Gambar 4.11 Potongan C *class diagram* pada *package view*



Gambar 4.12 Potongan D *class diagram* pada *package view*



Gambar 4.13 Potongan *class diagram* pada *package model*

4.4 Implementasi Sistem

Tahap implementasi sistem, mengimplementasikan desain perancangan ke dalam bahasa pemrograman. Bahasa pemrograman yang dipakai adalah bahasa pemrograman *Java* dan menggunakan *database MySql*. Dalam mengimplementasikan pengembangan sistem aplikasi analisis konsetrasi glukosa ini menggunakan arsitektur MVC (*Model View Controller*) untuk memudahkan di dalam pengembangan dan penulisan *coding*. Pada tahap implementasi sistem ini membangun fitur – fitur yang terdapat pada aplikasi analisis konsentrasi glukosa. Fitur-fitur tersebut, meliputi mengelola data *user*, menginputkan *datatrain*, menguji klasifikasi larutan, melihat hasil klasifikasi, melihat *datatrain*, dan mengubah password. Pada tahap menguji klasifikasi larutan glukosa menggunakan metode *K-Nearest Neighbor* di dalam barisan kode program yang dapat dilihat pada lampiran D.

4.5 Pengujian Sistem

Tahapan pengujian aplikasi merupakan suatu tahapan yang dilakukan secara sistematis untuk menguji dan mengevaluasi sistem dengan menggunakan sebuah metode pengujian sistem. Hal tersebut dilakukan dengan tujuan untuk mengevaluasi apakah kebutuhan sistem telah terpenuhi dan sistem layak untuk digunakan oleh pengguna. Agar pengujian yang dilakukan lebih valid, maka tahap pengujian sistem ini dilakukan dengan menggunakan dua metode, yaitu *white box* dan *black box*.

4.5.1 Metode *White-box*

Pengujian sistem dengan metode *white box* dilakukan untuk menguji sistem dari segi desain dan kode program. Hal tersebut bertujuan untuk mengevaluasi apakah sistem mampu menghasilkan fungsi-fungsi, inputan, dan keluaran yang sesuai dengan spesifikasi dari kebutuhan sistem itu sendiri. Pengujian dengan metode *white box* dilakukan oleh penulis dengan cara menghitung *independent path* yaitu dengan

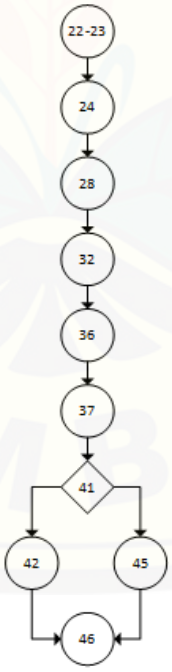
menggunakan suatu pengukuran kuantitatif *cyclomatic complexity*, *listing program*,penentuan jalur independen, dan *test case*. Tahapan-tahapan pengujian dengan metode *white box* ini akan diterapkan pada fitur yang dinilai dapat mewakili sistem ini. Tahapan pengujian jalur dasar meliputi:

a. *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.

b. Diagram Alir

Diagram alir merupakan notasi sederhana yang digunakan untuk merepresentasikan aliran kontrol. Aliran kontrol yang digambarkan merupakan hasil penomoran dari *listing* program. Diagram alir digambarkan dengan *node-node* (simpul) yang dihubungkan dengan *edge-edge* (garis) yang menggambarkan alur jalannya program. Diagram alir analisis konsentrasi dapat dilihat pada Gambar 4.7.



Gambar 4.7 Diagram alir analisis konsetrasi

c. Kompleksitas Siklomatik

Kompleksitas siklomatik merupakan metrik perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program. Bila digunakan dalam konteks teknik pengujian jalur dasar, nilai yang dihitung untuk kompleksitas siklomatik mendefinisikan jumlah jumlah jalur independen dalam basis set suatu program. Kompleksitas siklomatik prediksi tanah longsor berdasarkan diagram alir sebagai berikut:

CC = EDGE-NODE+2

CC = 10-10+2

CC = 2

d. Jalur Program Independen

Jalur independen adalah setiap jalur yang melalui program yang memperkenalkan setidaknya satu kumpulan pernyataan-pernyataan pemrosesan atau kondisi baru. Bila dinyatakan dalam grafik alir, jalur independen harus bergerak setidaknya sepanjang satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisi. Jalur program independen klasifikasi konsentrasi glukosa sebagai berikut:

Jalur 1 = 22-23-24-28-32-36-37-41-42-43-46

Jalur 2 = 22-23-24-28-32-36-37-41-45-46

e. 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 perangkat lunak jika salah satu jalur yang dieksekusi setidaknya satu kali. Pengujian basis set klasifikasi konsentrasi glukosa dapat dilihat pada Tabel 4.1.

Tabel 4.1 Test *case* analisis konsentrasi

Jalur 1	
Test Case	Jika nilai klasifikasi tidak cocok dengan nilai klasifikasi maka hasil gagal
Target yang diharapkan	Menampilkan hasil klasifikasi
HasilPengujian	Benar
Path/Jalur	22-23-24-28-32-36-37-41-42-43-46
Jalur 2	
Test Case	Jika nilai klasifikasi sama dengan nilai klasifikasi maka hasil sesuai dengan klasifikasi
Target yang diharapkan	Menampilkan hasil klasifikasi
HasilPengujian	Benar
Path/Jalur	22-23-24-28-32-36-37-41-45-46

4.5.2 Metode *Black Box*

Pengujian *black box* berfungsi untuk menguji aplikasi dari segi spesifikasi fungsional sistem dengan tujuan mengetahui apakah fungsi-fungsi, inputan, dan keluaran sistem sesuai dengan spesifikasi yang dibutuhkan oleh pengguna. Hasil pengujian dengan metode *black box* dapat dilihat pada Lampiran E.

BAB 6. PENUTUP

Bab ini berisi mengenai kesimpulan dan saran dari peneliti tentang penelitian yang telah dilakukan. Kesimpulan dan saran tersebut diharapkan dapat digunakan sebagai acuan pada penelitian selanjutnya.

6.1 Kesimpulan

Kesimpulan dari hasil penelitian yang telah dilakukan oleh peneliti adalah sebagai berikut:

1) Penelitian ini menggunakan spektrofotometer dengan tipe UV-5200PC UV/VIS untuk mendapatkan nilai absorbansi, kamera webcam Logitech C920 untuk mendapatkan ruang warna citra, *mini studio* sebagai media penempatan larutan, dan pencayahaan dengan lampu led berdaya 3 watt. Dari 3 ruang warna RGB, HSV, dan *grayscale* hanya ruang warna *grayscale* mampu merepresentasikan nilai absorbansi data dengan menggunakan persamaan linier *exponential* yang dapat digunakan untuk membentuk kelas klasifikasi konsentrasi glukosa.
2) Metode *K-Nearest Neighbor* mampu digunakan untuk mengklasifikasikan konsentrasi glukosa dengan *k*-tetangga bernilai 3. Hasil pengujian berdasarkan algoritma metode *K-Nearest Neighbor* dapat mengklasifikasikan konsentrasi glukosa dengan tepat. Persentase ketepatan akurasi klasifikasi sebesar 81,8%.
3) Keakuratan klasifikasi konsentrasi juga dipengaruhi oleh inputan citra. Kamera webcam Logitech C920 sendiri masiih memiliki pengaturan ISO yang otomatis sehingga tingkat sensitifitas sensor terhadap cahaya berubah-ubah. Pencahayaan menggunakan lampu led 3 watt, dengan panjang gelombang yang tidak diketahui. Berbeda dengan spektrofotometer yang memiliki lampu dengan panjang gelombang stabil dan dapat diatur. Maka klasifikasi konsentrasi glukosa menggunakan rentang yang cukup lebar dalam menentukan tiap kelas klasifikasi.

4) Sistem klasifikasi konsentrasi glukosa menggunakan metode *K-Nearest Neighbor* hanya mampu menghasilkan 13 rentang klasifikasi, Tidak terdefinisi di bawah 0, 0mg/ml – 5mg/ml, 6mg/ml – 15mg/ml, 16mg/ml - 25mg/ml, 26mg/ml - 35mg/ml, 36mg/ml - 45mg/ml, 46mg/ml - 55mg/ml, 56mg/ml - 65mg/ml, 66mg/ml - 75mg/ml, 76mg/ml - 85mg/ml, 86mg/ml - 95mg/ml, 96mg/ml - 100mg/ml, dan Tidak Terdefinisi diatas 100.

6.2 Saran

Beberapa saran dan masukan berikut diharapkan dapat memberikan perbaikan sistem dalam penelitian selanjutnya, antara lain:

1) Diharapkan pada penelitian lebih lanjut menggunakan kamera dan lampu yang memiliki spesifikasi lebih dari pada kamera dan lampu yang digunakan pada penelitian ini.
2) Dapat mempersempit rentang klasifikasi konsentrasi glukosa untuk mendapatkan hasil klasifikasi yang lebih akurat.

DAFTAR PUSTAKA

Bassett, J., Denney, R. C., Jeffery, G. H., & Mendham, J. (1994). *Buku Ajar Vogel: Kimia Analisis Kuantitatif Anorganik*. (I. L. S. A. Handyana P, Ed.). Jakarta: EGC.

Basuki, A., Palandi, J., & Fatchurrrochman. (2005). *Pengolahan Citra Digital Menggunakan Visual Basic*. Yogyakarta: Graha Ilmu.

Day Jr., R. A., & Underwood, A. L. (1996). *Analisis Kimia Kuantitatif* (5th ed.). Jakarta: Erlangga.

Hermawati, F. (2013). *Data Mining*. Yogyakarta: C.V. Andi Offset.

Kadir, A., & Susanto, A. (2012). *Teori dan Aplikasi Pengolahan Citra*. Yogyakarta: C.V. Andi Offset.

Krisandi, N., Prihandono, B., & Helmi. (2013). Algoritma K - Nearest Neighbor dalam Klasifikasi Data Hasil Produksi Kelapa Sawit Pada Pt. Minamas Kecamatan Parindu. *Buletin Ilmiah Math.Stat.dan Terapannya(Bimaster)*, *2*(1), 33–38.

Marks, D. B., Marks, A. D., & Smith, C. M. (2000). *Biokimia Kedokteran Dasar: Sebuah Pendekatan Klinis*. (L. I. M. Joko Suyono, Vivi Sadikin, Ed.) (I). Jakarta: EGC.

Munir, R. (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritmik* (1st ed.). Bandung: Informatika Bandung.

Mustakim, & F, G. (2016). Algoritma K-Nearest Neighbor Classification Sebagai Sistem Prediksi Predikat Prestasi Mahasiswa, *13*(2), 195–202.

Naufal, A. R., Wahono, R. S., & Syukur, A. (2015). Penerapan Bootstrapping dan Weighted Information Gain Untuk Optimasi Parameter Pada Algoritma Support Vector Machine Untuk Prediksi Loyalitas Pelanggan, *1*(2), 98–108.

