PENGEMBANGAN SISTEM PRESENSI MAHASISWA BERBASIS FACE RECOGNITION DENGAN MENGGUNAKAN ALGORITMA EIGENFACE

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

Noviana Putri Pradnyawati

NIM. 0710683019

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

PENGEMBANGAN SISTEM PRESENSI MAHASISWA BERBASIS FACE RECOGNITION DENGAN MENGGUNAKAN ALGORITMA EIGENFACE

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

Noviana Putri Pradnyawati

NIM. 0710683019

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Himawat Aryadita, ST, MSc	**Eriq M. Adams J, ST, MKom**
NIP. 19801018 200801 1 003	**NIK. 850410 06 1 1 0027**

LEMBAR PENGESAHAN

PENGEMBANGAN SISTEM PRESENSI MAHASISWA BERBASIS FACE RECOGNITION DENGAN MENGGUNAKAN ALGORITMA EIGENFACE

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

Noviana Putri Pradnyawati

NIM. 0710683019

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 8 April 2013

Penguji I	Penguji II
Drs. Marji., M.T.	**Budi Darma Setiawan, S.Kom.,M.Cs**
NIP. 19670801 199203 1 001	**NIK. 841015 06 1 1 0090**

Penguji III

Imam Cholissodin, S.Si.,M.Kom
NIK. 850719 16 1 1 0286

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji., M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Maret 2013
Mahasiswa,

Noviana Putri Pradnyawati
NIM 0710683019

KATA PENGANTAR

Puji syukur kehadirat Allah SWT karena atas limpahan rahmat dan hidayahNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "Pengembangan Sistem Presensi Mahasiswa Berbasis Face Recognition Dengan Menggunakan Algoritma Eigenface". Skripsi ini disusun untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Dalam menyelesaikan skripsi ini, banyak bantuan, bimbingan, dan dorongan yang diterima oleh penulis. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Ibunda Siti Asiyah, Almarhum Ayahanda Susiono, Adik Ainurrizky Putri Robbitasari dan Rizal Akbar Fauzani, serta seluruh keluarga besar atas doa dan semangat demi terselesaikannya skripsi ini.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
4. Bapak Himawat Aryadita, S.T, M.Sc dan Bapak Eriq Muhammad Adams J, S.T, M.Kom selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
5. Bapak Himawat Aryadita, S.T, M.Sc selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
7. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi

di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

8. Faiza Alif Fakhrina, Arif Mudjahidah, Ratih Kartika Dewi, Adam Hendra Brata, Wisnu Aditya dan Deta Pratama yang telah banyak memberikan bantuan dan dukungan dalam menyelesaikan skripsi ini.
9. Sahabat Legendary TPL07, terimakasih atas segala bantuannya selama menempuh studi di Teknik Informatika Universitas Brawijaya.
10. Teman – teman kos Watugong No. 1 yang telah memberi semangat dan menghibur dalam menyelesaikan skripsi ini.
11. Agung Budi Utomo yang selalu memberi motivasi dalam pengerjaan skripsi ini.
12. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga skripsi ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 18 Maret 2013

Penulis

ABSTRAK

Noviana Putri Pradnyawati. 2013. : Pengembangan Sistem Presensi Mahasiswa Berbasis Face Recognition Dengan Menggunakan Algoritma Eigenface. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.
Dosen Pembimbing : Himawat Aryadita, S.T, M.Sc dan Eriq M. Adams J, ST, M.Kom

Pengenalan wajah telah dikembangkan dan menjadi alternatif dalam berbagai aplikasi yang membutuhkan identifikasi seseorang, karena wajah merupakan bagian langsung dari tubuh manusia yang tidak mudah dicuri atau diduplikasi. Salah satu aplikasi yang dibangun dengan menggunakan wajah sebagai *identifier*-nya adalah sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*. *Eigenface* merupakan salah satu algoritma pengenalan wajah yang berdasar pada *Principal Component Analysis* (PCA).

Implementasi sistem presensi mahasiswa ini menggunakan bahasa pemrograman java. Pada saat pengujian unit, disimpulkan bahwa unit modul dari sistem sudah memenuhi kebutuhan fungsional yang telah dirancang. Pada pengujian validasi dapat disimpulkan bahwa implementasi dan fungsionalitas sistem presensi mahasiswa telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan. Berdasarkan hasil pengujian kinerja sistem, tingkat akurasi sistem dalam melakukan pengenalan wajah saat presensi menggunakan *input* pose wajah standar menghadap ke depan memiliki tingkat akurasi 100% dengan rata – rata waktu yang diperlukan saat proses presensi adalah 1005.467 ms. Untuk *input* dengan kondisi wajah menghadap ke samping kanan memiliki tingkat akurasi 80%, *error rate* 20% dan rata – rata waktu yang diperlukan 1542.267 ms. Sedangkan untuk *input* dengan kondisi wajah menghadap ke samping kiri memiliki tingkat akurasi 73.3%, *error rate* 26.7% dan rata – rata waktu yang diperlukan saat proses presensi adalah 1730.2 ms.

Kata Kunci : Sistem Presensi Mahasiswa, *Face Recognition, Eigenface.*

ABSTRACT

Noviana Putri Pradnyawati. 2013. : *Student Presence System Development Based On Face Recognition Using Eigenface Algorithm.*
***Advisor* : Himawat Aryadita, S.T, M.Sc *and* Eriq M. Adams J, ST, M.Kom**

Face recognition has been developed and become an alternative in many applications that require a person's identification, because the face is a direct part of the human body that are not easily to be stolen or duplicated. One of the applications that built using face as the identifier is student presence system using face recognition algorithm Eigenface. Eigenface is a face recognition algorithm based on Principal Component Analysis (PCA).

Implementation of this student presence system using Java programming language. Unit testing, show that the unit module and the integration module meets the functional requirements that have been designed. In validation testing can be concluded that the implementation and functionality of this student presence system meets the needs that have been described in the requirement analysis phase. Based on the results of performance system testing, the level of accuracy system in performing face recognition when using standard poses facing forward has an accuracy rate of 100% with an average of face recognition 1005.467 ms. For the input with conditions face to the right side has an accuracy rate of 80%, error rate 20% and an average of face recognition 1542,267 ms. For the input with the condition face to the left side has an accuracy rate of 73.3%, error rate 26.7% and an average of face recognition 1730.2 ms.

Keywords *: Student Presence System, Face Recognition, Eigenface.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Wajah merupakan salah satu *identifier* yang digunakan secara luas sebagai kunci pembeda dan identitas personal yang memiliki keunikan atau ciri-ciri khusus. Hal ini menyebabkan pengenalan wajah (*face recognition*) terus dikembangkan dan menjadi alternatif dalam berbagai aplikasi yang membutuhkan identifikasi seseorang, karena wajah merupakan bagian langsung dari tubuh manusia yang tidak mudah dicuri atau diduplikasi seperti halnya metode dengan menggunakan tanda tangan, *password* maupun kartu. Salah satu aplikasi yang dibangun dengan menggunakan wajah sebagai *identifier*-nya adalah aplikasi sistem presensi.

Sistem presensi saat ini telah berkembang dalam berbagai jenis, diantaranya sistem presensi secara manual, *barcode*, sidik jari dan pengenalan wajah, dimana setiap sistem presensi memiliki kelemahan dan kelebihan tersendiri. Kelemahan sistem presensi secara manual dan *barcode* dapat terjadi saat mahasiswa yang tidak hadir menitipkan absen dengan memalsukan tanda tangan atau menitipkan kartu kepada mahasiswa lainnya. Hal ini menyebabkan tingkat keakuratan dan keaslian absensi menjadi tidak terjamin. Selain itu, pada kegiatan administrasinya sistem presensi secara manual juga memerlukan waktu yang panjang, terutama dalam proses perhitungan rekap absensi sehingga menimbulkan ketidakefektifan. Oleh karena itu, perlu adanya alternatif sistem presensi mahasiswa baru yang mampu menutupi kekurangan dari sistem sebelumnya.

Salah satu solusinya adalah dengan mengembangkan sistem presensi mahasiswa sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*. *Eigenface* merupakan salah satu algoritma pengenalan wajah yang berdasar pada *Principal Component Analysis* (PCA) untuk mereduksi citra berdimensi tinggi menjadi berdimensi rendah yang akan memaksimalkan jarak antara semua citra wajah dalam database. Berdasarkan hasil penelitian yang dilakukan Turk dan Pentland, selain memiliki kompleksitas

komputasi yang cukup sederhana, algoritma ini menunjukkan hasil yang cukup optimal dalam pengenalan wajah manusia, yaitu 96% untuk gambar dengan variasi pencahayaan, 85% untuk gambar wajah dengan berbagai variasi orientasi dan 64% untuk gambar wajah dengan ukuran yang bervariasi [TUR-91:10].

Sistem presensi ini di implementasikan dengan menggunakan kamera digital atau *webcam* sebagai media *user interface*. Proses kerja dari *user interface* menggunakan *webcam* yaitu dengan menangkap gambar berupa video yang kemudian hasil tangkapan *webcam* tersebut akan diproses dan diolah sebagai masukan dari sistem presensi mahasiswa ini. Karena masukan yang digunakan adalah tangkapan wajah secara langsung dengan *webcam*, maka tingkat kecurangan bukti kehadiran mahasiswa dapat dikurangi. Selain itu, rekapitulasi yang terdapat pada sistem presensi ini diharapkan dapat mempersingkat waktu dalam proses perhitungan rekap absensi.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka permasalahan yang akan dibahas adalah sebagai berikut:

1. Bagaimana mengembangkan sebuah sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*?
2. Bagaimana menguji fungsionalitas dan non fungsionalitas sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*?

1.3 Tujuan

Tujuan penyusunan skripsi ini adalah mengembangkan sebuah sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*.

1.4 Batasan masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, maka aspek yang dibahas berkaitan dengan penelitian ini antara lain:

1. Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Menggunakan Algoritma *Eigenface* menerima masukan berupa citra wajah

yang di-*capture* menggunakan *webcam* pada saat proses presensi dilakukan.

2. Resolusi webcam yang digunakan untuk mengambil citra wajah yang digunakan sebagai *training image* maupun pada saat proses presensi dilakukan adalah 1,3 Mpx.
3. Citra masukan wajah yang digunakan sebagai *training image* memiliki format *.png*

1.5 Manfaat

Manfaat penelitian ini antara lain:

1. Bagi penulis
    1. Mengaplikasikan ilmu yang didapat selama mengikuti perkuliahan di Teknik Informatika dan Ilmu Komputer (PTIIK) Universitas Brawijaya.
    2. Mendapatkan pemahaman tentang pengembangan sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*.
b. Bagi pengguna
    1. Meningkatkan keakuratan, keefektifan dan kemudahan dalam kegiatan presensi maupun administrasi presensi mahasiswa.

1.6 Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam skripsi ini adalah sebagai berikut:

BAB I Pendahuluan

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

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam pengembangan dan perancangan aplikasi sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface*.

BAB III Metodologi Penelitian

Membahas metode yang digunakan dalam penulisan, yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian dan analisis.

BAB IV Analisis Kebutuhan dan Perancangan

Membahas analisis kebutuhan dari aplikasi sistem presensi mahasiswa berbasis *face recognition* dengan menggunakan algoritma *eigenface* dan kemudian merancang hal-hal yang berhunbungan dengan analisa tersebut.

BAB V Implementasi dan Pembahasan

Membahas tentang hasil perancangan dari analisis kebutuhan dan implementasi dari sistem aplikasi.

BAB VI Pengujian

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VII Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Pada bab ini berisi tentang pembahasan dan teori tentang perangkat lunak yang akan dilakukan. Teori yang akan dibahas dalam bab ini meliputi penjelasan tentang pengenalan wajah, *eigenface*, pengembangan perangkat lunak, dan pengujian yang akan dilakukan.

2.1 Pengenalan wajah

Pengenalan wajah merupakan bagian pengenalan pola yang dilakukan secara khusus pada wajah. Pengenalan wajah telah menjadi alternatif solusi dalam berbagai bidang yang membutuhkan identifikasi seseorang, dimana wajah merupakan bagian dari identifikasi *biometric* karena merupakan bagian langsung dari tubuh manusia yang tidak mudah untuk dicuri atau diduplikasi seperti halnya metode konvensional yang menggunakan *password* ataupun kartu. Sistem pengenalan wajah digunakan untuk membandingkan satu citra wajah masukan dengan suatu *database* wajah dan menghasilkan wajah yang paling cocok dengan citra tersebut jika ada.

Dalam visi komputer terdapat 3 macam pendekatan dalam melakukan ekstraksi ciri pada citra wajah, yaitu ekstraksi ciri berbasis *appearance*, *feature* dan *hybrid*. Pada pendekatan berbasis *appearance*, semua area wajah digunakan sebagai data input untuk ekstraksi fitur, sebagai contoh adalah PCA, *Linear Descriminant Analysis* (LDA), *Independent Component Analysis* (ICA), Kernel ICA, Kernel PCA, *Laplacianfaces*, Kernel *Fisherface*, Kernel *Direct Descriminant Analysis* (KDDA), DCT dan DST [PUR-10:220].

Pada pendekatan berbasis *feature*, fitur lokal seperti kelengkungan wajah, alis mata, mata, hidung, bibir dan ciri yang ada pada wajah digunakan sebagai data input. Jika lokasi fitur yang diinginkan sudah ditemukan, maka kebanyakan menggunakan parameter geometris untuk mencirikan fitur wajah. Beberapa contoh yang menggunakan metode ini adalah ASM, *Hidden Markov Model* dan *Maximum Line Gradient* [PUR-10:220].

Pada pendekatan terakhir yaitu pendekatan berbasis *hybrid*, merupakan gabungan pendekatan berbasis *appearance* dan *feature*. Pendekatan tersebut berasal dari sistem visi pada manusia. Pada pendekatan ini ada Modular *Eigenface*s, *Hybrid Local Feature*, *Shape Normalized* dan *Component Based Method* [PUR-10:220].

2.2 Eigenface

Teknik *eigenface* yang pertama kali dikemukakan oleh Turk dan Pentland pada tahun 1991 sangat handal untuk menyelesaikan masalah pengenalan wajah. Untuk menghasilkan *eigenface*, sekumpulan citra digital dari wajah manusia diambil pada kondisi pencahayaan yang sama kemudian dinormalisasikan dan diproses pada resolusi yang sama (misal *m* x *n*), kemudian citra tadi diperlakukan sebagai vektor dimensi *m* x *n* dimana komponennya diambil dari nilai piksel citra. Komputasi algoritma *eigenface* dijelaskan sebagai berikut [TUR-91:2]:

1. Menyiapkan gambar wajah (*training faces*) $I_1, I_2, ..., I_M$ dan usahakan gambar wajah berada ditengah serta memiliki ukuran yang sama.
2. Representasikan tiap gambar I_i sebagai vektor Γ_i
3. Hitung rata-rata vektor wajah (Ψ):

   $$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

4. Cari selisih (Φ) antara *training face* Γ_i dengan nilai tengah (Ψ):

   $$\Phi_i = \Gamma_i - \Psi$$

5. Hitung nilai *matrix* kovarian (C):

   $$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \qquad (N^2 \times N^2 \ matrix)$$

   Dimana $A = [\Phi_1, \Phi_2, ..., \Phi_M]$ $\qquad (N^2 \times M \ matrix)$

6. Hitung *eigenvector* u_i pada AA^T

Karena *matrix* AA^T sangat besar, maka asumsikan *matrix* $A^T A$ ($M \times M$ *matrix*)

Hitung *eigenvector* v_i pada $A^T A$

$$A^T A v_i = \mu_i v_i$$

Hubungan antara u_i dengan v_i :

$$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow$$

$$CA v_i = \mu_i A v_i \text{ atau } C u_i = \mu_i u_i \text{ dimana } u_i = A v_i$$

AA^T dan $A^T A$ memiliki *eigenvalue* yang sama dan *eigenvector* nya berhubungan : $u_i = A v_i$

Note 1 : AA^T dapat memiliki hingga N^2 *eigenvalue* dan *eigenvector*.

Note 2 : $A^T A$ dapat memiliki hingga M *eigenvalue* dan *eigenvector*.

Note 3 : *eigenvalue* M pada $A^T A$ (selama berhubungan dengan *eigenvector*) dihubungkan dengan *eigenvalue* terbesar M pada AA^T (selama berhubungan dengan *eigenvector*).

Hitung *eigenvector* terbaik M pada AA^T : $u_i = A v_i$, normalisasi u_i seperti $\|u_i\| = 1$

7. Ambil K *eigenvector* (sesuai dengan nilai *eigen* atau *eigenvalues* terbesar K).

2.2.1 Menghasilkan *Eigenface*

Eigenface dibangun menggunakan teknik matematika yang disebut *Principal Components Analysis* (PCA). Langkah – langkah yang dilakukan untuk melakukan PCA pada suatu kumpulan data, yaitu [SMI-02:12] :

1. Mengumpulkan data

   Misalkan dimiliki dua data set numerik x dan y yang disebut sebagai data awal seperti pada Tabel 2.1.

Tabel 2.1 Data awal x dan y

x	y
2.5	2.4
0.5	0.7
2.2	2.9

1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Sumber: [SMI-02:13]

2. Kurangi dengan *Mean*

Hitung *mean* dari masing – masing dimensi data, dengan mengurangi nilai masing – masing elemen data dengan *mean* dari dimensinya sendiri. Dari tabel 2.1 didapatkan nilai $\bar{x} = 1.81$ dan $\bar{y} = 1.91$. Sehingga nilai masing – masing elemen data x dan y setelah dikurangi dengan $\bar{x}$ dan $\bar{y}$ ditampilkan pada Tabel 2.2.

Tabel 2.2 Data x dan y Setelah Dikurangi $\bar{x}$ dan $\bar{y}$

x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

Sumber: [SMI-02:13]

3. Hitung *Covariance matrix*

Untuk membandingkan dua data set (yaitu antara data x dan y) digunakan *covariance matrix* (C).

$$\operatorname{cov}(x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

n merupakan jumlah dari elemen data (10 menurut tabel 2.1). Karena data yang digunakan berupa data 2 dimensi, maka C akan berbentuk matriks 2 dimensi.

$$\begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) \\ \text{cov}(y,x) & \text{cov}(y,y) \end{pmatrix}$$

Sehingga x dan y akan menjadi *covariance matrix* 2x2 :

$$\begin{pmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{pmatrix}$$

Karena didapatkan hasil *covariance* positif, artinya kedua data set sama – sama meningkat. Apabila ada yang bernilai negatif artinya satu data set meningkat, sedangkan yang lainnya berkurang.

4. Hitung *Eigenvector* dan *Eigenvalue* dari *Covariance matrix*

Karena *covariance matrix* (C) berupa *square matrix* 2 x 2, maka dapat dihasilkan 2 *eigenvalue* dan 2 *eigenvector* . *eigenvalue* dan *eigenvector* dari C adalah :

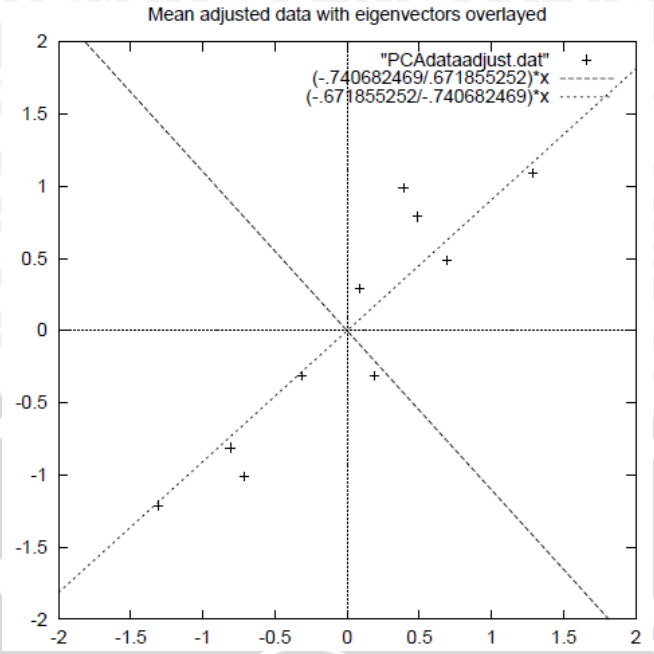
$$eigenvalue = \begin{pmatrix} 0.0490833989 \\ 1.2840277100 \end{pmatrix}$$

$$eigenvector = \begin{pmatrix} -0.735178656 & -0.677873399 \\ 0.67787399 & -0.735178656 \end{pmatrix}$$

Eigenvector yang diperoleh masing masing memiliki panjang satu. Bila data data x dan y yang telah dikurangi *mean* diatas digambarkan bersama dengan *eigenvector* yang diperoleh, maka akan diperoleh plot data seperti pada gambar 2.1. Pada gambar 2.1, dua *eigenvector* yang diperoleh dari C juga digambarkan pada plot data berupa garis putus – putus. Dari *eigenvector* yang diperoleh, membentuk dua buah garis diagonal pada plot.

Salah satu *eigenvector* tergambarkan tepat di tengah – tengah plot data, *eigenvector* ini menggambarkan bagaimana kedua data set memiliki hubungan sepanjang garis tersebut. Dari pencarian *covariance matrix* dan *eigenvector* ini, dapat dilakukan transformasi data menjadi hanya sebuah garis vektor.

5. Memilih Komponen dan Membentuk *Feature Vector*



Gambar 2.1 Plot data yang telah di normalisasi (dikurangi *mean*) dengan *eigenvector* dari *covariance matrix* yang di dapatkan sebelumnya.
Sumber: [SMI-02:15]

Dari *eigenvalue* yang didapat, diperoleh informasi bahwa *eigenvector* yang melintas di tengah – tengah data memiliki *eigenvalue* terbesar. *Eigenvector* ini disebut dengan nama principal component dari data. Di langkah ini dilakukan pengurutan *eigenvector* berdasarkan besar *eigenvalue* yang dimiliki masing – masing *eigenvector* secara menurun. Sehingga *eigenvector* dengan *eigenvalue* terbesar di urutan teratas.

Singkatnya, jika dimiliki data dengan dimensi *n*, maka dapat dibentuk *covariance matrix* berdimensi *n*. lalu dari *covariance matrix* yang diperoleh, dapat dihasilkan *n* buah *eigenvector* dan *eigenvalue*. Lalu dipilih *p* buah *eigenvector* yang memiliki *eigenvalue* terbesar yang selanjutnya disebut *Feature Vector*.

Dari *eigenvector* yang diperoleh pada tahap 4, kedua *eigenvector* dapat digunakan sebagai *Feature Vector* atau hanya menggunakan *eigenvector* yang memiliki *eigenvalue* terbesar sebagai *Feature Vector*. Sehingga *eigenvector* yang dipilih sebagai *Feature Vector* adalah :

$$\begin{pmatrix} -0.677873399 \\ -0.735178656 \end{pmatrix}$$

6. Kumpulan Data Baru

Dari *Feature Vector* yang diperoleh, dapat ditentukan kmpulan data baru dengan cara mengalikan *vector transpose* dari *Feature Vector* dengan *matrix transpose* dari data normal. Dibawah ini merupakan kumpulan data hasil perkalian dengan hanya satu *eigenvector* yang paling signifikan sebagai *Feature Vector* sehingga menyebabkan data normalisasi terkompresi menjadi data satu dimensi saja, seperti pada tabel 2.3.

Tabel 2.3 Data hasil perkalian dengan hanya satu *eigenvector* yang paling signifikan.

x
-0.827970186
1.77758033
-0.992197494
-0.274210416
-1.67580142
-0.912949103
0.0991094375
1,14457216
0.438046137
1.22382056

Sumber: [SMI-02:19]

2.3 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan [SOM-11:7]. Pada definisi ini, ada dua istilah kunci :

1. *'disiplin rekayasa'*

   Perekayasa membuat suatu alat bekerja. Perekayasa menerapkan teori, metode, dan alat bantu yang sesuai. Perekayasa menggunakannya dengan selektif dan selalu mencoba mencari solusi terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung. Perekayasa juga menyadari bahwa mereka harus bekerja dalam batasan organisasi dan keuangan, sehingga mereka berusaha mencari solusi dalam batasan – batasan ini.

2. *'semua aspek produksi perangkat lunak'*

Rekayasa perangkat lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak tetapi juga dengan kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode dan teori untuk mendukung produksi perangkat lunak.

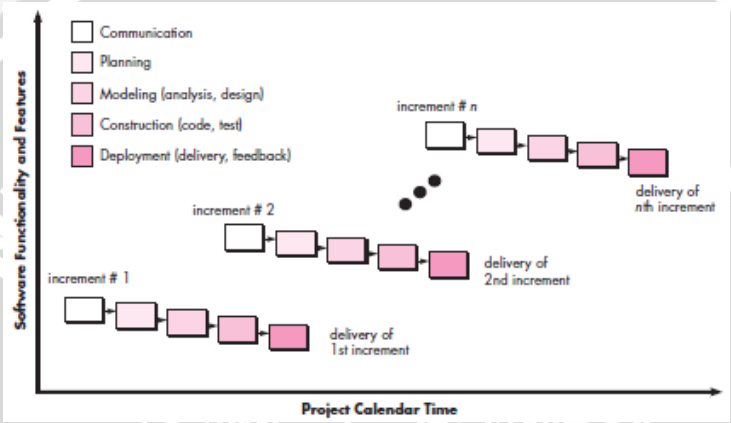
Perekayasa perangkat lunak memakai pendekatan yang sistematis dan terorganisir terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Rekayasa ini mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekatan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan [SOM-11:7].

Seorang *software engineer* atau sekumpulan *software engineer* harus menggabungkan strategi pengembangan perangkat lunak yang meliputi proses, metode, dan alat bantu yang digunakan dalam proses pengembangan. Strategi ini disebut sebagai model proses (*process model*) atau paradigma rekayasa perangkat lunak (*software engineering paradigm*). Model proses untuk rekayasa perangkat lunak dipilih sesuai dengan sifat dari proyek dan aplikasi yang akan dibuat. Terdapat beberapa model proses untuk rekayasa perangkat lunak antara lain *linear sequential model* (*waterfall*), *prototyping, Rapid Application Development* (RAD), *incremental model* dan *spiral model* [PRE-10:31-48]. Model proses yang digunakan dalam skripsi ini adalah *incremental model*.

Incremental model merupakan gabungan dari elemen *linear sequensial model* dan filosofi dari *prototyping* sehingga memudahkan seorang pengembang (*developer*) dalam mengidentifikasi kebutuhan pengguna [PRE-10:41]. *Incremental model* merupakan metodologi pengembangan perangkat lunak yang efisien dan efektif untuk *project* dengan skala kecil hingga skala menengah. *Incremental model* menawarkan strategi pengembangan perangkat lunak yang memungkinkan pengguna (*user*) dapat menggunakan versi awal (*increment* pertama) sebagai *prototype* untuk memperoleh informasi tentang persyaratan kebutuhan (*requirement*) mereka untuk pengembangan sistem selanjutnya [SOM-11:47]. Kelebihan dari model proses ini antara lain adalah proses *development* yang lebih cepat, lebih mudah dalam mengetahui kebutuhan pengguna, dan *resource* yang dibutuhkan untuk melakukan perubahan terhadap perangkat lunak

yang dikembangkan lebih sedikit [SOM-11:33]. Proses pengembangan menggunakan model proses *incremental* ini dapat dilihat pada Gambar 2.2.

Versi awal dari perangkat lunak (*increment* 1) dievaluasi oleh pengguna sehingga pihak pengembang mengetahui dengan jelas apa yang dibutuhkan oleh pengguna. Pihak pengembang kemudian mengembangkan perangkat lunak untuk *increment* selanjutnya sesuai dengan saran dan permintaan pengguna. Proses ini berulang sampai perangkat lunak selesai atau mencapai tahap yang diinginkan oleh pengguna [PRE-10:41].



Gambar 2.2 Incremental model
Sumber : [PRE-10:42]

2.4 Pengujian Perangkat Lunak

Pengujian (*testing*) arsitektur dari perangkat lunak berorientasi objek menghasilkan sekumpulan *layered subsystems* yang mengenkapsulasi kelas-kelas yang berkolaborasi. Setiap elemen sistem (subsistem dan *class*) melakukan fungsi yang membantu untuk mencapai kebutuhan sistem. Hal ini sangat penting untuk menguji sebuah *object-oriented system* pada berbagai macam level yang berbeda dalam sebuah usaha untuk menemukan kesalahan-kesalahan yang mungkin terjadi dari kolaborasi kelas-kelas dan komunikasi subsistem melewati *architetural layer* [PRE-10:631].

2.4.1 Teknik Pengujian

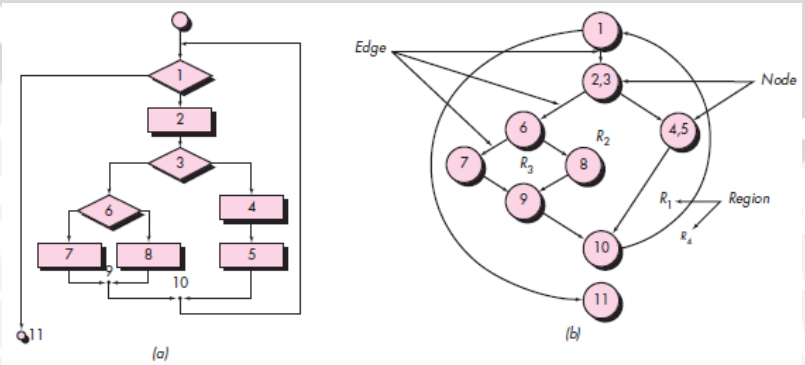
Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Berbagai macam metode perancangan kasus uji telah berevolusi. Metode-metode ini menyediakan *developer* pendekatan sistematis untuk pengujian. Terlebih lagi metode-metode ini menyediakan mekanisme yang dapat membantu memastikan

kelengkapan dari pengujian dan menyediakan kemungkinan tertinggi untuk menemukan kesalahan-kesalahan dalam perangkat lunak [PRE-10:484]. Teknik atau metode perancangan kasus uji yang digunakan adalah *black-box testing* dan *white-box testing*.

1. *White-Box Testing*

White-box testing atau *glass-box testing* merupakan sebuah metode perancangan kasus uji yang menggunakan struktur kontrol dari perancangan prosedural untuk memperoleh kasus uji [PRE-10:485]. Ada dua jenis pengujian yang termasuk *white-box testing* yaitu *basis path testing* dan *control structure testing.*

Pada skripsi ini menggunakan *basis path testing* yang diusulkan pertama kali oleh Tom McCabe [PRE-10:485]. *Basis path testing* ini memungkinkan perancang kasus uji memperoleh ukuran kompleksitas logis dari sebuah perancangan prosedural dan menggunakan pengukuran ini sebagai pedoman untuk mendefinisikan *basis set* dari jalur eksekusi (*execution path*). *Test case* yang dilakukan untuk menggunakan *basis set* tersebut dijamin untuk menggunakan setiap *statement* di dalam program paling tidak sekali selama pengujian. Sebelum metode *basis path* dapat diperkenalkan, notasi sederhana untuk representasi aliran kontrol yang disebut diagram alir (*flow graph*) harus diperkenalkan. Setiap representasi desain prosedural yang berupa *flow chart* dapat diterjemahkan ke dalam *flow graph*. Gambar 2.3 menunjukkan transformasi *flow chart* ke *flow graph.* Setelah *flow graph* didefinisikan maka harus ditentukan ukuran kompleksitas (*cyclomatic complexity*).



Gambar 2.3 Transformasi *flow chart* ke *flow graph*
Sumber : [PRE-10:486]

Cyclomatic complexity adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian *basis path*, maka nilai yang terhitung untuk *cyclomatic complexity* menentukan jumlah jalur independen (*independent path*) dalam *basis set* suatu program dan memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua *statement* telah dieksekusi sedikitnya satu kali.

