REKAYASA PERANGKAT LUNAK PENCATATAN KEHADIRAN BERBASIS FACE RECOGNITION DAN GEOLOCATION

TUGAS AKHIR
Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan
Program Pendidikan Sarjana

Oleh:
Agus Juliyanto
2014130051

JURUSAN TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER – LIKMI
BANDUNG
2020

REKAYASA PERANGKAT LUNAK PENCATATAN KEHADIRAN BERBASIS FACE RECOGNITION DAN GEOLOCATION

Oleh:
Agus Juliyanto
2014130051

Bandung, 1 Februari 2020
Menyetujui,

Dr. Hery Heryanto S.Kom., M.Kom.	Dhanny Setiawan, S.T.,M.T.
Pembimbing	Ketua Jurusan

JURUSAN TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER – LIKMI
BANDUNG
2020

ABSTRAK

Manusia mampu mengenali banyak wajah, sebab wajah merupakan salah satu hal yang dapat membedakan satu sama lain. Sistem komputer tidak dapat mengenali wajah dengan mudah seperti manusia, karena itu diperlukan perangkat lunak yang mendukung hal itu. Sistem pengenalan wajah merupakan salah satu sistem identifikasi dimana suatu sistem komputer bekerja dengan cara mengenali identitas wajah seseorang. Penelitian ini dilakukan untuk menjawab bagaimana merancang sistem pengenalan wajah menggunakan metode *Local Binary Patterns*, dan seberapa besar tingkat akurasinya. Berdasarkan hal tersebut, penelitian ini berjudul "REKAYASA PERANGKAT LUNAK PENCATATAN KEHADIRAN BERBASIS *FACE RECOGNITION* DAN *GEOLOCATION*".

Sistem pengenalan wajah dalam penelitian ini diterapkan dalam sistem pencatatan kehadiran. Untuk melakukan absensi, dibutuhkan verifikasi wajah pengguna, dan pengambilan koordinat untuk menentukan posisi. Jika kamera mengalami masalah, pengguna dapat melakukan *login* untuk mencatat kehadiran. Perangkat lunak ini membutuhkan koneksi internet untuk mengambil koordinat melalui GPS yang tersedia dalam PC / *notebook*.

Dalam tahap perancangan, perangkat lunak ini menggunakan bahasa pemrograman C# dan menggunakan metode *Local Binary Patterns* (LBP) untuk sistem pengenalan wajah. Metode LBP memanfaatkan nilai piksel pada gambar, membagi gambar menjadi beberapa *grid*, kemudian mencari nilai tengah dengan menggunakan nilai piksel di sekitarnya. Lalu nilai pada tiap *grid* disatukan menjadi nilai histogram yang baru yang nantinya akan dibandingkan dengan data wajah yang masuk ke dalam sistem. Jika sistem mengenal wajah pengguna, maka sistem akan menentukan posisi dimana absensi dilakukan, dan mencatat kehadiran pengguna tersebut. Jika sistem tidak mengenali wajah pengguna, maka sistem akan menyarankan pengguna untuk *login username* dan *password* untuk dicatat kehadirannya.

Setelah melakukan beberapa pengujian, menentukan nilai FAR dan FRR serta menggunakan *Confusion Matrix* untuk menentukan tingkat akurasi sistem dalam mengenali wajah seseorang, tingkat akurasi yang didapatkan dari perangkat lunak yang dirancang adalah 92.90% dengan nilai *threshold* 150.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus atas kasih dan rahmat-Nya yang telah memimpin dan membimbing sehingga penulis mampu menyelesaikan penulisan tugas akhir dengan judul "REKAYASA PERANGKAT LUNAK PENCATATAN KEHADIRAN BERBASIS *FACE RECOGNITION* DAN *GEOLOCATION*" sebagai salah satu syarat kelulusan program pendidikan sarjana Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI.

Penulisan tugas akhir ini tidak dapat terselesaikan tanpa dukungan beberapa pihak. Secara khusus ucapan terimakasih disampaikan kepada :

1. Bapak Dr. Hery Heryanto S.Kom., M.Kom. selaku dosen pembimbing yang selalu mengingatkan untuk pengerjaan tugas akhir ini dan memberikan solusi ketika ada masalah dalam pengerjaannya.
2. Bapak Dhanny Setiawan S.T., M.T. selaku ketua jurusan S1 Teknik Informatika STMIK LIKMI.
3. Misto dan Sukati selaku orang tua dan Adi Mulyo selaku kakak yang selalu mendukung dengan memberikan semangat , nasihat dan dukungan moril agar pembuatan tugas akhir ini dapat terselesaikan.
4. Theofilus, Albert dan Stefanus serta semua teman gereja dan teman satu angkatan serta pihak lain yang tidak dapat disebutkan satu persatu yang membantu dan menyediakan waktu untuk membantu pengerjaan tugas akhir ini.
5. Seluruh staf dan karyawan STMIK LIKMI yang secara tidak langsung membantu kelancaran pembuatan tugas akhir ini.

Penulis juga berterimakasih kepada pembaca tugas akhir ini. Penulis berharap bahwa tugas akhir ini dapat membantu dan bermanfaat bagi banyak pihak serta menambah wawasan pembaca.