Pressman, R. S. (2009). *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*. https://doi.org/10.1017/CBO9781107415324.004

Rosa A.S, and M. S. (2011). Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Obyek). *Bandung: Modula*, *53*(1), 160.

https://doi.org/10.1017/CBO9781107415324.004

Sommerville, I. (2003). *Software Engineering: Rekayasa Perangkat Lunak* (6th ed.). Jakarta.

Sudarmadji, S. (1996). *Analisa Bahan Makanan dan Pertanian*. Yogyakarta: Liberty.

Widhiarso, W. (2010). Produser Uji Linieritas Pada Hubungan Antar Variabel. Retrieved May 6, 2017, from http://wahyupsy.blog.ugm.ac.id/2010/08/30/prosedur-uji-linieritas-pada-hubungan-antar-variabel/

LAMPIRAN A. SKENARIO

A.1 Skenario Mengelola Data User

Nomor *Use Case*	UC-01
Nama	Mengolah Data User
Aktor	Admin
Pre Condition	Admin harus sudah *login* kedalam sistem.
Post Condition	Admin berhasil mengelola data user
SKENARIO NORMAL MENGOLAH DATA USER	
TAMBAH DATA USER	
Aktor	**Sistem**
1. Klik menu home	
	2. Menampilkan halaman home
3. Klik tombol user	
	4. Menampilkan data user dan form user
5. Menginputkan data user baru	
6. Klik tombol simpan	
	7. Menyimpan input data user ke dalam database
	8. Menampilkan kotak dialog “User berhasil ditambahkan”
9. Klik OK	
	10. Menampilkan data user dan form user
SKENARIO ALTERNATIF MENGOLAH DATA USER	
DATA PARAMETER TIDAK DAPAT MENYIMPAN	
6a. Klik tombol simpan	
	7a. Menyimpan input data user ke dalam database
	8a. Menampilkan kotak dialog “Database bermasalah silahkan coba lagi nanti”
SKENARIO NORMAL MENGOLAH DATA USER	
UBAH DATA USER	
Aktor	**Sistem**
1. Klik menu home	
	2. Menampilkan halaman home
3. Klik tombol user	

	4. Menampilkan data user dan form user
5. Klik data user	
6. Mengubah data user	
7. Klik tombol ubah	
	8. Mengubah data user ke dalam database
	9. Menampilkan kotak dialog "User berhasil diubah"
10. Klik OK	
	11. Menampilkan data user dan form user
SKENARIO ALTERNATIF MENGOLAH DATA USER	
DATA PARAMETER TIDAK DAPAT MENGUBAH	
7a. Klik tombol ubah	
	8a. Mengubah data user ke dalam database
	9a. Menampilkan kotak dialog "Database bermasalah silahkan coba lagi nanti"
SKENARIO NORMAL MENGOLAH DATA USER	
HAPUS DATA USER	
Aktor	**Sistem**
1. Klik menu home	
	2. Menampilkan halaman home
3. Klik tombol user	
	4. Menampilkan data user dan form user
5. Klik data user	
6. Klik tombol hapus	
	7. Menghapus data user di dalam database
	8. Menampilkan kotak dialog "User berhasil dhapus"
9. Klik OK	
	10. Menampilkan data user dan form user
SKENARIO ALTERNATIF MENGOLAH DATA USER	
DATA PARAMETER TIDAK DAPAT MENGHAPUS	
6a. Klik tombol hapus	

	7a. Menghapus data user di dalam database
	8a. Menampilkan kotak dialog "Database bermasalah silahkan coba lagi nanti"

A.2 Skenario Menginputkan Datatrain

Nomor *Use Case*	UC-02
Nama	Menginputkan datatrain
Aktor	Admin
Pre Condition	Admin harus sudah *login* kedalam sistem.
Post Condition	Admin berhasil menginputkan datatrain
SKENARIO NORMAL MENGINPUTKAN DATATRAIN	
TAMBAH DATATRAIN	
Aktor	**Sistem**
1. Klik menu datatrain	
	2. Menampilkan halaman datatrain
3. Klik datatrain 1	
	4. Menampilkan halaman datatrain 1
5. Klik tombol data baru	
	6. Menampilkan kotak dialog "Apakah Anda yakin akan mereset semua datatrain?"
7. Klik tombol yes	
	8. Menampilkan halaman pengambilan data
9. Memasukkan larutan glukosa bagian 1 ke dalam "mini studio"	
10. Menginputkan data glukosa bagian 1	
11. Klik tombol foto untuk mengambil citra glukosa	
	12. Memproses pengambilan citra
	13. Menghitung ruang warna rgb, hsv, dan grayscale
14. Mengulangi no. 9 – 13 sampai larutan bagian 1 selesai diambil citra	
15. Klik tombol simpan	
	16. Menyimpan ke dalam database sementara
	17. Menampilkan kotak dialog "Silahkan lanjutkan data selanjutnya"
18. Klik OK	

	19. Menampilkan halaman pengambilan data
20. Memasukkan larutan glukosa bagian 2 ke dalam “mini studio”	
21. Menginputkan data glukosa bagian 2	
22. Klik tombol foto untuk mengambil citra glukosa	
	23. Memproses pengambilan citra
	24. Menghitung ruang warna rgb, hsv, dan grayscale
25. Mengulangi no. 20 – 24 sampai larutan bagian 2 selesai diambil citra	
26. Klik tombol simpan	
	27. Menyimpan ke dalam database sementara
	28. Menampilkan kotak dialog “Silahkan lanjutkan data selanjutnya”
29. Klik OK	
	30. Menampilkan halaman pengambilan data
31. Memasukkan larutan glukosa bagian 3 ke dalam “mini studio”	
32. Menginputkan data glukosa bagian 3	
33. Klik tombol foto untuk mengambil citra glukosa	
	34. Memproses pengambilan citra
	35. Menghitung ruang warna rgb, hsv, dan grayscale
36. Mengulangi no. 31 – 35 sampai larutan bagian 3 selesai diambil citra	
37. Klik tombol simpan	
	38. Menyimpan ke dalam database sementara
	39. Menyalin database sementara ke dalam database perhitungan awal

	40. Menghitung datatrain baru dengan rumus linier dari absorbansi dan ruang warna grayscale
	41. Menyimpan ke dalam database datatrain
	42. Menampilkan halaman datatrain
SKENARIO ALTERNATIF MENGINPUTKAN DATATRAIN	
DATA PARAMETER KLIK TOMBOL NO	
7a. Klik tombol NO	
	8a. Menampilkan halaman datatrain 1

A.3 Skenario Menguji Klasifikasi Larutan

Nomor *Use Case*	UC-03
Nama	Menguji Klasifikasi Larutan
Aktor	Admin dan Asisten Lab
Pre Condition	Aktor harus sudah *login* kedalam sistem.
Post Condition	Aktor berhasil mengklasifikasikan larutan glukosa
SKENARIO NORMAL MENGUJI KLASIFIKASI LARUTAN	
MENGUJI KLASIFIKASI LARUTAN GLUKOSA	
Aktor	**Sistem**
10. Klik menu uji	
	11. Menampilkan halaman uji larutan glukosa
12. Memasukkan larutan ke dalam "mini studio"	
13. Klik tombol test data	
	14. Mengambil citra gambar
	15. Mendapatkan nilai grayscale
	16. Menguji data dengan metode K-Nearest Neighbor
	17. Menampilkan hasil klasifikasi
	18. Menyimpan ke dalam database

A.4 Skenario Melihat Hasil Klasifikasi

Nomor *Use Case*	UC-04
Nama	Melihat Hasil Klasifikasi
Aktor	Admin dan Asisten Lab
Pre Condition	Aktor harus sudah *login* kedalam sistem.
Post Condition	Aktor berhasil melihat hasil klasifikasi larutan glukosa
SKENARIO NORMAL MENGUJI KLASIFIKASI LARUTAN MELIHAT HASIL KLASIFIKASI LARUTAN GLUKOSA	
Aktor	**Sistem**
A.1Klik menu hasil	
	A.2Menampilkan halaman hasil
A.3Klik data dalam tabel	
	A.4Menampilkan citra dan hasil klasifikasi

A.5 Skenario Melihat Datatrain

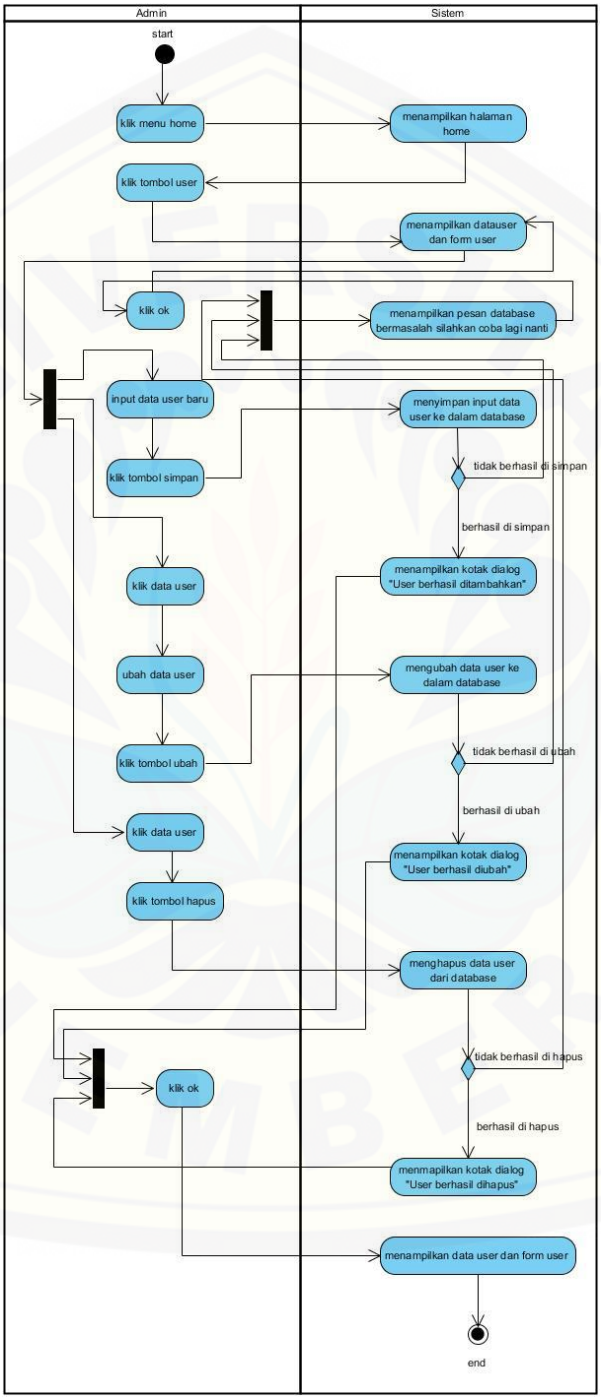
Nomor *Use Case*	UC-05
Nama	Melihat Datatrain
Aktor	Admin dan Asisten Lab
Pre Condition	Aktor harus sudah *login* kedalam sistem.
Post Condition	Aktor berhasil melihat datatrain
SKENARIO NORMAL MELIHAT DATATRAIN	
MELIHAT DATATRAIN 1	
Aktor	**Sistem**
1. Klik menu datatrain	
	2. Menampilkan halaman datatrain
3. Klik datatrain 1	
	4. Menampilkan halaman data datatrain 1
SKENARIO NORMAL MELIHAT DATATRAIN	
MELIHAT DATATRAIN 2	
Aktor	**Sistem**
1. Klik menu datatrain	
	2. Menampilkan halaman datatrain
3. Klik datatrain 2	
	4. Menampilkan halaman data datatrain 2
SKENARIO NORMAL MELIHAT DATATRAIN	
MELIHAT DATATRAIN 3	
Aktor	**Sistem**
1. Klik menu datatrain	
	2. Menampilkan halaman datatrain
3. Klik datatrain 3	
	4. Menampilkan halaman data datatrain 3