Jalur independen adalah jalur yang melalui program yang mengenalkan sedikitnya satu rangkaian *statement* proses baru atau suatu kondisi baru. Untuk menentukan *cyclomatic complexity* bisa dilakukan dengan beberapa cara, diantaranya [PRE-10:488]:

1. Jumlah *region* pada *flow graph* sesuai dengan *cyclomatic complexity*.
2. *Cyclomatic complexity* V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node*.
3. V(G) = P + 1, dimana P adalah jumlah *predicate node* yaitu *node* yang merupakan kondisi (ada 2 atau lebih *edge* akan keluar *node* ini).

2. *Black-Box Testing*

Black-box testing atau *behavioral testing* berfokus pada persyaratan fungsional perangkat lunak [PRE-10:495]. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk semua program. Pengujian *black-box* bukan merupakan alternatif dari teknik *white-box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white-box*.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori berikut :

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan kinerja

5. Inisialisasi dan kesalahan terminasi.

Tidak seperti pengujian *white-box*, yang dilakukan pada saat awal proses pengujian, pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi. Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut :

- Bagaimana validitas fungsional diuji ?
- Kelas input apa yang akan membuat *test case* menjadi baik ?
- Apakah sistem sangat sensitif terhadap harga input tertentu ?
- Bagaimana batasan dari suatu data diisolasi ?
- Kecepatan dan volume data apa yang dapat ditolerir oleh sistem ?
- Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem ?

2.4.2 Strategi Pengujian

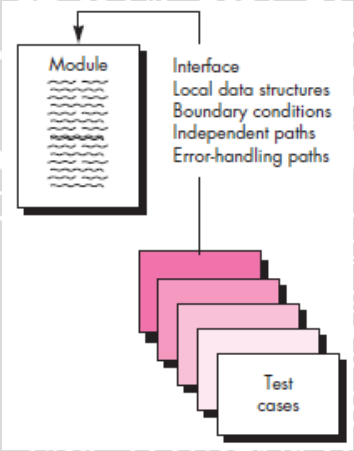
Strategi untuk pengujian perangkat lunak mengintegrasikan metode desain *test case* perangkat lunak ke dalam sederetan langkah yang direncanakan dengan baik, dan hasilnya adalah konstruksi perangkat lunak yang berhasil [PRE-10:477]. Sejumlah strategi pengujian perangkat lunak telah diusulkan di dalam literatur. Strategi pengujian harus mengakomodasi pengujian tingkat rendah yang diperlukan untuk membuktikan bahwa segmen kode sumber yang kecil telah diimplementasikan dengan tepat, demikian juga pengujian tingkat tinggi yang memvalidasi fungsi-fungsi sistem mayor yang berlawanan dengan kebutuhan pelanggan. Proses pengujian dimulai dengan pengujian yang berfokus pada setiap modul secara individual (*unit testing*), dilanjutkan dengan pengujian integrasi (*integration testing*) dan berakhir pada pengujian validasi (*validation testing*) [PRE-10:481].

1. Pengujian Unit

Pengujian unit berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak, yakni modul. Dengan menggunakan gambaran desain prosedural sebagai panduan, jalur kontrol yang penting diuji untuk mengungkap kesalahan di dalam batas modul tersebut. Kompleksitas

relatif dari pengujian dan kesalahan yang diungkap dibatasi oleh ruang lingkup batasan yag dibangun untuk pengujian unit. Pengujian unit biasanya berorientasi pada *white-box*, dan langkahnya dapat dilakukan secara paralel untuk model bertingkat [PRE-10:457].

Pengujian yang terjadi sebagai bagian dari inti digambarkan secara skematis pada Gambar 2.4 *Interface* modul diuji untuk memastikan bahwa informasi secara tepat mengalir masuk dan keluar dari inti program yang diuji. Struktur data lokal diuji untuk memastikan bahwa data yang tersimpan secara temporal dapat tetap menjaga integritasnya selama semua langkah di dalam suatu algoritma dieksekusi. Kondisi batas diuji untuk memastikan bahwa modul beroperasi dengan tepat pada batas yang ditentukan untuk membatasi pemrosesan. Semua jalur independen (jalur dasar) yang melalui struktur kontrol dipakai sedikitnya satu kali. Dan akhirnya, penanganan kesalahan uji [PRE-10:457].



Gambar 2.4 Pengujian unit
Sumber : [PRE-10:457]

2. Pengujian Validasi

Pada kulminasi pengujian terintegrasi, perangkat lunak secara lengkap dirakit sebagai suatu paket, kesalahan *interfacing* telah diungkap dan dikoreksi, dan seri akhir dari pengujian perangkat lunak, yaitu pengujian validasi dapat dimulai. Validasi dapat ditentukan dengan berbagai cara, tetapi definisi yang sederhana adalah bahwa validasi berhasil bila perangkat lunak berfungsi dengan cara yang dapat diharapkan secara bertanggung jawab oleh pelanggan. Validasi perangkat lunak

dicapai melalui sederetan pengujian *black-box* yang memperlihatkan konformitas dengan persyaratan. Rencana pengujian menguraikan kelas-kelas pengujian yang akan dilakukan, dan prosedur pengujian menentukan *test case* spesifik yang akan digunakan untuk mengungkap kesalahan dalam konformitas dengan persyaratan. Baik rencana dan prosedur didesain untuk memastikan apakah semua persyaratan fungsional dipenuhi; semua persyaratan kinerja dicapai; dokumentasi benar dan direkayasa oleh manusia: dan persyaratan lainnya dipenuhi (transportabilitas, kompatibilitas, pembetulan kesalahan, maintanabilitas) [PRE-10:468].

2.5 Classification

Klasifikasi merupakan suatu tugas mempelajari suatu fungsi target *f* yang memetakan masing-masing atribut *x* ke dalam suatu kelas yang bernama *y* yang telah didefinisikan sebelumnya. Data masukan untuk pengklasifikasian berupa kumpulan *record*. Masing-masing *record* diketahui sebagai suatu contoh, yang digolongkan oleh *(x,y)*, dimana *x* merupakan atribut set dan *y* merupakan nama kelas [NIN-06:146].

Teknik klasifikasi merupakan pendekatan yang sistematis dalam membuat model pengklasifikasian dari masukan data set. Misalnya *decision tree classfiers, rule based clasifiers, neural networks, support vector machines*, dan *naive bayes classifiers.* Masing-masing teknik tersebut menggunakan algoritma pembelajaran untuk mengidentifikasi model hubungan yang cocok antara kumpulan atribut dengan nama kelas dari data masukan. Model yang dihasilkan oleh algoritma pembelajaran harus sesuai dengan data masukan dan secara benar memprediksi nama kelas dari *record* yang tidak pernah diketahui sebelumnya. Oleh karena itu, tujuan dari algoritma pembelajaran adalah untuk membangun suatu model dengan kemampuan *generalization* yang bagus, yaitu model yang secara akurat memprediksi nama kelas yang tidak diketahui *record*-nya sebelumnya [NIN-06:148]. Untuk memecahkan masalah pengklasifikasian, maka dapat dipecahkan dengan menyediakan kumpulan data pelatihan (*training set*) yang terdiri atas *record* yang telah diketahui nama kelasnya. Data pelatihan tersebut digunakan untuk membuat model pengklasifikasian yang kemudian digunakan oleh

kumpulan data tes, yang terdiri atas *record* yang tidak diketahui nama kelasnya [NIN-06:149].

Pengukuran performa dari model pengklasifikasian didasarkan pada jumlah data yang diprediksi benar dan jumlah yang diprediksi salah oleh model. Jumlah tersebut disusun kedalam suatu tabel yang disebut sebagai *confusion matrix* [NIN-06:149].

Tabel 2.4 *Confusion Matrix* Untuk Permasalahan *2-Class*

		Predicted Class	
		Class = 1	***Class = 0***
Actual Class	***Class = 1***	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Sumber: [NIN-06:149]

Tabel 2.4 menggambarkan *confusion matrix* untuk dua masalah pengklasifikasian. Masing-masing masukan f_{ij} pada tabel merupakan jumlah *record* dari kelas *i* yang diprediksi menjadi kelas *j*. Sebagai contoh, f_{01} merupakan jumlah dari *record* kelas 0 yang secara salah diprediksi sebagai kelas 1. Berdasarkan jumlah masukan dalam *confusion matrix*, jumlah total prediksi benar yang dibuat oleh model adalah (f_{11}+f_{00}) dan jumlah total yang diprediksi salah yaitu (f_{10}+f_{01}) [NIN-06:149].

Meskipun *confusion matrix* menyediakan informasi yang dibutuhkan untuk menentukan bagaimana performa model pengklasifikasian, ringkasan informasi dengan suatu angka dapat membuat lebih tepat untuk membandingkan performa dari beberapa model.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Performa dari suatu model dapat dinyatakan dalam hubungan dari tingkat *error*-nya.

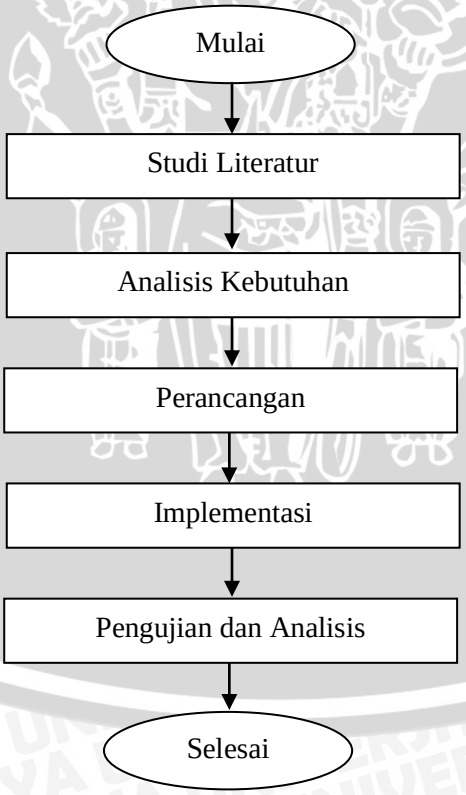
$$Error\ rate = \frac{Number\ of\ wrong\ predictions}{Total\ number\ of\ predictions} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari sistem presensi mahasiswa berbasis *face recognition* dengan algoritma *eigenface*. Metodologi penelitian yang dilakukan melibatkan beberapa langkah berikut:

1. Studi Literatur
2. Analisis Kebutuhan
3. Perancangan
4. Implementasi
5. Pengujian dan analisis

Keterkaitan antar langkah-langkah tersebut dapat dilihat pada gambar 3.1. Setiap langkah ini akan dijelaskan pada sub bab berikutnya.



Gambar 3.1 Diagram Alir Metodologi Penelitian
Sumber : Perancangan

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

1. Pengenalan wajah
2. *Eigenface*
    a. *Face Recognition* dengan *Eigenfaces*
3. Rekayasa Perangkat Lunak
    a. *Incremental Model*
4. Pengujian Perangkat Lunak
    a. Teknik Pengujian
        i. *White Box Testing*
        ii. *Black Box Testing*
    b. Strategi Pengujian
        i. Pengujian Unit
        ii. Pengujian Validasi
5. *Classification*

3.2 Analisis Kebutuhan

Kegiatan analisis kebutuhan perangkat lunak meliputi analisis spesifikasi perangkat lunak. Metode analisis menggunakan bahasa pemodelan UML *(Unified Modeling Language). Use Case Diagram* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem dari perspektif *user*. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem yang kemudian akan dimodelkan dalam diagram *use case.* Kebutuhan fungsional yang nantinya akan disediakan oleh aplikasi ini antara lain adalah :

1. Masukan pada sistem presensi adalah citra wajah mahasiswa yang diambil menggunakan *webcam* pada saat presensi dilakukan.
2. Sistem mencatat waktu presensi mahasiswa secara otomatis.
3. Sistem menyediakan rekapitulasi atau laporan kehadiran mahasiswa.

3.3 Perancangan Sistem

Perancangan aplikasi dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan. Perancangan aplikasi berdasarkan

Obeject Oriented Analysis dan *Object Oriented Design* menggunakan pemodelan UML (*Unified Modeling Language*). Perancangan subsistem aplikasi dilakukan dengan mengidentifikasi kelas-kelas dan *interface-interface* yang dibutuhkan yang dimodelkan dalam *class diagram.* Hubungan interaksi antar elemen (obyek) yang telah diidentifikasi dimodelkan dalam *Sequence Diagram* yang menggambarkan interaksi antar objek yang disusun dalam urutan waktu. Perancangan sistem presensi mahasiswa ini secara garis besar meliputi:

- Perancangan *Database*
- Perancangan *User Interface*
- Penerapan algoritma *eigenface*

Pada saat perancangan sistem, terdapat beberapa kali perubahan maupun penambahan fitur – fitur sistem, diantaranya :

1. *Prototype* 1 (26/12/2012): penambahan fitur update pada halaman mahasiswa, matakuliah, jadwal dan dosen.
2. *Prototype* 2 (22/1/2013): penambahan halaman laporan presensi manual.
3. *Prototype* 3 (09/04/2013) : penambahan fitur *print* pada halaman laporan presensi.

3.4 Implementasi

Implementasi dilakukan dengan mengacu kepada perancangan sistem. Implementasi sistem dilakukan dengan menggunakan bahasa pemrograman berorientasi objek yaitu menggunakan bahasa pemrograman *Java*. Implementasi sistem presensi mahasiswa ini meliputi:

- Pembuatan *user interface* dan penerapan algoritma *Eigenface* dalam program yang dibuat dengan *NetBeans IDE 7.1.1*.
- Mengambil data citra wajah yang digunakan sebagai data *training image*, melakukan proses perhitungan *eigenvalue* dari koleksi *training image*.
- Melakukan proses *matching* atau pencocokan *eigenvalue* citra wajah yang dimasukkan pada saat proses presensi dengan nilai *eigenvalue* dari koleksi *training image* dan mencari nilai yang paling mendekati serta mencari data mahasiswa yang berkorespondensi dengan nilai tadi.

- Melakukan rekapitulasi proses presensi mahasiswa yang dibentuk menjadi laporan absensi.

3.5 Pengujian dan Analisis

Pengujian perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya. Pengujian dilakukan dengan menggunakan metode *white box testing* dan *black box testing*. Dalam pengujian ini, strategi pengujian yang digunakan meliputi pengujian unit dan pengujian validasi. Pengujian unit digunakan untuk melihat jalur yang mungkin dilakukan oleh program. Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Selain itu untuk menguji kinerja dari sistem, maka dilakukan uji kinerja sistem. Uji kinerja sistem meliputi menguji tingkat akurasi dan *eror rate* sistem dengan menggunakan *confusion matrix* untuk menentukan performa model pengklasifikasian, serta menghitung rata – rata waktu yang diperlukan sistem dalam melakukan pengenalan wajah.

BAB IV
PERANCANGAN

Bab ini membahas mengenai perancangan perangkat lunak Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*. Perancangan yang dilakukan meliputi dua tahap. Proses analisis kebutuhan dilakukan pada tahap pertama dan proses perancangan sistem dilakukan pada tahap kedua.

Proses analisis kebutuhan terdiri atas empat tahap. Tahap pertama melakukan penjabaran tentang gambaran umum sistem. Tahap kedua melakukan proses identifikasi aktor yang terlibat dalam sistem. Tahap ketiga melakukan proses analisis data yang diperlukan. Tahap keempat melakukan analisis *use case* berdasarkan penjabaran gambaran umum Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*.

Pada proses perancangan sistem terdiri dari empat tahap, yaitu perancangan basis data, pemodelan kelas untuk menggambarkan perancangan struktur *class-class* yang menyusun sistem, perancangan diagram *sequence* dan perancangan antarmuka pengguna dari Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*.

4.1 Analisis Kebutuhan

Proses analisis kebutuhan mengacu pada gambaran umum dan hasil pengumpulan, pemahaman dan penetapan kebutuhan-kebutuhan yang ingin didapatkan oleh pengguna. Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*, identifikasi aktor yang terlibat, analisis data yang akan disimpan, penjabaran tentang daftar kebutuhan dan kemudian memodelkannya ke dalam diagram *use case*. Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna.

4.1.1 Gambaran Umum Sistem

Pembahasan gambaran umum Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* terdiri atas dua bagian, yaitu deskripsi umum sistem dan cara kerja sistem.

4.1.1.1 Deskripsi Umum Sistem

Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* berfungsi untuk membantu menutupi kelemahan pada sistem presensi mahasiswa yang masih menggunakan sistem presensi manual, dimana mahasiswa menandatangani berkas presensi pada saat mengikuti perkuliahan. Sistem presensi ini terdiri dari subsistem aplikasi mahasiswa dan subsistem aplikasi *administrator*.

a. Aplikasi Mahasiswa

Fungsi utama yang dapat dilakukan oleh aplikasi mahasiswa adalah :

1. Melakukan proses presensi mahasiswa

   Operasi ini bertujuan untuk mengenali citra wajah mahasiswa yang diambil saat proses presensi, apabila wajah mahasiswa dapat dikenali dengan benar, maka proses presensi berhasil.

2. Menambah data wajah baru

   Operasi ini bertujuan agar mahasiswa dapat menambahkan data diri supaya dapat dikenali oleh sistem. Data mahasiswa yang dimasukkan berupa citra wajah sebagai *training image*.

b. Aplikasi *Administrator*

Aplikasi *administrator* dari sistem ini adalah aplikasi yang digunakan untuk mengakses dan mengelola elemen – elemen perangkat lunak secara lebih lanjut. Proses – proses utama yang terdapat dalam aplikasi *administrator* adalah sebagai berikut :

1. *Login*

   Operasi ini bertujuan untuk memberi seleksi kepada pengguna sehingga hanya *administrator* saja yang dapat mengakses keseluruhan modul yang disediakan sistem.

2. Mengelola Data Mahasiswa

Operasi ini bertujuan untuk melakukan manipulasi data mahasiswa yang disimpan pada tabel mahasiswa pada *database*. Operasi ini terdiri atas beberapa operasi, yaitu:

- Melakukan tambah pada tabel mahasiswa.
- Melakukan hapus pada tabel mahasiswa
- Melakukan ubah pada tabel mahasiswa.

3. Mengelola Data Dosen

   Operasi ini bertujuan untuk melakukan manipulasi data dosen yang disimpan pada tabel dosen pada *database*. Operasi ini terdiri atas beberapa operasi, yaitu :

   - Melakukan tambah pada tabel dosen.
   - Melakukan hapus pada tabel dosen.
   - Melakukan ubah pada tabel dosen.

4. Mengelola Data Matakuliah

   Operasi ini bertujuan untuk melakukan manipulasi data matakuliah yang disimpan pada tabel mata_kuliah pada *database*. Operasi ini terdiri atas beberapa operasi, yaitu :

   - Melakukan tambah pada tabel mata_kuliah.
   - Melakukan hapus pada tabel mata_kuliah.
   - Melakukan ubah pada tabel mata_kuliah.

5. Mengelola Data Jadwal

   Operasi ini bertujuan untuk melakukan manipulasi data jadwal yang disimpan pada tabel mengajar pada *database*. Operasi ini terdiri atas beberapa operasi, yaitu :

   - Melakukan tambah pada tabel mengajar.
   - Melakukan hapus pada tabel mengajar.
   - Melakukan ubah pada tabel mengajar.

6. Mengelola Data Absensi

   Operasi ini bertujuan untuk melakukan manipulasi data presensi mahasiswa yang disimpan pada tabel absensi pada *database*. Operasi ini terdiri atas beberapa operasi, yaitu :

   - Melakukan tambah pada tabel absensi.

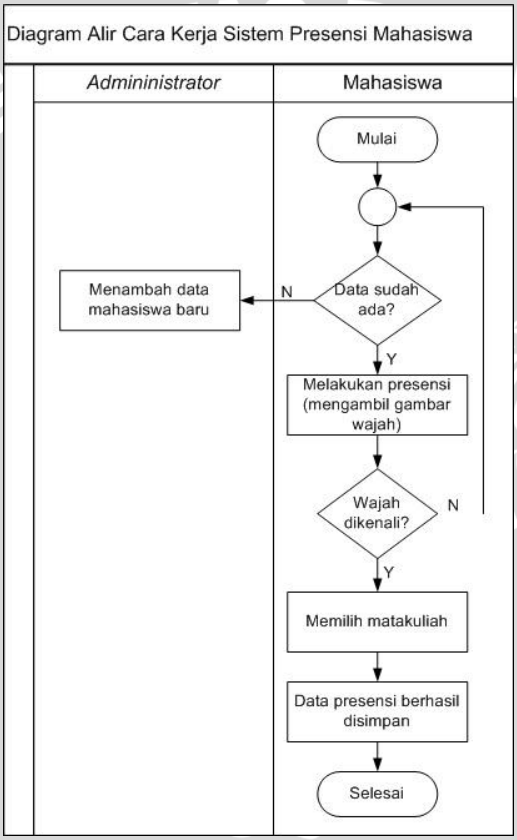
- Melakukan hapus pada tabel absensi.
- Melakukan ubah pada tabel absensi.

7. *Logout.*

   Operasi *Logout* dilakukan bila pengguna yang terdaftar sebagai *Administrator* akan keluar dari sistem.

4.1.1.2 Cara Kerja Sistem

Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* memiliki urutan langkah kerja seperti yang ditunjukkan gambar 4.1.



Gambar 4.1 Diagram Alir Cara Kerja Sistem Presensi Mahasiswa
Sumber : Perancangan

Penjelasan diagram alir cara kerja sistem presensi mahasiswa pada Gambar 4.1 sebagai berikut:

1. Jika saat mahasiswa menjalankan sistem datanya belum ada dalam *database*, maka mahasiswa dapat menemui *administrator*. *Administrator* akan memasukkan data diri mahasiswa dan mendampingi mahasiswa saat mengambil gambar wajah yang akan disimpan menjadi *training image*.

2. Jika saat mahasiswa menjalankan sistem datanya sudah ada, maka dapat dilanjutkan untuk melakukan proses presensi dengan cara mengambil gambar wajah mahasiswa.
3. Jika wajah mahasiswa tidak berhasil dikenali saat proses presensi, maka kembali pada proses awal (No. 1).
4. Jika wajah mahasiswa berhasil dikenali, selanjutnya mahasiswa dapat memilih matakuliah yang ingin dilakukan proses presensi.
5. Setelah selesai memilih matakuliah, maka data presensi mahasiswa akan disimpan ke dalam database.

4.1.2 Identifikasi Aktor

Tahap ini adalah tahap untuk melakukan identifikasi terhadap aktor – aktor yang akan berinteraksi dengan Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*. Tabel 4.1 memperlihatkan aktor – aktor yang terlibat beserta penjelasannya masing-masing yang merupakan hasil dari proses identifikasi aktor.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Mahasiswa	Mahasiswa adalah pengguna biasa yang hanya dapat melakukan proses presensi dan menambah data wajah baru.
Administrator	*Administrator* merupakan aktor yang berhak mengelola serta mengakses semua modul yang terdapat pada sistem.

Sumber : Perancangan

4.1.3 Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional. Pada daftar kebutuhan fungsional akan dispesifikasikan menjadi dua yaitu spesifikasi kebutuhan fungsional mahasiswa dan spesifikasi kebutuhan fungsional *administrator*. Spesifikasi kebutuhan fungsional mahasiswa ditunjukkan pada Tabel 4.2.

Tabel 4.2 Spesifikasi Kebutuhan Fungsional Mahasiswa

Nomor SRS	Kebutuhan	*Use Case*
SRS_001_01	Sistem presensi ini harus mampu melakukan proses pengenalan wajah saat mahasiswa melakukan proses presensi.	Melakukan Presensi

SRS_001_02	Sistem presensi ini harus menyediakan fasilitas menambahkan data wajah baru untuk mahasiswa.	Menambah Wajah Baru

Sumber : Perancangan

Spesifikasi kebutuhan fungsional *administrator* ditunjukkan pada Tabel 4.3.

Tabel 4.3 Spesifikasi Kebutuhan Fungsional *Administrator*

Nomor SRS	Kebutuhan	*Use Case*
SRS_002_01	Perangkat lunak harus menyediakan fasilitas untuk *login* sehingga hanya *administrator* saja yang dapat mengelola serta mengakses semua modul yang terdapat pada sistem.	*Login*
SRS_002_02	Perangkat lunak harus menyediakan fasilitas untuk mengelola data mahasiswa.	Mengelola Data Mahasiswa
SRS_002_03	Perangkat lunak harus menyediakan fasilitas untuk mengelola data dosen.	Mengelola Data Dosen
SRS_002_04	Perangkat lunak harus menyediakan fasilitas untuk mengelola data matakuliah.	Mengelola Data Matakuliah
SRS_002_05	Perangkat lunak harus menyediakan fasilitas untuk mengelola data jadwal.	Mengelola Data Jadwal
SRS_002_06	Perangkat lunak harus menyediakan fasilitas untuk mengelola data presensi mahasiswa.	Mengelola Data Presensi
SRS_002_07	Perangkat lunak harus menyediakan fasilitas untuk *logout* sehingga *admin* dapat meninggalkan aplikasi sistem.	*Logout*

Sumber : Perancangan

Daftar kebutuhan non-fungsional Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* ditunjukkan pada Tabel 4.4.

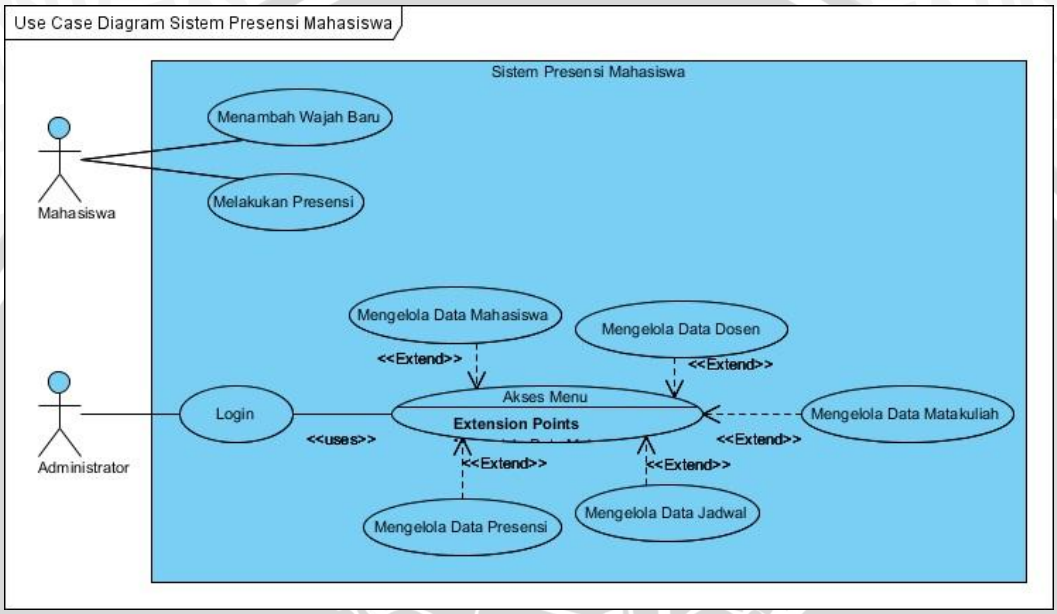
Tabel 4.4 Spesifikasi Kebutuhan Non-Fungsional

Parameter	Deskripsi Kebutuhan
Availability	Sistem harus dapat beroperasi terus – menerus selama waktu yang diinginkan (24 jam atau bahkan melebihi jam kerja).
Response Time	Sistem harus dapat melakukan proses pengenalan wajah dengan waktu kurang dari 5 detik.
Control	Membatasi akses penggunaan sistem dengan menyediakan 2 sisi aplikasi, yaitu aplikasi mahasiswa dan *administrator*.
Usability	Sistem harus dapat dengan mudah digunakan pengguna.

Sumber : Perancangan

4.1.4 Diagram *Use Case*

Diagram *use case* adalah salah satu diagram untuk memodelkan aspek perilaku sistem. Diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas dari sistem yang diinisialisasi oleh aktor. Pemodelan diagram *use case* yang menggambarkan fungsionalitas. Diagram *use case* untuk Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* ditunjukkan pada Gambar 4.2 berikut ini :



Gambar 4.2 Diagram *Use Case* Sistem
Sumber : Perancangan

4.1.4.1 Skenario *Use Case* Melakukan Presensi

Tabel 4.5 *Use case* Melakukan Presensi

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_01
Nama	Melakukan Presensi.
Tujuan	Untuk melakukan proses presensi mahasiswa.
Deskripsi	*Use Case* ini menjelaskan bagaimana mahasiswa melakukan proses presensi.
Aktor	Mahasiswa.
Pra-kondisi	Aplikasi sistem presensi telah berjalan dan data mahasiswa sebelumnya telah terdapat pada *database*.
Proses Skenario	
Aksi Aktor	**Reaksi Sistem**

1. Aktor menekan tombol "Deteksi Wajah"	2. Sistem melakukan proses pengenalan terhadap wajah mahasiswa yang melakukan presensi. Apabila dikenali sistem menampilkan halaman Pilih Matakuliah.
3. Aktor memilih kode matakuliah yang akan dilakukan proses presensi, selanjutnya menekan tombol "Pilih".	4. Sistem melakukan pengecekan pada basis data. Apabila valid, sistem akan menampilkan form yang memberitahukan bahwa absensi berhasil dilakukan, dimana form berisi data mahasiswa yang berhasil melakukan presensi.
Kondisi Akhir	Data mahasiswa yang berhasil melakukan proses presensi akan disimpan ke dalam tabel absensi dalam *database*.

Sumber : Perancangan

4.1.4.2 Skenario *Use Case* Menambah Data Wajah Baru

Tabel 4.6 *Use case* Menambah Data Wajah Baru

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_001_02
Nama	Menambahkan Data Wajah Baru.
Tujuan	Untuk menambahkan data wajah baru mahasiswa.
Deskripsi	*Use Case* ini menjelaskan bagaimana menambahkan data citra wajah mahasiswa.
Aktor	Mahasiswa
Pra-kondisi	Aplikasi sistem presensi telah berjalan dan wajah mahasiswa belum terdapat pada database.
Proses Skenario	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol "Mahasiswa".	2. Sistem mengambil gambar mahasiswa dan menampilkan form tambah data mahasiswa.
3. Aktor mengisi *field* id_gambar dan memilih nim kemudian menekan tombol "Tambah".	4. Sistem melakukan proses penyimpaan data wajah baru dalam basis data.
	5. Sistem menampilkan pesan "Berhasil Tambah Data."
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman utama sisi mahasiswa.
Kondisi Akhir	Data mahasiswa disimpan ke dalam tabel gambar dalam *database*.

Sumber : Perancangan

4.1.4.3 Skenario Use Case Login

Tabel 4.7 *Use case Login*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_01
Nama	*Login*
Deskripsi	*Use case* ini menjelaskan proses melakukan *Login*
Tujuan	Untuk menyeleksi *administrator* yang sah.
Aktor	*Administrator*
Pra-kondisi	Aplikasi sistem presensi telah berjalan. Aktor memilih Halaman *Administrator* dan sistem menampilkan *form login*.
Proses Skenario	
Aksi Aktor	**Reaksi Sistem**
1. Aktor memasukkan *username* dan *password*, kemudian menekan tombol "*Login*".	2. Sistem melakukan pengecekan pada basis data, jika *username* dan *password* valid, maka sistem akan menampilkan halaman utama sisi *administrator*.
Eksespsi Jika Gagal *Log In*	
	3. Sistem menampilkan pesan peringatan "*Username* atau *password* salah".
Kondisi Akhir	Sistem menampilkan halaman utama sisi *administrator*.