Penulis

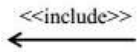
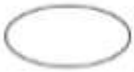
DAFTAR ISI

DAFTAR GAMBAR

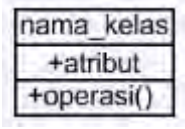
DAFTAR TABEL

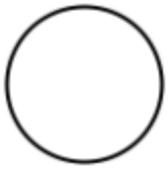
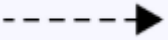
DAFTAR SIMBOL

Use Case Diagram

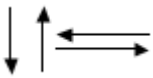
Nama	Simbol	Keterangan
Actor		Himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
Dependency		Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
Generalization		Hubungan dimana objek anak (*descendent*) berbagu perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
Include		Merincikan bahwa *use case* sumber secara eksplisit.
Extend		Merincikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
Association		Apa yang menghubungkan antara objek satu dengan objek lainnya.
System		Merincikan paket yang menampilkan sistem secara terbatas.
Use Case		Deskripsi urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
Collaboration		Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemenya.

Class Diagram

Nama	Simbol	Keterangan
Class		Simbol sebuah kelas pada struktur sistem. Penulisan tidak diperbolehkan menggunakan spasi. Memiliki 3 bagian yaitu nama kelas, atribut dan operasi.

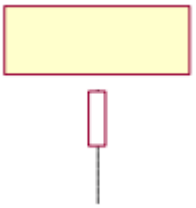
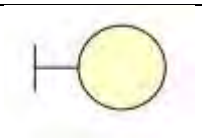
Nama	Simbol	Keterangan
Interface		Simbol untuk interface (antarmuka), konsep yang digunakan sama dengan pemrograman berorientasi objek (OOP).
Association		Garis yang digunakan untuk menghubungkan atau merelasikan kelas satu dengan kelas yang lainnya dengan makna umum.
Indirected Association		Simbol yang sama seperti association, namun digunakan jika kelas satu digunakan oleh kelas lainnya.
Generalization		Digunakan untuk menghubungkan antar kelas dengan arti umum-khusus.
Aggregation		Digunakan untuk menghubungkan antar kelas dengan makan untuk semua bagian.
Dependency		Digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* lainnya.

Activity Diagram

Nama	Simbol	Keterangan
Activity		Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
Action		Status dari sistem yang mencerminkan eksekusi dari suatu aksi.
Initial Node		Bagaimana objek dibentuk atau diawali.
Final Node		Bagaimana objek dibentuk dan diakhiri.
Decision		Digunakan untuk menggambarkan suatu keputusan/tindakan yang harus diambil pada kondisi tertentu.
Line Connector		Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

Sequence Diagram

Nama	Simbol	Keterangan
Actor		Mempresentasikan entitas yang berada di luar sistem dan berinteraksi dengan sistem. Bisa berupa manusia, perangkat keras maupun sistem yang lain.
Lifeline		Mengeksekusi objek selama sequence.

Nama	Simbol	Keterangan
General		Merepresentasikan entitas tunggal dalam sequence diagram . Entitas ini memiliki nama , stereotype atau berupa instance (class).
Boundary		biasanya berupa tepi dari sistem , seperti user interface atau suatu alat yang berinteraksi dengan sistem yang lain.
Control		Mengatur aliran dari informasi untuk sebuah skenario . Objek ini umumnya mengatur perilaku dan perilaku bisnis.
Entity		Elemen yang bertanggung jawab menyimpan data atau informasi . Ini dapat berupa beans atau model object.
Activation		Suatu titik dimana sebuah objek mulai berpartisipasi di dalam sebuah sequence yang menunjukkan kapan sebuah objek mengirim atau menerima objek.
Message Entry		Berfungsi untuk menggambarkan pesa/hubungan antar objek yang menunjukkan urutan kejadian yang terjadi.
Message to Self		Menggambarkan pesan/hubungan objek itu sendiri , yang menunjukkan urutan kejadian yang terjadi.
Message Return		nggambarkan hasil dari pengiriman message ı digambarkan dengan arah dari kanan ke kiri.

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Manusia sebagai makhluk yang cerdas, memiliki kemampuan untuk mengenal banyak wajah selama hidupnya. Manusia dengan mudah dapat mengenali sesorang melalui wajah meskipun hanya sekilas bertemu, maupun tidak bertemu dalam kurun waktu yang cukup lama. Manusia juga masih mampu mengenali seseorang walau terjadi perubahan karena perubahan usia atau perubahan fisik (kurus atau gemuk).

Wajah merupakan salah satu bagian dari anggota tubuh manusia yang mudah diingat jika dibandingkan dengan nama. Dalam beberapa kasus, terdapat orang-orang dengan nama yang persis sama, namun memiliki wajah yang berbeda. Oleh karena itu wajah dapat digunakan sebagai pembeda antara wajah satu dengan wajah lainnya. Sistem komputer tidak dapat mengenali wajah dengan mudah seperti manusia tanpa ada aplikasi yang mendukung. Sistem pengenalan wajah merupakan salah satu sistem identifikasi dimana suatu sistem komputer bekerja dengan cara mengenali identitas wajah seseorang. Tentu sebelum mengenali identitas wajah seseorang, sistem komputer harus belajar mengenali wajah tersebut. Dalam perkembangannya, sistem pengenalan wajah dapat menangani beberapa masalah keamanan seperti : pencarian orang dalam video pengawasan, kontrol akses pintu ruangan dan baru – baru ini mulai dikembangkan untuk kontrol perangkat *smartphone*.