A.6 Skenario Mengubah *Password*

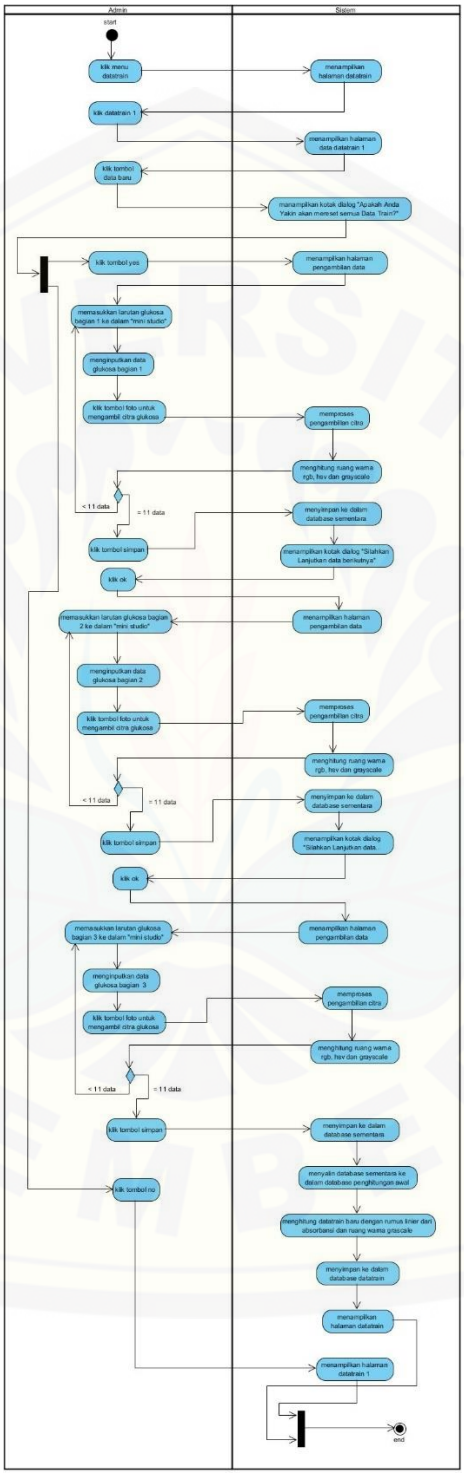
Nomor *Use Case*	UC-06
Nama	Mengubah *Password*
Aktor	Asisten Lab
Pre Condition	Asisten Lab harus sudah *login* kedalam sistem.
Post Condition	Asisten Lab berhasil mengubah password untuk *login*
SKENARIO NORMAL MENGUBAH PASSWORD	
MENGUBAH PASSWORD	
Aktor	**Sistem**
1. Klik menu home	
	2. Menampilkan halaman home
3. Klik tombol ubah password	
	4. Menampilkan kotak dialog ubah password
5. Isi form ubah password	
6. Klik simpan	
	7. Menyimpan ke dalam database
	8. Menampilkan kotak dialog "Password telah di rubah"
9. Klik OK	
	10. Menampilkan halaman home
SKENARIO ALTERNATIF MENGUBAH PASSWORD	
DATA PARAMETER PASSWORD LAMA TIDAK SESUAI	
6a. Klik simpan	
	7a. Menampilkan kotak dialog "Password lama Anda tidak sesuai"
SKENARIO ALTERNATIF MENGUBAH PASSWORD	
DATA PARAMETER PASSWORD ULANGAN TIDAK SAMA	
6a. Klik simpan	
	7a. Menampilkan kotak dialog "Ulangan Password tidak sama"

LAMPIRAN B. *ACTIVITY DIAGRAM*

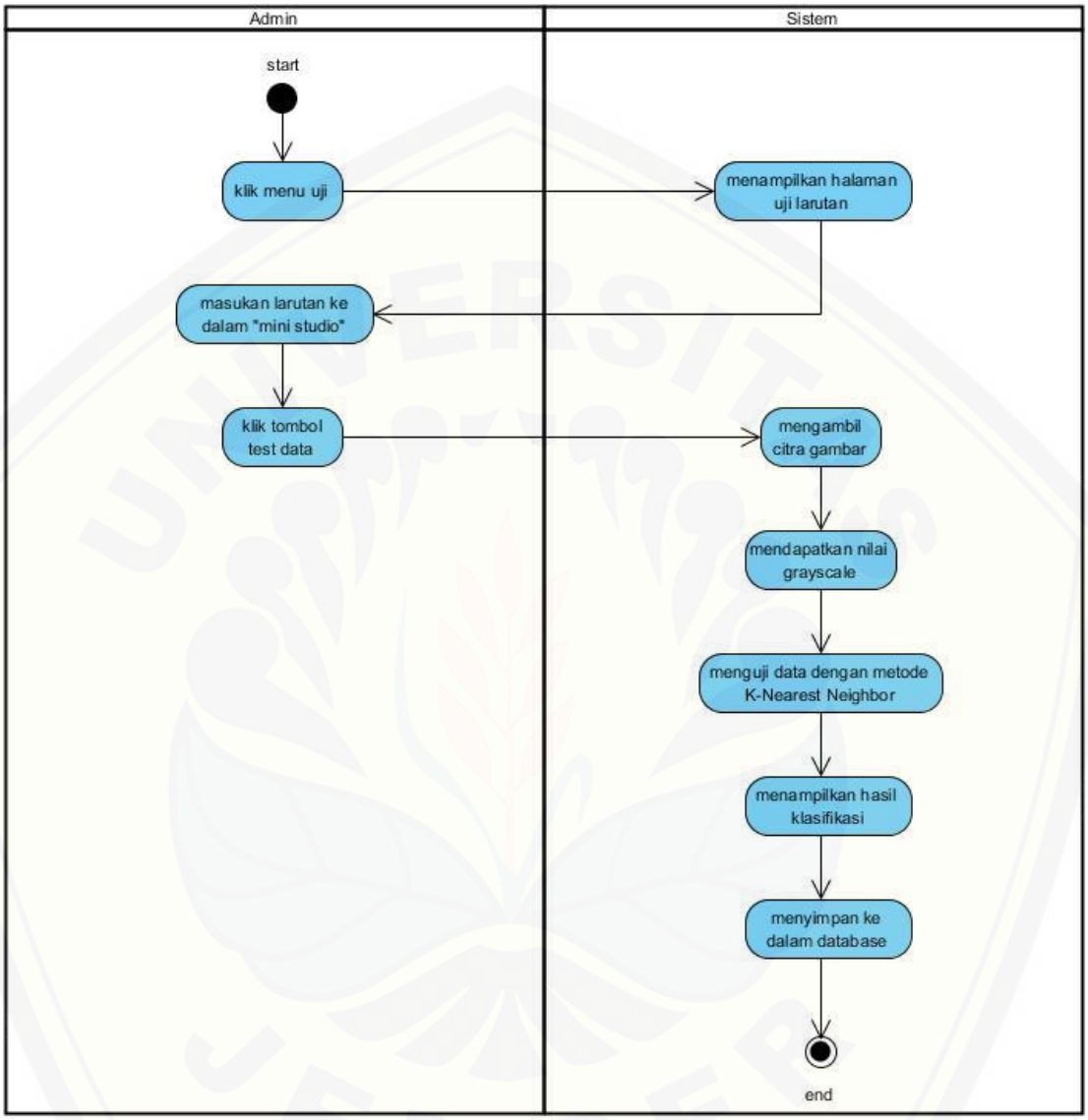
B.1 *Activity Diagram* Mengelola Data *User*



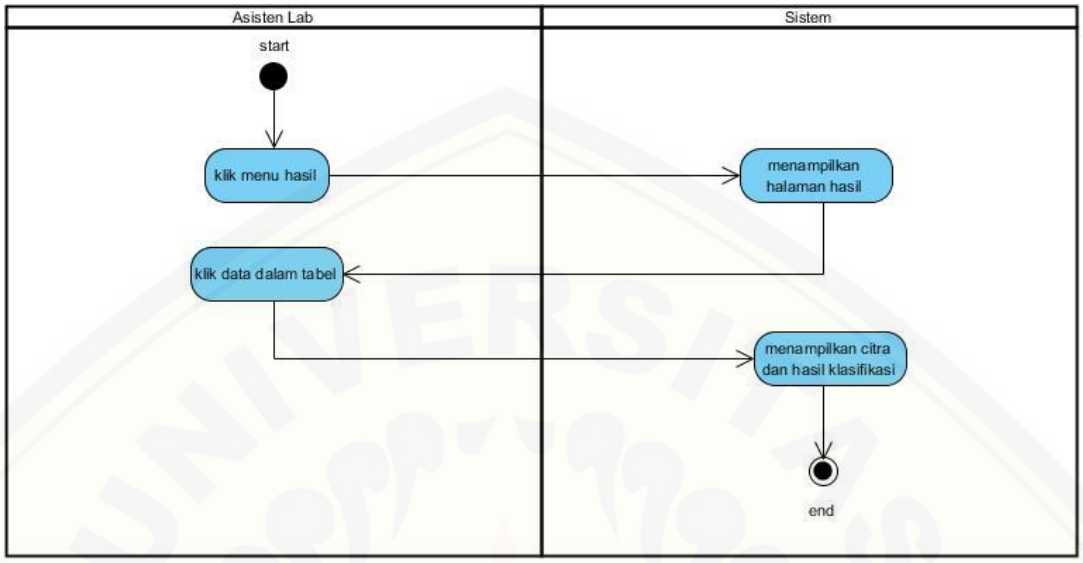
B.2 *Activity Diagram* Menginputkan *Datatrain*