Sumber : Perancangan

4.1.4.4 Skenario *Use Case* Mengelola Data Mahasiswa

Tabel 4.8 *Use case* Mengelola Data Mahasiswa

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_02
Nama	Mengelola Data Mahasiswa
Tujuan	Untuk melakukan pengelolaan data mahasiswa.
Deskripsi	*Use case* ini menjelaskan bagaimana *administrator* dapat melakukan pengelolaan data mahasiswa.
Aktor	*Administrator*
Pra-kondisi	Sistem menampilkan halaman utama sisi *administrator*.
Skenario: Tambah Data Mahasiswa	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol	2. Sistem menampilkan halaman Mahasiswa.

"Mahasiswa".	
3. Aktor memilih untuk memasukkan data mahasiswa baru dengan mengisi *field* nim, nama, memilih jenis kelamin dan mengisi *field* alamat. Selanjutnya menekan tombol "Tambah".	4. Sistem melakukan proses penyimpaan data mahasiswa baru dalam basis data.
	5. Sistem menampilkan pesan "Data mahasiswa telah tersimpan"
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman Mahasiswa.
Skenario: Ubah Data Mahasiswa	
Aksi Aktor	**Reaksi Sistem**
1. Aktor memilih data mahasiswa yang akan diubah, selanjutnya menekan tombol "Ubah".	2. Sistem menampilkan data yang diubah ke dalam form masukkan data
3. Aktor mengubah data, selanjutnya menekan tombol "Simpan".	4. Sistem melakukan proses penyimpaan data mahasiswa baru dalam basis data.
	5. Sistem menampilkan pesan "Data mahasiswa telah tersimpan".
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman Mahasiswa.
Skenario: Hapus Data Mahasiswa	
1. Aktor memilih data mahasiswa yang akan dihapus. Kemudian menekan tombol "Hapus".	2. Sistem melakukan proses penghapusan data mahasiswa dari basis data.
	3. Sistem menampilkan pesan "Data mahasiswa telah terhapus"
4. Aktor menutup pesan	5. Sistem menampilkan kembali halaman mahasiswa
Kondisi Akhir	Sistem akan menyimpan hasil pengolahan data mahasiswa ke dalam tabel mahasiswa dalam *database*.

Sumber : Perancangan

4.1.4.5 Skenario *Use Case* Mengelola Data Dosen

Tabel 4.9 *Use case* Mengelola Data Dosen

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_03
Nama	Mengelola Data Dosen
Tujuan	Untuk melakukan pengelolaan data dosen.
Deskripsi	*Use case* ini menjelaskan bagaimana *administrator* dapat melakukan pengelolaan data dosen
Aktor	*Administrator*
Pra-kondisi	Sistem menampilkan halaman utama sisi *administrator*.
Skenario: Tambah Data Dosen	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol "Dosen".	2. Sistem menampilkan halaman Dosen.
3. Aktor memilih untuk memasukkan data dosen baru dengan mengisi *field* kode dosen dan nama dosen. Selanjutnya menekan tombol "Tambah".	4. Sistem melakukan proses penyimpaan data dosen baru dalam basis data.
	5. Sistem menampilkan pesan "Data dosen telah tersimpan"
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman Dosen.
Skenario: Ubah Data Dosen	
1. Aktor memilih data dosen yang akan diubah, selanjutnya menekan tombol "Ubah".	2. Sistem menampilkan data yang diubah ke dalam form masukkan data
3. Aktor mengubah data, selanjutnya menekan tombol "Simpan".	4. Sistem melakukan proses penyimpaan data dosen baru dalam basis data.
	5. Sistem menampilkan pesan "Data dosen telah tersimpan".
6. Aktor menutup pesan.	7. Sistem menampilkan kembali halaman Dosen.
Skenario: Hapus Data Dosen	
1. Aktor memilih data dosen yang akan	2. Sistem melakukan proses penghapusan data dosen dari basis data.

dihapus. Kemudian menekan tombol "Hapus".	
	3. Sistem menampilkan pesan "Data dosen telah terhapus".
4. Aktor menutup pesan.	5. Sistem menampilkan kembali halaman dosen
Kondisi Akhir	Sistem akan menyimpan hasil pengolahan data dosen ke dalam tabel dosen dalam *database*.

Sumber : Perancangan

4.1.4.6 Skenario *Use Case* Mengelola Data Matakuliah

Tabel 4.10 *Use case* Mengelola Data Matakuliah

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_04
Nama	Mengelola Data Matakuliah
Tujuan	Untuk melakukan pengelolaan data matakuliah.
Deskripsi	*Use case* ini menjelaskan bagaimana *administrator* dapat melakukan pengelolaan data matakuliah
Aktor	*Administrator*
Pra-kondisi	Sistem menampilkan halaman utama sisi *administrator*.
Skenario: Tambah Data Matakuliah	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol "Matakuliah".	2. Sistem menampilkan halaman Matakuliah.
3. Aktor memilih untuk memasukkan data matakuliah baru dengan mengisi *field* kode matakuliah dan nama matakuliah, sks dan semester. Selanjutnya menekan tombol "Tambah".	4. Sistem melakukan proses penyimpaan data matakuliah baru dalam basis data.
	5. Sistem menampilkan pesan "Data matakuliah telah tersimpan"
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman Matakuliah.
Skenario: Ubah Data Matakuliah	

1. Aktor memilih data matakuliah yang akan diubah, selanjutnya menekan tombol "Ubah".	2. Sistem menampilkan data yang diubah ke dalam form masukkan data
3. Aktor mengubah data, selanjutnya menekan tombol "Simpan".	4. Sistem melakukan proses penyimpaan data matakuliah baru dalam basis data.
	5. Sistem menampilkan pesan "Data matakuliah telah tersimpan".
6. Aktor menutup pesan.	7. Sistem menampilkan kembali halaman Matakuliah.
Skenario: Hapus Data Matakuliah	
1. Aktor memilih data matakuliah yang akan dihapus. Kemudian menekan tombol "Hapus".	2. Sistem melakukan proses penghapusan data matakuliah dari basis data.
	3. Sistem menampilkan pesan "Data matakuliah telah terhapus".
4. Aktor menutup pesan.	5. Sistem menampilkan kembali halaman Matakuliah.
Kondisi Akhir	Sistem akan menyimpan hasil pengolahan data matakuliah ke dalam tabel mata_kuliah dalam *database*.

Sumber : Perancangan

4.1.4.7 Skenario *Use Case* Mengelola Data Jadwal

Tabel 4.11 *Use case* Mengelola Data Jadwal

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_05
Nama	Mengelola Data Jadwal
Tujuan	Untuk melakukan pengelolaan data jadwal
Deskripsi	*Use case* ini menjelaskan bagaimana *administrator* dapat melakukan pengelolaan data jadwal
Aktor	*Administrator*
Pra-kondisi	Sistem menampilkan halaman utama sisi *administrator*.
Skenario: Tambah Data Jadwal	

Aksi Aktor	Reaksi Sistem
1. Aktor menekan tombol "Jadwal".	2. Sistem menampilkan halaman Jadwal.
3. Aktor memilih untuk memasukkan data jadwal baru dengan memilih kode dosen, kode matakuliah dan mengisi *field* kelas. Selanjutnya menekan tombol "Tambah".	4. Sistem melakukan proses penyimpaan data jadwal baru dalam basis data.
	5. Sistem menampilkan pesan "Data jadwal telah tersimpan"
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman Jadwal.
Skenario: Ubah Data Jadwal	
1. Aktor memilih data jadwal yang akan diubah, selanjutnya menekan tombol "Ubah".	2. Sistem menampilkan data yang diubah ke dalam form masukkan data
3. Aktor mengubah data, selanjutnya menekan tombol "Simpan".	4. Sistem melakukan proses penyimpaan data jadwal baru dalam basis data.
	5. Sistem menampilkan pesan "Data jadwal telah tersimpan".
6. Aktor menutup pesan.	7. Sistem menampilkan kembali halaman Jadwal.
Skenario: Hapus Data Jadwal	
1. Aktor memilih data jadwal yang akan dihapus. Kemudian menekan tombol "Hapus".	2. Sistem melakukan proses penghapusan data jadwal dari basis data.
	3. Sistem menampilkan pesan "Data jadwal telah terhapus".
4. Aktor menutup pesan.	5. Sistem menampilkan kembali halaman Jadwal.
Kondisi Akhir	Sistem akan menyimpan hasil pengolahan data jadwal ke dalam tabel mengajar dalam *database*.

Sumber : Perancangan

4.1.4.8 Skenario *Use Case* Mengelola Data Presensi

Tabel 4.12 *Use case* Mengelola Data Presensi

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_06
Nama	Mengelola Data Presensi
Tujuan	Untuk melakukan pengelolaan data presensi
Deskripsi	*Use case* ini menjelaskan bagaimana *administrator* dapat melakukan pengelolaan data presensi
Aktor	*Administrator*
Pra-kondisi	Sistem menampilkan halaman utama sisi *administrator*.
Skenario: Tambah Data Presensi	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol "Laporan".	2. Sistem menampilkan halaman Laporan.
3. Aktor memilih untuk memasukkan data presensi baru dengan memilih nim dan kode matakuliah. Selanjutnya menekan tombol "Tambah".	4. Sistem melakukan proses penyimpaan data presensi baru dalam basis data.
	5. Sistem menampilkan pesan "Data presensi telah tersimpan"
6. Aktor menutup pesan	7. Sistem menampilkan kembali halaman Laporan.
Skenario: Ubah Data Presensi	
1. Aktor memilih data presensi yang akan diubah, selanjutnya menekan tombol "Ubah".	2. Sistem menampilkan data yang diubah ke dalam form masukkan data
3. Aktor mengubah data, selanjutnya menekan tombol "Simpan".	4. Sistem melakukan proses penyimpaan data presensi baru dalam basis data.
	5. Sistem menampilkan pesan "Data presensi telah tersimpan".
6. Aktor menutup pesan.	7. Sistem menampilkan kembali halaman Laporan.

Skenario: Hapus Data Presensi	
1. Aktor memilih data presensi yang akan dihapus. Kemudian menekan tombol "Hapus".	2. Sistem melakukan proses penghapusan data presensi dari basis data.
	3. Sistem menampilkan pesan "Data presensi telah terhapus".
4. Aktor menutup pesan.	5. Sistem menampilkan kembali halaman Laporan.
Kondisi Akhir	Sistem akan menyimpan hasil pengolahan data presensi ke dalam tabel absensi dalam *database*.

Sumber : Perancangan

4.1.4.9 Skenario Use Case Logout

Tabel 4.13 *Use case Logout*

Skenario Kasus Pada Sistem	
Nomor *Use Case*	SRS_002_07
Nama	*Logout*
Tujuan	*Use case* ini menjelaskan bagaimana Admistrator keluar dari sistem
Deskripsi	*Use case* ini menjelaskan bagaimana *administrator* melakukan *logout* untuk keluar dari sistem.
Aktor	*Administrator*
Pra-kondisi	Sistem menampilkan halaman utama sisi *administrator*.
Proses Skenario	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol "*Logout"*.	2. Sistem menutup halaman utama
	3. Sistem menampilkan kembali halaman *Login*
Kondisi Akhir	Sistem menampilkan halaman *Login*

Sumber : Perancangan

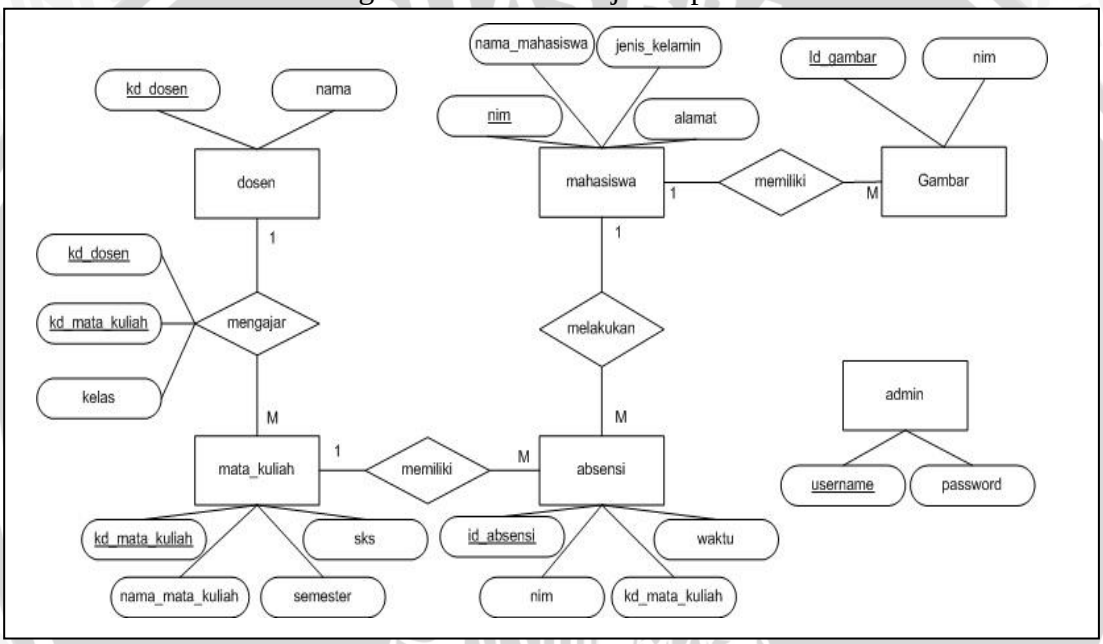
4.2 Perancangan Sistem

Perancangan sistem dilakukan dalam beberapa tahap, yaitu perancangan basis data, pemodelan diagram *class* untuk menggambarkan perancangan struktur *class-class* yang menyusun sistem, pemodelan diagram *sequence* untuk

menggambarkan interaksi antar objek atau *class* dan perancangan antarmuka pengguna dari Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*.

4.3 Perancangan Basis Data

Basis data berfungsi sebagai tempat menyimpan data. Perancangan basis data digunakan untuk merancang basis data yang akan dibuat agar masukan dan keluaran program sesuai dengan apa yang diharapkan. Perancangan basis data mengambil acuan dari proses analisis data yang dilakukan pada tahap analisis kebutuhan. Gambar rancangan basis data ditunjukan pada Gambar 4.3.



Gambar 4.3 Diagram *Entity Relationship* Sistem Presensi Mahasiswa
Sumber : Perancangan

Berikut ini merupakan struktur tabel serta keterangan masing masing tabel dan *field* yang ada pada *database*. Tabel dosen terdiri dari kd_dosen yang merupakan *primary key* serta nama. Struktur tabel dosen ditunjukkan pada Tabel 4.14.

Tabel 4.14 Struktur Tabel Dosen

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	kd_dosen	Varchar	25	Kode dosen
2	Nama	Varchar	50	Nama dosen

Sumber : Perancangan

Himpunan relasi yang menghubungkan antara himpunan entitas atau tabel dosen dan mata_kuliah dinamakan mengajar. Pada relasi ini setiap dosen dapat mengajar lebih dari satu mata kuliah, sedangkan setiap mata kuliah diajar hanya oleh paling banyak satu orang dosen.

Pada himpunan relasi mengajar terdapat satu atribut tambahan yang bukan berasal dari salah satu himpunan entitas yang dihubungkan, yaitu atribut ruang. Sedangkan atribut kd_dosen dan kd_mata_kuliah merupakan *foreign key* yang masing – masing berasal dari tabel dosen dan mata_kuliah. Struktur himpunan relasi mengajar ditunjukkan pada Tabel 4.15.

Tabel 4.15 Struktur Himpunan Relasi Mengajar

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	kd_dosen	Varchar	25	Kode dosen
2	kd_mata_kuliah	Varchar	15	Kode matakuliah
3	Kelas	Varchar	25	Kelas mengajar

Sumber : Perancangan

Entitas mata_kuliah meliputi kd_mata_kuliah yang merupakan *primary key*, nama_mata_kuliah, sks dan semester. Struktur tabel mata_kuliah ditunjukkan pada Tabel 4.16.

Tabel 4.16 Struktur Tabel Mata_Kuliah

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	kd_mata_kuliah	Varchar	15	Kode matakuliah
2	nama_mata_kuliah	Varchar	35	Nama matakuliah
3	Sks	Tinyint	3	Jumlah sks matakuliah
4	Semester	Varchar	25	Semester matakuliah

Sumber : Perancangan

Entitas mahasiswa terdiri dari nim yang merupakan *primary key*, nama_mahasiswa, jenis_kelamin dan alamat. Struktur tabel mahasiswa ditunjukkan pada Tabel 4.17.

Tabel 4.17 Struktur Tabel Mahasiswa

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	nim	Varchar	15	Nim mahasiswa
2	nama_mahasiswa	Varchar	60	Nama mahasiswa
3	jenis_kelamin	Char	1	Jenis kelamin mahasiswa
4	Alamat	Text		Alamat mahasiswa

Sumber : Perancangan

Entitas absensi terdiri dari id_absensi yang merupakan *primary key*, nim dan kd_mata_kuliah yang masing – masing merupakan *foreign key* dari tabel mahasiswa dan mata_kuliah serta waktu. Struktur tabel absensi ditunjukkan pada Tabel 4.18.

Tabel 4.18 Struktur Tabel Absensi

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	id_absensi	Integer	11	*Id* absensi
2	Nim	Varchar	15	Nim mahasiswa
3	kd_mata_kuliah	Varchar	15	Kode matakuliah
4	Waktu	Timestamp		Waktu melakukan absensi

Sumber : Perancangan

Entitas admin terdiri dari *username* yang merupakan *primary key* dan *password*. Struktur tabel admin ditunjukkan pada Tabel 4.19.

Tabel 4.19 Struktur Tabel Admin

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	*username*	Varchar	20	*Username administrator*
2	*password*	Varchar	10	*Password administrator*

Sumber : Perancangan

Entitas gambar terdiri dari *id_gambar* yang merupakan *primary key* dan nim merupakan *foreign key* yang berasal dari tabel mahasiswa. Struktur tabel gambar ditunjukkan pada Tabel 4.20.

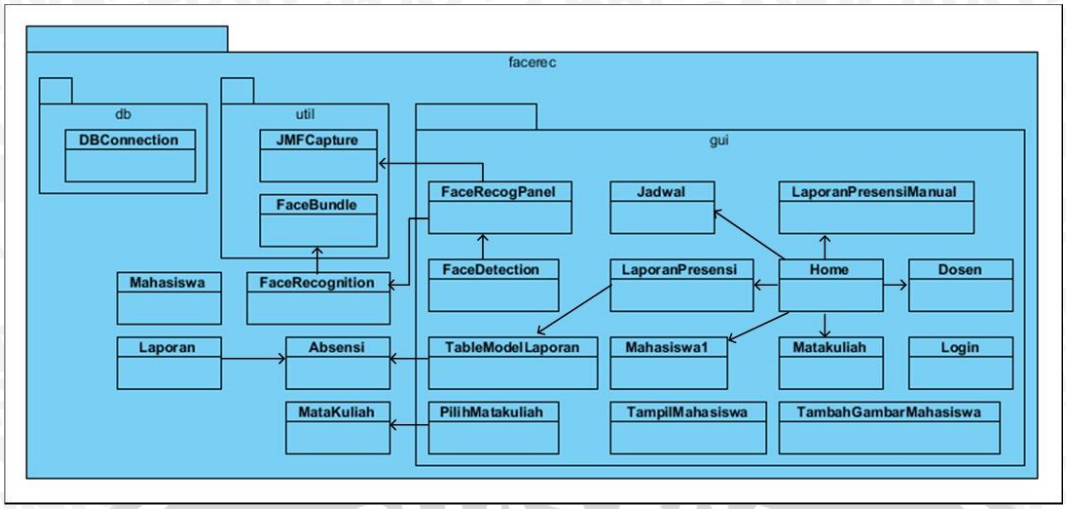
Tabel 4.20 Struktur Tabel Gambar

No.	Nama *Field*	Tipe	Lebar	Keterangan
1	id_gambar	Varchar	35	id gambar wajah mahasiswa
2	Nim	Varchar	15	Nim mahasiswa

Sumber : Perancangan

4.3.1 Pemodelan Kelas (*Class Diagram*)

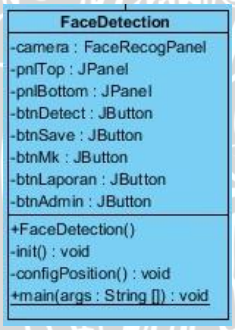
Pemodelan kelas memberikan gambaran pemodelan elemen-elemen kelas yang membentuk sebuah sistem. Kelas bisa didapatkan dengan menganalisis secara detail terhadap *use case* yang dimodelkan. Gambar 4.4 menunjukkan pemodelan diagram kelas dari sistem presensi mahasiswa yang akan dibuat.



Gambar 4.4 Pemodelan Kelas Sistem Presensi Mahasiswa
Sumber : Perancangan

4.3.1.1 Class FaceDetection

Class FaceDetection merupakan *class* yang digunakan saat sistem pertama kali dijalankan, dimana *webcam* melakukan deteksi terhadap wajah. Saat *webcam* dijalankan *class FaceDetection* akan memanggil *class FaceRecogPanel*. Gambar 4.5 menunjukkan diagram *class FaceDetection*.



Gambar 4.5 Diagram *Class FaceDetection*
Sumber : Perancangan

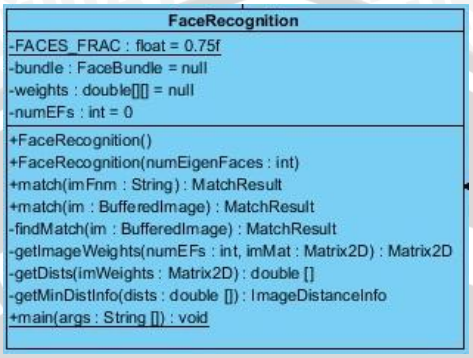
Tabel 4.21 Deskripsi *Class FaceDetection*

Nama *class* : *FaceDetection*
Deskripsi : *Class FaceDetection* merupakan *class* yang digunakan saat sistem pertama kali dijalankan, dimana *webcam* melakukan deteksi terhadap wajah.
Nama *Method* : init() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur *user* interface.
Nama *Method* : configPosition() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur komposisi komponen.

Nama *Method* : *FaceDetection*() **Fungsi :** *Method* ini adalah *method* yang berfungsi sebagai *constructor* dari *class FaceDetection*.

Sumber : Perancangan

4.3.1.2 Class FaceRecognition



Gambar 4.6 Diagram *Class FaceRecognition*
Sumber : Perancangan

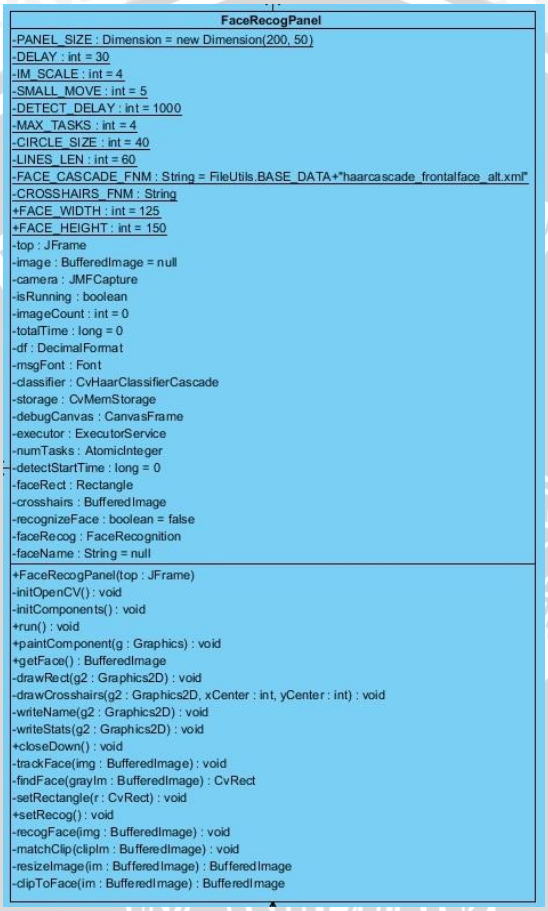
Tabel 4.22 Deskripsi *Class FaceRecognition*

Nama *class* : *FaceRecognition*
Deskripsi : *Class FaceRecognition* merupakan *class* yang menerima data baru kemudian melakukan pencocokan untuk menemukan *training image* yang terdekat
Nama *Method* : *FaceRecognition*(numEigenFaces : int) **Fungsi :** *Method* ini adalah *method* yang memuat cache dan mengubahnya kembali menjadi objek *FaceBundle*.
Nama *Method* : match(imFnm : String) **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mencocokkan gambar pada file terhadap gambar training.
Nama *Method* : findMatch(im : BufferedImage) **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melaksanakan pembandingan antara gambar baru dengan gambar training dengan menggunakan eigenvector dan eigenvalue yang diambil dari objek *FaceBundle*
Nama *Method* : getImageWeights(numEFs : int, imMat : Matrix2D) **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memetakan gambar kedalam *eigenspace*, yang akan mengembalikan hasil koordinat (atau bobot).
Nama *Method* : getDist(imWeights : Matrix2D) **Fungsi :** *Method* ini adalah *method* yang mengembalikan sebuah array dari jumlah kuadrat *Euclidian distance* antara bobot *input image* dan seluruh bobot *training image*.
Nama *Method* : getMinDistInfo(dists : double[]) **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menemukan jarak

terpendek pada array.
Nama *Method* : match(im : BufferedImage) **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mencocokkan gambar yang dimuat dengan gambar training.

Sumber : Perancangan

4.3.1.3 Class FaceRecogPanel



Gambar 4.7 Diagram *Class FaceRecogPanel*
Sumber : Perancangan

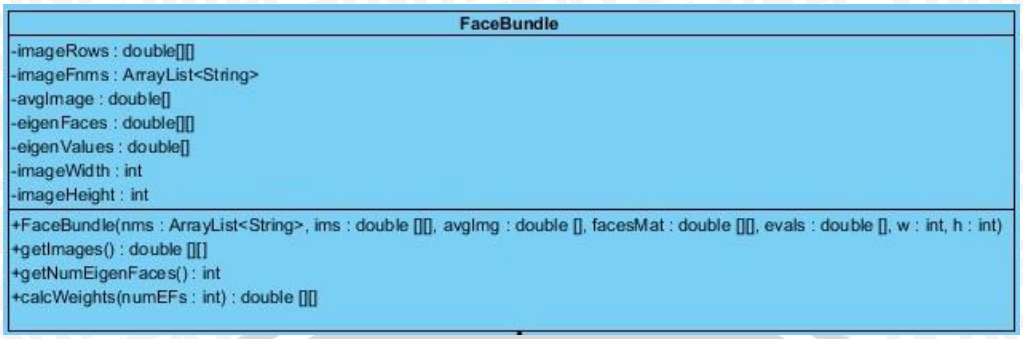
Tabel 4.23 Deskripsi *Class FaceRecogPanel*

Nama *class* : *FaceRecogPanel*
Deskripsi : *Class FaceRecogPanel* merupakan panel yang menampung fungsi untuk mengenali wajah.
Nama *Method* : initOpenCV() **Fungsi :** *Method* ini adalah *method* yang dijalankan untuk inisiasi object yang terdapat pada *OpenCV*, yaitu inisiasi *Haar Classifier Cascade* untuk deteksi wajah.
Nama *Method* : initComponents() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur komposisi

komponen.
Nama *Method* : paintComponent() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengambil gambar *webcam* di panel.
Nama *Method* : getFace() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mendapatkan referensi simpul wajah.
Nama *Method* : drawRect() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menggambar bentuk segiempat berwarna biru.
Nama *Method* : drawCrosshairs() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menggambar lingkaran dan crosshair (bidikan *cross*) di tengah segiempat.
Nama *Method* : closeDown() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengakhiri aplikasi.
Nama *Method* : trackFace() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melacak wajah.
Nama *Method* : findFace() **Fungsi :** *Method* ini adalah *method* yang mengandung semua proses *image* processing OpenCV yang meliputi *image equalization*, pemanggilan *Haar classifier* dan ekstraksi hasil dari gambar segiempat wajah ((x, y), width, height).
Nama *Method* : setRectangle() **Fungsi :** *Method* ini adalah *method* yang mengekstrak koordinat segiempat wajah dari struktur data OpenCV dan menyimpannya dalam object *Java Rectangle*. Dalam prosesnya, data diperbesar sehingga memiliki skala yang sama dengan gambar asli yang diambil.
Nama *Method* : recogFace() **Fungsi :** *Method* ini adalah *method* untuk melakukan manipulasi *image* melalui buffered*Image*d, yaitu jenis *image* yang dapat dimodifikasi.
Nama *Method* : matchClip() **Fungsi :** *Method* ini adalah *method* untuk menampilkan hasil pencocokkan wajah hasil manipulasi *bufferedImage*.
Nama *Method* : resize*Image*() **Fungsi :** *Method* ini adalah *method* untuk mengubah gambar *input* berwarna menjadi grayscale sekaligus merubah ukurannya.
Nama *Method* : clipToFace() **Fungsi :** *Method* ini adalah *method* untuk memotong gambar wajah.
Nama *Method* : *FaceRecogPanel*() **Fungsi :** *Method* ini adalah *method* yang berfungsi sebagai *constructor* dari *class* *FaceRecogPanel*.

Sumber : Perancangan

4.3.1.4 Class FaceBundle



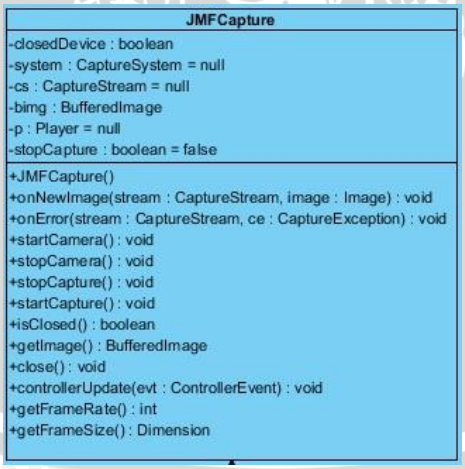
Gambar 4.8 Diagram *Class FaceBundle*
Sumber : Perancangan

Tabel 4.24 Deskripsi *Class FaceBundle*

Nama *class* : *FaceBundle*
Deskripsi : *Class FaceBundle* merupakan *class* yang menghasilkan *eigenvectors* dan *eigenvalues* dari data training.
Nama *Method* : calcWeights() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghitung bobot subset yang dipilih dari *eigenface*.
Nama *Method* : *FaceBundle*() **Fungsi :** *Method* ini adalah *method* yang berfungsi sebagai *constructor* dari *class FaceBundle*.

Sumber : Perancangan

4.3.1.5 Class JMFCapture



Gambar 4.9 Diagram *Class JMFCapture*
Sumber : Perancangan

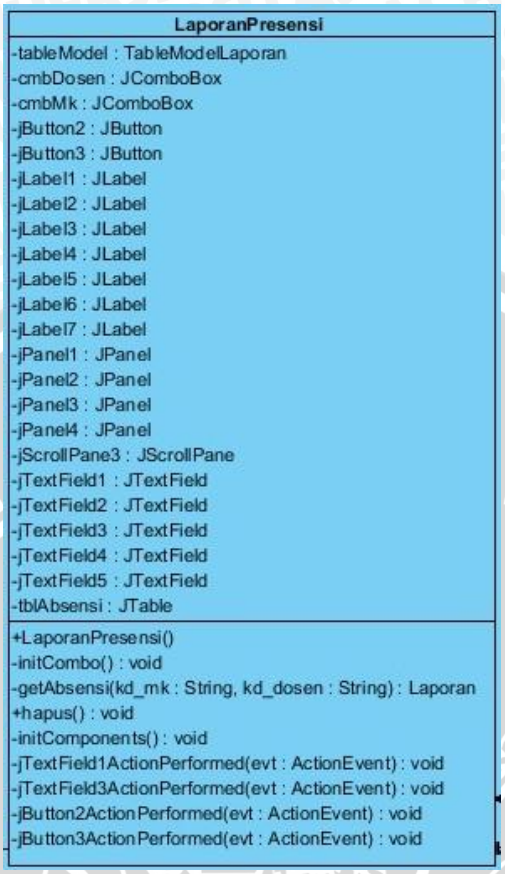
Tabel 4.25 Deskripsi *Class JMFCapture*

Nama *class* : *JMFCapture*
Deskripsi : *Class JMFCapture* merupakan *class* yang menyimpan kode untuk menangkap gambar wajah.