Sistem absensi menggunakan identitas unik seseorang untuk mencatat kehadiran. Wajah merupakan salah satu identitas unik yang dapat digunakan dalam sistem absensi. Dalam beberapa kasus, absensi harus dilakukan di tempat dimana identitas terdaftar, yang mana cukup menghabiskan banyak waktu jika tempat absensi cukup jauh. Oleh karena itu diperlukan sistem absensi yang dapat mencatat kehadiran tanpa harus ke tempat dimana identitas didaftarkan. Pada penelitian ini, penulis merancang sistem teknik pengenalan wajah dengan pendekatan fitur menggunakan *Local Binary Patterns* (LBP) dan menggunakan teknik *Geolocation* untuk menentukan posisi absensi yang dilakukan.

Berdasarkan analisa di atas, maka penelitian ini mengusulkan judul "REKAYASA PERANGKAT LUNAK PENCATATAN KEHADIRAN BERBASIS *FACE RECOGNITION* DAN *GEOLOCATION*".

1.2 Rumusan Masalah

Penelitian ini diarahkan untuk menyelesaikan beberapa permasalahan berikut :

1. Bagaimana merancang sistem absensi berbasis pengenalan wajah menggunakan metode *Local Binary Patterns*?
2. Berapa besar tingkat akurasi sistem dalam mengenali wajah seseorang?

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Menerapkan metode *Local Binary Patterns* pada sistem pengenalan pola wajah
2. Menganalisis kinerja, akurasi dan kecepatan sistem dalam mengenali wajah

1.4 Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah :

1. Perangkat lunak yang dirancang hanya untuk windows pc (*personal computer*)
2. Metode yang digunakan adalah *Local Binary Patterns*

1.5 Kegunaan Hasil

Penelitian ini diharapkan dapat menghasilkan suatu sistem pengenalan wajah berupa perangkat lunak yang mampu mengidentifikasi wajah seseorang. Selain itu untuk menambah ilmu pengetahuan mengenai bidang sistem biometrika, pengenalan pola, dan pengolahan citra khususnya pengenalan pola wajah dengan metode *Local Binary Patterns.*

1.6 Metodologi Penelitian

Metode – metode yang digunakan dalam pembuatan tugas akhir ini adalah sebagai berikut :

1. Studi Pustaka

   Memperoleh informasi dengan membaca, mengumpulkan, dan mempelajari berbagai referensi buku jurnal dan sumber – sumber bacaan lainnya yang merupakan landasan teori dan sumber inspirasi dalam menyelesaikan tugas akhir ini.

2. Perancangan Sistem

   Merancang bagaimana sistem absensi dapat mencatat kehadiran dengan mengenali wajah seseorang dan menentukan posisi absensi.

3. Desain dan Penulisan Program

   Membuat desain antarmuka dan menuliskan program.

4. Testing

   Setelah aplikasi dibuat, aplikasi akan diuji untuk mengukur ketepatan program dalam pencatatan kehadiran, mengenali wajah seseorang dan menetukan posisi absensi.

1.7 Sistematika Penulisan

Adapun sistematika penulisan ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika yang digunakan adalah sebagai berikut :

1. BAB I : Pendahuluan

   Dalam bab ini diuraikan mengenai latar belakang yang meliputi alasan pemilihan topik, perumusan masalah, tujuan penulisan, batasan masalah yang akan dibahas, kegunaan hasil, metodologi penelitian, serta sistematika penulisan.

2. BAB II : Landasan Teori

   Dalam bab ini berisi landasan teori yang akan digunakan untuk mendukung penulisan tugas akhir yang meliputi penjelasan tentang rekayasa perangkat lunak, pengenalan pola wajah, pengolahan citra, metode *Local Binary Patterns*, *Unified Modeling Language* dan *Object Oriented Analysis and Design*, dan Visual Studio.

3. BAB III : Analisis dan Perancangan

   Dalam bab ini akan diuraikan mengenai teknik – teknik analisis dan perancangan perangkat lunak mulai dari pembuatan *use case diagram*, *class diagram*, *activity diagram*, *sequence diagram*, dan perancangan antarmuka.

4. BAB IV : Implementasi dan Pengujian

   Dalam bab ini menjelaskan tentang implementasi pengenalan pola wajah yang dibuat serta pengujian atas bagian – bagian dan komponen yang terdapat dalam aplikasi ini.

5. BAB V : Kesimpulan dan Saran

   Dalam bab terakhir ini memuat tentang kesimpulan dari identifikasi masalah yang ada serta menjelaskan hal yang diperoleh dari penelitian dan pengembangan sistem pengenalan wajah yang dilakukan. Juga berisi tentang saran bagi pengembangan aplikasi sejenis pada masa yang akan datang.

BAB II

LANDASAN TEORI

2.1 Rekayasa Perangkat Lunak

Dalam buku yang berjudul *Software Engineering – Third Revised Edition*, rekayasa perangkat lunak didefinisikan sebagai :

"*Sofware engineering is a discipline in which theories, methods and tools are applied to develop professional software. In software engineering a systematic and organized approach is adopted. Based on the nature of problem and development constraints various tools and techniques are applied in order to develop quality software."* (Puntambekar, 2009:1).