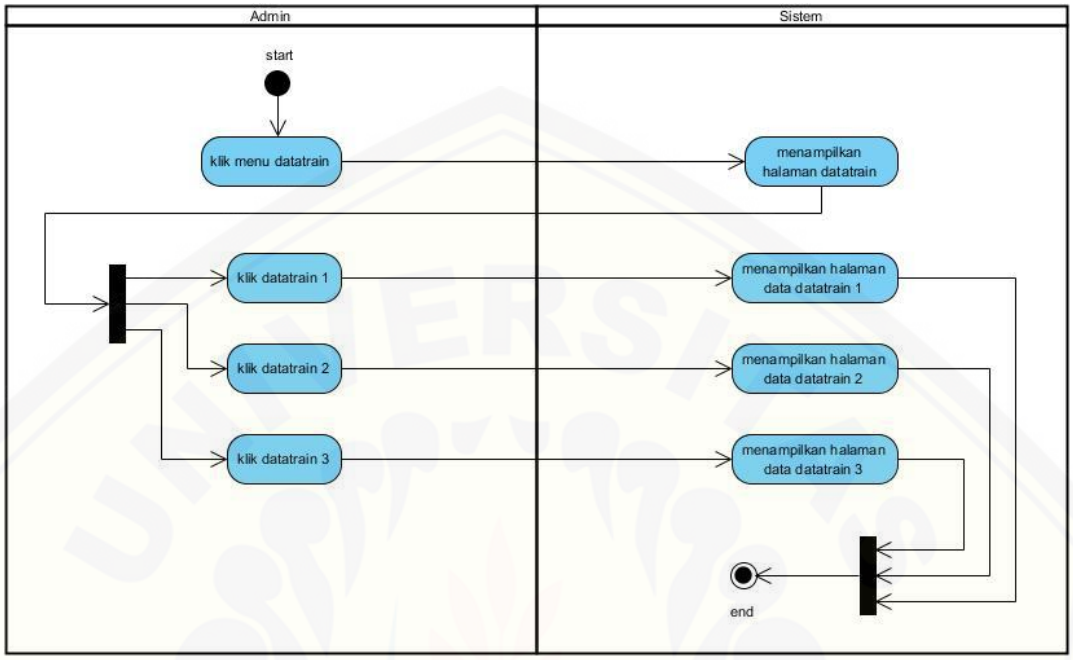
B.3 *Activity Diagram* Menguji Klasifikasi Larutan



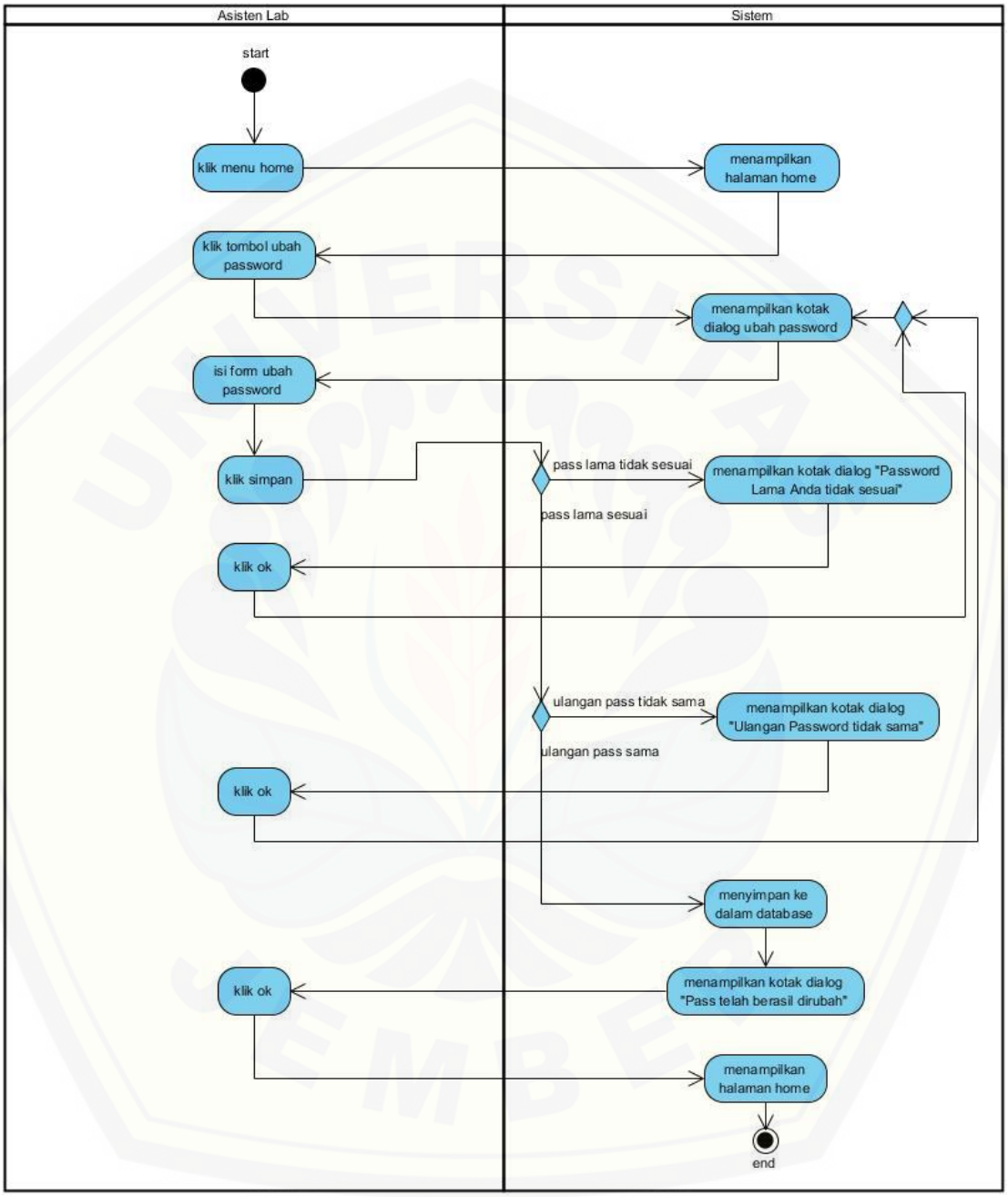
B.4 *Activity Diagram* Melihat Hasil Klasifikasi