Nama *Method* : onNew*Image*() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menangkap atau mengambil gambar dasar dari *webcam*.
Nama *Method* : start*Kamera*() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memulai kamera
Nama *Method* : stop*Kamera*() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghentikan kamera
Nama *Method* : startCapture() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memulai menangkap gambar
Nama *Method* : isClosed() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menutup *device*
Nama *Method* : get*Image*() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memulai kamera
Nama *Method* : close() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengakhiri link ke *device capture*
Nama *Method* : controllerUpdate() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengupdate *ControllerEvent* yang merupakan kelas dasar untuk kejadian yang dihasilkan oleh *cotroller*.
Nama *Method* : getFrameRate() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menyetting *frame rate*
Nama *Method* : getFrameSize() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mendapatkan *frame size* yang sesuai.
Nama *Method* : *JMFCapture*() **Fungsi :** *Method* ini adalah *method* yang berfungsi sebagai *constructor* dari *class JMFCapture*.

Sumber : Perancangan

4.3.1.6 *Class* LaporanPresensi



Gambar 4.10 Diagram *Class* LaporanPresensi
Sumber : Perancangan

Tabel 4.26 Deskripsi *Class* LaporanPresensi

Nama *class* : LaporanPresensi
Deskripsi : *Class* LaporanPresensi merupakan *class* yang menampilkan hasil laporan absensi.
Nama *Method* : initCombo() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur list matakuliah dan dosen.
Nama *Method* : getAbsensi() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk mengambil nilai presensi dari *database*.

Sumber : Perancangan

4.3.1.7 Class Home

Gambar 4.11 Diagram *Class Home*
Sumber : Perancangan

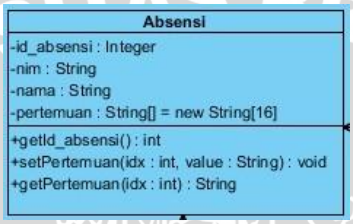
Tabel 4.27 Deskripsi *Class Home*

Nama *class* : *Home*
Deskripsi : *Class Home* merupakan *class interface* yang berfungsi sebagai halaman utama administrator untuk mengakses halaman lainnya.
Nama *Method* : dataMahasiswa() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan halaman data mahasiswa.
Nama *Method* : dataMatakuliah() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan halaman data matakuliah.
Nama *Method* : dataDosen() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan halaman data dosen.
Nama *Method* : dataJadwal() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan halaman data jadwal.

Nama *Method* : dataLaporan() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan halaman data laporan.
Nama *Method* : dataLaporanmanual() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan halaman data laporan manual.
Nama *Method* : logOut() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk keluar dari *class home*.

Sumber : Perancangan

4.3.1.8 *Class* Absensi



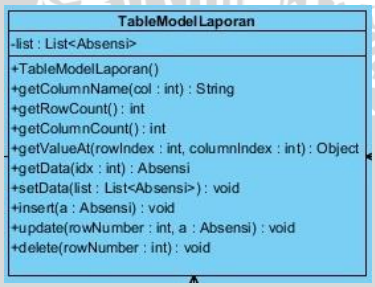
Gambar 4.12 Diagram *Class* Absensi
Sumber : Perancangan

Tabel 4.28 Deskripsi *Class* Absensi

Nama *class* : Absensi
Deskripsi : *Class* Absensi merupakan sebuah *class* yang berisi *setter getter* untuk absensi.

Sumber : Perancangan

4.3.1.9 *Class* TableModelLaporan



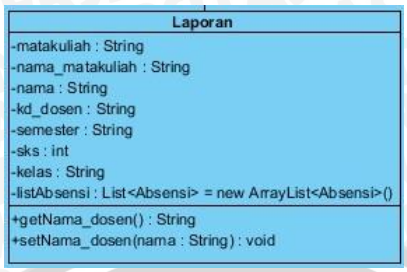
Gambar 4.13 Diagram *Class* TableModelLaporan
Sumber : Perancangan

Tabel 4.29 Deskripsi *Class* TableModelLaporan

Nama *class* : TableModelLaporan
Deskripsi : *Class* TabelModelLaporan merupakan sebuah interface yang mengekstrak *class* AbstractTabelModel.

Sumber : Perancangan

4.3.1.10 *Class* Laporan



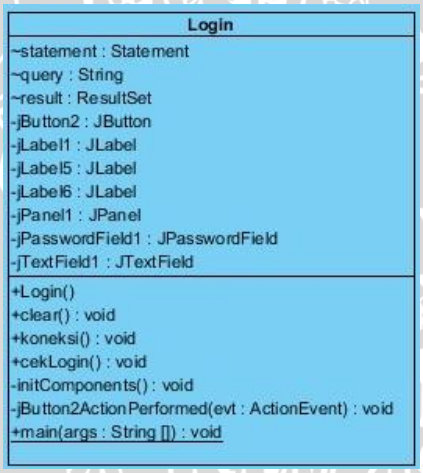
Gambar 4.14 Diagram *Class* Laporan
Sumber : Perancangan

Tabel 4.30 Deskripsi *Class* Laporan

Nama *class* : Laporan
Deskripsi : *Class* Laporan merupakan *class* yang berisi *setter getter* untuk laporan

Sumber : Perancangan

4.3.1.11 Class Login



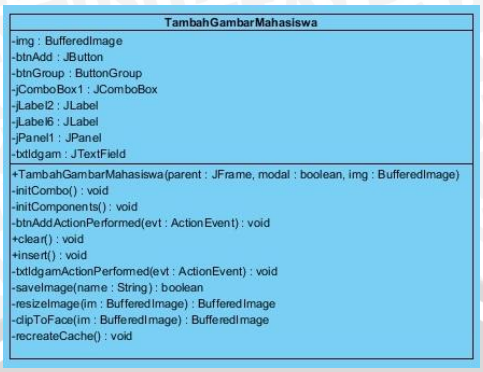
Gambar 4.15 Diagram *Class Login*
Sumber : Perancangan

Tabel 4.31 Deskripsi *Class Login*

Nama *class* : *Login*
Deskripsi : *Class Login* merupakan *class interface* yang berfungsi untuk melakukan pengecekan *login administrator*.
Nama *Method* : cek*Login*() **Fungsi :** *Method* ini adalah *method* yang berfungsi menjalankan proses pengecekan data administrator untuk dapat mengakses *class home*.

Sumber : Perancangan

4.3.1.12 *Class* TambahGambarMahasiswa



Gambar 4.16 Diagram *Class* TambahGambarMahasiswa
Sumber : Perancangan

Tabel 4.32 Deskripsi *Class* TambahGambarMahasiswa

Nama *class* : TambahGambarMahasiswa
Deskripsi : *Class* TambahGambarMahasiswa *class interface* yang berfungsi untuk melakukan penambahan gambar mahasiswa.
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data gambar mahasiswa ke dalam database.
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data gambar mahasiswa ke dalam database.
Nama *Method* : clipToFace() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengambil gambar.
Nama *Method* : resizeImage() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengubah ukuran gambar yang diambil.
Nama *Method* : saveImage() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menyimpan gambar.

Sumber : Perancangan

4.3.1.13 *Class* TampilMahasiswa

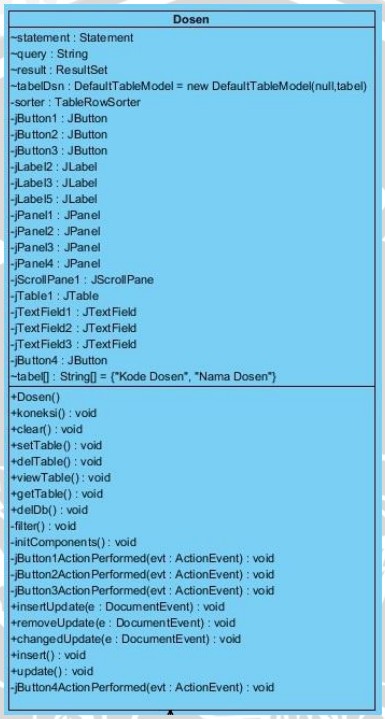
Gambar 4.17 Diagram *Class* TampilMahasiswa
Sumber : Perancangan

Tabel 4.33 Deskripsi *Class* TampilMahasiswa

Nama *class* : TampilMahasiswa
Deskripsi : *Class* TampilMahasiswa *class interface* yang berfungsi untuk menampilkan data mahasiswa yang berhasil melakukan proses presensi.

Sumber : Perancangan

4.3.1.14 Class Dosen



Gambar 4.18 Diagram *Class* Dosen
Sumber : Perancangan

Tabel 4.34 Deskripsi *Class* Dosen

Nama *class* : Dosen
Deskripsi : *Class* dosen merupakan *class interface* yang berfungsi untuk mengelola data yang berhubungan dengan data dosen.
Nama *Method* : koneksi() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melakukan koneksi dengan *database*.
Nama *Method* : clear() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur agar *form input* data di atur kosong.
Nama *Method* : setTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data ke tabel.
Nama *Method* : delTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghapus data dari tabel.

Nama *Method* : viewTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan data pada tabel.
Nama *Method* : getTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengubah data yang dipilih dari tabel.
Nama *Method* : delDb() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk menghapus data dari *database*.
Nama *Method* : filter() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk melakukan pencarian di tabel.
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk memasukkan data ke *database*.
Nama *Method* : update() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk mengubah data pada *database*.

Sumber : Perancangan

4.3.1.15 *Class* Mahasiswa1

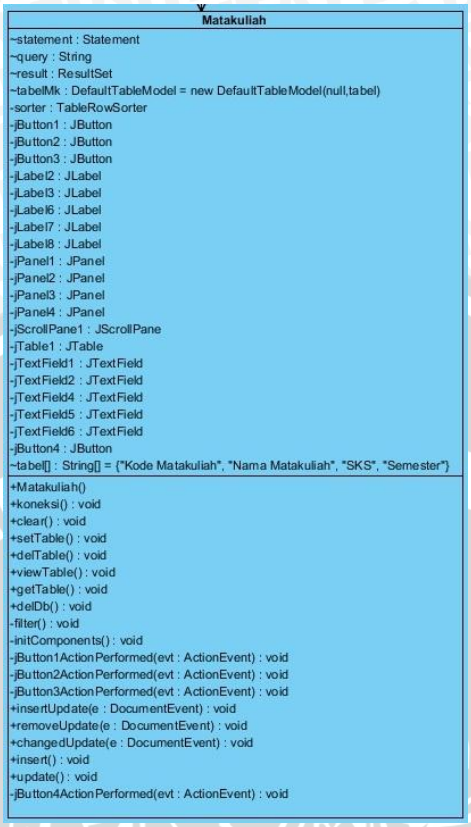
Gambar 4.19 Diagram *Class* Mahasiswa1
Sumber : Perancangan

Tabel 4.35 Deskripsi *Class* Mahasiswa1

Nama *class* : Mahasiswa1
Deskripsi : *Class* mahasiswa1 merupakan *class interface* yang berfungsi untuk mengelola data yang berhubungan dengan data mahasiswa.
Nama *Method* : koneksi() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melakukan koneksi dengan *database*.
Nama *Method* : clear() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur agar *form input* data di atur kosong.
Nama *Method* : setTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data ke tabel.
Nama *Method* : delTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghapus data dari tabel.
Nama *Method* : viewTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan data pada tabel.
Nama *Method* : getTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengubah data yang dipilih dari tabel.
Nama *Method* : delDb() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk menghapus data dari *database.*
Nama *Method* : filter() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk melakukan pencarian di tabel..
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk memasukkan data ke *database*.
Nama *Method* : update() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk mengubah data pada *database*.

Sumber : Perancangan

4.3.1.16 *Class* Matakuliah



Gambar 4.20 Diagram *Class* Matakuliah
Sumber : Perancangan

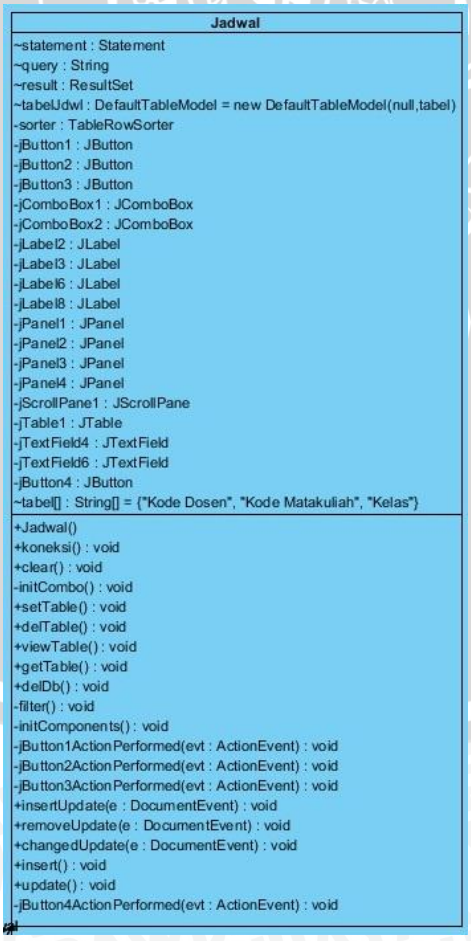
Tabel 4.36 Deskripsi *Class* Matakuliah

Nama *class* : Matakuliah
Deskripsi : *Class* matakuliah merupakan *class interface* yang berfungsi untuk mengelola data yang berhubungan dengan data matakuliah.
Nama *Method* : koneksi() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melakukan koneksi dengan *database*.
Nama *Method* : clear() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur agar *form input* data di atur kosong.
Nama *Method* : setTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data ke tabel.
Nama *Method* : delTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghapus data dari tabel.
Nama *Method* : viewTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan data pada tabel.

Nama *Method* : getTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengubah data yang dipilih dari tabel.
Nama *Method* : delDb() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk menghapus data dari *database.*
Nama *Method* : filter() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk melakukan pencarian di tabel..
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk memasukkan data ke *database*.
Nama *Method* : update() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk mengubah data pada *database*.

Sumber : Perancangan

4.3.1.17 *Class* Jadwal



Gambar 4.21 Diagram *Class* Jadwal
Sumber : Perancangan

Tabel 4.37 Deskripsi *Class* Jadwal

Nama *class* : Jadwal
Deskripsi : *Class* Jadwal merupakan *class interface* yang berfungsi untuk mengelola data yang berhubungan dengan data jadwal.
Nama *Method* : koneksi() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melakukan koneksi dengan *database*.
Nama *Method* : clear() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur agar *form input* data di atur kosong.
Nama *Method* : setTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data ke tabel.
Nama *Method* : delTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghapus data dari tabel.
Nama *Method* : viewTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan data pada tabel.
Nama *Method* : getTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengubah data yang dipilih dari tabel.
Nama *Method* : delDb() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk menghapus data dari *database*.
Nama *Method* : filter() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk melakukan pencarian di tabel..
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk memasukkan data ke *database*.
Nama *Method* : update() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk mengubah data pada *database*.

Sumber : Perancangan

4.3.1.18 *Class* PilihMataKuliah

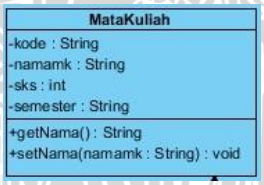
Gambar 4.22 Diagram *Class* PilihMatakuliah
Sumber : Perancangan

Tabel 4.38 Deskripsi *Class* PilihMatakuliah

Nama *class* : PilihMatakuliah
Deskripsi : *Class* PilihMatakuliah merupakan *class interface* yang berfungsi untuk melakukan proses pemilihan matakuliah saat dan menyimpan data presensi.
Nama *Method* : insertAbsensi() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data presensi ke dalam database.

Sumber : Perancangan

4.3.1.19 *Class* MataKuliah



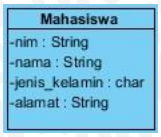
Gambar 4.23 Diagram *Class* MataKuliah
Sumber : Perancangan

Tabel 4.39 Deskripsi *Class* MataKuliah

Nama *class* : MataKuliah
Deskripsi : *Class* MataKuliah merupakan sebuah *class* yang berisi *setter getter* untuk matakuliah.

Sumber : Perancangan

4.3.1.20 *Class* Mahasiswa



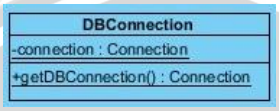
Gambar 4.24 Diagram *Class* Mahasiswa
Sumber : Perancangan

Tabel 4.40 Deskripsi *Class* Mahasiswa

Nama *class* : Mahasiswa
Deskripsi : *Class* Matahasiswa merupakan sebuah *class* yang berisi *setter getter* untuk mahasiswa.

Sumber : Perancangan

4.3.1.21 Class DBConnection



Gambar 4.25 Diagram *Class DBConnection*
Sumber : Perancangan

Tabel 4.41 Deskripsi *Class DBConnection*

Nama *class* : *DBConnection*
Deskripsi : ***Class DBConnection*** *class interface* yang berfungsi melakukan koneksi dengan *database*.
Nama *Method* : get*DBConnection*() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk membuat koneksi dengan *database*.

Sumber : Perancangan

4.3.1.22 *Class* LaporanPresensiManual

Gambar 4.26 Diagram *Class* LaporanPresensiManual
Sumber : Perancangan

Tabel 4.42 Deskripsi *Class* LaporanPresensiManual

Nama *class* : LaporanPresensiManual
Deskripsi : *Class* LaporanPresensiManual merupakan *class interface* yang berfungsi untuk mengelola data yang berhubungan dengan data presensi mahasiswa.
Nama *Method* : koneksi() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk melakukan koneksi dengan *database*.
Nama *Method* : clear() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengatur agar *form input* data di atur kosong.
Nama *Method* : setTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk memasukkan data ke tabel.
Nama *Method* : delTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menghapus data dari tabel.
Nama *Method* : viewTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk menampilkan data pada tabel.
Nama *Method* : getTable() **Fungsi :** *Method* ini adalah *method* yang digunakan untuk mengubah data yang dipilih dari tabel.
Nama *Method* : delDb() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk menghapus data dari *database.*
Nama *Method* : filter() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk melakukan pencarian di tabel.
Nama *Method* : insert() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk memasukkan data ke *database*.
Nama *Method* : update() **Fungsi :** *Method* ini adalah *method* yang berfungsi untuk mengubah data pada *database*.

Sumber : Perancangan

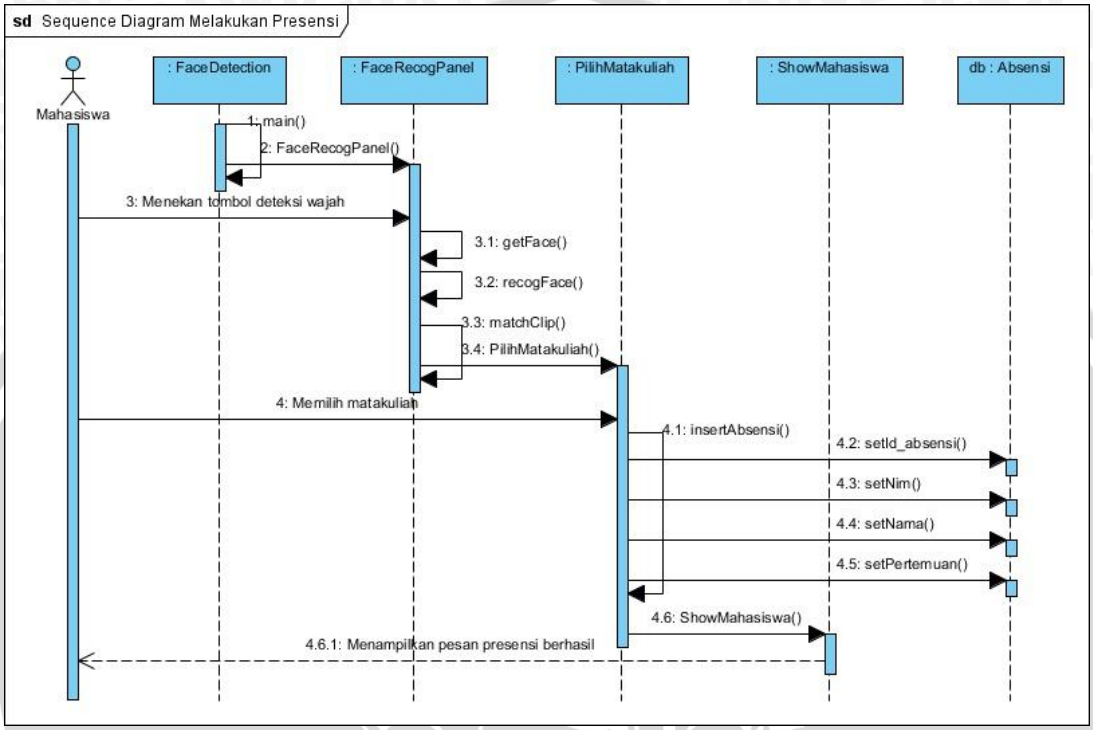
4.3.2 Diagram Sekuensial (*Sequence Diagram*)

Dalam metodologi berorientasi objek untuk menunjukkan bagaimana suatu objek berkomunikasi dengan objek lain dengan memperhatikan urutan waktu di modelkan dalam *sequence diagram.*

4.3.2.1 Diagram Sekuensial Mahasiswa

a. Melakukan Presensi

Diagram sekuensial ini digunakan untuk menggambarkan saat mahasiswa melakukan proses presensi. Diagram melakukan presensi ditunjukkan pada gambar 4.27.



Gambar 4.27 Diagram Sekuensial Melakukan Presensi
Sumber : Perancangan

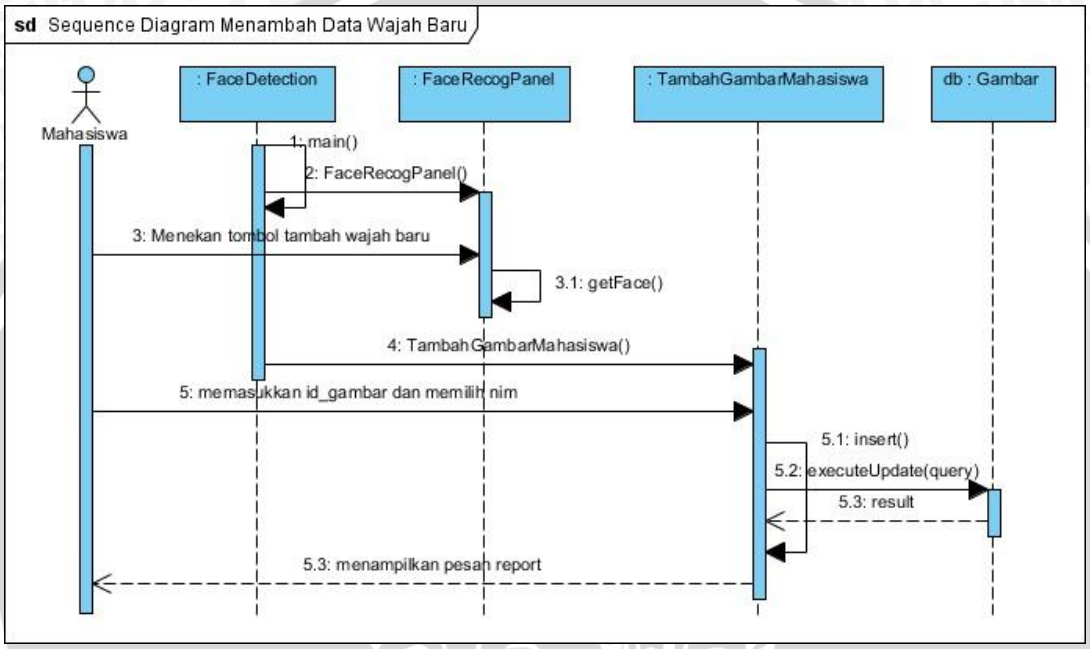
Penjelasan diagram sekuensial dari Gambar 4.27 sebagai berikut:

1. Saat aplikasi presensi mahasiswa pertama kali dijalankan, pada main dari *class FaceDetection* akan memanggil *FaceRecogPanel*.
2. Menampilkan *FaceRecogPanel* yang merupakan halaman utama sisi mahasiswa.
3. Mahasiswa menekan tombol deteksi wajah, kemudian sistem akan mengambil gambar wajah mahasiswa dan mencocokkan dengan data training wajah mahasiswa yang sebelumnya sudah terdapat pada database. Apabila dikenali, sistem akan menampilkan halaman PilihMatakuliah.
4. Mahasiswa memillih kode matakuliah yang akan dilakukan proses presensi. Setelah proses presensi berhasil, sistem akan menampilkan

halaman yang berisi data mahasiswa dan pesan bahwa proses presensi berhasil.

b. Menambah Data Wajah Baru

Diagram sekuensial ini menggambarkan bagaimana mahasiswa melakukan penambahan data wajah baru agar dapat dikenali oleh sistem. Diagram ini ditunjukkan pada gambar 4.28.



Gambar 4.28 Diagram Sekuensial Menambah Data Wajah Baru
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.28 sebagai berikut:

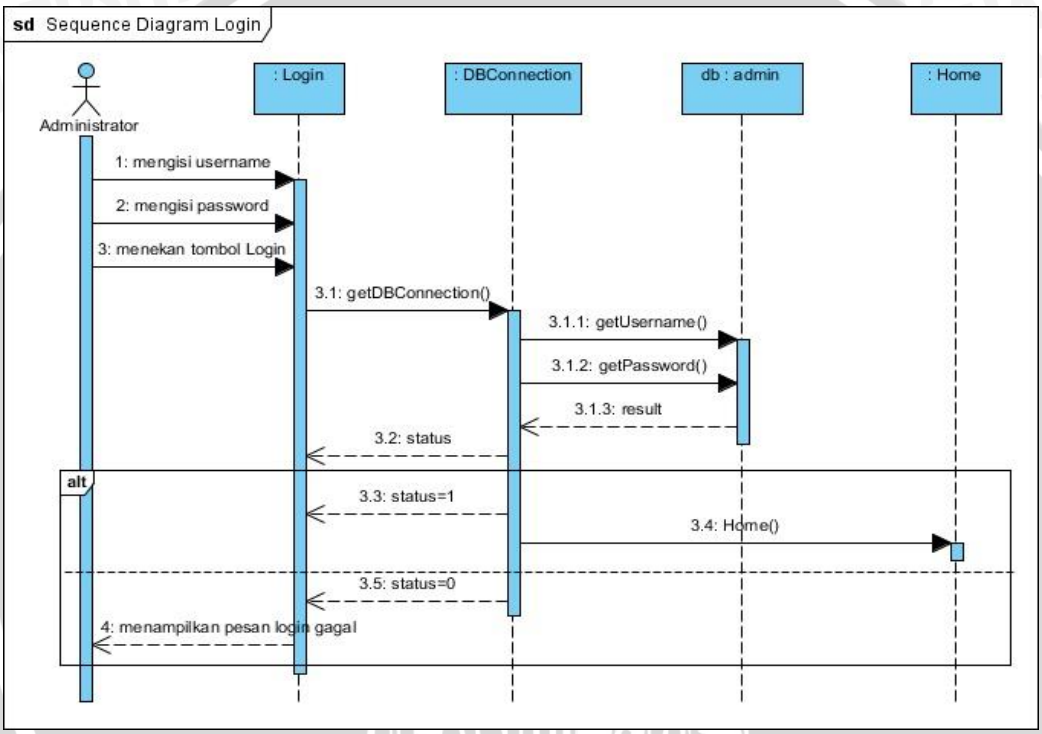
1. Saat aplikasi presensi mahasiswa pertama kali dijalankan, pada main dari *class FaceDetection* akan memanggil *FaceRecogPanel*.
2. Menampilkan *FaceRecogPanel* yang merupakan halaman utama sisi mahasiswa.
3. Mahasiswa menekan tombol tambah wajah baru, kemudian sistem akan mengambil gambar wajah mahasiswa.
4. *Class FaceDetection* akan memanggil *class* TambahGambarMahasiswa.
5. Mahasiswa memasukkan id gambar dan memilih nim. Sistem akan menjalankan operasi insert untuk memasukkan data pada basis data, sekaligus menyimpan gambar pada direktori *training image*. Basis data

akan mengirimkan result kepada sistem. Kemudian sistem akan menampilkan pesan bahwa data telah tersimpan.

4.3.2.2 Diagram Sekuensial *Administrator*

a. *Login*

Diagram ini sekuensial menggambarkan bagaimana *administrator* melakukan *login* untuk mengakses sistem. Diagram ini ditunjukkan pada gambar 4.29.



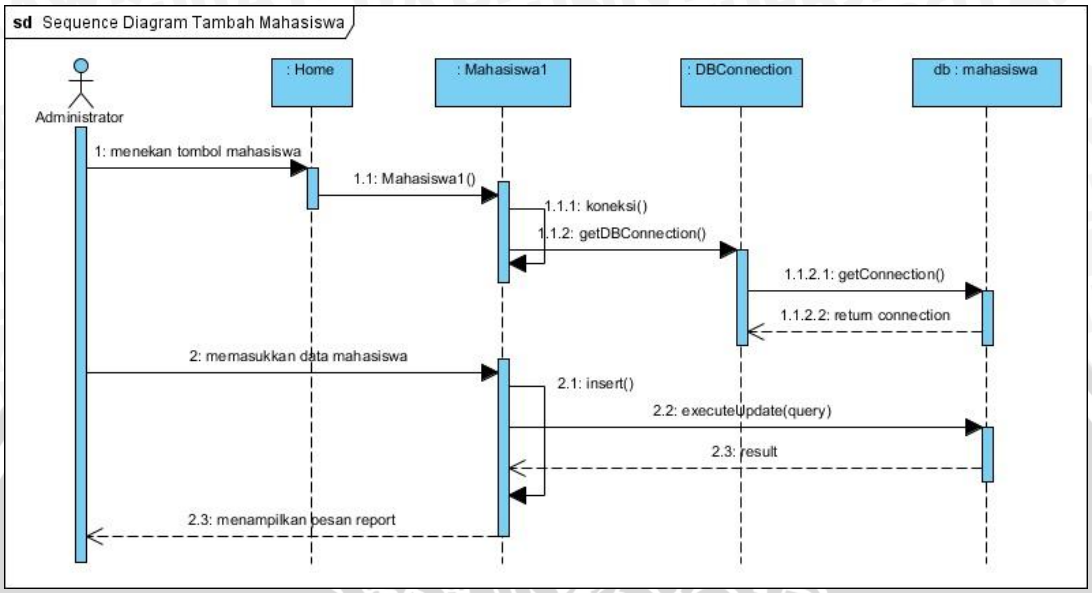
Gambar 4.29 Diagram Sekuensial *Login*
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.29 sebagai berikut:

1. *Administrator* memilih halaman *administrator*, kemudian sistem menampilkan halaman *Login*. *Administrator* mengisi *field username*.
2. *Administrator* mengisi *field password*.
3. Saat *administrator* menekan tombol *login*, sistem akan melakukan koneksi dengan basis data dan mengecek *username* dan *password* yang dimasukkan. Apabila valid, sistem akan menampilkan halaman utama sisi *administrator*.
4. Jika tidak valid, maka sistem akan menampilkan pesan *login* gagal.

b. Tambah Mahasiswa

Diagram ini menggambarkan proses menambahkan data mahasiswa. Diagram sekuensial tambah mahasiswa ditunjukkan pada gambar 4.30.