Berdasarkan definisi tersebut, maka dapat digambarkan bahwa rekayasa perangkat lunak merupakan suatu disiplin dimana teori, metode dan alat, diterapkan untuk mengembangkan suatu perangkat lunak yang profesional. Dalam pengembangan perangkat lunak yang berkualitas, berbagai alat dan teknik yang digunakan bergantung kepada permasalahan yang muncul.

Rekayasa perangkat lunak menurut IEEE (*Institute of Electrical and Electronics Engineers*) yang disertakan dalam buku "*Software Engineering A Practitioner's Approach Eight Edition*" adalah sebagai berikut

"*Software Engineering: (1) The application of a sistematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study off approaches as in (1)*
(Pressman, 2015:15)."

Berdasarkan definisi tersebut, dapat digambarkan bahwa rekayasa perangkat lunak merupakan penerapan terhadap pengembangan, pengerjaan dan pemeliharaan perangkat lunak yang bersifat sistematis, disiplin, dan terukur.

Dari beberapa definisi para ahli, maka dapat disimpulkan bahwa rekayasa perangkat lunak merupakan penerapan segala aspek yang digunakan untuk mengembangkan dan memelihara sistem perangkat lunak .

2.1.1 Model Pengembangan

Dalam buku yang berjudul "*Software Engineering (A Lifecycle Approach)*", Pratap K.J. Mohapatra menjelaskan bahwa proses pengembangan perangkat lunak mengambil rute yang berbeda pada waktu yang berbeda di masa lalu. Seseorang dapat melihat model ideal dari proses pengembangan perangkat lunak : (Mohapatra, 2009:17)

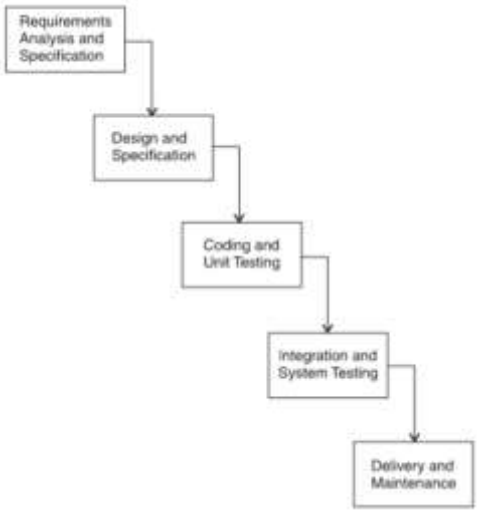
1. *The Code-and-Fix Model*

   Model ini merupakan pengembangan yang dilakukan satu orang, yang ditandai sebagai berikut :

   a. Aplikasi yang dikembangkan merupakan aplikasi teknik atau sains.
   b. Pengembang juga merupakan pengguna aplikasi.
   c. Persyaratan sepenuhnya diketahui.
   d. Pengembangan produk perangkat lunak melibatkan *coding* dan *fixing bugs*, jika ada.

2. *The Waterfall Model*

   Perangkat lunak digambarkan seperti makhluk hidup dengan urutan perngembangan yang jelas, mulai dari konseptualisasi masalah (kemunculan ide di fase pertama) hingga membuang perangkat lunak (kematian perangkat lunak di fase terakhir). Pekerjaan dalam pengembangan dapat dibagi menurut fase di antara kelas spesialis yang berbeda.

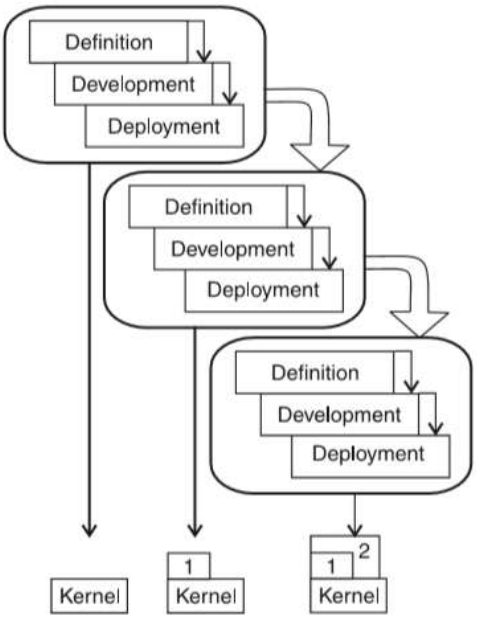


Gambar 2.1
Model *Waterfall* dari Royce (1970)
(Sumber : Mohapatra, 2009:19)

Dalam perkembangannya, terdapat modifikasi yang dibuat oleh Boehm pada tahun 1981 dengan memberikan umpan balik ke fase sebelumnya.

3. The Evolutionary Model

Dalam pendekatan evolusioner, model kerja perangkat lunak dikembangkan dan disajikan kepada pelanggan untuk umpan baliknya, sebagai penggabungan dan pengiriman perangkat lunak. Pendekatan evolusi dapat diimplementasikan dalam dua bentuk, *Incremental Implementation* dan *Prototyping*. Dalam *Incremental Implementation*, perangkat lunak dikembangan dalam pengingkatan kemampuan fungsional, yaitu dengan membagi pengembangan menjadi beberapa langkah, menunda beberapa bagian dengan maksud menghasilkan fungsi kerja yang berguna sebelumnya dalam pengembangan proyek. Fungsi-fungsi lain secara perlahan akan ditambahkan sebagai peningkatan.