B.5 *Activity Diagram* Melihat *Datatrain*

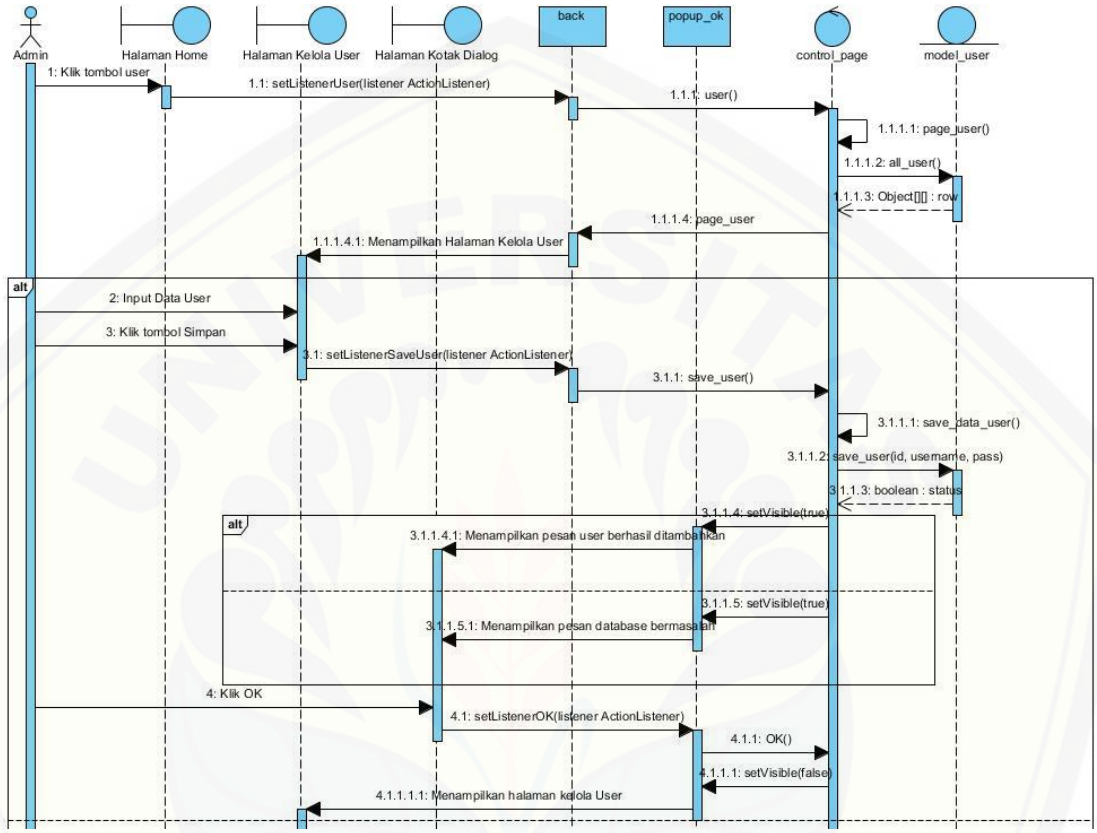


B.6 *Activity Diagram* Mengubah *Password*

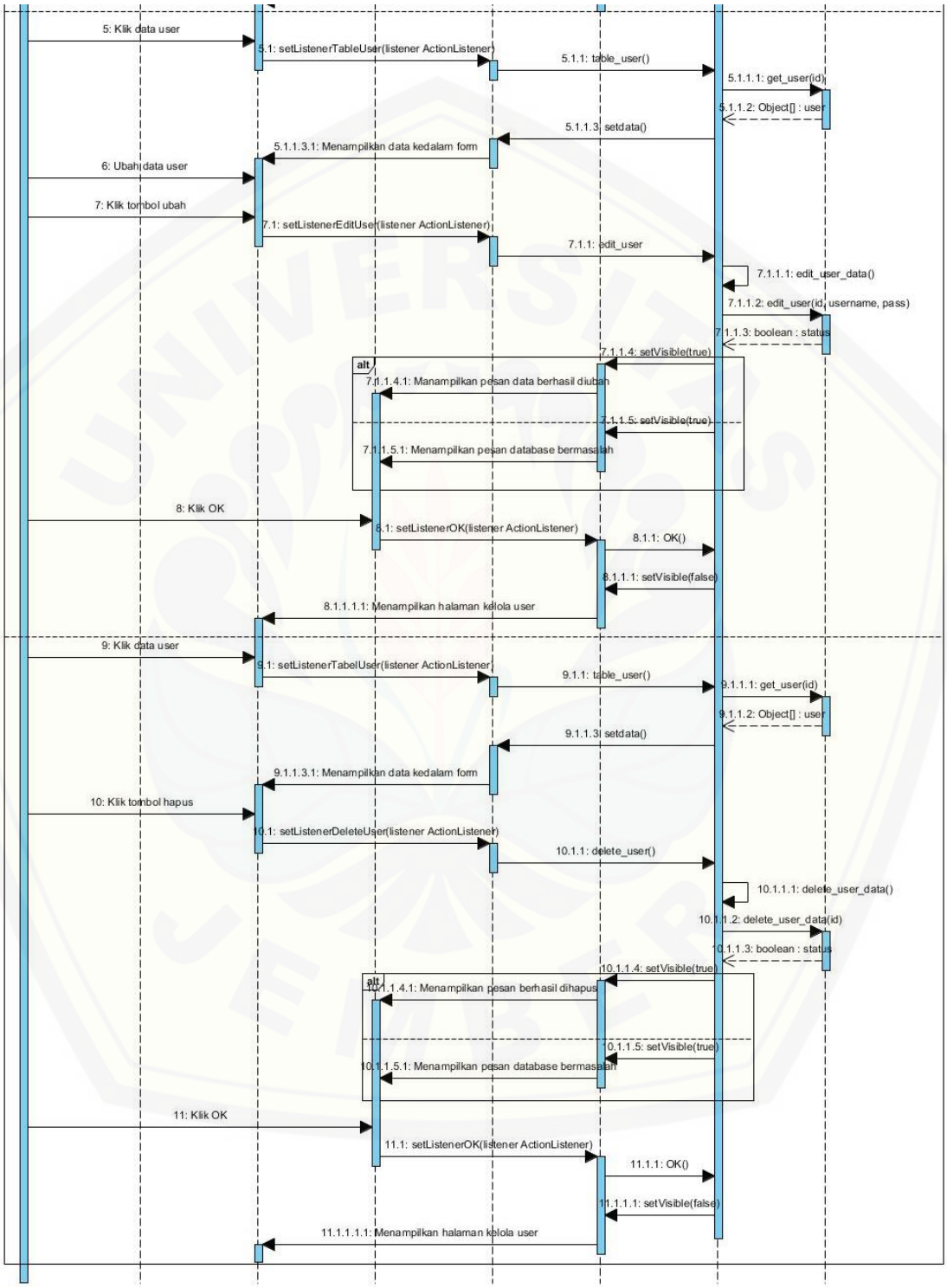


LAMPIRAN C. *SEQUENCE DIAGRAM*

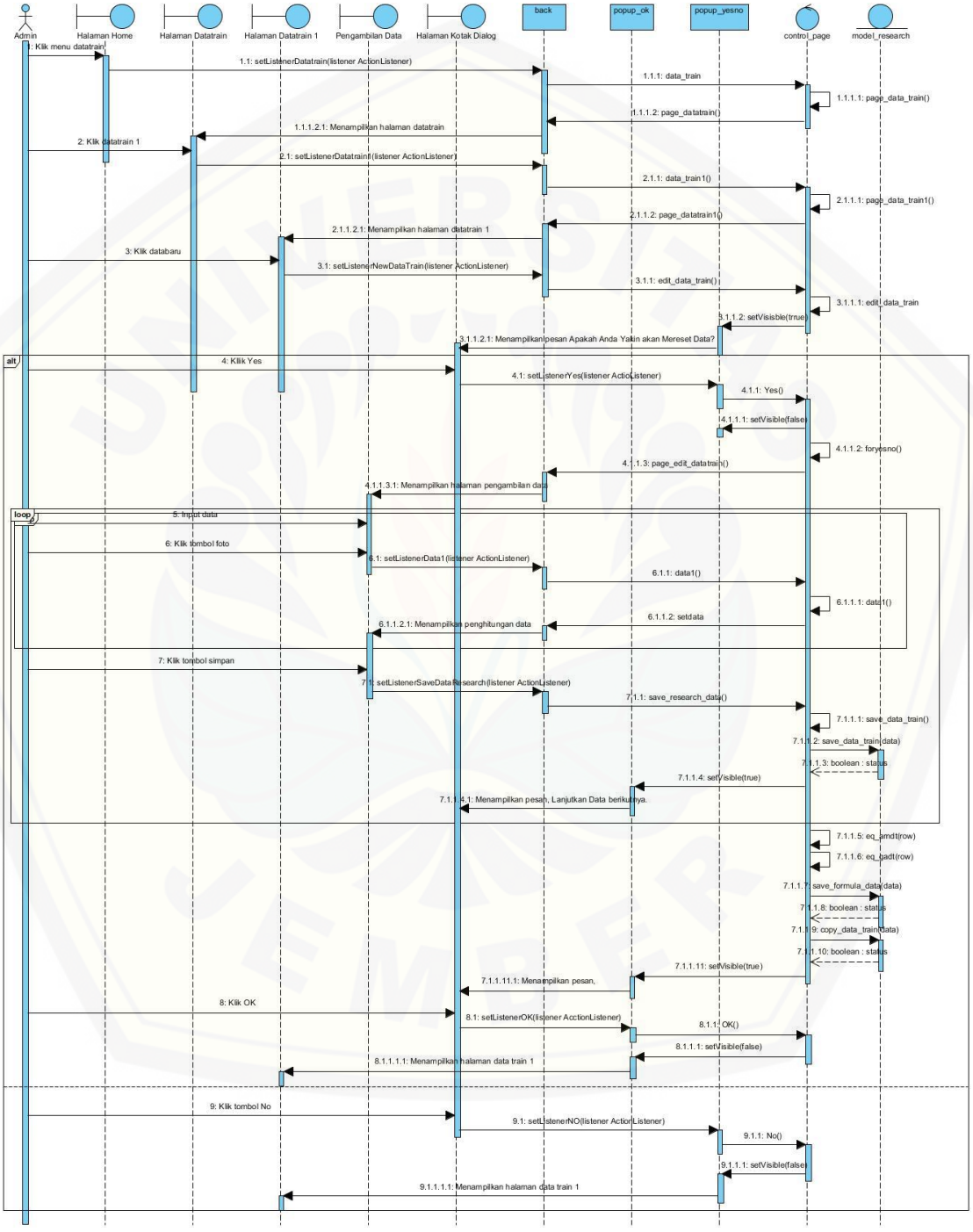
C.1a *Sequence Diagram* Mengelola Data *User*



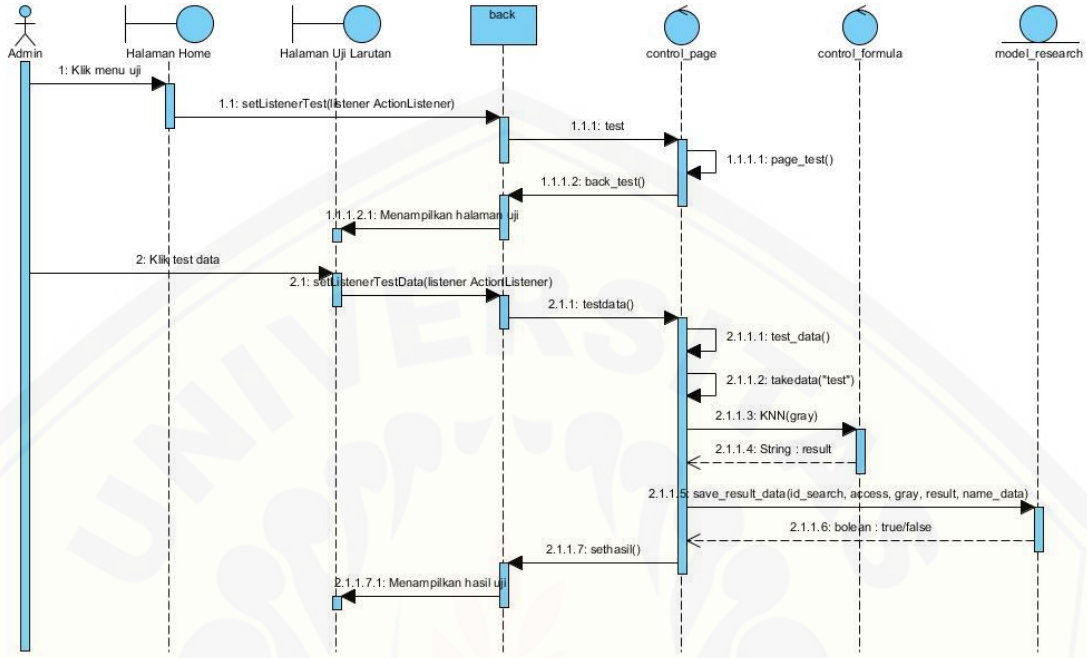
C.2a *Sequence Diagram* Mengelola Data *User*



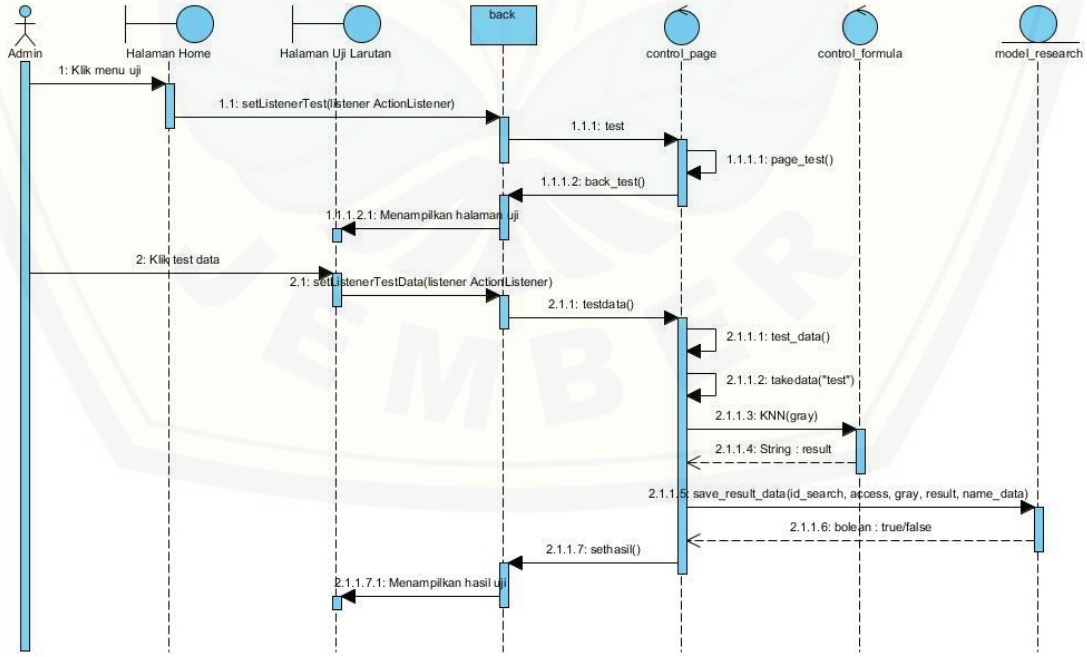
C.2 *Sequence Diagram* Menginputkan *Datatrain*