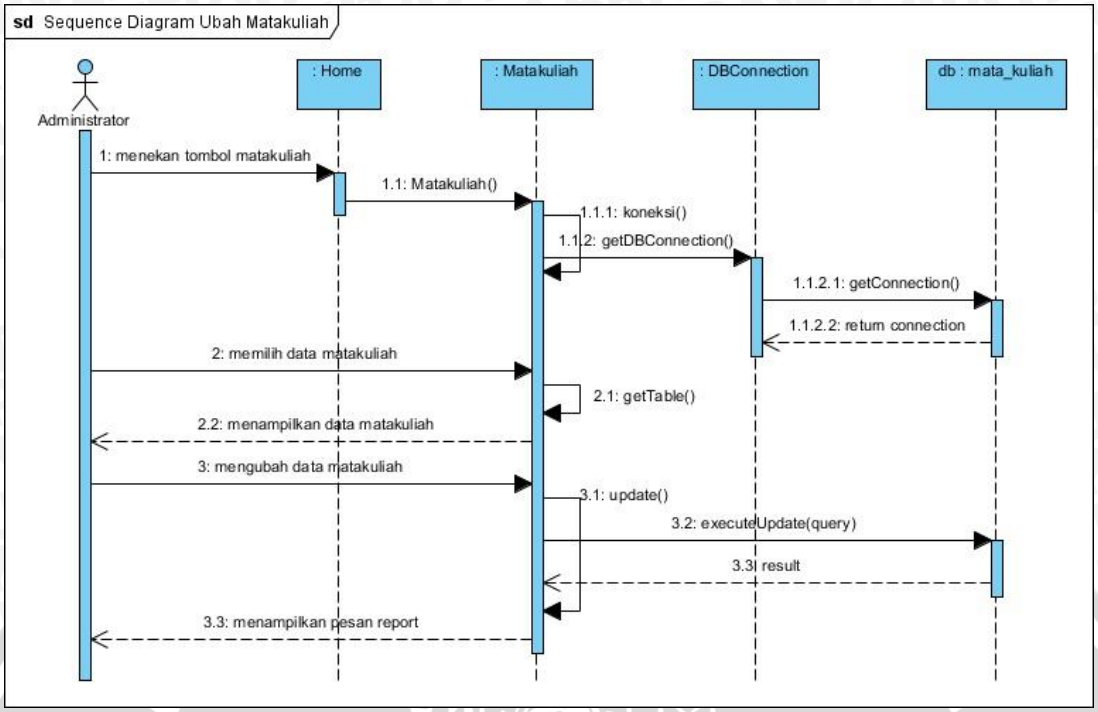
Gambar 4.30 Diagram Sekuensial Tambah Mahasiswa
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.30 sebagai berikut:

1. *Administrator* menekan tombol mahasiswa, kemudian sistem menampilkan halaman mahasiswa dan melakukan koneksi dengan basis data.
2. *Administrator* memasukkan data mahasiswa baru. Sistem akan menjalankan operasi insert untuk memasukkan data pada basis data. Basis data akan mengirimkan result kepada sistem. Kemudian sistem akan menampilkan pesan bahwa data telah tersimpan.

c. Ubah Matakuliah

Diagram ini menggambarkan proses untuk mengubah data matakuliah pada *database*. Diagram sekuensial ubah matakuliah ditunjukkan pada gambar 4.31.



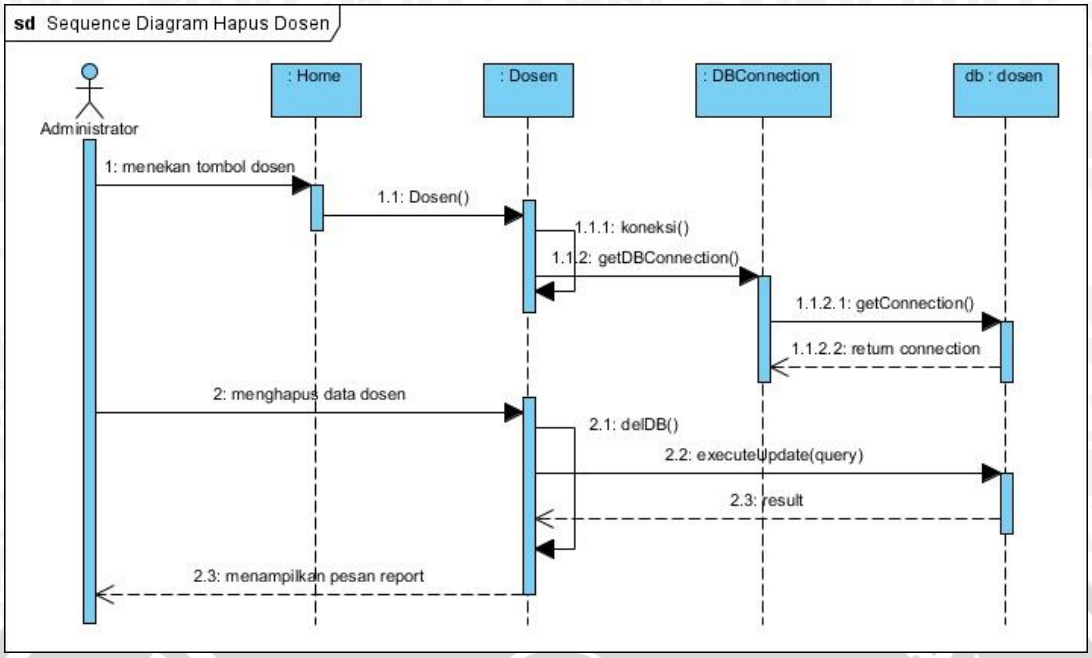
Gambar 4.31 Diagram Sekuensial Ubah Matakuliah
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.31 sebagai berikut:

1. *Administrator* menekan tombol matakuliah, kemudian sistem menampilkan halaman matakuliah dan melakukan koneksi dengan basis data.
2. *Administrator* memilih data matakuliah yang akan diubah. Sistem akan menampilkan data yang dipilih pada *form input* data.
3. *Administrator* mengubah data matakuliah. Sistem akan menjalankan operasi update dan mengeksekusi query ke basis data. Basis data akan mengirimkan result kepada sistem. Kemudian sistem akan menampilkan pesan bahwa data telah tersimpan.

d. Hapus Dosen

Diagram ini menggambarkan proses untuk menghapus data dosen pada *database*. Diagram sekuensial hapus dosen dijelaskan pada gambar 4.32.



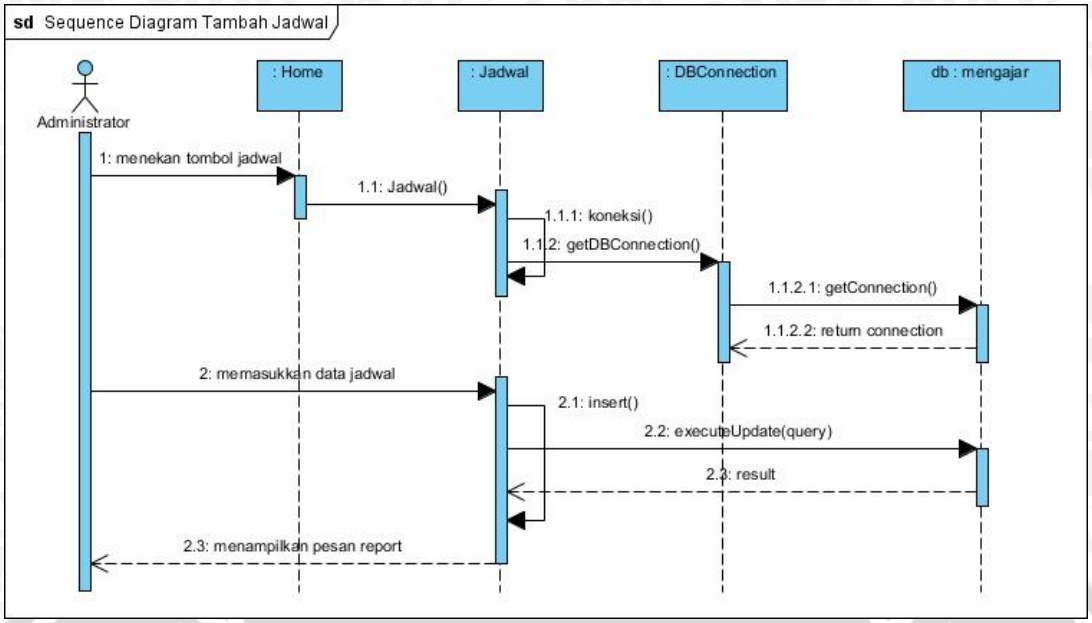
Gambar 4.32 Diagram Sekuensial Hapus Dosen
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.32 sebagai berikut:

1. *Administrator* menekan tombol dosen, kemudian sistem menampilkan halaman dosen dan melakukan koneksi dengan basis data.
2. *Administrator* memilih data matakuliah yang akan dihapus. Sistem akan menjalankan operasi delDb untuk menghapus data pada basis data. Basis data akan mengirimkan result kepada sistem. Kemudian sistem akan menampilkan pesan bahwa data telah terhapus.

e. Tambah jadwal

Diagram ini menggambarkan proses untuk melakukan penambahan data jadwal pada *database*. Diagram sekuensial tambah jadwal ditunjukkan pada gambar 4.33.



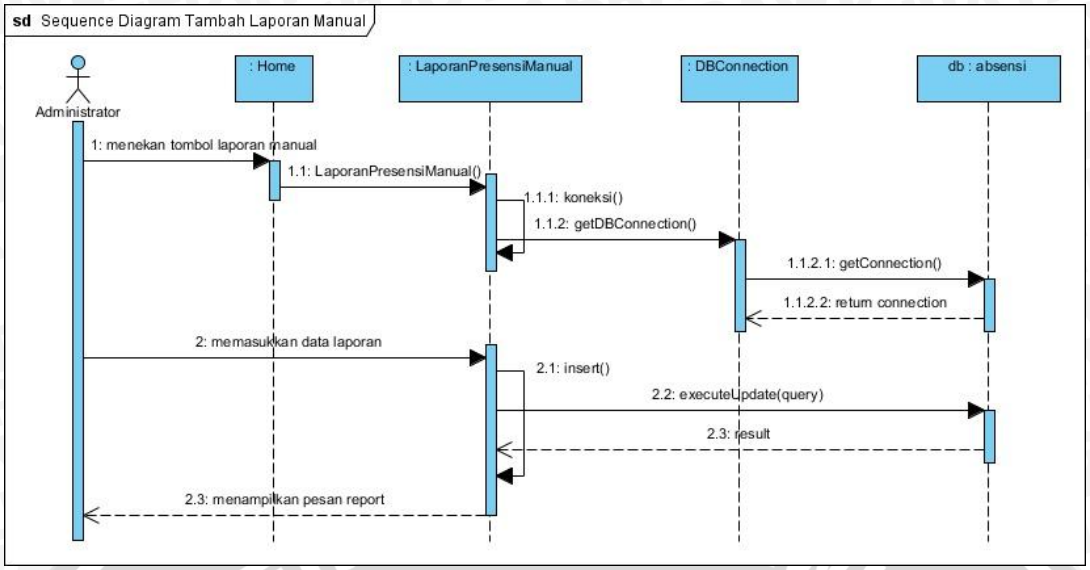
Gambar 4.33 Diagram Sekuensial Tambah Jadwal
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.33 sebagai berikut:

1. *Administrator* menekan tombol jadwal, kemudian sistem menampilkan halaman jadwal dan melakukan koneksi dengan basis data.
2. *Administrator* memasukkan data jadwal baru. Sistem akan menjalankan operasi insert untuk memasukkan data pada basis data. Basis data akan mengirimkan result kepada sistem. Kemudian sistem akan menampilkan pesan bahwa data telah tersimpan.

f. Tambah Laporan Manual

Diagram ini menggambarkan proses untuk menambahkan data presensi mahasiswa secara manual. Diagram sekuensial tambah laporan manual ditunjukkan pada gambar 4.34.



Gambar 4.34 Diagram Sekuensial Tambah Laporan Manual
Sumber : Perancangan

Penjelasan diagram sekuensial dari Gambar 4.34 sebagai berikut:

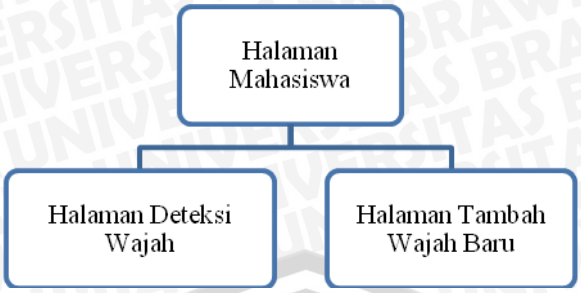
1. *Administrator* menekan tombol laporan, kemudian sistem menampilkan halaman laporan presensi manual dan melakukan koneksi dengan basis data.
2. *Administrator* memasukkan data presensi baru. Sistem akan menjalankan operasi insert untuk memasukkan data pada basis data. Basis data akan mengirimkan result kepada sistem. Kemudian sistem akan menampilkan pesan bahwa data telah tersimpan.

4.3.3 Perancangan Perangkat Lunak

Pada bagian ini akan dijelaskan tentang perancangan antarmuka Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*. Antarmuka perangkat lunak ini akan digunakan oleh pengguna untuk berinteraksi dengan sistem.

4.3.3.1 Antarmuka Mahasiswa

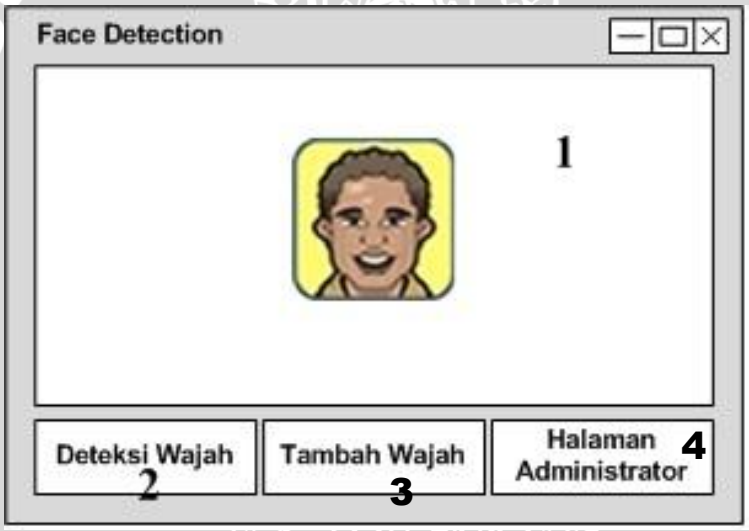
Antarmuka mahasiswa merupakan halaman yang disediakan untuk mahasiswa yang digunakan untuk melakukan presensi pada halaman deteksi wajah dan melakukan penambahan data wajah baru pada halaman tambah mahasiswa. *Site map* halaman mahasiswa ditunjukkan pada Gambar 4.35.



Gambar 4.35 *Site Map* Halaman Mahasiswa
Sumber : Perancangan

a. Melakukan Presensi

Antarmuka pengguna untuk mahasiswa terdiri dari halaman utama, dimana terdapat sebuah tombol yang digunakan untuk melakukan proses presensi. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_001_01. Gambar 4.36 akan menunjukkan perancangan tampilan antarmuka dari halaman utama mahasiswa.

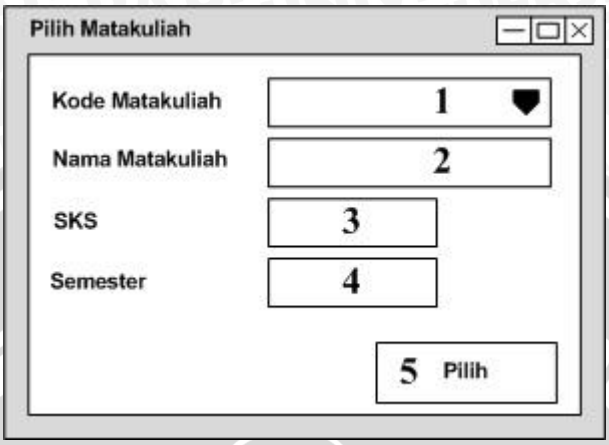


Gambar 4.36 Tampilan Antarmuka Halaman Utama Mahasiswa
Sumber : Perancangan

Gambar 4.36 memiliki keterangan sebagai berikut :

1. Layar yang sudah tersambung dengan *webcam.*
2. Tombol Deteksi Wajah untuk mulai melakukan proses presensi dan mengambil gambar wajah mahasiswa.
3. Tombol Tambah Wajah untuk menambah citra wajah baru bagi mahasiswa agar dapat dikenali sistem.
4. Tombol Halaman Administrator untuk menuju halaman *login administrator*.

Untuk melakukan proses presensi mahasiswa, mahasiswa menekan tombol Deteksi Wajah pada halam utama Mahasiswa. Setelah gambar mahasiswa berhasil dikenali maka sistem akan menampilkan form untuk memilih matakuliah apa yang akan dilakukan proses presensi.



Gambar 4.37 Tampilan Antarmuka Halaman Pilih Matakuliah
Sumber : Perancangan

Gambar 4.37 memiliki keterangan sebagai berikut :

1. *Combo Box* Kode Matakuliah untuk memilih matakuliah apa yang akan dilakukan presensi.
2. *Field* Nama Matakuliah akan otomatis terisi sesuai dengan kode matakuliah yang dipilih.
3. *Field* SKS akan otomatis terisi sesuai dengan kode matakuliah yang dipilih.
4. *Field* Semester akan otomatis terisi sesuai dengan kode matakuliah yang dipilih.
5. Tombol Plih wajah untuk menjalankan proses memilih matakuliah.

b. Halaman Tambah Wajah Mahasiswa

Halaman tambah mahasiswa merupakan antarmuka yang digunakan untuk menambah data mahasiswa. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_001_02. Halaman ini akan ditampilkan setelah mahasiswa menekan tombol Tambah Wajah pada halaman utama mahasiswa. Gambar 4.38 akan menunjukkan perancangan tampilan antarmuka dari halaman tambah mahasiswa.

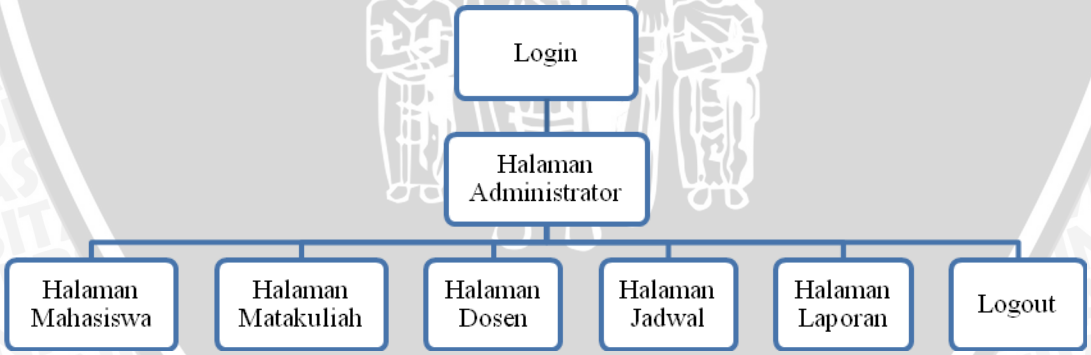
Gambar 4.38 Tampilan Antarmuka Halaman Tambah Wajah Mahasiswa
Sumber : Perancangan

Gambar 4.38 memiliki keterangan sebagai berikut :

1. *Field Id* Gambar untuk mengisi id gambar mahasiswa.
2. *Combo Box* nim untuk memilih nim mahasiswa.
3. Tombol Simpan untuk menjalankan proses menambah mahasiswa baru.

4.3.3.2 Antarmuka Administrator

Antarmuka *administrator* merupakan halaman yang disediakan untuk *administrator* yang digunakan untuk mengelola data mahasiswa, matakuliah, dosen, jadwal dan laporan presensi. *Site map* halaman mahasiswa ditunjukkan pada Gambar 4.39.

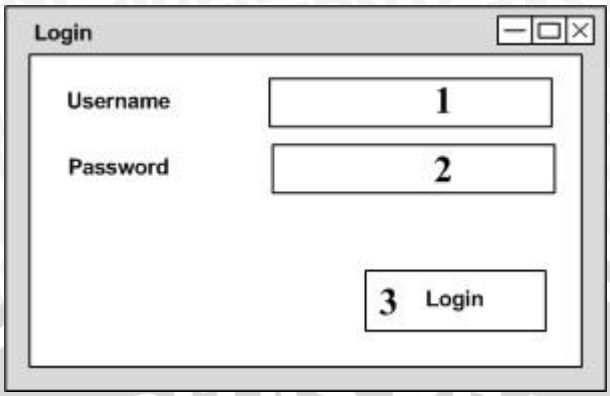


Gambar 4.39 *Site Map* Halaman *Administrator*
Sumber : Perancangan

a. Halaman *Login*

Halaman *login* merupakan salah satu antarmuka pengguna untuk aplikasi *administrator*. Perancangan antarmuka ini mengacu pada spesifikasi

kebutuhan SRS_002_01. Gambar 4.40 akan menunjukkan perancangan tampilan antarmuka dari halaman *login*.



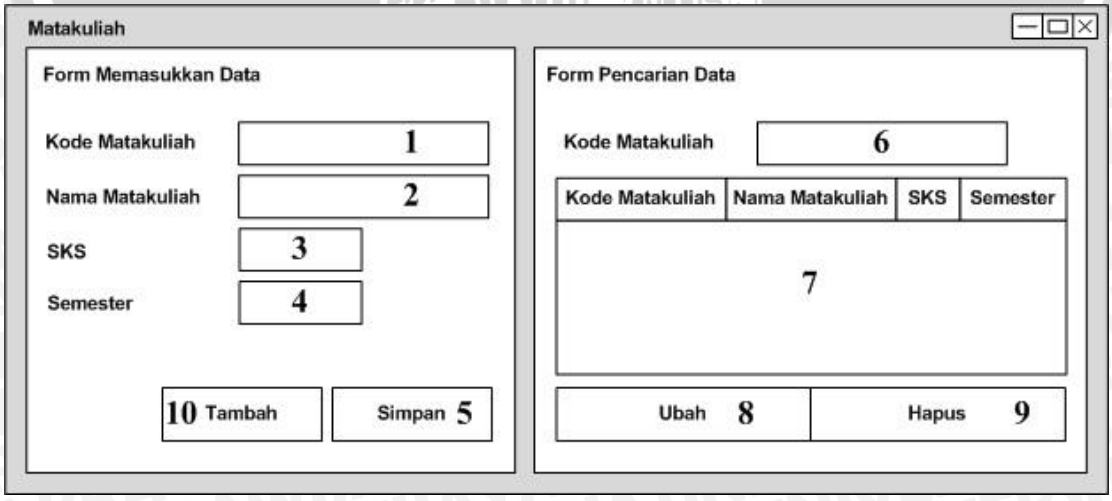
Gambar 4.40 Tampilan Antarmuka Halaman *Login*
Sumber : Perancangan

Gambar 4.40 memiliki keterangan sebagai berikut :

1. *Field Username* untuk mengisi *username administrator*.
2. *Field Password* untuk mengisi *password administrator*.
3. Tombol *Login* untuk menjalankan proses *login* menuju halaman utama sisi *administrator*.

b. Halaman Matakuliah

Halaman matakuliah merupakan antarmuka yang digunakan *administrator* untuk mengelola data matakuliah. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_002_04. Gambar 4.41 akan menunjukkan perancangan tampilan antarmuka halaman matakuliah.



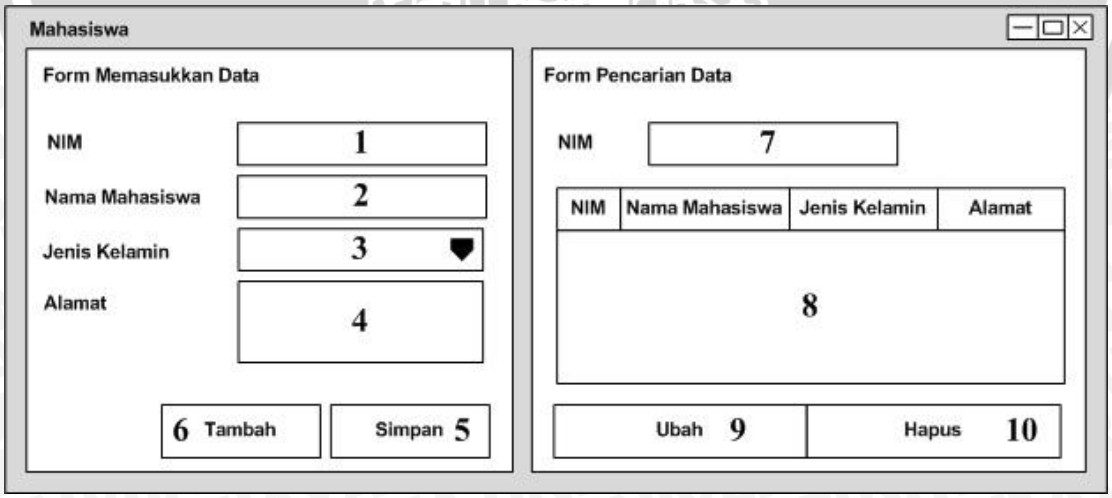
Gambar 4.41 Tampilan Antarmuka Halaman Matakuliah
Sumber : Perancangan

Gambar 4.41 memiliki keterangan sebagai berikut :

1. *Field* Kode Matakuliah untuk memasukkan kode matakuliah.
2. *Field* Nama Matakuliah untuk memasukkan nama matakuliah.
3. *Field* SKS untuk mengisi jumlah sks matakuliah.
4. *Field* Semester untuk mengisi semester matakuliah.
5. Tombol Simpan untuk menjalankan proses menyimpan data matakuliah setelah di ubah.
6. *Field* Kode Matakuliah untuk memasukkan kode matakuliah yang dicari.
7. Tabel yang akan menampilkan data matakuliah yang terdapat pada database. Terdiri dari kode matakuliah, nama matakuliah, sks dan semester.
8. Tombol Ubah untuk menjalankan proses mengubah data matakuliah.
9. Tombol Hapus untuk menghapus data matakuliah.
10. Tombol Tambah untuk menambah data matakuliah.

c. Halaman Mahasiswa

Halaman mahasiswa merupakan antarmuka yang digunakan *administrator* untuk mengelola data mahasiswa. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_002_02. Gambar 4.42 akan menunjukkan perancangan tampilan antarmuka halaman mahasiswa.



Gambar 4.42 Tampilan Antarmuka Halaman Mahasiswa
Sumber : Perancangan

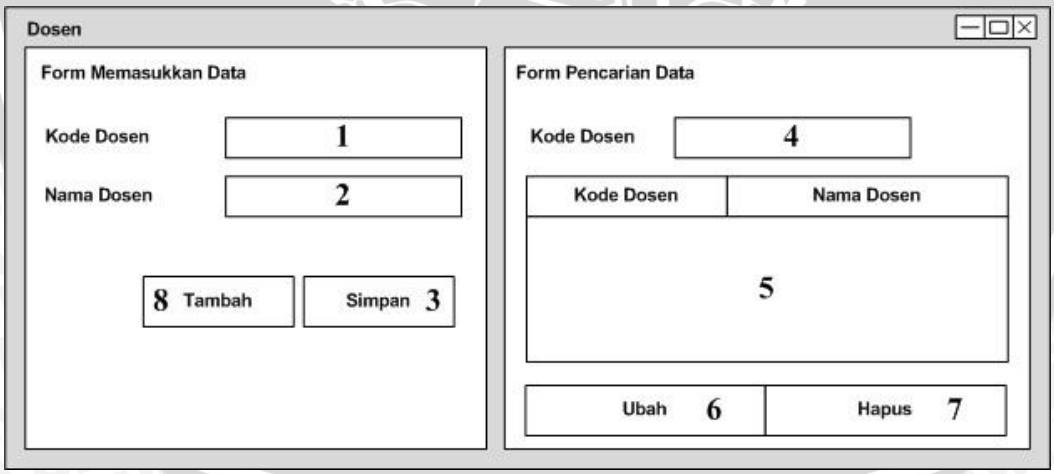
Gambar 4.42 memiliki keterangan sebagai berikut :

1. *Field* NIM untuk memasukkan nim mahasiswa

2. *Field* Nama Mahasiswa untuk memasukkan nama mahasiswa.
3. *Radio Button* Laki-laki dan Perempuan untuk memilih jenis kelamin mahasiswa.
4. *Field* Alamat untuk mengisi alamat mahasiswa.
5. Tombol Simpan untuk menyimpan data mahasiswa setelah di ubah.
6. Tombol Tambah untuk menjalankan proses menambah data mahasiswa.
7. *Field* NIM untuk memasukkan nim yang dicari.
8. Tabel yang akan menampilkan data mahasiswa yang terdapat pada database. Terdiri dari nim, nama mahasiswa, jenis kelamin, alamat dan no telp.
9. Tombol Ubah untuk menjalankan proses mengubah data mahasiswa.
10. Tombol Hapus untuk menghapus data mahasiswa.

d. Halaman Dosen

Halaman dosen merupakan antarmuka yang digunakan *administrator* untuk mengelola data dosen. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_002_03. Gambar 4.43 akan menunjukkan perancangan tampilan antarmuka halaman dosen.



Gambar 4.43 Tampilan Antarmuka Halaman Dosen
Sumber : Perancangan

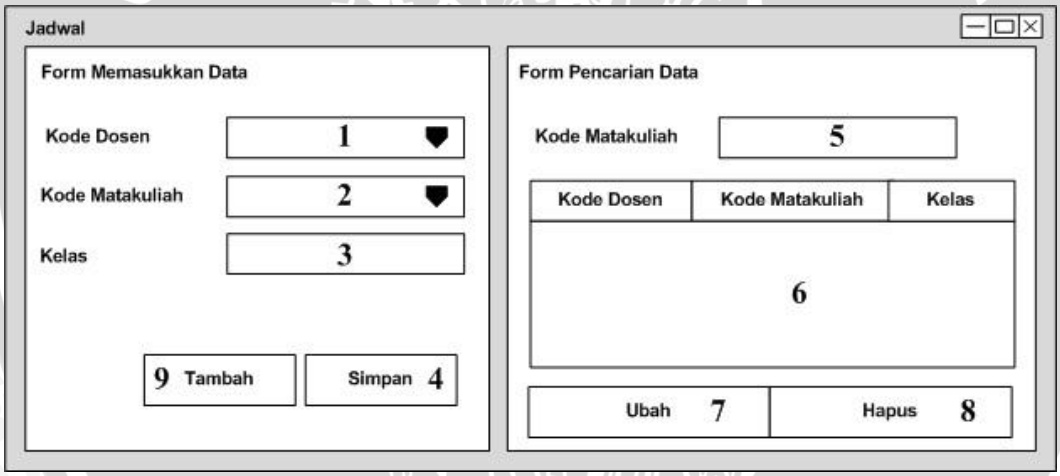
Gambar 4.43 memiliki keterangan sebagai berikut :

1. *Field* Kode Dosen untuk memasukkan kode dosen.
2. *Field* Nama Dosen untuk memasukkan nama dosen.