Gambar 2.2
Model Pengembangan *Incremental*
(Sumber : Mohapatra, 2009:26)

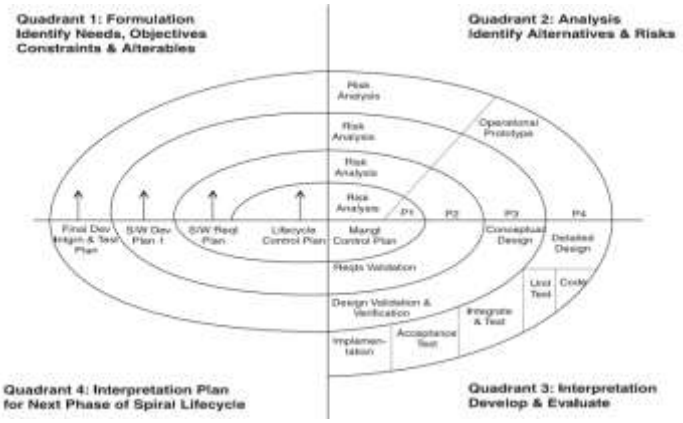
Metode *Prototyping* didasarkan pada prosedur eksperimental dimana prototipe kerja perangkat lunak diberikan kepada pengguna untuk komentar dan umpan balik. Ini membantu pengguna untuk mengekspresikan persyaratan dalam istilah yang lebih definitif dan konkret. Model prototipe dapat dibagi menjadi dua jenis,

The Rapid Throwaway Prototyping (Scaffolding) dan *Evolutionary Prototyping*. *Throwaway prototyping* melakukan prinsip dua kali kerja. Versi awal perangkat lunak dikembangkan hanya untuk memperoleh persyaratan informasi dari pengguna, setelah itu perangkat lunak dibuang dan dibuat versi dua yang dikembangkan dengan skala penuh. Dalam *Evolutionary Prototyping,* prototipe awal tidak dibuang, tetapi dimodifikasi secara progresif sehingga menjadi aplikasi akhir.

Gambar 2.3
Model Pengembangan Prototipe
(Sumber : Mohapatra, 2009:28)

4. *The Spiral Model*

Model pengembangan ini dapat digambarkan dalam bentuk spiral yang searah jarum jam. Setiap siklus menggambarkan fase tertentu dari siklus hidup pengembangan perangkat lunak. Dengan demikian, siklus bagian dalam mungkin berhubungan dengan analisis kebutuhan, berikutnya dengan siklus desain, dan seterusnya. Manajemen menentukan fase-fase, sehingga jumlah siklus dalam model spiral bervariasi antar organisasi, antar proyek, bahkan antar proyek dalam organisasi yang sama.

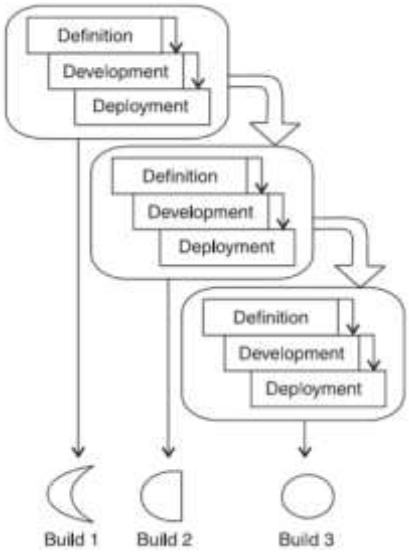


Gambar 2.4
Model Pengembangan Spiral
(Sumber : Mohapatra, 2009:31)

Dalam perancangan perangkat lunak sistem pengenalan wajah ini, digunakan *Evolutionary Model* dengan jenis *Evolutionary Prototyping* sebagai model pengembangan.

2.1.2 Evolutionary Prototyping

Evolutionary Prototyping merupakan model pengembangan perangkat lunak dengan sistem terus dimodifikasi dan disempurnakan sesuai dengan umpan balik pengguna sampai pengguna merasa puas.



Gambar 2.5
Model Pengembangan *Evolutionary Prototyping*
(Sumber : Mohapatra, 2009:29)

Terdapat beberapa keuntungan dari model pengembangan ini :

1. Dapat mengidentifikasi salah paham antara pihak pengembang dan pengguna.
2. Dapat mendeteksi layanan pengguna yang hilang.
3. Layanan yang sulit digunakan atau membingungkan dapat diidentifikasi dan disempurnakan.
4. Pengembang mungkin menemukan persyaratan yang tidak konsisten.

2.2 Absensi

Absensi merupakan suatu kegiatan atau rutinitas yang dilakukan oleh pegawai sebagai bukti kehadiran maupun tidak hadir dalam bekerja di suatu perusahaan tertentu. Absensi berkaitan dengan penerapan disiplin yang ditentukan oleh masing-masing perusahaan atau institusi.