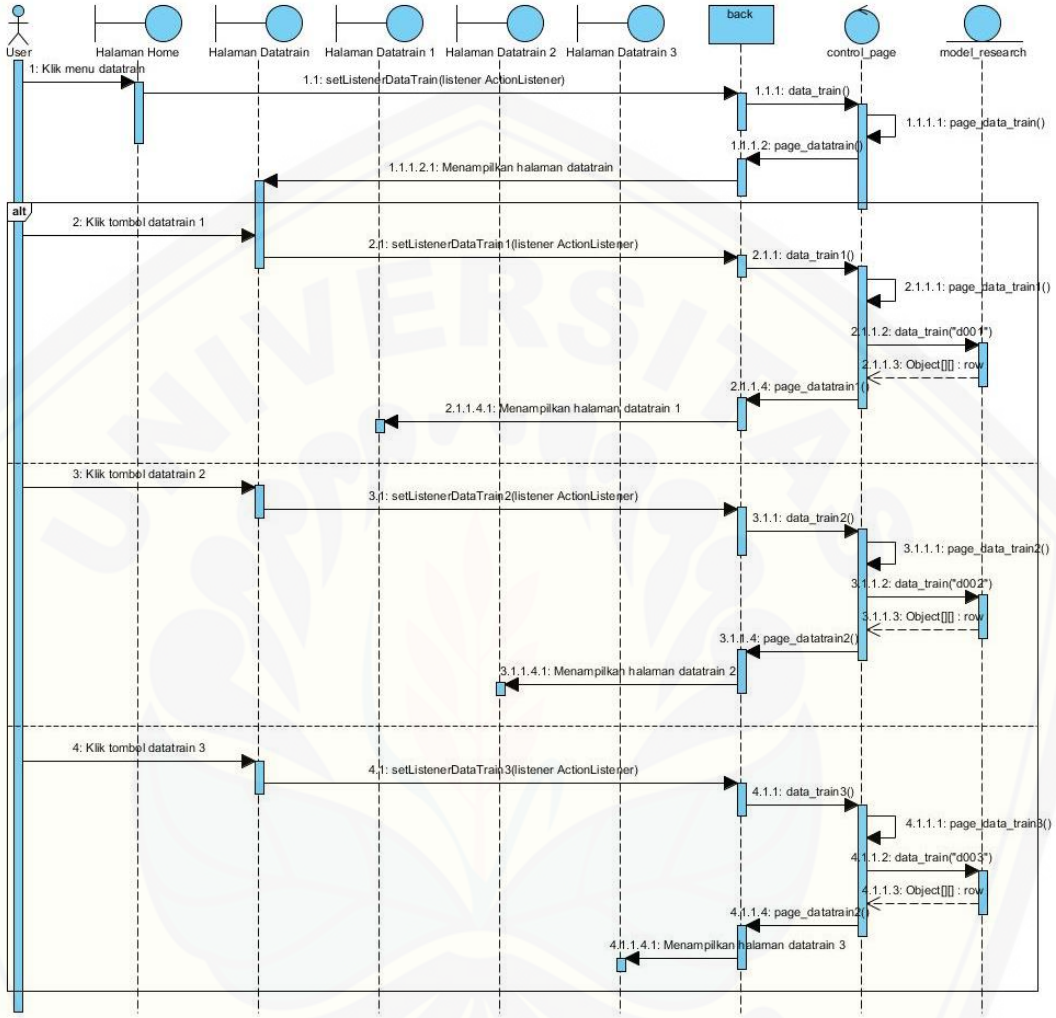
C.3 *Sequence Diagram* Menguji Klasifikasi Larutan



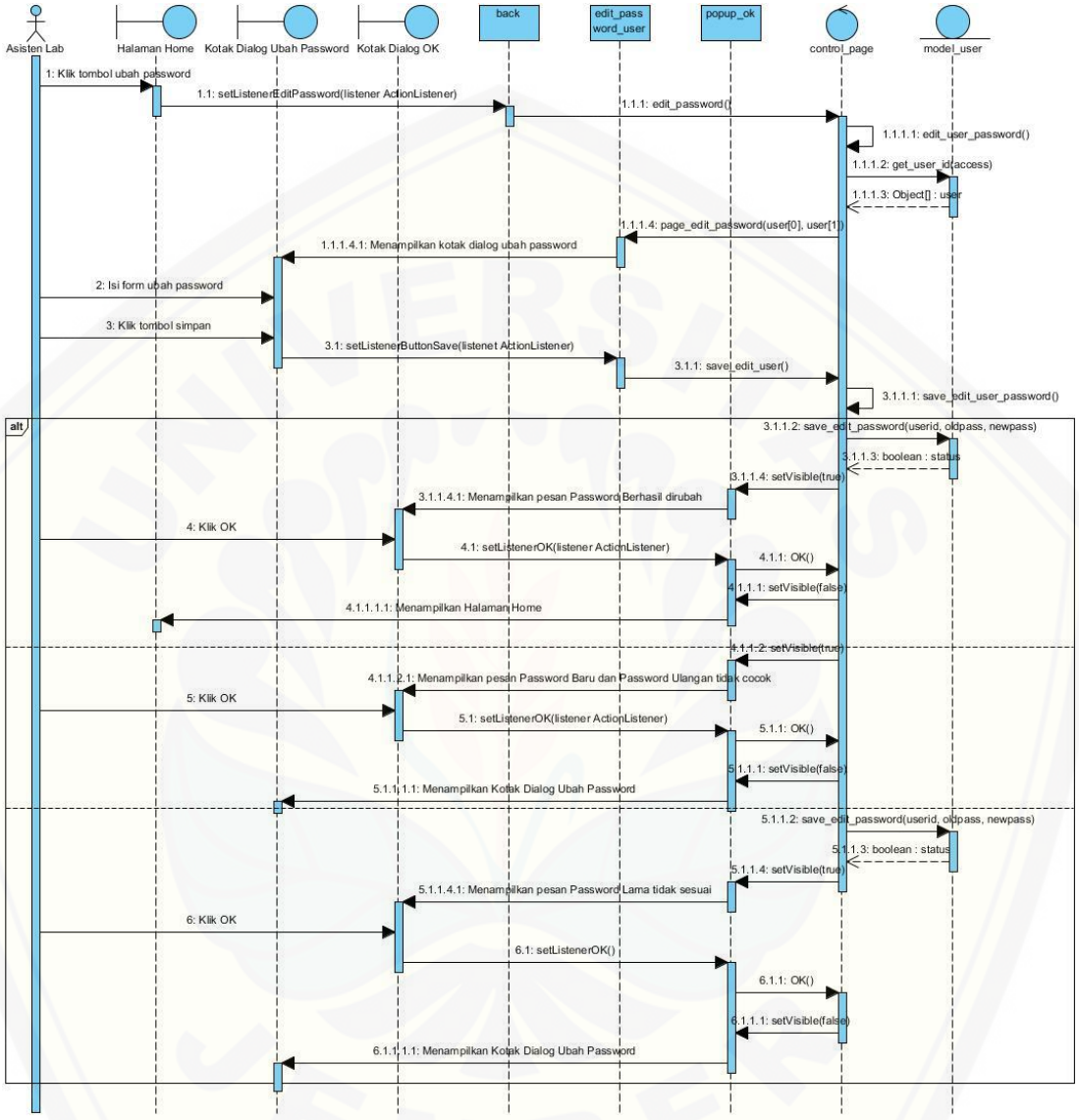
C.4 *Sequence Diagram* Melihat Hasil Klasifikasi



C.5 *Sequence Diagram* Melihat *Datatrain*



C.6 *Sequence Diagram* Mengubah *Password*



LAMPIRAN D. IMPLEMENTASI *CODING*

D.1a Tambah Data *User*

```
991     public void save_data_user() {
992         if (back.getusername().equals("") || back.getpassword().equals("")) {
993             popup_ok.setText("Silahkan tambahkan user!");
994             popup_ok.setVisible(true);
995         } else {
996             boolean status = mod_user.save_user(back.getiduser(), back.getusername(), back.getpassword());
997             if (status) {
998                 popup_ok.setText("User berhasil ditambahkan!");
999                 popup_ok.setVisible(true);
1000            } else {
1001                popup_ok.setText("Database bermasalah silahkan coba lagi nanti!");
1002                popup_ok.setVisible(true);
1003            }
1004            page_user();
1005        }
1006    }
```

D.1b Ubah Data *User*

```
1008    public void edit_user_data() {
1009        if (back.getusername().equals("") || back.getpassword().equals("")) {
1010            popup_ok.setText("Tidak Boleh ada Data Kosong!");
1011            popup_ok.setVisible(true);
1012        } else {
1013            boolean status = mod_user.edit_user(back.getiduser(), back.getusername(), back.getpassword());
1014            if (status) {
1015                popup_ok.setText("User berhasil diubah!");
1016                popup_ok.setVisible(true);
1017            } else {
1018                popup_ok.setText("Database bermasalah silahkan coba lagi nanti!");
1019                popup_ok.setVisible(true);
1020            }
1021            page_user();
1022        }
1023    }
```

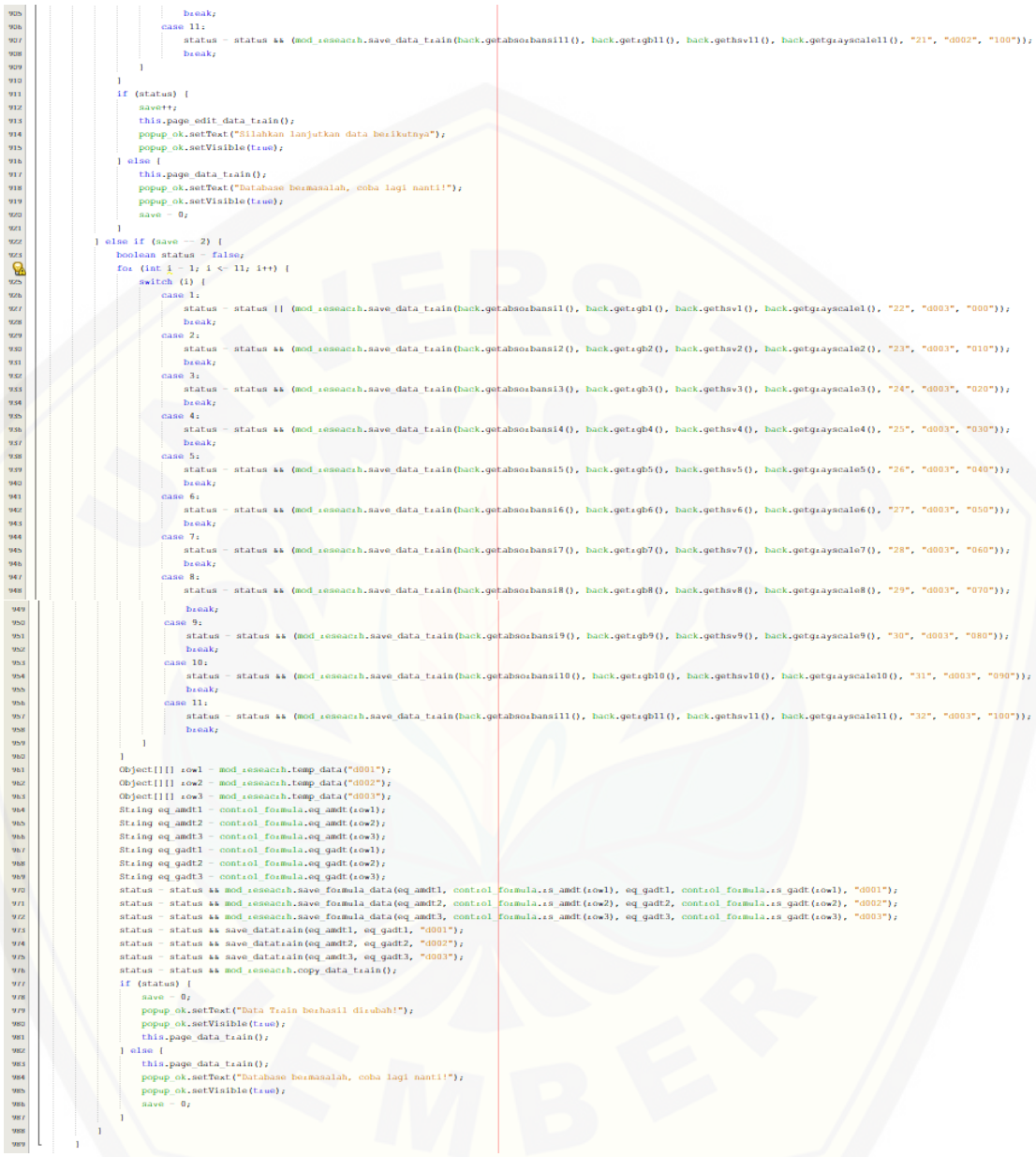
D.1c Hapus Data *User*

```
1025    public void delete_user_data() {
1026        if (back.getusername().equals("") || back.getpassword().equals("")) {
1027            popup_ok.setText("Silahkan pilih user!");
1028            popup_ok.setVisible(true);
1029        } else {
1030            popup_yesno.setText("Apakah Anda yakin akan menghapus user ini??");
1031            popup_yesno.setVisible(true);
1032            foryesno = "hapus user";
1033        }
1034    }
```