3. Tombol Simpan untuk menjalankan proses menyimpan data Dosen setelah di ubah.
4. *Field* Kode Dosen untuk memasukkan kode dosen yang dicari.
5. Tabel yang akan menampilkan data dosen yang terdapat pada database. Terdiri dari kode dosen dan nama dosen.
6. Tombol Ubah untuk menjalankan proses mengubah data dosen.
7. Tombol Hapus untuk menghapus data dosen.
8. Tombol Tambah untuk menambah data dosen.

e. Halaman Jadwal

Halaman jadwal merupakan antarmuka yang digunakan *administrator* untuk mengelola data jadwal. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_002_05. Gambar 4.44 akan menunjukkan perancangan tampilan antarmuka halaman jadwal.



Gambar 4.44 Tampilan Antarmuka Halaman Jadwal
Sumber : Perancangan

Gambar 4.44 memiliki keterangan sebagai berikut :

1. *Combo Box* Kode Dosen untuk memilih kode dosen.
2. *Combo Box* Kode Matakuliah untuk memilih kode matakuliah.
3. *Field* Kelas untuk memasukkan kelas.
4. Tombol Simpan untuk menjalankan proses menyimpan data jadwal setelah di ubah.
5. *Field* Kode Matakuliah untuk memasukkan kode matakuliah yang dicari.

6. Tabel yang akan menampilkan data jadwal yang terdapat pada database. Terdiri dari kode dosen, kode matakuliah dan kelas.
7. Tombol Ubah untuk menjalankan proses mengubah data jadwal.
8. Tombol Hapus untuk menghapus data jadwal.
9. Tombol tambah untuk menambah data jadwal

f. Halaman Laporan

Halaman laporan merupakan antarmuka yang digunakan *administrator* untuk mengelola data presensi. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_002_06. Gambar 4.45 akan menunjukkan perancangan tampilan antarmuka halaman laporan.

Gambar 4.45 Tampilan Antarmuka Halaman Laporan
Sumber : Perancangan

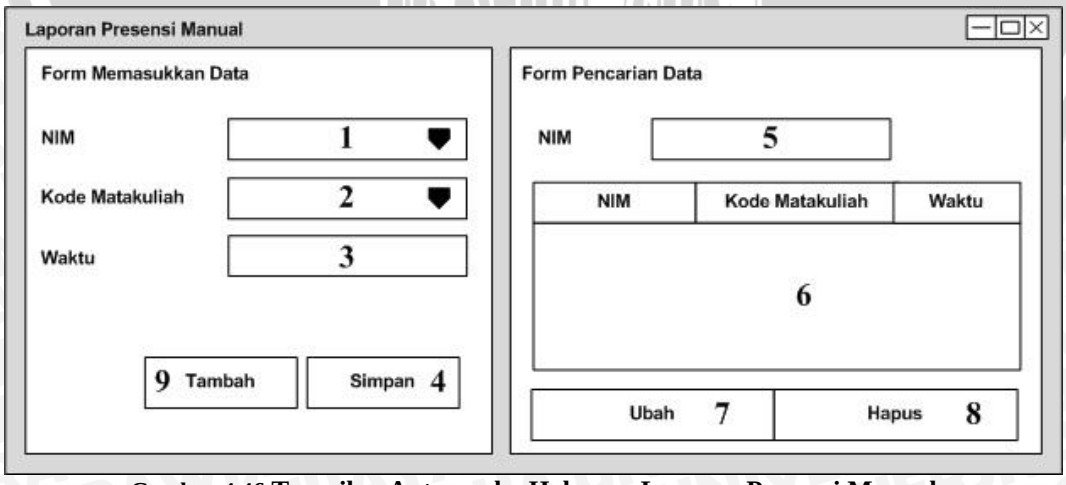
Gambar 4.45 memiliki keterangan sebagai berikut :

1. *Combo Box* Kode Matakuliah untuk memilih kode matakuliah.

2. *Field* Nama Matakuliah akan otomatis terisi saat memilih kode matakuliah.
3. *Field* Semester akan otomatis terisi saat memilih kode matakuliah.
4. *Field* SKS untuk akan otomatis terisi saat memilih kode matakuliah.
5. *Combo Box* Kode Dosen untuk memilih kode dosen.
6. *Field* Nama Dosen akan otomatis terisi saat memilih kode dosen.
7. *Field* Kelas akan otomatis terisi saat memilih kode dosen.
8. Tombol Simpan untuk menjalankan proses menyimpan data presensi.
9. *Field* NIM untuk memasukkan nim yang dicari.
10. Tabel yang akan menampilkan data mahasisa yang melakukan presensi. Terdiri dari nim, nama mahasiswa, pertemuan 1 sampai pertemuan 16.
11. Tombol Hapus untuk menghapus data mahasiswa.

g. Halaman Laporan Presensi Manual

Halaman laporan presensi manual merupakan antarmuka yang digunakan *administrator* untuk mengelola data presensi secara manual. Halaman ini digunakan apabila sistem melakukan kesalahan dalam mengenali wajah, sehingga data presensi mahasiswa harus dimasukkan secara manual. Perancangan antarmuka ini mengacu pada spesifikasi kebutuhan SRS_002_06. Gambar 4.46 akan menunjukkan perancangan tampilan antarmuka halaman laporan presensi manual.



Gambar 4.46 Tampilan Antarmuka Halaman Laporan Presensi Manual
Sumber : Perancangan

Gambar 4.46 memiliki keterangan sebagai berikut :

1. *Combo Box* NIM untuk memilih nim mahasiswa.
2. *Combo Box* Kode Matakuliah untuk memilih kode matakuliah.
3. *Field* Waktu untuk memasukkan waktu presensi, berupa tanggal dan jam.
4. Tombol Simpan untuk menjalankan proses menyimpan data presensi mahasiswa setelah di ubah.
5. *Field* NIM untuk memasukkan nim mahasiswa yang dicari.
6. Tabel yang akan menampilkan data presensi mahasiswa yang terdapat pada database. Terdiri dari NIM, kode matakuliah dan waktu.
7. Tombol Ubah untuk menjalankan proses mengubah data presensi mahasiswa.
8. Tombol Hapus untuk menghapus data presensi mahasiwa.
9. Tombol Tambah untuk menambah data presensi mahasiswa.

BAB V
IMPLEMENTASI

Bab ini membahas mengenai tahapan implementasi Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan sistem. Pembahasan terdiri atas penjelasan tentang spesifikasi sistem, batasan–batasan dalam implementasi, implementasi basis data, implementasi tiap *class* pada *file* program, implementasi algoritma, dan implementasi antarmuka.

5.1 Spesifikasi Sistem

Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Compaq Presario CQ40	
Processor	AMD Turion(tm) X2 Dual-Core Mobile RM-75 2.20 GHz
Memory (RAM)	1.00 GB
Harddisk	FUJITSU MHZ2320BH G2 ATA
Graphic Card	ATI Radeon HD 3200 Graphics
Monitor	14.1" High-definition widescreen
PC Camera	
Model	HP Webcam 101
Image Resolution	1,3 Mpx

Sumber : Implementasi

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Komputer

Compaq Presario CQ40	
Operating System	Microsoft Windows 7 Ultimate 32-bit
Programming Language	Java
Software Development Kit	JDK 1.7
Database Management System	My SQL
Integrated Development Environment	Oracle NetBeans IDE 7.1.1

Sumber : Implementasi

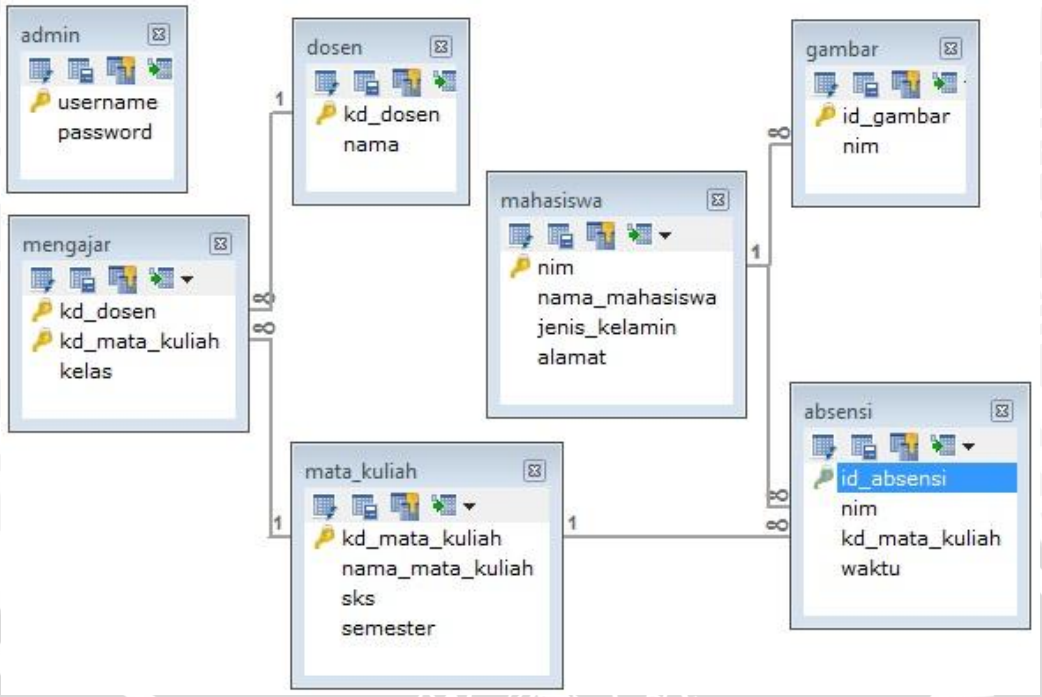
5.2 Batasan – Batasan Implementasi

Beberapa batasan dalam mengimplementasikan Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* sebagai berikut :

1. Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dirancang dan dijalankan dengan menggunakan *Java Application*.
2. *Database Management System* yang digunakan adalah MySQL.
3. Citra masukan wajah yang digunakan sebagai *training image* memiliki format *.png*
4. *Training image* yang digunakan diambil dari 5 orang mahasiswa, dengan masing-masing mahasiswa memiliki 3 pose, yaitu pose standar menghadap ke depan, samping kiri dan samping kanan.

5.3 Implementasi Basis Data

Implementasi penyimpanan data dilakukan dengan *database management system* MySQL. Hasil implementasi penyimpanan data ini berupa *script – script* SQL. Hasil implementasi SQL pada *database* ini dimodelkan dalam diagram konseptual *entity relationship.* Gambar 5.1 menggambarkan diagram konseptual *entitiy relationship* dari Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface.*



Gambar 5.1 Diagram ER konseptual dari Sistem Presensi Mahasiswa
Sumber : Implementasi

5.3.1 Data Definition Table (DDL) Create Table Admin

Data definition table create table admin berfungsi untuk membuat tabel *admin*.

```
Create Table

CREATE TABLE `admin` (
  `username` varchar(20) NOT NULL,
  `password` varchar(10) NOT NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Gambar 5.2 *Data Definition Table Create Table Admin*
Sumber : Implementasi

5.3.2 Data Definition Table (DDL) Create Table Dosen

Data definition table create table dosen berfungsi untuk membuat tabel dosen.

```
Create Table

CREATE TABLE `dosen` (
  `kd_dosen` varchar(25) NOT NULL,
  `nama` varchar(50) NOT NULL,
  PRIMARY KEY (`kd_dosen`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Gambar 5.3 *Data Definition Table Create Table* Dosen
Sumber : Implementasi

5.3.3 Data Definition Table (DDL) Create Table Mengajar

Data definition table create table mengajar berfungsi untuk membuat tabel mengajar.

```
Create Table

CREATE TABLE `mengajar` (
  `kd_dosen` varchar(25) NOT NULL,
  `kd_mata_kuliah` varchar(15) NOT NULL,
  `kelas` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`kd_dosen`,`kd_mata_kuliah`),
  KEY `kd_mata_kuliah` (`kd_mata_kuliah`),
  CONSTRAINT `mengajar_ibfk_1` FOREIGN KEY (`kd_dosen`) REFERENCES `dosen` (`kd_dosen`)
 ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `mengajar_ibfk_2` FOREIGN KEY (`kd_mata_kuliah`) REFERENCES `mata_kuliah` (`kd_mata_kuliah`)
 ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Gambar 5.4 *Data Definition Table Create Table* Mengajar
Sumber : Implementasi

5.3.4 Data Definition Table (DDL) Create Table Mata_kuliah

Data definition table create table mata_kuliah berfungsi untuk membuat tabel Mata_Kuliah.

```
Create Table

CREATE TABLE `mata_kuliah` (
  `kd_mata_kuliah` varchar(15) NOT NULL,
  `nama_mata_kuliah` varchar(35) NOT NULL,
  `sks` tinyint(3) NOT NULL,
  `semester` varchar(25) NOT NULL,
  PRIMARY KEY (`kd_mata_kuliah`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Gambar 5.5 *Data Definition Table Create Table* Mata_kuliah
Sumber : Implementasi

5.3.5 Data Definition Table (DDL) Create Table Mahasiswa

Data definition table create table mahasiswa berfungsi untuk membuat tabel mahasiswa.

```
Create Table

CREATE TABLE `mahasiswa` (
  `nim` varchar(15) NOT NULL,
  `nama_mahasiswa` varchar(60) NOT NULL,
  `jenis_kelamin` char(1) NOT NULL,
  `alamat` text NOT NULL,
  PRIMARY KEY (`nim`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Gambar 5.6 *Data Definition Table Create Table* Mahasiswa
Sumber : Implementasi

5.3.6 Data Definition Table (DDL) Create Table Gambar

Data definition table create table gambar berfungsi untuk membuat tabel gambar.

```
Create Table

CREATE TABLE `gambar` (
  `id_gambar` varchar(35) NOT NULL,
  `nim` varchar(15) NOT NULL,
  PRIMARY KEY (`id_gambar`),
  KEY `nim` (`nim`),
  CONSTRAINT `gambar_ibfk_1` FOREIGN KEY (`nim`) REFERENCES `mahasiswa` (`nim`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Gambar 5.7 *Data Definition Table Create Table* Gambar
Sumber : Implementasi

5.3.7 Data Definition Table (DDL) Create Table Absensi

Data definition table create table absensi berfungsi untuk membuat tabel absensi.

```
Create Table

CREATE TABLE `absensi` (
  `id_absensi` int(11) NOT NULL AUTO_INCREMENT,
  `nim` varchar(15) NOT NULL,
  `kd_mata_kuliah` varchar(15) NOT NULL,
  `waktu` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id_absensi`),
  KEY `nim` (`nim`),
  KEY `kd_mata_kuliah` (`kd_mata_kuliah`),
  CONSTRAINT `absensi_ibfk_1` FOREIGN KEY (`nim`) REFERENCES `mahasiswa` (`nim`)
 ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `absensi_ibfk_2` FOREIGN KEY (`kd_mata_kuliah`) REFERENCES `mata_kuliah` (`kd_mata_kuliah`)
 ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=latin1
```

Gambar 5.8 *Data Definition Table Create Table* Absensi
Sumber : Implementasi

5.4 Implementasi *Class* Dan *Interface* Pada File Program

Setiap *class* yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program dengan ekstensi *.java. Tabel 5.3 menjelaskan mengenai pasangan antara *class* dengan *file* program yang digunakan untuk mengimplementasikannya.

Tabel 5.3 Implementasi *Class* Pada Kode Program

No.	*Package*	Nama *Class* atau *Interface*	Nama *File* Program
1	org.facerec	Absensi	Absensi.java
2	org.facerec	EigenvalueDecomp	EigenvalueDecomp.java
4	org.facerec	FaceRecognition	FaceRecognition.java
5	org.facerec	MataKuliah	MataKuliah.java

6	org.facerec	Laporan	Laporan.java
7	org.facerec	MatchResult	MatchResult.java
8	org.facerec	Mahasiswa	Mahasiswa.java
9	org.facerec	Matrix2D	Matrix2D.java
10	org.facerec.db	DBConnection	DBConnection.java
11	org.facerec.gui	Dosen	Dosen.java
12	org.facerec.gui	FaceDetection	FaceDetection.java
13	org.facerec.gui	FaceRecogPanel	FaceRecogPanel.java
14	org.facerec.gui	Home	Home.java
15	org.facerec.gui	Jadwal	Jadwal.java
16	org.facerec.gui	LaporanPresensi	LaporanPresensi.java
17	org.facerec.gui	*Login*	*Login*.java
18	org.facerec.gui	Mahasiswa1	Mahasiswa1.java
19	org.facerec.gui	Matakuliah	Matakuliah.java
20	org.facerec.gui	MotionDetectionControl	MotionDetectionControl.java
21	org.facerec.gui	MotionDetectionEffect	MotionDetectionEffect.java
22	org.facerec.gui	PilihMatakuliah	PilihMatakuliah.java
23	org.facerec.gui	TampilMahasiswa	TampilMahasiswa.java
24	org.facerec.gui	TableModelLaporan	TableModelLaporan.java
25	org.facerec.gui	TambahGambarMahasiswa	TambahGambarMahasiswa.java
26	org.facerec.util	BuildEigenFaces	BuildEigenFaces.java
27	org.facerec.util	FaceBundle	FaceBundle.java
28	org.facerec.util	FileUtils	FileUtils.java
29	org.facerec.util	ImageDistanceInfo	ImageDistanceInfo.java
30	org.facerec.util	ImageUtils	ImageUtils.java
31	org.facerec.util	JMFCapture	JMFCapture.java

Sumber : Implementasi

5.5 Implementasi Prosedur atau *Method*

Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* mempunyai beberapa proses (*method*) utama yang terbagi dalam beberapa *class*. Pada penulisan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja sehingga tidak semua implementasi algoritma *method* akan dicantumkan. Implementasi algoritma ini akan direpresentasikan dalam bentuk *code* dengan bahasa pemrograman Java.

5.5.1 Implementasi *Method* Membangun Bundle

Sebagian besar kode PCA penting dipanggil dari *method* `makeBundle()` di kelas BuildEigenFaces. `makeBundle()` menciptakan objek *eigenvector/eigenvalue* FaceBundle untuk file gambar pelatihan tertentu, dan juga menyimpan tiap eigenvector sebagai gambar di *eigenfaces/*. Gambar 5.9 merupakan implementasi dari *method* `makeBundle()`.

```
1.  private static FaceBundle makeBundle(ArrayList<String>
2.        fnms)
3.
4.    {
5.      BufferedImage[] ims =
6.        FileUtils.loadTrainingIms(fnms);
7.
8.      Matrix2D imsMat = convertToNormMat(ims);
9.      double[] avgImage =
10.       imsMat.getAverageOfEachColumn();
11.     imsMat.subtractMean();
12.
13.     Matrix2D imsDataTr = imsMat.transpose();
14.     Matrix2D covarMat = imsMat.multiply(imsDataTr);
15.
16.     EigenvalueDecomp egValDecomp =
17.       covarMat.getEigenvalueDecomp();
18.     double[] egVals = egValDecomp.getEigenValues();
19.     double[][] egVecs = egValDecomp.getEigenVectors();
20.
21.     sortEigenInfo(egVals, egVecs);
22.
23.     Matrix2D egFaces = getNormEgFaces(imsMat, new
24.       Matrix2D(egVecs));
25.
26.     System.out.println("\nSaving Eigenfaces as
27.       images...");
28.     FileUtils.saveEFIms(egFaces, ims[0].getWidth());
29.     System.out.println("Saving done\n");
30.
31.     return new FaceBundle(fnms, imsMat.toArray(),
32.       avgImage,
33.                   egFaces.toArray(), egVals,
        ims[0].getWidth(), ims[0].getHeight());
    }
```

Gambar 5.9 Implementasi *Method* `makeBundle()`
Sumber : Implementasi

Penjelasan implementasi *method* `makebundle()` pada Gambar 5.9 yaitu:

1. Baris 5 merupakan proses meload seluruh gambar menjadi gambar *grayscale*.
2. Baris 7-9 merupakan proses merubah gambar ke dalam matriks. Setiap baris merupakan gambar yang telah dinormalisasi. Hitung rata-rata dari tiap gambar, kemudian kurangi nilai tiap gambar dengan rata-rata yang didapat.
3. Baris 11-12 merupakan proses menghitung matriks kovarian.
4. Baris 14-17 merupakan proses menghitung *eigenvalue* dan *eigenvector* dari matriks kovarian.

5. Baris 19 merupakan proses mengurutkan *eigenvariable* dan *eigenvector* secara menurun (dari besar ke kecil) sesuai nilai *eigenvalue*.
6. Baris 21-22 merupakan proses mengalikan transpose dari *eigenvector* dengan matriks gambar untuk mendapatkan *eigenfaces*.
7. Baris 24-27 merupakan proses menyimpan *eigenfaces* sebagai *file* gambar.
8. 29-32 merupakan proses pengembalian nilai faceBundle baru.

5.5.2 Implementasi *Method* Menghitung Bobot *Training Image*

Konstrukstor pada *class* FaceRecognition menggunakan *method* `calcWeights()` di *class* FaceBundle untuk membuat bobot pada training image. Gambar 5.10 merupakan implementasi dari *method* `calcWeights()`.

```
1.  public double[][] calcWeights(int numEFs)
2.    {
3.      Matrix2D imsMat = new Matrix2D(imageRows);
4.
5.      Matrix2D facesMat = new Matrix2D(eigenFaces);
6.      Matrix2D facesSubMatTr =
7.        facesMat.getSubMatrix(numEFs).transpose();
8.
9.      Matrix2D weights  = imsMat.multiply(facesSubMatTr);
10.     return weights .toArray();
11.   }
```

Gambar 5.10 Implementasi *Method* `calcWeights()`
Sumber : Implementasi

Penjelasan implementasi *method* `calcWeights()` pada Gambar 5.10 yaitu:

1. Baris 3 merupakan inisiasi kelas dimana setiap baris berisi gambar pelatihan.
2. Baris 5-7 merupakan proses untuk mendapatkan submatriks *eigenfaces*.
3. Baris 9-10 merupakan proses mengalikan matriks gambar dengan hasil *transpose* submatriks dari *eigenfaces* untuk mendapatkan array bobot.

5.5.3 Implementasi *Method* Membandingkan Gambar

Method `findMatch()` merupakan method yang melaksanakan proses membandingkan gambar baru dengan *training image*. Proses ini menggunakan *eigenfaces* dan *eigenvector* yang diambil dari objek FaceBundle. Gambar 5.11 merupakan implementasi dari *method* `findMatch()`.

```
1.      private MatchResult findMatch(BufferedImage im)
```

```
2.      {
3.          double[] imArr =
4.      ImageUtils.createArrFromIm(im);
5.
6.          Matrix2D imMat = new Matrix2D(imArr, 1);
7.          imMat.normalise();
8.
9.          imMat.subtract(new
10.     Matrix2D(bundle.getAvgImage(), 1));
11.         Matrix2D imWeights = getImageWeights(numEFs,
12.     imMat);
13.
14.         double[] dists = getDists(imWeights);
15.         ImageDistanceInfo distInfo =
16.     getMinDistInfo(dists);
17.
18.         ArrayList<String> imageFNms =
19.     bundle.getImageFnms();
20.         String matchingFNm = imageFNms.get(
21.     distInfo.getIndex() );
22.
23.         double minDist = Math.sqrt( distInfo.getValue()
24.     );
25.
26.         return new MatchResult(matchingFNm, minDist);
    }
```

Gambar 5.11 Implementasi *Method* `findMatch()`
Sumber : Implementasi

Penjelasan implementasi *method* `findMatch()` pada Gambar 5.11 yaitu:

1. Baris 3 merupakan proses mengubah gambar menjadi array.
2. Baris 5-6 merupakan proses mengubah array menjadi matriks 1D dan di normalisasi.
3. Baris 8-11 merupakan proses mengurangi hasil normalisasi gambar dengan rata-rata gambar. Gambar kemudian dipetakan ke eigenspace yang akan mengembalikan hasil koordinat (bobot).
4. Baris 13-15 merupakan proses untuk menentukan jarak Euclidian terdekat antara gambar dengan *training image*.
5. Baris 17-22 merupakan proses untuk mencari nama file dari *training image* yang terdekat.

5.5.4 Implementasi *Method* Mengelola Data Matakuliah

Algoritma mengelola data matakuliah bertujuan untuk melakukan manipulasi data matakuliah yang disimpan di *database*. Pengguna yang terdaftar sebagai *administrator* dapat melihat, menambah, mengubah dan menghapus data matakuliah pada sistem presensi mahasiswa ini. Gambar 5.12 merupakan implementasi dari *method* `insert()` yang berfungsi untuk menambah data matakuliah.

```
1.      public void insert(){
2.              String kode_mk= jTextField1.getText();
3.              String nama_mk= jTextField2.getText();
4.              String sks= jTextField4.getText();
5.              String semester= jTextField5.getText();
6.              try{
7.                  query = "select * from mata_kuliah
8.      where kd_mata_kuliah= '"+kode_mk+"'";
9.                  result = statement.executeQuery(query);
10.
11.                 if (result.next()){
12.                     JOptionPane.showMessageDialog(null,
13.      "Data sudah ada!");
14.                     clear();
15.                 }else{
16.                     query= "insert into mata_kuliah
17.      (kd_mata_kuliah, nama_mata_kuliah, sks, semester)
18.      values ('"+kode_mk+"', "
19.                                 + "'"+nama_mk+"',
20.      '"+sks+"',"
21.                                 + "'"+semester+"')";
22.                     statement.executeUpdate(query);
23.                     setTable();
24.                     clear();
25.                 }
26.             } catch (SQLException ex){
27.                 ex.printStackTrace();
28.             }
    }
```

Gambar 5.12 Implementasi *Method* `insert()`
Sumber : Implementasi

Penjelasan implementasi *method* `insert()` pada Gambar 5.12 yaitu:

1. Baris 2-5 merupakan proses mengambil nilai variabel yang dibutuhkan dalam *method*.
2. Baris 6 merupakan '*try'* yang akan dijalankan pertama kali dalam *method*.
3. Baris 7-9 merupakan proses menjalankan query mengambil data matakuliah melalui kode matakuliah.

4. Baris 11-14 jika kode matakuliah sudah ada, maka ditampilkan pesan bahwa "Data sudah ada!".
5. Baris 16-23 jika data tidak ada, maka akan menjalankan query untuk menambah data matakuliah.
6. Baris 25-26 merupakan peringatan bahwa method gagal menjalankan '*try'*, sehingga langsung menjalankan '*catch'* dalam method.

Gambar 5.13 merupakan *method* `update()` yang berfungsi untuk mengubah data matakuliah.

```
1.      public  void update(){
2.              String kode_mk= jTextField1.getText();
3.              String nama_mk= jTextField2.getText();
4.              String sks= jTextField4.getText();
5.              String semester= jTextField5.getText();
6.              try{
7.                  query = "select * from mata_kuliah
8.      where kd_mata_kuliah= '"+kode_mk+"'";
9.                  result = statement.executeQuery(query);
10.
11.                 query= "update mata_kuliah set
12.     nama_mata_kuliah = '"+nama_mk+"',"
13.                         + "sks= '"+sks+"',"
14.                         + "semester= '"+semester+"'
15.     where kd_mata_kuliah = '"+kode_mk+"'";
16.                 statement.executeUpdate(query);
17.                 setTable();
18.                 delTable();
19.                 clear();
20.
21.             } catch (SQLException ex){
22.                 ex.printStackTrace();
23.             }
24.     }
```

Gambar 5.13 Implementasi *Method* `update()`
Sumber : Implementasi

Penjelasan implementasi *method* `update()` pada Gambar 5.13 yaitu:

7. Baris 2-5 merupakan proses mengambil nilai variabel yang dibutuhkan dalam *method*.
8. Baris 6 merupakan '*try'* yang akan dijalankan pertama kali dalam *method*.
9. Baris 7-9 merupakan proses menjalankan query mengambil data matakuliah melalui kode matakuliah.
10. Baris 11-19 merupakan proses mengubah data matakuliah.

11. Baris 21-22 merupakan peringatan bahwa method gagal menjalankan '*try'*, sehingga langsung menjalankan '*catch'* dalam method.

Gambar 5.14 merupakan *method* `delDb()` yang berfungsi untuk melakukan proses penghapusan data matakuliah.

```
1.      public void delDb(){
2.              try{
3.
4.                  if (jTable1.getSelectedRow()!=-1){
5.                  query = "delete from mata_kuliah where
6.      kd_mata_kuliah =
7.      '"+jTable1.getValueAt(jTable1.getSelectedRow(),
8.      0)+"'";
9.                  statement.executeUpdate(query);
10.                 delTable();
11.
12.                 } else {
13.                     JOptionPane.showMessageDialog(null,
14.     "Tidak ada item yang akan dihapus!");
15.                 }
16.
17.             }catch (SQLException ex){
18.                 ex.printStackTrace();
19.             }
20.     }
```

Gambar 5.14 Implementasi *Method* `delDb()`
Sumber : Implementasi

Penjelasan implementasi *method* `delDb()` pada Gambar 5.14 yaitu:

1. Baris 2 merupakan '*try'* yang akan dijalankan pertama kali dalam *method*.
2. Baris 4-10 merupakan proses percabangan. Jika baris tabel yang dipilih tidak sama dengan -1, maka jalankan *query* untuk menghapus data matakuliah.
3. Baris 12-15 jika baris tabel yang dipilih sama dengan -1, maka ditampilkan pesan "Tidak ada item yang akan dihapus!".
4. Baris 17-19 merupakan peringatan bahwa method gagal menjalankan '*try'*, sehingga langsung menjalankan '*catch'* dalam method.

5.6 Implementasi Antarmuka Aplikasi

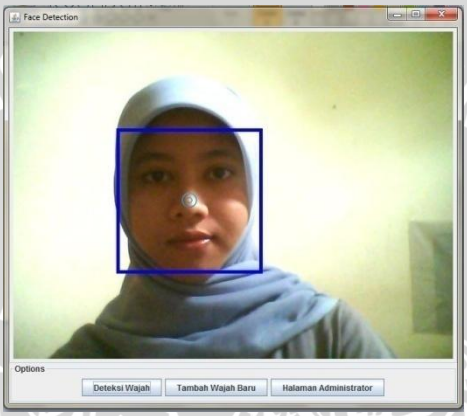
Antarmuka Aplikasi Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* digunakan oleh pengguna untuk berinteraksi dengan sistem perangkat lunak. Antarmuka perangkat lunak sistem

ini dibagi menjadi dua, yaitu aplikasi untuk sisi mahasiswa dan aplikasi sisi *administrator*.

5.6.1 Implementasi Antarmuka Aplikasi Mahasiswa

Antarmuka pengguna untuk aplikasi mahasiswa terdiri dari halaman utama mahasiswa, halaman deteksi wajah, halaman tambah wajah baru dan halaman *administrator*.

1. Halaman Utama



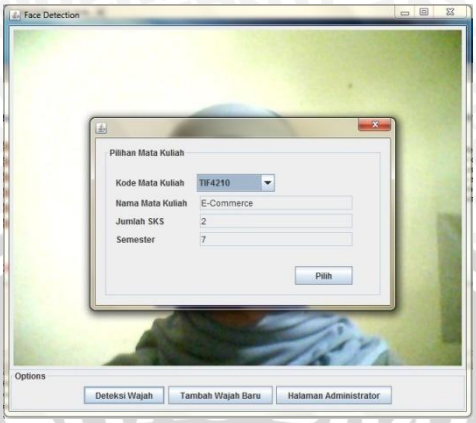
Gambar 5.15 Tampilan Antarmuka Halaman Utama
Sumber : Implementasi

Halaman utama merupakan halaman yang pertama kali ditampilkan pada saat aplikasi dijalankan. Pada halaman ini terdapat beberapa tombol yang akan menghubungkan halaman utama dengan halaman lainnya pada sistem presensi mahasiswa ini. Gambar 5.15 akan menunjukkan implementasi tampilan antarmuka dari halaman utama.