Absensi dapat dikatakan suatu pendataan kehadiran yang merupakan bagian dari aktifitas pelaporan yang ada adalam sebuah institusi. Secara umum jenis-jenis absensi menurut cara penggunaannya dapat dikelompokkan menjadi dua : (Budi S., 2015:44)

1. Absensi manual

   Merupakan cara penulisan kehadiran dengan cara menggunakan alat tulis berupa tanda tangan.

2. Absensi non manual

   Merupakan cara penulisan kehadiran dengan menggunakan bantuan sistem komputerisasi, bisa menggunakan kartu RFID ataupun penggunan data biometrik.

2.3 Biometrik

Sistem biometrik menurut Anil K. Jain, Arun A. Ross, dan Karthik Nandakumar dalam buku yang berjudul "*Introduction to Biometrics*" adalah sebagai berikut :

"*A biometric system measures one or more physical or behavioral characteristics, including fingerprint, palmprint, face, iris, retina, ear, voice, signature, gait, hand vein,*

odor, or the DNA information of an individual to determine or verify his identity" (Jain, 2011:3).

Berdasarkan definisi diatas, biometrik merupakan sistem pengukuran informasi satu atau lebih fisik atau perilaku karakter (termasuk sidik jari, telapak tangan, wajah, iris mata, retina mata, telinga, suara, tanda tangan, gaya berjalan, pembuluh vena di tangan, aroma atau *Deoxyribonucleic Acid* (DNA)) dari seseorang untuk menentukan atau membuktikan indentitasnya.

Dalam buku lain yang berjudul "*Biometric Technology Authentication, Biocryptography, and Cloud-Based Architecture*", biometrik didefinisikan sebagai berikut "*the use of computer science technology to extract the unique features of an individual, whether it be physical or behavioral traits, in order to positively verify and/or identify the identity of an individual, so that they may have access to certain resources*"
(Ravindra, 2014:6).

Berdasarkan definisi diatas, biometrik merupakan penggunaan teknologi komputer sains untuk mengambil fitur unik dari seseorang baik itu fisik maupun karakteristik, dengan maksud untuk mengidentifikasi, sehingga ia dapat menggunakan sumber daya tertentu.

Biometrik menurut Peter Gregory dan Michael A. Simon dalam bukunya yang berjudul "*Biometric for Dummies*" adalah sebagai berikut
"*The term biometrics comes from the ancient Greek bios = "life" and metron = "measure." Biometrics refers to the entire class of technologies and techniques to uniquely identify humans*" (Gregory, 2008:9).
Berdasarkan definisi di atas, biometrik mengacu pada keseluruhan teknologi dan teknik untuk mengidentifikasi manusia secara unik.

Berdasarkan beberapa definisi diatas, dapat disimpulkan bahwa biometrik merupakan penggunaan teknologi untuk menghitung fitur unik dari manusia baik fisik maupun karakter yang berfungsi untuk mengidentifikasi identitas seseorang .

Biometrik sendiri terbagi menjadi 2 jenis : (Gregory, 2008:11)

3. Physiological

Physiological Biometric mengukur bagian spesifik dari struktur atau bentuk dari porsi tubuh. *Physiological Biometric* terbagi menjadi beberapa bagian :

a. Fingerprint

b. *Hand Scan*

c. Iris Scan

d. Retina Scan

e. Facial Scan

4. Behavioral

Behavioral Biometric lebih berfokus kepada bagaimana manusia melakukan sesuatu, dibanding hanya mengukur bagian spesifik dari tubuh. *Behavioral Biometric* terbagi menjadi beberapa bagian :

a. Handwriting

b. Keystroke Dynamic

c. Voice Recognition

d. Gait

Tujuh faktor yang menentukan kesesuaian dari sifat fisik, perilaku yang unik untuk sistem biometrik, sebagai berikut :

1. *Uniqueness*, yang berarti harus cukup berbeda antar individual damal sebuah populasi.
2. *Measurability*, tang diartikan sebagai kemungkinan untuk mendapatkan sifat biologis melalui perangkat.
3. *Universality of techniques*, yang berarti bahwa akses tersebut (mendapatkan sifat biologis) dapat diterima secara umum.
4. *Acceptability*, yang mengindikasikan kesediaan masyarakat untuk memanfaatkan sistem tersebut.
5. Kinerja harus menunjukkan keakuratan dan keterulangan berdasarkan batasan yang diberikan.
6. *Permanence*, yang menunjukkan bahwa sifat biometrik tidak berubah ke tingkat tertentu selama periode waktu tertentu.

7. *Circumvention*, yang menunjukkan bahwa sistem tidak responsif artefak palsu dan menolak peniruan ciri perilaku.

2.3.1 Sistem Pengenalan Wajah

Pengenalan wajah merupakan nilai komersial yang tinggi karena keterlibatan dari beberapa tantangan dalam prosesnya. Ciri atau sifat gambar wajah dapat berubah dalam kondisi yang berbeda, bahkan sebelum atau sesudah pengambilan gambar menggunakan kamera. Kualitas gambar wajah juga tergantung dari sensor kamera yang digunakan dalam mengambil gambar. Tantangan utama dalam pengenalan wajah yaitu bagaimana meneliti perubahan besar pada penampilan wajah dalam pose, perubahan cahaya dan ekspresi wajah yang berbeda, sedangkan bentuk dan refleksi adalah milik dasar dari objek wajah. Penampilan wajah juga dipengaruhi oleh beberapa faktor lain, seperti pose wajah, sudut pandang kamera, pencahayaan dan ekspresi wajah (Karande, 2014:4).