D.2a Menginputkan *Datatrain* Bagian I

```
public void save_data_train() {
    if (save == 0) {
        boolean status = false;
        for (int i = 1; i <= 11; i++) {
            switch (i) {
                case 1:
                    status = status || (mod_research.save_data_train(back.getabso:bansi1(), back.get:gb1(), back.gethsv1(), back.getg:rayscale1(), "00", "d001", "000"));
                    break;
                case 2:
                    status = status && (mod_research.save_data_train(back.getabso:bansi2(), back.get:gb2(), back.gethsv2(), back.getg:rayscale2(), "01", "d001", "010"));
                    break;
                case 3:
                    status = status && (mod_research.save_data_train(back.getabso:bansi3(), back.get:gb3(), back.gethsv3(), back.getg:rayscale3(), "02", "d001", "020"));
                    break;
                case 4:
                    status = status && (mod_research.save_data_train(back.getabso:bansi4(), back.get:gb4(), back.gethsv4(), back.getg:rayscale4(), "03", "d001", "030"));
                    break;
                case 5:
                    status = status && (mod_research.save_data_train(back.getabso:bansi5(), back.get:gb5(), back.gethsv5(), back.getg:rayscale5(), "04", "d001", "040"));
                    break;
                case 6:
                    status = status && (mod_research.save_data_train(back.getabso:bansi6(), back.get:gb6(), back.gethsv6(), back.getg:rayscale6(), "05", "d001", "050"));
                    break;
                case 7:
                    status = status && (mod_research.save_data_train(back.getabso:bansi7(), back.get:gb7(), back.gethsv7(), back.getg:rayscale7(), "06", "d001", "060"));
                    break;
                case 8:
                    status = status && (mod_research.save_data_train(back.getabso:bansi8(), back.get:gb8(), back.gethsv8(), back.getg:rayscale8(), "07", "d001", "070"));
                    break;
                case 9:
                    status = status && (mod_research.save_data_train(back.getabso:bansi9(), back.get:gb9(), back.gethsv9(), back.getg:rayscale9(), "08", "d001", "080"));
                    break;
                case 10:
                    status = status && (mod_research.save_data_train(back.getabso:bansi10(), back.get:gb10(), back.gethsv10(), back.getg:rayscale10(), "09", "d001", "090"));
                    break;
                case 11:
                    status = status && (mod_research.save_data_train(back.getabso:bansi11(), back.get:gb11(), back.gethsv11(), back.getg:rayscale11(), "10", "d001", "100"));
                    break;
            }
        }
        if (status) {
            save++;
            this.page_edit_data_train();
            popup_ok.setText("Silahkan lanjutkan data berikutnya");
            popup_ok.setVisible(true);
        } else {
            this.page_data_train();
            popup_ok.setText("Database bermasalah, coba lagi nanti!");
            popup_ok.setVisible(true);
            save = 0;
        }
    } else if (save == 1) {
        boolean status = false;
        for (int i = 1; i <= 11; i++) {
            switch (i) {
                case 1:
                    status = status || (mod_research.save_data_train(back.getabso:bansi1(), back.get:gb1(), back.gethsv1(), back.getg:rayscale1(), "11", "d002", "000"));
                    break;
                case 2:
                    status = status && (mod_research.save_data_train(back.getabso:bansi2(), back.get:gb2(), back.gethsv2(), back.getg:rayscale2(), "12", "d002", "010"));
                    break;
                case 3:
                    status = status && (mod_research.save_data_train(back.getabso:bansi3(), back.get:gb3(), back.gethsv3(), back.getg:rayscale3(), "13", "d002", "020"));
                    break;
                case 4:
                    status = status && (mod_research.save_data_train(back.getabso:bansi4(), back.get:gb4(), back.gethsv4(), back.getg:rayscale4(), "14", "d002", "030"));
                    break;
                case 5:
                    status = status && (mod_research.save_data_train(back.getabso:bansi5(), back.get:gb5(), back.gethsv5(), back.getg:rayscale5(), "15", "d002", "040"));
                    break;
                case 6:
                    status = status && (mod_research.save_data_train(back.getabso:bansi6(), back.get:gb6(), back.gethsv6(), back.getg:rayscale6(), "16", "d002", "050"));
                    break;
                case 7:
                    status = status && (mod_research.save_data_train(back.getabso:bansi7(), back.get:gb7(), back.gethsv7(), back.getg:rayscale7(), "17", "d002", "060"));
                    break;
                case 8:
                    status = status && (mod_research.save_data_train(back.getabso:bansi8(), back.get:gb8(), back.gethsv8(), back.getg:rayscale8(), "18", "d002", "070"));
                    break;
                case 9:
                    status = status && (mod_research.save_data_train(back.getabso:bansi9(), back.get:gb9(), back.gethsv9(), back.getg:rayscale9(), "19", "d002", "080"));
                    break;
                case 10:
                    status = status && (mod_research.save_data_train(back.getabso:bansi10(), back.get:gb10(), back.gethsv10(), back.getg:rayscale10(), "20", "d002", "090"));
```

D.2b Menginputkan *Datatrain* Bagian II



```
                        break;
                    case 11:
                        status = status && (mod_research.save_data_train(back.getabso:bansi11(), back.get:gb11(), back.gethsv11(), back.getg:rayscale11(), "21", "d002", "100"));
                        break;
                }
            }
            if (status) {
                save++;
                this.page_edit_data_train();
                popup_ok.setText("Silahkan lanjutkan data berikutnya");
                popup_ok.setVisible(true);
            } else {
                this.page_data_train();
                popup_ok.setText("Database bermasalah, coba lagi nanti!");
                popup_ok.setVisible(true);
                save = 0;
            }
        } else if (save == 2) {
            boolean status = false;
            for (int i = 1; i <= 11; i++) {
                switch (i) {
                    case 1:
                        status = status || (mod_research.save_data_train(back.getabso:bansi1(), back.get:gb1(), back.gethsv1(), back.getg:rayscale1(), "22", "d003", "000"));
                        break;
                    case 2:
                        status = status && (mod_research.save_data_train(back.getabso:bansi2(), back.get:gb2(), back.gethsv2(), back.getg:rayscale2(), "23", "d003", "010"));
                        break;
                    case 3:
                        status = status && (mod_research.save_data_train(back.getabso:bansi3(), back.get:gb3(), back.gethsv3(), back.getg:rayscale3(), "24", "d003", "020"));
                        break;
                    case 4:
                        status = status && (mod_research.save_data_train(back.getabso:bansi4(), back.get:gb4(), back.gethsv4(), back.getg:rayscale4(), "25", "d003", "030"));
                        break;
                    case 5:
                        status = status && (mod_research.save_data_train(back.getabso:bansi5(), back.get:gb5(), back.gethsv5(), back.getg:rayscale5(), "26", "d003", "040"));
                        break;
                    case 6:
                        status = status && (mod_research.save_data_train(back.getabso:bansi6(), back.get:gb6(), back.gethsv6(), back.getg:rayscale6(), "27", "d003", "050"));
                        break;
                    case 7:
                        status = status && (mod_research.save_data_train(back.getabso:bansi7(), back.get:gb7(), back.gethsv7(), back.getg:rayscale7(), "28", "d003", "060"));
                        break;
                    case 8:
                        status = status && (mod_research.save_data_train(back.getabso:bansi8(), back.get:gb8(), back.gethsv8(), back.getg:rayscale8(), "29", "d003", "070"));
                        break;
                    case 9:
                        status = status && (mod_research.save_data_train(back.getabso:bansi9(), back.get:gb9(), back.gethsv9(), back.getg:rayscale9(), "30", "d003", "080"));
                        break;
                    case 10:
                        status = status && (mod_research.save_data_train(back.getabso:bansi10(), back.get:gb10(), back.gethsv10(), back.getg:rayscale10(), "31", "d003", "090"));
                        break;
                    case 11:
                        status = status && (mod_research.save_data_train(back.getabso:bansi11(), back.get:gb11(), back.gethsv11(), back.getg:rayscale11(), "32", "d003", "100"));
                        break;
                }
            }
            Object[][] row1 = mod_research.temp_data("d001");
            Object[][] row2 = mod_research.temp_data("d002");
            Object[][] row3 = mod_research.temp_data("d003");
            String eq_amdt1 = control_formula.eq_amdt(row1);
            String eq_amdt2 = control_formula.eq_amdt(row2);
            String eq_amdt3 = control_formula.eq_amdt(row3);
            String eq_gadt1 = control_formula.eq_gadt(row1);
            String eq_gadt2 = control_formula.eq_gadt(row2);
            String eq_gadt3 = control_formula.eq_gadt(row3);
            status = status && mod_research.save_formula_data(eq_amdt1, control_formula.rs_amdt(row1), eq_gadt1, control_formula.rs_gadt(row1), "d001");
            status = status && mod_research.save_formula_data(eq_amdt2, control_formula.rs_amdt(row2), eq_gadt2, control_formula.rs_gadt(row2), "d002");
            status = status && mod_research.save_formula_data(eq_amdt3, control_formula.rs_amdt(row3), eq_gadt3, control_formula.rs_gadt(row3), "d003");
            status = status && save_datatrain(eq_amdt1, eq_gadt1, "d001");
            status = status && save_datatrain(eq_amdt2, eq_gadt2, "d002");
            status = status && save_datatrain(eq_amdt3, eq_gadt3, "d003");
            status = status && mod_research.copy_data_train();
            if (status) {
                save = 0;
                popup_ok.setText("Data Train berhasil diubah!");
                popup_ok.setVisible(true);
                this.page_data_train();
            } else {
                this.page_data_train();
                popup_ok.setText("Database bermasalah, coba lagi nanti!");
                popup_ok.setVisible(true);
                save = 0;
            }
        }
    }
```