2. Halaman Deteksi Wajah

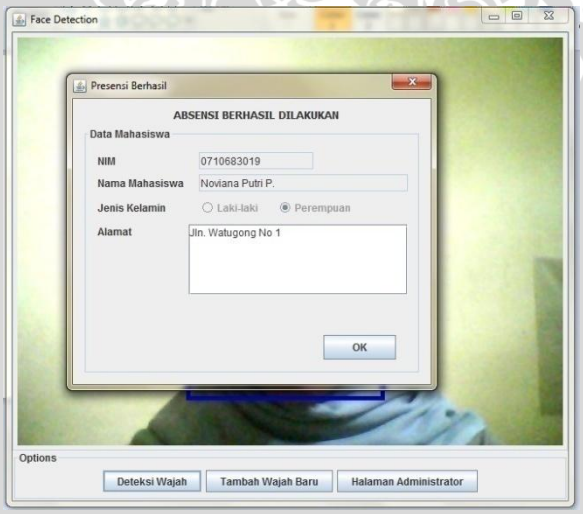
Halaman deteksi wajah merupakan halaman yang dapat diakses mahasiswa setelah menekan tombol Deteksi Wajah pada halaman utama. Halaman ini berfungsi untuk melakukan proses presensi mahasiswa. Tampilan halaman deteksi wajah sama dengan tampilan halaman utama, hanya saja setelah menekan tombol deteksi wajah dan wajah dideteksi, ditampilkan *form* untuk memilih matakuliah yang akan dilakukan presensi dan data mahasiswa

yang berhasil melakukan presensi. Gambar 5.16 menunjukkan implementasi tampilan antarmuka dari *form* pilih matakuliah.



Gambar 5.16 Tampilan Antarmuka *Form* Pilih Matakuliah
Sumber : Implementasi

Gambar 5.17 menunjukkan implementasi tampilan antarmuka presensi berhasil, yang menampilkan data mahasiswa yang berhasil melakukan proses presensi.

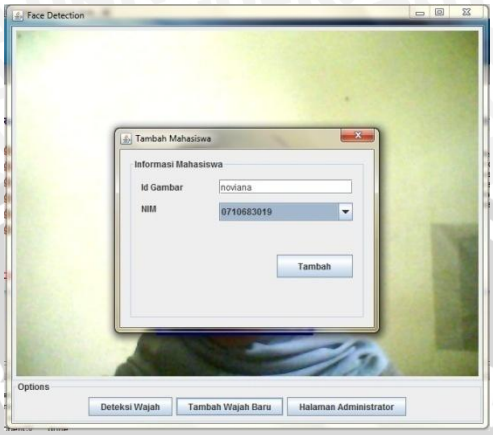


Gambar 5.17 Tampilan Antarmuka *Form* Presensi Berhasil
Sumber : Implementasi

3. Halaman Tambah Wajah

Halaman tambah wajah merupakan halaman yang dapat diakses mahasiswa setelah menekan tombol Tambah Wajah pada halaman utama. Halaman ini berfungsi untuk melakukan proses menambah data mahahasiswa berupa gambar wajah mahasiswa. Tampilan halaman deteksi wajah sama dengan tampilan halaman utama, hanya saja setelah menekan tombol tambah

wajah ditampilkan *form* untuk mengisi id gambar dan memilih nim. Gambar 5.18 akan menunjukkan implementasi tampilan antarmuka dari form tambah wajah mahasiswa.



Gambar 5.18 Tampilan Antarmuka Halaman Tambah Wajah Mahasiswa
Sumber : Implementasi

5.6.2 Implementasi Antarmuka Aplikasi *Administrator*

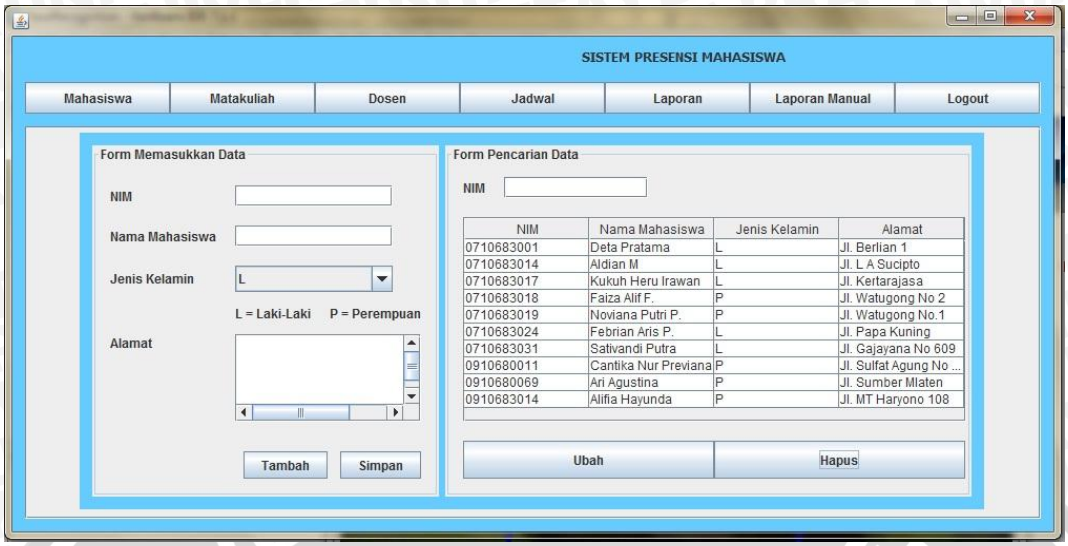
Antarmuka pengguna untuk aplikasi *administrator* terdiri dari halaman *login*, halaman mahasiswa, halaman matakuliah, halaman dosen, halaman jadwal dan halaman laporan.

1. **Halaman *Login***

   Halaman *login* merupakan halaman yang dapat diakses oleh *administrator* setelah menekan tombol Halaman *Administrator* pada halaman utama mahasiswa. Halaman ini berfungsi untuk menyeleksi pengguna, sehingga hanya pengguna yang terdaftar sebagai *administrator* saja yang dapat mengakses aplikasi *administrator*. Gambar 5.19 akan menunjukkan implementasi tampilan antarmuka dari halaman *login*.

Gambar 5.19 Tampilan Antarmuka Halaman *Login*
Sumber : Implementasi

2. Halaman Mahasiswa



Gambar 5.20 Tampilan Antarmuka Halaman Mahasiswa
Sumber : Implementasi

Halaman mahasiswa merupakan halaman yang pertama kali ditampilkan saat *administrator* berhasil melakukan *login* untuk mengakses aplikasi sisi *administrator*. Halaman ini secara tidak langsung juga merupakan halaman utama dari aplikasi sisi *administrator*. Di halaman mahasiswa ini *administrator* dapat melakukan pengolahan data mahasiswa, seperti menambah, mengubah serta menghapus data mahasiswa. Gambar 5.20 menunjukkan implementasi tampilan antarmuka dari halaman mahasiswa.

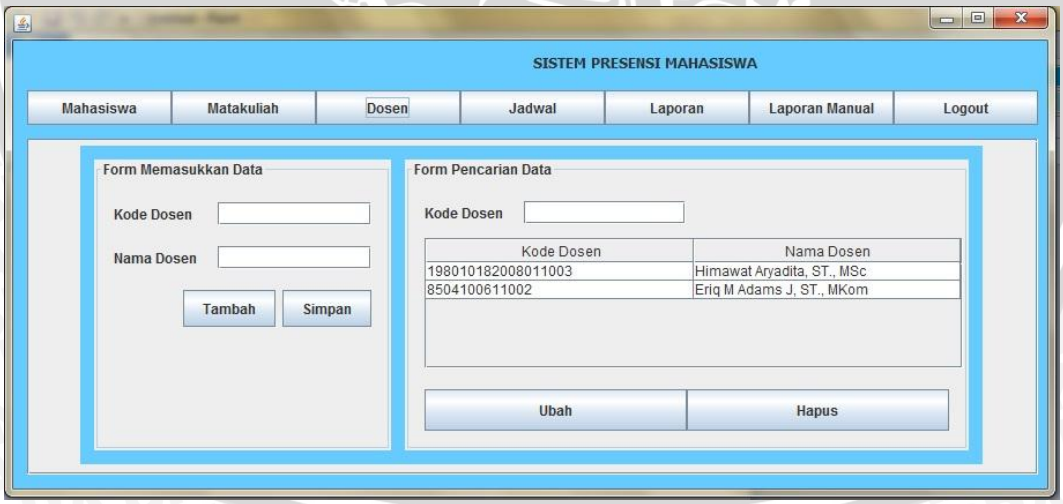
3. Halaman Matakuliah

Halaman matakuliah merupakan halaman yang dapat diakses oleh *administrator* setelah menekan tombol Matakuliah. Halaman ini digunakan *administrator* untuk mengolah data matakuliah, yaitu menambah, mengubah dan menghapus data matakuliah. Gambar 5.21 akan menunjukkan implementasi tampilan antarmuka dari halaman matakuliah.

Gambar 5.21 Tampilan Antarmuka Halaman Matakuliah
Sumber : Implementasi

4. Halaman Dosen

Halaman dosen merupakan halaman yang dapat diakses oleh *administrator* setelah menekan tombol Dosen. Halaman ini digunakan *administrator* untuk mengolah data dosen, yaitu menambah, mengubah dan menghapus data dosen. Gambar 5.22 akan menunjukkan implementasi tampilan antarmuka dari halaman dosen.

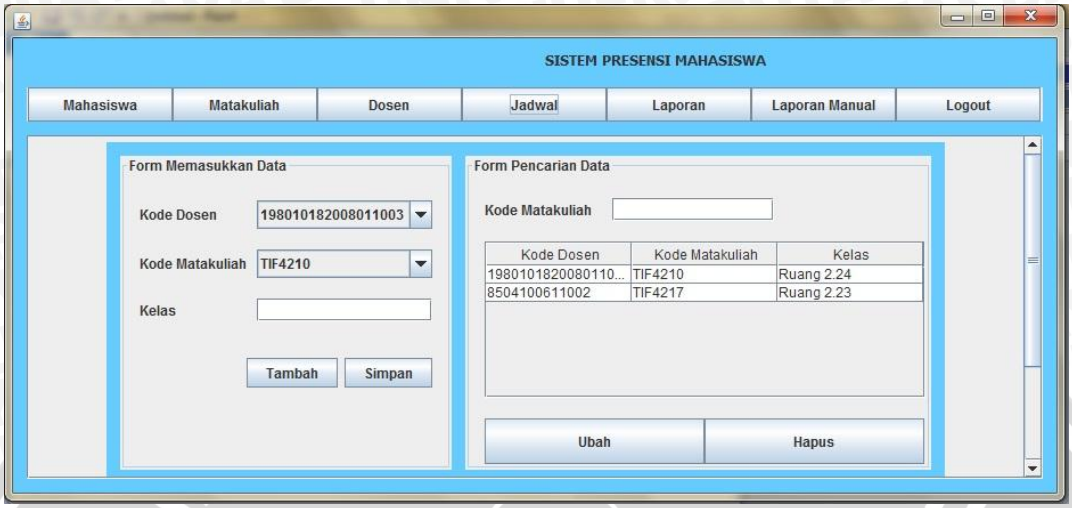


Gambar 5.22 Tampilan Antarmuka Halaman Dosen
Sumber : Implementasi

5. Halaman Jadwal

Halaman jadwal merupakan halaman yang dapat diakses oleh *administrator* setelah menekan tombol Jadwal. Halaman ini digunakan

administrator untuk mengolah data jadwal, yaitu menambah, mengubah dan menghapus data jadwal. Gambar 5.23 akan menunjukkan implementasi tampilan antarmuka dari halaman jadwal.



Gambar 5.23 Tampilan Antarmuka Halaman Jadwal
Sumber : Implementasi

6. Halaman Laporan Manual

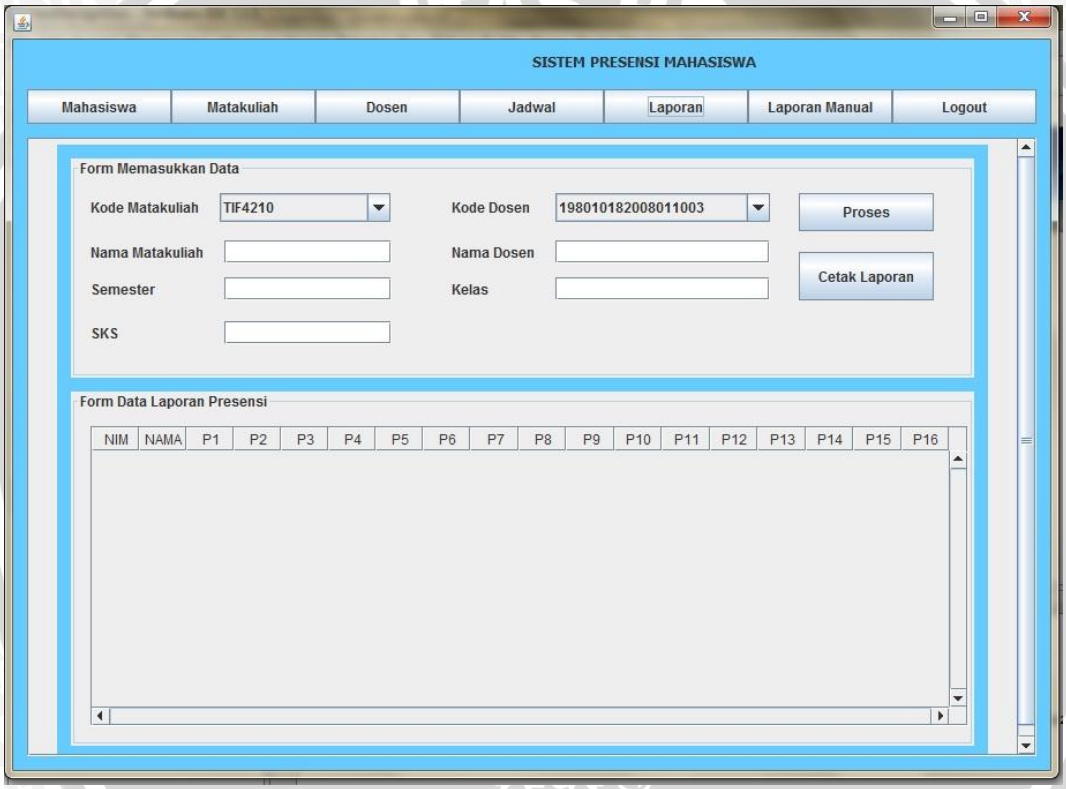
Gambar 5.24 Tampilan Antarmuka Halaman Laporan Manual
Sumber : Implementasi

Halaman laporan manual merupakan halaman yang dapat diakses oleh *administrator* setelah menekan tombol Laporan Manual. Halaman ini digunakan *administrator* untuk mengolah data laporan presensi mahasiswa, yaitu menambah, mengubah dan menghapus data laporan presensi. Gambar

5.24 menunjukkan implementasi tampilan antarmuka dari halaman laporan manual.

7. Halaman Laporan

Halaman laporan merupakan halaman yang dapat diakses oleh *administrator* setelah menekan tombol Laporan. Halaman ini digunakan *administrator* untuk menampilkan rekap data laporan presensi mahasiswa berdasarkan matakuliah yang di pilih. Gambar 5.25 menunjukkan implementasi tampilan antarmuka dari halaman laporan presensi mahasiswa.



Gambar 5.25 Tampilan Antarmuka Halaman Laporan
Sumber : Implementasi

BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis terhadap aplikasi Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* yang telah dikembangkan. Proses pengujian dilakukan melalui dua tahapan (strategi) yaitu pengujian unit dan pengujian validasi. Pada pengujian unit akan digunakan teknik pengujian *White Box* (*White Box Testing*). Pada pengujian validasi digunakan teknik pengujian *Black Box (Black Box Testing)*.

6.1 Pengujian

Proses pengujian dilakukan melalui dua tahapan (strategi) yaitu pengujian unit dan pengujian validasi dari aplikasi Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*.

6.1.1 Pengujian Unit

Perangkat lunak yang dikembangkan dengan paradigma *object-oriented programming* menerapkan pengujian unit untuk suatu *method* (operasi) dari suatu *class.* Pada pengujian unit aplikasi Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* digunakan teknik *White-Box Testing* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)*, menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan.

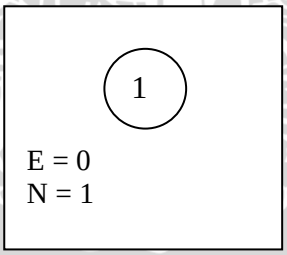
6.1.1.1 Pengujian Unit Untuk *Method* `findMatch()`

Method `findMatch()` merupakan method yang melaksanakan proses membandingkan gambar baru dengan *training image* dengan menggunakan *eigenfaces* dan *eigenvector* yang diambil dari objek FaceBundle. *Method* `findMatch()` terdapat pada kelas FaceRecognition. Gambar 6.1 menggambarkan proses pengujian unit dari *method* `findMatch()` dan gambar 6.2 menunjukkan pemodelan *flow graph* pada *method* `findMatch()`.

```
private MatchResult findMatch(BufferedImage im)
  {
    double[] imArr =
 ImageUtils.createArrFromIm(im);

    Matrix2D imMat = new Matrix2D(imArr, 1);
    imMat.normalise();

    imMat.subtract(new
 Matrix2D(bundle.getAvgImage(), 1));
    Matrix2D imWeights =
 getImageWeights(numEFs, imMat);

    double[] dists = getDists(imWeights);
    ImageDistanceInfo distInfo =
 getMinDistInfo(dists);

    ArrayList<String> imageFNms =
 bundle.getImageFnms();
    String matchingFNm = imageFNms.get(
 distInfo.getIndex() );

    double minDist = Math.sqrt(
 distInfo.getValue() );

    return new MatchResult(matchingFNm,
 minDist);
}
```

1

Gambar 6.1 Pengujian Unit *Method* `findMatch()`
Sumber : Pengujian



Gambar 6.2 *Flow Graph Method* `findMatch()`
Sumber : Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `findMatch()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 0 – 1 + 2

= 1

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka didapatkan sebuah basis set dari jalur *independent*, yaitu :

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.1.

Tabel 6.1 *Test Case* Untuk Pengujian Unit *Method* `findMatch()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Terdapat gambar baru	Mendapatkan jarak terdekat dengan *training image.*	Mendapatkan jarak terdekat dengan *training image.*

Sumber : Pengujian

6.1.1.2 Pengujian Unit Untuk Algoritma Mengelola Data Matakuliah

Algoritma mengelola data matakuliah bertujuan untuk melakukan manipulasi data matakuliah, yaitu operasi untuk menambahkan, mengubah dan menghapus data matakuliah. Algoritma ini terdapat pada kelas Matakuliah. Gambar 6.3 menggambarkan proses pengujian unit dari *method* `insert()` dan gambar 6.4 menunjukkan pemodelan *flow graph* pada *method* `insert()`.

```
public void insert(){
        String kode_mk= jTextField1.getText();
        String nama_mk= jTextField2.getText();
        String sks= jTextField4.getText();
        String semester=
jTextField5.getText();
        try{
            query = "select * from mata_kuliah
where kd_mata_kuliah= '"+kode_mk+"'";
            result =
statement.executeQuery(query);

            if (result.next()){

JOptionPane.showMessageDialog(null, "Data
sudah ada!");
                clear();
            }else{
```

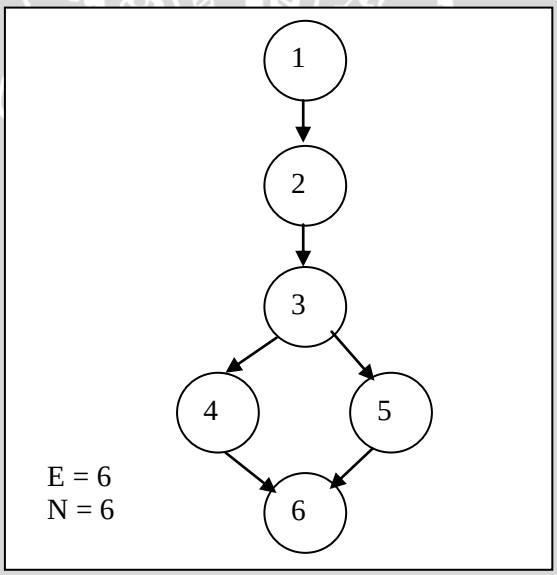
1
2
3
4

```
                query= "insert into
mata_kuliah (kd_mata_kuliah,
nama_mata_kuliah, sks, semester) values
('"+kode_mk+"', "
                        + "'"+nama_mk+"',
'"+sks+"',"
                        + "'"+semester+"')";

statement.executeUpdate(query);
                setTable();
                clear();
            }
        }
        catch (SQLException ex){
            ex.printStackTrace();
        }
}
```

5

6

Gambar 6.3 Pengujian Unit *Method* `insert()`
Sumber : Pengujian



Gambar 6.4 *Flow Graph Method* `insert()`
Sumber : Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `insert()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 6 – 6 + 2

= 2

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka didapatkan 2 buah basis set dari jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 6

Jalur 2 : 1 – 2 – 3 – 5 - 6

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.2.

Tabel 6.2 *Test Case* untuk Pengujian Unit *Method* `insert()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Data matakuliah sudah ada.	Muncul pesan Data Sudah Ada.	Muncul pesan Data Sudah Ada.
2	Data matakuliah belum ada.	Menampilkan data matakuliah baru pada tabel.	Menampilkan data matakuliah baru pada tabel.

Sumber : Pengujian

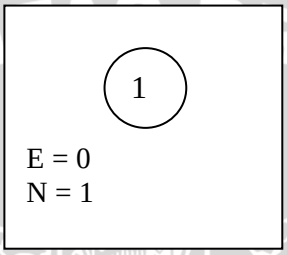
Gambar 6.5 menggambarkan proses pengujian unit dari *method* `update()` dan gambar 6.6 menunjukkan pemodelan *flow graph* pada *method* `update()`.

```
public  void update(){
        String kode_mk= jTextField1.getText();
        String nama_mk= jTextField2.getText();
        String sks= jTextField4.getText();
        String semester= jTextField5.getText();
        try{
            query = "select * from mata_kuliah
  where kd_mata_kuliah= '"+kode_mk+"'";
            result =
  statement.executeQuery(query);

                query= "update mata_kuliah set
  nama_mata_kuliah = '"+nama_mk+"',"
                    + "sks= '"+sks+"',"
                    + "semester=
  '"+semester+"' where kd_mata_kuliah =
  '"+kode_mk+"'";
```

1

```
                    statement.executeUpdate(query);
                    setTable();
                    delTable();
                    clear();

                } catch (SQLException ex){
                    ex.printStackTrace();
                }
    }
```

Gambar 6.5 Pengujian Unit *Method* `update()`
Sumber : Pengujian



Gambar 6.6 *Flow Graph Method* `update()`
Sumber : Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* update()menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 0 – 1 + 2
= 1

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka didapatkan sebuah basis set dari jalur *independent*, yaitu :

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.3.

Tabel 6.3 *Test Case* untuk Pengujian Unit *Method* `update()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Mengubah data	Data berhasil dirubah dan ditampilkan di tabel.	Data berhasil dirubah dan ditampilkan di tabel.

	matakuliah		

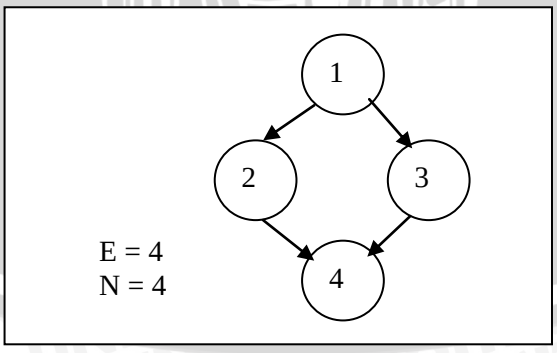
Sumber : Pengujian

Gambar 6.7 menggambarkan proses pengujian unit dari *method* `delDb()` dan gambar 6.8 menunjukkan pemodelan *flow graph* pada *method* `delDb()`.

```
public void delDb(){
        try{
            if (jTable1.getSelectedRow()!=-1){
            query = "delete from mata_kuliah
  where kd_mata_kuliah =
  '"+jTable1.getValueAt(jTable1.getSelectedRow(
  ), 0)+"'";
            statement.executeUpdate(query);
            delTable();

            } else {

  JOptionPane.showMessageDialog(null, "Tidak
  ada item yang akan dihapus!");
            }

        }
catch (SQLException ex){
            ex.printStackTrace();
        }
 }
```

1
2
3
4

Gambar 6.7 Pengujian Unit *Method* `delDb()`
Sumber : Pengujian



Gambar 6.8 *Flow Graph Method* `delDb()`
Sumber : Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `delDb()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)*

melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2
= 4 – 4 + 2
= 2

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka didapatkan 2 buah basis set dari jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 4

Jalur 2 : 1 – 3 – 4

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.4.

Tabel 6.4 *Test Case* untuk Pengujian Unit *Method* `delDb()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Jika baris tabel matakuliah yang dipilih tidak sama dengan -1	Data berhasil dihapus dari tabel.	Data berhasil dihapus dari tabel.
2	Jika baris tabel matakuliah yang dipilih sama dengan -1	Ditampilkan pesan Tidak item yang akan dihapus.	Ditampilkan pesan Tidak item yang akan dihapus.

Sumber : Pengujian

6.1.2 Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Item - item yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Pada skripsi ini dilakukan pengujian validasi terhadap Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface*.

a. Kasus Uji Untuk Melakukan Presensi Mahasiswa

Tabel 6.5 Kasus Uji Untuk Pengujian Melakukan Presensi Mahasiswa

Nama Kasus Uji	Kasus Uji Melakukan Presensi Mahasiswa
Objek Uji	SRS_001_01
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk melakukan kegiatan presensi mahasiswa.
Prosedur Uji	1. Mahasiswa menekan tombol Deteksi Wajah 2. *Form* pilih matakuliah ditampilkan 3. Mahasiswa memilih matakuliah yang ingin dilakukan presensi. 4. Mahasiswa menekan tombol Pilih.
Hasil yang diharapkan	Aplikasi dapat melakukan pengenalan wajah mahasiswa, menambahkan data presensi pada *database* serta menampilkan *form* yang berisi data mahasiswa yang berhasil melakukan presensi.

Sumber : Pengujian

b. Kasus Uji Untuk Menambah Wajah Baru

Tabel 6.6 Kasus Uji Untuk Pengujian Menambah Wajah Baru

Nama Kasus Uji	Kasus Uji Menambah Wajah Baru
Objek Uji	SRS_001_02
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk menambah gambar wajah baru bagi mahasiswa agar dapat dikenali oleh sistem.
Prosedur Uji	1. Mahasiswa menekan tombol Tambah Wajah Baru. 2. *Form* tambah gambar mahasiswa ditampilkan. 3. Mahasiswa mengisi id gambar dan memilih nim. 4. Mahasiswa menekan tombol Tambah.
Hasil yang diharapkan	Aplikasi dapat melakukan penyimpanan gambar baru, menyimpan data gambar pada *database* dan menampilkan pesan Berhasil Tambah Data.

Sumber : Pengujian

c. Kasus Uji Untuk *Login* Sukses

Tabel 6.7 Kasus Uji Untuk Pengujian Validasi *Login* Sukses

Nama Kasus Uji	Kasus Uji *Login* Sukses
Objek Uji	SRS_002_01
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas *login* bagi *administrator*.
Prosedur Uji	1. *Administrator* menekan tombol Halaman *administrator*. 2. *Form login* ditampilkan. 3. *Administrator* memasukkan *username* dan *password* di dalam *form login*. 4. *Administrator* menekan tombol *Login*.
Hasil yang diharapkan	Aplikasi dapat melakukan penyeleksian kondisi *login* pada *database* berdasar data yang dimasukkan dan jika penyeleksian kondisi ini benar, maka *administrator* dapat mengakses halaman utama aplikasi sisi *administrator*.

Sumber : Pengujian

d. Kasus Uji Untuk *Login* Gagal

Tabel 6.8 Kasus Uji Untuk Pengujian Validasi *Login* Gagal

Nama Kasus Uji	Kasus Uji *Login* Gagal
Objek Uji	SRS_002_01
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas *login* bagi *administrator*.
Prosedur Uji	1. *Administrator* menekan tombol Halaman *administrator*. 2. *Form login* ditampilkan. 3. *Administrator* memasukkan *username* dan *password* di dalam *form login*. 4. *Administrator* menekan tombol *Login*.
Hasil yang diharapkan	Aplikasi dapat melakukan penyeleksian kondisi *login* pada *database* berdasar data yang dimasukkan dan jika penyeleksian kondisi ini salah, maka *administrator* tidak dapat mengakses halaman utama

	aplikasi sisi *administrator* dan ditampilkan pesan *error*.

Sumber : Pengujian

e. Kasus Uji Untuk *Logout*

Tabel 6.9 Kasus Uji Untuk Pengujian Validasi *Logout*

Nama Kasus Uji	Kasus Uji *Logout*
Objek Uji	SRS_002_07
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas *logout* bagi *administrator*.
Prosedur Uji	1. Aplikasi dijalankan pada keadaan *administrator* sudah *login* 2. *Administrator* menekan tombol *Logout*.
Hasil yang diharapkan	Aplikasi melakukan penghapusan session yang aktif dan menampilkan halaman *login*.

Sumber : Pengujian

f. Kasus Uji Untuk Mengelola Data Mahasiswa

Tabel 6.10 Kasus Uji Untuk Pengujian Validasi Mengelola Data Mahasiswa

Nama Kasus Uji	Kasus Uji Mengelola Data Mahasiswa
Objek Uji	SRS_002_02
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk mengelola data mahasiswa bagi *administrator*.
Prosedur Uji	1. Aplikasi dijalankan pada keadaan sudah *login* 2. *Administrator* memilih tab mahasiswa pada aplikasi. 3. *Administrator* mengisi field nim, nama mahasiswa, alamat serta memilih jenis kelamin. Selanjutnya *administrator* menekan tombol Tambah. 4. *Administrator* memilih data mahasiswa yang akan diubah pada tabel dan menekan tombol Ubah. *Administrator* mengubah isi field data

	kemudian menekan tombol Simpan. 5. *Administrator* memilih data mahasiswa yang akan dihapus pada tabel dan menekan tombol Hapus.
Hasil yang diharapkan	Aplikasi menampilkan data mahasiswa pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data mahasiswa sesuai dengan operasi yang dipilih oleh *administrator*.

Sumber : Pengujian

g. Kasus Uji Untuk Mengelola Data Dosen

Tabel 6.11 Kasus Uji Untuk Pengujian Validasi Mengelola Data Dosen

Nama Kasus Uji	Kasus Uji Mengelola Data Dosen
Objek Uji	SRS_002_03
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk mengelola data dosen bagi *administrator*.
Prosedur Uji	1. Aplikasi dijalankan pada keadaan sudah *login* 2. *Administrator* memilih tab dosen pada aplikasi. 3. *Administrator* mengisi field kode dosen dan nama dosen. Selanjutnya *administrator* menekan tombol Tambah. 4. *Administrator* memilih data dosen yang akan diubah pada tabel dan menekan tombol Ubah. *Administrator* mengubah isi field data kemudian menekan tombol Simpan. 5. *Administrator* memilih data dosen yang akan dihapus pada tabel dan menekan tombol Hapus.
Hasil yang diharapkan	Aplikasi menampilkan data dosen pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data dosen sesuai dengan operasi yang dipilih oleh *administrator*.