2.3.2 Pengolahan Citra

Pengolahan citra mengacu kepada segala proses pengolahan citra, khusunya menggunakan komputer. Pengolahan citra dapat dibedakan menjadi beberapa bagian berikut :

1. *Low-level Processing*

   Dalam *low-level processing*, masukan(*input*) dan keluaran(*output*) merupakan citra digital.

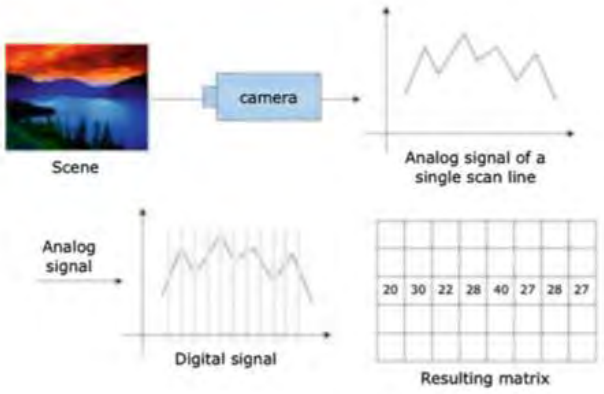
2. *Medium-level Processing*

   Dalam *medium-level processing*, masukan merupakan citra digital, dan beberapa fitur dapat diambil seperti bidang atau titik tepi.

3. *High-level Processing*

   Dalam *high-level processing*, masukan merupakan citra digital, sedangkan keluaran merupakan deskripsi konten dari masukan citra digital.

Citra digital merupakan hasil representasi dua dimensi dari sebuah objek atau layar tiga dimensi yang dihasilkan dari proses digitalisasi. Sebuah gambar dapat dianggap sebagai sinyal digital dua dimensi yang diperoleh dari dua proses dasar : sampling dalam domain spasial dan kuantisasi di domain nilai / level (Caponetti, 2017:3).



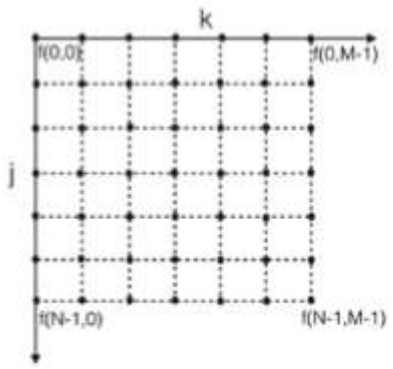
Gambar 2.6
Proses Perolehan Gambar dan Proses Digitalisasi
(Sumber : Caponetti, 2017:4)

Dengan kata lain, citra digital didefinisikan sebagai fungsi digital dua dimensi *f(x, y)* yang merupakan pemetaan dari kordinat spasial domain D ke nilai instensitas domain D' :

$$f = D \rightarrow D' \tag{2.1}$$

Dimana D merupakan domain terbatas yang terdiri dari koordinat diskrit (*x, y*) dan D' merupakan nilai-nilai dari domain diskrit yang disebut *gray levels*.

Citra digital *f* memiliki nilai yang tidak negatif dengan jumlah yang terbatas, khususnya dalam jarak antara 0 sampai 255. Dapat direpresentasikan menggunakan matriks elemen N x M, dimana N merupakan jumlah baris dan M merupakan jumlah kolom. Setiap elemen *f(j, k)* untuk $0 \leq j \leq N$ dan $0 \leq k \leq M$ merupakan *pixel* yang memiliki *abscissa* x = k dan *ordinate* y = - j. Kemudian citra digital dapat diwakili menggunakan sistem koordinat logis *(x, y)* atau sistem koordinat gambar *(j, k)*, dimana x = k adalah kolom dan y = - j adalah baris.



Gambar 2.7
Sistem Koordinat Gambar
(Sumber : Caponetti, 2017:5)

Berikut merupakan unsur citra digital :

1. *Dimension*, merupakan jumlah *pixel* dari sebuah gambar.
2. *Size*, merupakan jumlah baris dan kolom dari sebuah gambar atau juga merupakan sebuah ukuran (lebar dan tinggi) yang dinyatakan dalam *cm/inch*.
3. *Spatial Resolution*, merupakan jumlah dari ukuran satuan pixel yang dinyatakan dalam *dot per inch (dpi)*.
4. *Dynamic Range* atau bisa juga disebut *gray-level range*, merupakan jumlah perbedaan dari *gray-level* yang terdapat dalam citra/gambar.

Citra digital dapat diklasifikasikan menjadi dua, citra *Raster* atau *Bitmap* dan citra *Vector*. Citra *bitmap* atau *raster* adalah himpunan persegi panjang dari nilai sampel atau piksel. Citra ini memiliki jumlah piksel yang tetap. Citra *vector* merupakan nilai yang disimpan dalam bentuk garis dan kurva matematika. Informasi seperti panjang, warna, ketebalan dan informasi lainnya disimpan dalam bentuk vektor. (Vipin, 2018:7) Terdapat beberapa tipe *file* citra digital :

1. *Tagged Image File Format* (TIFF)

   Format yang menggunakan teknik *lossless* dalam penyimpanan gambar, karena itu ukurannya cukup besar.
2. *Portable Network Graphics* (PNG)

   Format yang menggunakan teknik lossless dan menggunakan pola pada gambar untuk mengompres gambar.