D.3 Menguji Klasifikasi Larutan

```
159     public void test_data() {
160         double[] x;
161         x = this.takedata("test");
162         System.out.println("rata2 5 x ambil gambar argb : " + Math.round(x[0]) + ", " + Math.round(x[1]) +
163                 ", " + Math.round(x[2]) + ", " + Math.round(x[3]));
            double y = (Double.valueOf(Math.round(x[1])) + Double.valueOf(Math.round(x[2])) +
                    Double.valueOf(Math.round(x[3]))) / 3;
166         DecimalFormat df = new DecimalFormat("#.##");
167         double z = Double.parseDouble(df.format(y));
168         System.out.println("Grayscale : " + String.valueOf(z));
169         back.disabletestdata();
170         back.setgraytest(String.valueOf(z));
171         String result = control_formula.KNN(z);
172         String id_search = mod_reseacrh.get_primarykey_search();
173
174         if (mod_reseacrh.save_result_data(id_search, access, String.valueOf(z), result, "data" + id_search)) {
175             back.setResultPhoto("src/gambar_pencarian/data" + id_search + ".png");
176         } else {
177             popup_ok.setText("Database bermasalah, coba lagi nanti!");
178             popup_ok.setVisible(true);
179         }
180
181         back.sethasil(mod_reseacrh.data_result_class(result));
182     }
```

D.4 Melihat Hasil Klasifikasi Larutan

```
182     public void page_result() {
183         String[] column = {"No", "Nama Peneliti", "Waktu", "Grayscale", "Konsentrasi"};
184         Object[][] row = mod_reseacrh.search_data();
185         DefaultTableModel search = new DefaultTableModel(row, column);
186         back.setTableSearch(search);
187
188         back.page_result();
189     }
```

D.5a Melihat *Datatrain* 1

```
317     public void page_data_train1() {
318         if (!access.equals("admin")) {
319             back.setVisibleNewDataTrain1(false);
320         }
321         Object[][] row = mod_reseacrh.begin_data("d001");
322         Object[][] formula = mod_reseacrh.formula_data("d001");
323         String[] column1 = {"Konsentrasi (mg/ml)", "Absorbansi", "Grayscale"};
324         Object[][] row1 = mod_reseacrh.data_train("d001");
325         DefaultTableModel datatrain1 = new DefaultTableModel(row1, column1);
326         back.setTableDataTrain1(datatrain1);
327 
328         String[] formula_am = String.valueOf(formula[0][0]).split(",");
329         String[] formula_ga = String.valueOf(formula[0][1]).split(",");
330         double a = Double.valueOf(formula_am[0]);
331         double b = Double.valueOf(formula_am[1]);
332         double aa = Double.valueOf(formula_ga[0]);
333         double bb = Double.valueOf(formula_ga[1]);
334         back.seteaml(formula_am[0], formula_am[1]);
335         back.setrsaml(String.valueOf(formula[0][2]));
336         back.setegal(formula_ga[0], formula_ga[1]);
337         back.setrsgal(String.valueOf(formula[0][3]));
338 
339         //grafik absorbansi mgml
340         XYDataset dataset = control_chart.DatasetRegAbsorbansiMgml(row, a, b);
341         JFreeChart chart = ChartFactory.createXYLineChart("Konsentrasi x Absorbansi", "Kosentrasi Glukosa",
342                 "Absorbansi", dataset, PlotOrientation.VERTICAL, true, true, true);
343         ChartPanel chartpanel = new ChartPanel(chart);
344         chartpanel.setPreferredSize(new Dimension(350, 350));
345 
346         //grafik rgb absorbansi
347         XYDataset dataset2 = control_chart.DatasetRegGrayAbsorbansi(row, aa, bb);
348         JFreeChart chart2 = ChartFactory.createXYLineChart("Absorbansi x Grayscale", "Grayscale",
349                 "Absorbansi", dataset2, PlotOrientation.HORIZONTAL, true, true, true);
350         ChartPanel chartpanel2 = new ChartPanel(chart2);
351         chartpanel2.setPreferredSize(new Dimension(350, 350));
352 
353         back.page_datatrain1(chartpanel, chartpanel2);
354 
355     }
```

D.5b Melihat *Datatrain* 2

```
357     public void page_data_train2() {
358         if (!access.equals("admin")) {
359             back.setVisibleNewDataTrain2(false);
360         }
361         Object[][] row = mod_reseacrh.begin_data("d002");
362         Object[][] formula = mod_reseacrh.formula_data("d002");
363         String[] column1 = {"Konsentrasi (mg/ml)", "Absorbansi", "Grayscale"};
364         Object[][] row1 = mod_reseacrh.data_train("d002");
365         DefaultTableModel datatrain2 = new DefaultTableModel(row1, column1);
366         back.setTableDataTrain2(datatrain2);
367 
368         String[] formula_am = String.valueOf(formula[0][0]).split(",");
369         String[] formula_ga = String.valueOf(formula[0][1]).split(",");
370         double a = Double.valueOf(formula_am[0]);
371         double b = Double.valueOf(formula_am[1]);
372         double aa = Double.valueOf(formula_ga[0]);
373         double bb = Double.valueOf(formula_ga[1]);
374         back.seteam2(formula_am[0], formula_am[1]);
375         back.setrsam2(String.valueOf(formula[0][2]));
376         back.setega2(formula_ga[0], formula_ga[1]);
377         back.setrsga2(String.valueOf(formula[0][3]));
378 
379         //grafik absorbansi mgml
380         XYDataset dataset = control_chart.DatasetRegAbsorbansiMgml(row, a, b);
381         JFreeChart chart = ChartFactory.createXYLineChart("Konsentrasi x Absorbansi", "Kosentrasi Glukosa",
382                 "Absorbansi", dataset, PlotOrientation.VERTICAL, true, true, true);
383         ChartPanel chartpanel = new ChartPanel(chart);
384         chartpanel.setPreferredSize(new Dimension(350, 350));
385 
```

D.5c Melihat *Datatrain* 3

```
386             //grafik rgb absorbansi
387             XYDataset dataset2 = control_chart.DatasetRegGrayAbsorbansi(row, aa, bb);
388             JFreeChart chart2 = ChartFactory.createXYLineChart("Absorbansi x Grayscale", "Grayscale",
389                     "Absorbansi", dataset2, PlotOrientation.HORIZONTAL, true, true, true);
390             ChartPanel chartpanel2 = new ChartPanel(chart2);
391             chartpanel2.setPreferredSize(new Dimension(350, 350));
392 
393             back.page_datatrain2(chartpanel, chartpanel2);
394 
395         }
```

```
397         public void page_data_train3() {
398             if (!access.equals("admin")) {
399                 back.setVisibleNewDataTrain3(false);
400             }
401             Object[][] row = mod_reseacrh.begin_data("d003");
402             Object[][] formula = mod_reseacrh.formula_data("d003");
403             String[] column1 = {"Konsentrasi (mg/ml)", "Absorbansi", "Grayscale"};
404             Object[][] row1 = mod_reseacrh.data_train("d003");
405             DefaultTableModel datatrain3 = new DefaultTableModel(row1, column1);
406             back.setTableDataTrain3(datatrain3);
407 
408             String[] formula_am = String.valueOf(formula[0][0]).split(",");
409             String[] formula_ga = String.valueOf(formula[0][1]).split(",");
410             double a = Double.valueOf(formula_am[0]);
411             double b = Double.valueOf(formula_am[1]);
412             double aa = Double.valueOf(formula_ga[0]);
413             double bb = Double.valueOf(formula_ga[1]);
414             back.seteam3(formula_am[0], formula_am[1]);
415             back.setrsam3(String.valueOf(formula[0][2]));
416             back.setega3(formula_ga[0], formula_ga[1]);
417             back.setrsga3(String.valueOf(formula[0][3]));
418 
419             //grafik absorbansi mgml
420             XYDataset dataset = control_chart.DatasetRegAbsorbansiMgml(row, a, b);
421             JFreeChart chart = ChartFactory.createXYLineChart("Konsentrasi x Absorbansi", "Kosentrasi Glukosa",
422                     "Absorbansi", dataset, PlotOrientation.VERTICAL, true, true, true);
423             ChartPanel chartpanel = new ChartPanel(chart);
424             chartpanel.setPreferredSize(new Dimension(350, 350));
425 
426             //grafik rgb absorbansi
427             XYDataset dataset2 = control_chart.DatasetRegGrayAbsorbansi(row, aa, bb);
428             JFreeChart chart2 = ChartFactory.createXYLineChart("Absorbansi x Grayscale", "Grayscale",
429                     "Absorbansi", dataset2, PlotOrientation.HORIZONTAL, true, true, true);
430             ChartPanel chartpanel2 = new ChartPanel(chart2);
431             chartpanel2.setPreferredSize(new Dimension(350, 350));
432 
433             back.page_datatrain3(chartpanel, chartpanel2);
434 
435         }
```

D.6 Mengubah *Password*

```
1041    public void save_edit_user_password() {
1042        if (edit_password_user.getNewPassR().equals(edit_password_user.getNewPass())) {
1043            if (mod_user.save_edit_password(edit_password_user.getUserId(),
1044                    edit_password_user.getOldPass(), edit_password_user.getNewPass())) {
1045                edit_password_user.dispose();
1046                popup_ok.setText("Password berhasil dirubah!");
1047                popup_ok.setVisible(true);
1048            } else {
1049                popup_ok.setText("Password lama tidak sesuai!");
1050                popup_ok.setVisible(true);
1051            }
1052        } else {
1053            popup_ok.setText("Ulangan Password tidak sama!");
1054            popup_ok.setVisible(true);
1055        }
1056    }
```

LAMPIRAN E. *BLACK BOX TESTING*

No.	Menu	Fungsi	Kasus	Hasil	Keterangan
1.	Login	Menu ini berfungsi sebagai keamanan sistem. memilah *user* yang dapat menggunakan sistem ini.	Ketika *user* memasukan nama *user* dan password dengan benar	Menampilkan halaman sesuai level *user*	Berhasil
2	Home	Menampilkan tata cara penggunaan aplikasi	Ketika *user* telah melakukan login, sistem akan menampilkan halaman home yang berisi tata cara penggunaan aplikasi	Menampilkan halaman yang berisi tata cara penggunaan aplikasi	Berhasil
3.	Mengelola data *user*	Untuk tambah data *user*, ubah data *user*, dan hapus data *user*	Ketika Admin klik menu *users* menampilkan halaman kelola data *user*	Menampilkan halaman kelola data *user*	Berhasil
			Ketika Admin menambah dan klik tombol simpan	Menyimpan data *user*	Berhasil
			Ketika Admin merubah data dan klik tombol ubah	Merubah data *user*	Berhasil
			Ketika Admin klik tombol hapus	Menghapus data *user*	Berhasil
4.	Menginputkan *datatrain*	Melakukan Input *Datatrain*	Ketika Admin menginputkan *datatrain* dan klik tombol simpan	Menyimpan *datatrain*	Berhasil
5.	Menguji klasifikasi larutan	Melakukan uji klasifikasi larutan	Ketika *user* menguji larutan dan klik test data	Menampilkan hasil pengujian	Berhasil

No.	Menu	Fungsi	Kasus	Hasil	Keterangan
6.	Melihat hasil Klasifikasi	Menampilkan hasil klasifikasi	Ketika *user* klik menu hasil klasifikasi	Menampilkan hasil klasifikasi	Berhasil
7.	Melihat *Datatrain*	Menampilkan *datatrain*	Ketika *user* klik menu *datatrain*	Menampilkan halaman *datatrain*	Berhasil
8.	Mengubah *Password*	Untuk merubah *password* Asisten Lab	Ketika Asisten Lab merubah data dan klik ubah data	Menyimpan perubahan *password*	Berhasil