Sumber : Pengujian

h. Kasus Uji Untuk Mengelola Data Matakuliah

Tabel 6.12 Kasus Uji Untuk Pengujian Validasi Mengelola Data Matakuliah

Nama Kasus Uji	Kasus Uji Mengelola Data Matakuliah
Objek Uji	SRS_002_04
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk mengelola data matakuliah bagi *administrator*.
Prosedur Uji	1. Aplikasi dijalankan pada keadaan sudah *login* 2. *Administrator* memilih tab matakuliah pada aplikasi. 3. *Administrator* mengisi field kode matakuliah, nama matakuliah, sks dan semester. Selanjutnya *administrator* menekan tombol Tambah. 4. *Administrator* memilih data matakuliah yang akan diubah pada tabel dan menekan tombol Ubah. *Administrator* mengubah isi field data kemudian menekan tombol Simpan. 5. *Administrator* memilih data matakuliah yang akan dihapus pada tabel dan menekan tombol Hapus.
Hasil yang diharapkan	Aplikasi menampilkan data matakuliah pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data matakuliah sesuai dengan operasi yang dipilih oleh *administrator*.

Sumber : Pengujian

i. Kasus Uji Untuk Mengelola Data Jadwal

Tabel 6.13 Kasus Uji Untuk Pengujian Validasi Mengelola Data Jadwal

Nama Kasus Uji	Kasus Uji Mengelola Data Jadwal
Objek Uji	SRS_002_05
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk mengelola data jadwal bagi *administrator*.
Prosedur Uji	1. Aplikasi dijalankan pada keadaan sudah *login* 2. *Administrator* memilih tab jadwal pada aplikasi. 3. *Administrator* memilih kode dosen dan kode

	matakuliah serta mengisi field kelas. Selanjutnya *administrator* menekan tombol Tambah. 4. *Administrator* memilih data jadwal yang akan diubah pada tabel dan menekan tombol Ubah. *Administrator* mengubah isi field data kemudian menekan tombol Simpan. 5. *Administrator* memilih data jadwal yang akan dihapus pada tabel dan menekan tombol Hapus.
Hasil yang diharapkan	Aplikasi menampilkan data jadwal pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data jadwal sesuai dengan operasi yang dipilih oleh *administrator*.

Sumber : Pengujian

j. Kasus Uji Untuk Mengelola Data Presensi

Tabel 6.14 Kasus Uji Untuk Pengujian Validasi Mengelola Data Presensi

Nama Kasus Uji	Kasus Uji Mengelola Data Presensi
Objek Uji	SRS_002_06
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional dalam menyediakan fasilitas untuk mengelola data presensi mahasiswa bagi *administrator*.
Prosedur Uji	1. Aplikasi dijalankan pada keadaan sudah *login* 2. *Administrator* memilih tab laporan manual pada aplikasi. 3. *Administrator* memilih nim dan kode matakuliah serta mengisi field waktu. Selanjutnya *administrator* menekan tombol Tambah. 4. *Administrator* memilih data presensi yang akan diubah pada tabel dan menekan tombol Ubah. *Administrator* mengubah isi field data kemudian menekan tombol Simpan. 5. *Administrator* memilih data presensi yang akan dihapus pada tabel dan menekan tombol Hapus.
Hasil yang diharapkan	Aplikasi menampilkan data presensi mahasiswa pada

	tabel, melakukan penambahan data presensi mahasiswa sesuai dengan operasi yang dipilih oleh *administrator*.

Sumber : Pengujian

6.1.3 Hasil Pengujian Validasi

Tabel 6.15 Hasil Pengujian Validasi

No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	Kasus Uji Melakukan Presensi Mahasiswa	Aplikasi dapat melakukan pengenalan wajah mahasiswa, menambahkan data presensi pada *database* serta menampilkan *form* yang berisi data mahasiswa yang berhasil melakukan presensi.	Aplikasi dapat melakukan pengenalan wajah mahasiswa, menambahkan data presensi pada *database* serta menampilkan *form* yang berisi data mahasiswa yang berhasil melakukan presensi.	Valid
2	Kasus Uji Menambah Wajah Baru	Aplikasi dapat melakukan penyimpanan gambar baru, menyimpan data gambar pada *database* dan menampilkan pesan Berhasil Tambah Data.	Aplikasi dapat melakukan penyimpanan gambar baru, menyimpan data gambar pada *database* dan menampilkan pesan Berhasil Tambah Data.	Valid
3	Kasus Uji *Login* Sukses	Aplikasi dapat melakukan penyeleksian kondisi *login* pada *database* berdasar data yang dimasukkan dan jika penyeleksian kondisi ini benar, maka *administrator* dapat	Aplikasi dapat melakukan penyeleksian kondisi *login* pada *database* berdasar data yang dimasukkan dan jika penyeleksian kondisi ini benar, maka *administrator* dapat	Valid

		mengakses halaman utama aplikasi sisi *administrator*.	mengakses halaman utama aplikasi sisi *administrator*.	
4	Kasus Uji *Login* Gagal	Aplikasi dapat melakukan penyeleksian kondisi *login* pada *database* berdasar data yang dimasukkan dan jika penyeleksian kondisi ini salah, maka *administrator* tidak dapat mengakses halaman utama aplikasi sisi *administrator* dan ditampilkan pesan *error*.	Aplikasi dapat melakukan penyeleksian kondisi *login* pada *database* berdasar data yang dimasukkan dan jika penyeleksian kondisi ini salah, maka *administrator* tidak dapat mengakses halaman utama aplikasi sisi *administrator* dan ditampilkan pesan *error*.	Valid
5	Kasus Uji *Logout*	Aplikasi melakukan penghapusan session yang aktif dan menampilkan halaman *login*.	Aplikasi melakukan penghapusan session yang aktif dan menampilkan halaman *login*.	Valid
6	Kasus Uji Mengelola Data Mahasiswa	Aplikasi menampilkan data mahasiswa pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data mahasiswa sesuai dengan operasi yang dipilih oleh *administrator*.	Aplikasi menampilkan data mahasiswa pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data mahasiswa sesuai dengan operasi yang dipilih oleh *administrator*.	Valid
7	Kasus Uji Mengelola Data Dosen	Aplikasi menampilkan data dosen pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data dosen sesuai	Aplikasi menampilkan data dosen pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data dosen sesuai	Valid

		dengan operasi yang dipilih oleh *administrator*.	dengan operasi yang dipilih oleh *administrator*.	
8	Kasus Uji Mengelola Data Matakuliah	Aplikasi menampilkan data matakuliah pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data matakuliah sesuai dengan operasi yang dipilih oleh *administrator*.	Aplikasi menampilkan data matakuliah pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data matakuliah sesuai dengan operasi yang dipilih oleh *administrator*.	Valid
9	Kasus Uji Mengelola Data Jadwal	Aplikasi menampilkan data jadwal pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data jadwal sesuai dengan operasi yang dipilih oleh *administrator*.	Aplikasi menampilkan data jadwal pada tabel, melakukan penambahan data, pengubahan data serta penghapusan data jadwal sesuai dengan operasi yang dipilih oleh *administrator*.	Valid
10	Kasus Uji Mengelola Data Presensi	Aplikasi menampilkan data presensi mahasiswa pada tabel, melakukan penambahan, ubah dan penghapusan data presensi mahasiswa sesuai dengan operasi yang dipilih oleh *administrator*.	Aplikasi menampilkan data presensi mahasiswa pada tabel, melakukan penambahan, ubah dan penghapusan data presensi mahasiswa sesuai dengan operasi yang dipilih oleh *administrator*.	Valid

Sumber : Pengujian

6.1.4 Pengujian Kinerja Sistem

Untuk menguji kinerja dari sistem, maka dilakukan uji kinerja sistem terhadap data *training image* yang telah ada. Sistem dikatakan memiliki kinerja tinggi apabila output yang dihasilkan oleh Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* ini sesuai dengan data yang telah disimpan pada *database*.

6.1.4.1 Database Citra Wajah

Untuk menentukan akurasi dari Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* ini, sebelumnya simpan citra wajah mahasiswa yang akan digunakan sebagai pengenal dalam direktori *trainingImage*. Citra wajah yang disimpan terdiri dari 3 pose yang berbeda, yaitu menghadap ke depan, samping kiri dan samping kanan. Untuk pengambilan gambar citra wajah mahasiswa yang akan dinormalisasi pada saat presensi, idealnya terdapat beberapa kondisi yang dilakukan, diantaranya :

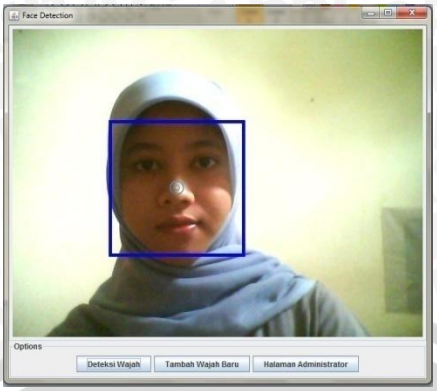
1. Pose wajah standar yaitu idealnya citra wajah yang disimpan memiliki pose lurus ke depan, menghadap ke samping kiri dan samping kanan dengan ekspresi standar.
2. Pencahayaan saat pengambilan semua citra mahasiswa dibuat sama.
3. Jarak kamera dan wajah yang di ambil dibuat konstan (± 30 cm).

Setelah diambil, citra wajah akan normalisasi menjadi citra *grayscale* dengan ukuran 125 x 150 dan bertipe PNG.

Gambar 6.9 Contoh Citra Wajah yang Disimpan Sebagai *Training Image*
Sumber : Pengujian

6.1.4.2 Hasil Pengujian Citra Wajah

Untuk keperluan pengujian dilakukan percobaan presensi mahasiswa dengan membandingkan citra uji yang terdiri dari 3 pose sebanyak 3 kali tiap pose, dengan data *training* yang terdiri dari 10 buah untuk masing – masing pose. Proses pengenalan wajah dilakukan pada halaman deteksi wajah.

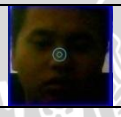


Gambar 6.10 Proses Presensi Dengan Pose Standar
Sumber : Pengujian

Hasil pengujian seluruh presensi citra wajah mahasiswa ditampilkan pada tabel – tabel berikut :

Tabel 6.16 Hasil Pengujian Kondisi Wajah Menghadap Ke Depan

No.	Nama Mahasiswa	Citra Uji	Hasil Citra Pengujian	Waktu Pengenalan Wajah (ms)
1.	Noviana Putri P.			463
				104
				3601
2.	Pricillia C. K			3833
				120
				2584
3.	Adam Hendra B.			303
				163

				136
4.	Firman Jati P.			577
				130
				118
5.	Deta Pratama			111
				2542
				97

Sumber : Pengujian

Perbandingan jumlah wajah yang dikenali antara kenyataan dengan sistem untuk kondisi wajah menghadap ke depan ditunjukkan pada tabel 6.17 berikut :

Tabel 6.17 Perbandingan Jumlah Wajah Dikenali Antara Kenyataan Dan Sistem Untuk Kondisi Wajah Menghadap Ke Depan

		Hasil Pengenalan sistem	
		Dikenali	Tidak dikenali
Kenyataan	Dikenali	15	0
	Tidak dikenali	0	0

Sumber : Pengujian

Nilai akurasi sistem dinilai dengan cara menghitung nilai f_{11}, f_{00}, f_{01}, dan f_{10} pada Tabel 6.17.

$f_{11} = 15,\ f_{00} = 0,\ f_{01} = 0,\ f_{10} = 0$

$$Accuracy = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}} \times 100\%$$

$$= \frac{15+0}{15+0+0+0} \times 100\%$$

$$= 100\%$$

$$Error\ rate = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}} \times 100\%$$

$$= \frac{0+0}{15+0+0+0} \times 100\%$$

$$= 0\%$$

Waktu rata – rata yang dibutuhkan sistem untuk mengenali wajah mahasiswa dengan kondisi wajah menghadap ke depan pada saat proses presensi :

Waktu rata – rata

$$= \frac{463+104+3601+3833+120+2584+303+163+136+777+130+118+111+2542+97}{15}$$

$$= 1005.467 \text{ ms}$$

Tabel 6.18 Hasil Pengujian Kondisi Wajah Menghadap Ke Samping Kanan

No.	Nama Mahasiswa	Citra Uji	Hasil Citra Pengujian	Waktu Pengenalan Wajah (ms)
1.	Noviana Putri P.			1119
				1359
				126
2.	Pricillia C. K			2929
				2081
				1007

3.	Adam Hendra B.			114
				104
				108
4.	Firman Jati P.			7220
				139
				131
5.	Deta Pratama			6464
				121
				112

Sumber : Pengujian

Perbandingan jumlah wajah yang dikenali antara kenyataan dengan sistem untuk kondisi wajah menghadap ke samping kanan ditunjukkan pada tabel 6.19 berikut :

Tabel 6.19 Perbandingan Jumlah Wajah Dikenali Antara Kenyataan Dan Sistem Untuk Kondisi Wajah Menghadap Ke Samping Kanan

		Hasil Pengenalan sistem	
		Dikenali	Tidak dikenali
Kenyataan	Dikenali	12	3
	Tidak dikenali	0	0

Sumber : Pengujian

Nilai akurasi sistem dinilai dengan cara menghitung nilai f_{11}, f_{00}, f_{01}, dan f_{10} pada Tabel 6.19.

$f_{11} = 12$, $f_{00} = 0$, $f_{01} = 0$, $f_{10} = 3$

$$Accuracy = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}} \times 100\%$$

$$= \frac{12 + 0}{12 + 3 + 0 + 0} \times 100\%$$

$$= 80\%$$

$$Error\ rate = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}} \times 100\%$$

$$= \frac{3 + 0}{12 + 3 + 0 + 0} \times 100\%$$

$$= 20\%$$

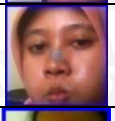
Waktu rata – rata yang dibutuhkan sistem untuk mengenali wajah mahasiswa dengan kondisi wajah menghadap ke samping kanan pada saat proses presensi :

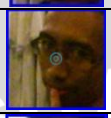
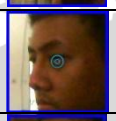
Waktu rata – rata

$$= \frac{1119 + 1359 + 126 + 2929 + 2081 + 1007 + 114 + 104 + 108 + 7220 + 139 + 131 + 6464 + 121 + 112}{15}$$

= 1542.267 ms

Tabel 6.20 Hasil Pengujian Kondisi Wajah Menghadap Ke Samping Kiri

No.	Nama Mahasiswa	Citra Uji	Hasil Citra Pengujian	Waktu Pengenalan Wajah (ms)
1.	Noviana Putri P.			1636
				127
				92
2.	Pricillia C. K			103

				180
				122
3.	Adam Hendra B.			365
				291
				273
4.	Firman Jati P.			12721
				134
				101
5.	Deta Pratama			8718
				981
				109

Sumber : Pengujian

Perbandingan jumlah wajah yang dikenali antara kenyataan dengan sistem untuk kondisi wajah menghadap ke samping kiri ditunjukkan pada tabel 6.21 berikut :

Tabel 6.21 Perbandingan Jumlah Wajah Dikenali Antara Kenyataan Dan Sistem Untuk Kondisi Wajah Menghadap Ke Samping Kiri

		Hasil Pengenalan sistem	
		Dikenali	Tidak dikenali
Kenyataan	Dikenali	11	4
	Tidak dikenali	0	0

Sumber : Pengujian

Nilai akurasi sistem dinilai dengan cara menghitung nilai f_{11}, f_{00}, f_{01}, dan f_{10} pada Tabel 6.21.

$f_{11} = 11,\ f_{00} = 0,\ f_{01} = 0,\ f_{10} = 4$

$$Accuracy = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}} \times 100\%$$

$$= \frac{11+0}{11+4+0+0} \times 100\%$$

$$= 73.3\%$$

$$Error\ rate = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}} \times 100\%$$

$$= \frac{4+0}{11+4+0+0} \times 100\%$$

$$= 26.7\%$$

Waktu rata – rata yang dibutuhkan sistem untuk mengenali wajah mahasiswa dengan kondisi wajah menghadap ke samping kiri pada saat proses presensi :

Waktu rata – rata

$$= \frac{1636+127+92+103+180+122+365+291+273+12721+134+101+8718+981+109}{15}$$

= 1730.2 ms

6.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* yang telah dilakukan. Proses analisis mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi

analisis hasil pengujian unit, analisis hasil pengujian validasi dan analisis pengujian akurasi.

6.2.1 Analisis Hasil Pengujian Unit

Proses analisis terhadap hasil pengujian unit dilakukan dengan melihat kesesuaian fungsi dari implementasi unit modul yang diuji dengan hasil perancangan perangkat lunak yang telah dirancang sebelumnya. Berdasarkan hal tersebut, maka dapat diambil kesimpulan bahwa unit modul dari program sudah memenuhi kebutuhan fungsional yang telah dirancang pada tahap perancangan.

6.2.2 Analisis Hasil Pengujian Validasi

Proses analisis terhadap hasil pengujian validasi dilakukan dengan melihat kecocokan antara hasil kinerja sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian validasi dapat disimpulkan bahwa implementasi dan fungsionalitas Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

6.2.3 Analisis Hasil Pengujian Kinerja Sistem

Berdasarkan hasil pengujian kinerja sistem dapat disimpulkan bahwa Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dalam melakukan pengenalan wajah saat presensi dengan *input* pose wajah standar menghadap ke depan memiliki tingkat akurasi 100% dengan rata – rata waktu yang diperlukan saat proses presensi adalah 1005.467 ms. Untuk *input* dengan kondisi wajah menghadap ke samping kanan memiliki tingkat akurasi 80%, *error rate* 20% dan rata – rata waktu yang diperlukan 1542.267 ms. Sedangkan untuk *input* dengan kondisi wajah menghadap ke samping kiri memiliki tingkat akurasi 73.3%, *error rate* 26.7% dan rata – rata waktu yang diperlukan saat proses presensi adalah 1730.2 ms.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengamatan selama perancangan, implementasi, dan proses pengujian sistem presensi mahasiswa berbasis *face recognition* dengan algoritma *eigenface*, diambil kesimpulan sebagai berikut:

1. Aplikasi Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dikembangkan melalui beberapa tahap. Tahap pertama yaitu perancangan sistem yang meliputi perancangan *database*, perancangan *user interface* dan penerapan algoritma *eigenface*. Tahap kedua, implementasi sistem, dan tahap ketiga pengujian dan analisis sistem yang meliputi pengujian *White Box*, *Black box*, pengujian unit, pengujian validasi dan pengujian kinerja sistem.
2. Hasil pengujian *White Box* dan pengujian *Black box* pada Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* telah valid. Hal ini telah dibuktikan dengan dilakukannya proses pengujian unit dengan menghitung kompeksitas siklomatis serta pengujian validasi.
3. Berdasarkan hasil pengujian kinerja sistem, tingkat akurasi dari Sistem Presensi Mahasiswa Berbasis *Face Recognition* Dengan Algoritma *Eigenface* dalam melakukan pengenalan wajah saat presensi menggunakan *input* pose wajah standar menghadap ke depan memiliki tingkat akurasi 100% dengan rata – rata waktu yang diperlukan saat proses presensi adalah 1005.467 ms. Untuk *input* dengan kondisi wajah menghadap ke samping kanan memiliki tingkat akurasi 80%, *error rate* 20% dan rata – rata waktu yang diperlukan 1542.267 ms. Sedangkan untuk *input* dengan kondisi wajah menghadap ke samping kiri memiliki tingkat akurasi 73.3%, *error rate* 26.7% dan rata – rata waktu yang diperlukan saat proses presensi adalah 1730.2 ms.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem presensi mahasiswa berbasis *face recognition* dengan algoritma *eigenface* lebih lanjut antara lain :

1. Untuk meningkatkan akurasi pada pengembangan sistem berikutnya sebaiknya ditambahkan fitur-fitur lain, seperti fitur morfologi wajah sehingga akurasi pengenalan wajah dapat ditingkatkan.
2. Untuk pengembangan lebih lanjut sistem ini dapat dikembangkan dengan algoritma pengenalan wajah lainnya seperti *Fisherface*, *Eigenhill*, *Eigenedge* maupun algoritma lainnya [YIL-01].
3. Untuk pengembangan lebih lanjut pada pengujian sistem, data citra wajah untuk *training image* maupun *testing* dapat ditambahkan jumlah maupun posenya.

DAFTAR PUSTAKA

[NIN-06] Ning Tan, Pang, Michael Steinbach, Vipin Kumar. 2006. *Introduction To Data Mining*. Addison Wesley.

[PRE-10] Pressman, Roger S. 2010. S*oftware Engineering : A Practitioner's Approach, Seventh Edition*. McGraw Hill.

[PUR-10] Purnomo, Mauridhi Hery, Arif Muntasa. 2010. Konsep Pengenalan citra Digital Dan Ekstraksi Fitur. Graha Ilmu.

[SMI-02] Smith, Lindsay I. 2002. A Tutorial On Principal Component Analysis. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. Tanggal Akses 08 April 2013.

[SOM-11] Sommerville, Ian. 2011. *Software Engineering, Ninth Edition.* Pearson Education, Inc.

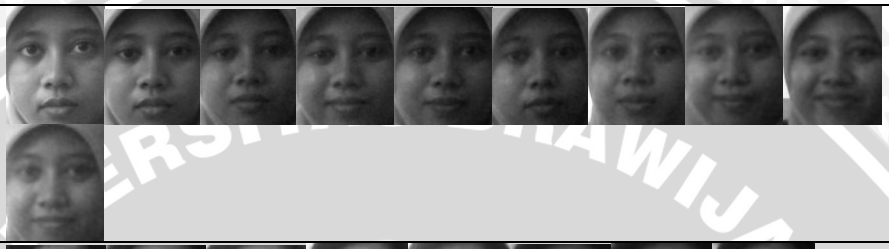
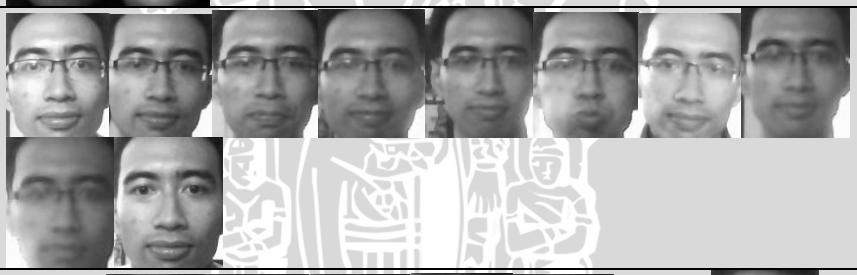
[TUR-91] Turk, M and A. Pentland. 1991. Eigenfaces for face Detection/Recognition. *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86.

[YIL-01] Yilmaz, Alper and Muhittin Gokmen. 2001. Eigenhill vs. Eigenface and Eigenedge. *Istanbul Technical University Department of Computer Science*. Pattern Recognition 34 (2001) 181-184.

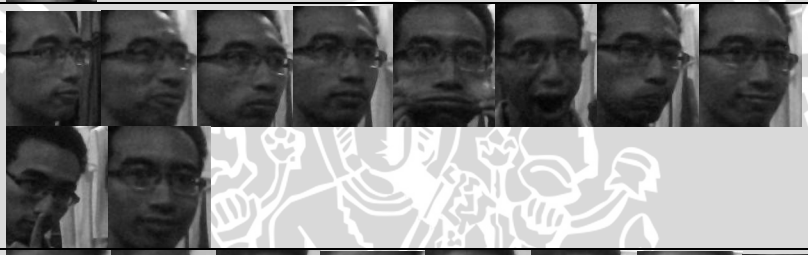
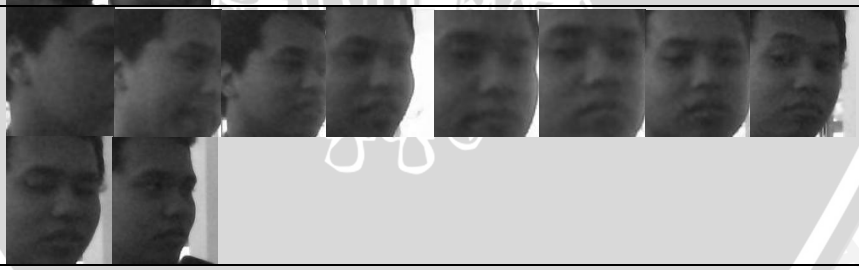
LAMPIRAN

LAMPIRAN 1

Data *Training image* Untuk pose Menghadap Ke Depan

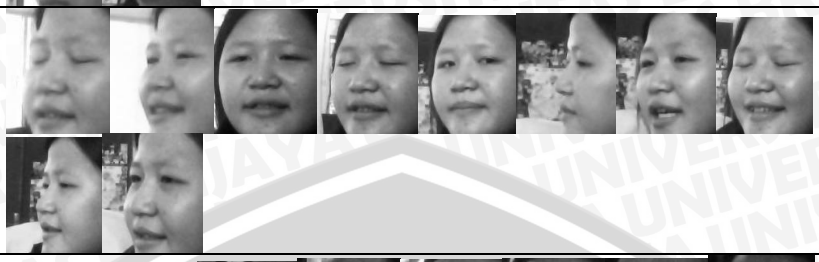
Nama Mahasiswa	**Data *Training image***
Noviana Putri P.	
Pricillia C. K.	
Adam Hendra B.	
Firman Jati P.	
Deta Pratama	

Data *Training image* Untuk pose Menghadap Ke Samping Kanan

Nama Mahasiswa	**Data *Training image***
Noviana Putri P.	
Pricillia C. K.	
Adam Hendra B.	
Firman Jati P.	
Deta Pratama	

Data *Training image* Untuk pose Menghadap Ke Samping Kiri

Nama Mahasiswa	**Data *Training image***
Noviana Putri P.	

	
Pricillia C. K.	
Adam Hendra B.	
Firman Jati P.	
Deta Pratama	

LAMPIRAN 2

Langkah – Langkah Menghasilkan Eigenface Menggunakan PCA

Terdapat 4 *training image* (merupakan grayscale image dengan nilai piksel antara 0-1) dan berukuran 2x2 piksel.

Training image 1

	1	2
1	1	0.727273
2	0.909091	0.636364

Training image 2

	1	2
1	0.9	1
2	0.5	0.7

Training image 3

	1	2
1	0.5	1
2	0.3125	0.75

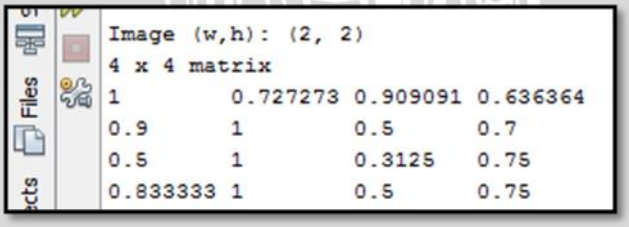
Training image 4

	1	2
1	0.833333	1
2	0.5	0.75

1. Masing – masing *training image* disusun menjadi *matrix* 1 baris.

Training image	X_1	X_2	X_3	X_4
1	1	0.727273	0.909091	0.636364
2	0.9	1	0.5	0.7
3	0.5	1	0.3125	0.75
4	0.833333	1	0.5	0.75

Tampilan Hasil Program



2. Mencari nilai rata – rata seluruh citra

Jumlah total masing-masing kolom (parameter) *training image*

3.233333	3.727273	2.221591	2.836364

Jumlah *training image*

4	4	4	4

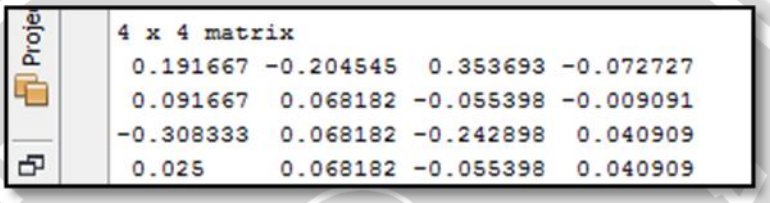
Nilai rata – rata seluruh citra

0.80833325	0.93181825	0.55539775	0.709091

3. Menghitung *Zero Mean* yaitu setiap nilai pada *training image* dikurangi dengan rata – rata tiap parameter yang terkait.

Training image	X_1	X_2	X_3	X_4
1	0.191667	-0.204545	0.353693	-0.072727
2	0.091667	0.068182	-0.055398	-0.009091
3	-0.308333	0.068182	-0.242898	0.040909
4	0.025	0.068182	-0.055398	0.040909

Tampilan Hasil Program



4. Hitung *Covariance Matrix*. Nilai *covariance matrix* bisa didapat dengan melakukan perkalian kombinasi dari setiap baris diatas.

					Total Jumlah	X-1 (4-1)	COV
$X_1 * X_1$	0.036736143	0.041838759	0.125098915	0.005289217	0.208963034	3	0.069654345
$X_1 * X_2$	0.017569468	-0.013946253	-0.01959381	0.000661161	-0.015309434	3	-0.005103145
$X_1 * X_3$	-0.059097232	-0.013946253	-0.085911295	-0.002975189	-0.161929969	3	-0.053976656
$X_1 * X_4$	0.004791621	-0.013946253	-0.01959381	-0.002975189	-0.031723631	3	0.010574544

					Total Jumlah	X-1 (4-1)	COV
$X_2 * X_1$	$X_1 * X_2$				$X_1 * X_2$		
$X_2 * X_2$	0.008402793	0.004648751	0.003068911	8.26463E-05	0.016203101	3	0.005401034
$X_2 * X_3$	-0.02826391	0.004648751	0.013455989	-0.00037190	-0.01053107	3	-0.00351036
$X_2 * X_4$	0.002291646	0.004648751	0.003068911	-0.00037190	0.009637404	3	0.003212468

					Total Jumlah	X-1 (4-1)	COV
$X_3 * X_1$	$X_1 * X_3$				$X_1 * X_3$		
$X_3 * X_2$	$X_2 * X_3$				$X_2 * X_3$		
$X_3 * X_3$	0.095069393	0.004648751	0.058999317	0.001673546	0.160391007	3	0.053463669
$X_3 * X_4$	-0.007708254	0.004648751	0.013455989	0.001673546	0.012070032	3	0.004023344

					Total Jumlah	X-1 (4-1)	COV
$X_4 * X_1$	$X_1 * X_4$				$X_1 * X_4$		
$X_4 * X_2$	$X_2 * X_4$				$X_2 * X_4$		
$X_4 * X_3$	$X_3 * X_4$				$X_3 * X_4$		
$X_4 * X_4$	0.000624988	0.004648751	0.003068911	0.001673546	0.010016196	3	0.003338732

	1	2	3	4
1	$X_1 * X_1$	$X_2 * X_1$	$X_3 * X_1$	$X_4 * X_1$
2	$X_1 * X_2$	$X_2 * X_2$	$X_3 * X_2$	$X_4 * X_2$
3	$X_1 * X_3$	$X_2 * X_3$	$X_3 * X_3$	$X_4 * X_3$
4	$X_1 * X_4$	$X_2 * X_4$	$X_3 * X_4$	$X_4 * X_4$

	1	2	3	4
1	0.208963034	-0.015309434	-0.161929969	-0.031723631
2	-0.015309434	0.016203101	-0.01053107	0.009637404
3	-0.161929969	-0.01053107	0.160391007	0.012070032
4	-0.031723631	0.009637404	0.012070032	0.010016196

Tampilan Hasil Program

Navigator

```
4 x 4 matrix
 0.208963 -0.015309 -0.16193  -0.031724
-0.015309  0.016203 -0.010531  0.009637
-0.16193  -0.010531  0.160391  0.01207
-0.031724  0.009637  0.01207   0.010016
```

5. Menghitung *eigenvalue* dan *eigenvector* dari *covariance matrix*.

Didapatkan nilai eigenvalue :

Services

```
1.341315E-017 0        0        0
0             0.00139  0        0
0             0        0.042683 0
0             0        0        0.351501
```

Sedangkan eigenvector :

Projects

```
-0.5 -0.026351 -0.419785 -0.757025
-0.5 -0.620481  0.603917  0.016954
-0.5 -0.126648 -0.562232  0.646417
-0.5  0.773479  0.3781    0.093653
```