3. *Graphical Interchange Format* (GIF)

   Format yang mengompres gambar menjadi 256 warna. Jika gambar yang dikompres memiliki kurang dari 256 warna, maka gambar GIF memiliki warna yang sama, jika lebih dari 256, maka GIF mendekati warna dalam tabel 256 warna yang tersedia.

4. *Joint Picture Experts Group* (JPG atau JPEG)

   Format yang dioptimalkan untuk foto dan gambar dengan *tone* yang berjumlah besar. File JPEG mencapai rasio kompresi yang tinggi dengan tetap menjaga kualitas gambar.

5. RAW

   Format gambar *lossless* yang tersedia pada beberapa kamera digital.

6. *Bitmapped Image* (BMP)

   Format eksklusif tanpa kompresi yang ditemukan oleh Microsoft.

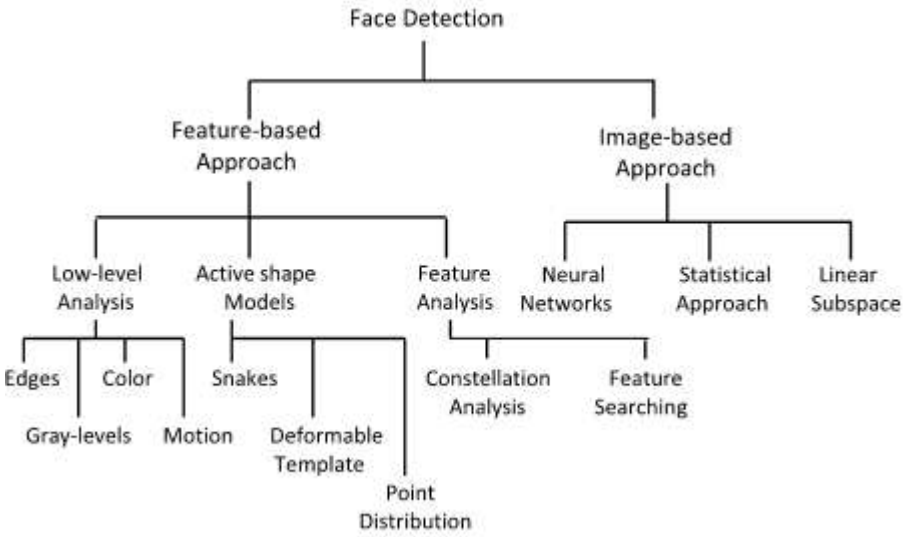
2.3.3 Deteksi Wajah

Deteksi wajah merupakan langkah pertama yang penting dalam sistem pengenalan wajah, dengan tujuan lokalisasi dan ekstraksi daerah bagian wajah dari *background*. Terdapat banyak jenis teknik yang digunakan dalam sistem pengenalan wajah, dimulai dari pendekatan yang sederhana berdasarkan batas tepi (*edge*) sampai pendekatan tingkat tinggi dengan menggabungkan beberapa metode pengenalan pola lanjutan (Datta, 2015:19).

Tantangan yang terkait dalam proses mendeteksi wajah dapat disebabkan oleh beberapa faktor berikut :

1. *Occlusion*, wajah mungkin saja terhalangi oleh benda atau objek lain. Dalam gambar dengan sekelompok orang, beberapa wajah mungkin dapat menutupi wajah lainnya.
2. *Facial expression*, penampilan wajah secara langsung dipengaruhi oleh ekspresi seseorang.
3. *Pose*, citra wajah dapat bervariasi berdasarkan pose.

4. *Illumination*, saat gambar wajah sudah terbentuk, faktor kondisi cahaya dan karakteristik kamera juga dapat mempengaruhi penampilan wajah.



Gambar 2.8
Macam – macam Teknik Dalam Pengenalan Wajah
(Sumber : Datta, 2015:20)

Segala teknik pengenalan wajah yang ada dapat diklasifikasikan menjadi lima jenis berdasarkan cara dalam mengidentifikasi wajah, yaitu :

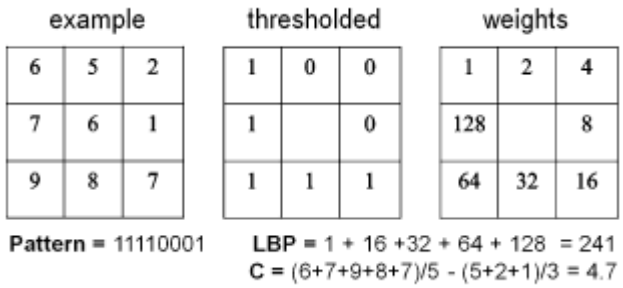
1. *Appearance-(feature) based*, yang menggunakan semuar fiture termasuk tekstur.
2. *Subspaced-based.*
3. *Techniques using neural networks.*
4. *Model-based,* yang bekerja dalam bentuk dan tekstur, juga informasi kedalaman tiga dimensi.
5. Teknik lainnya seperti hubungan antar teknik dan penggunaan mesin vektor yang mendukung.

2.3.4 *Local Binary Patterns*

Local Binary Patterns adalah operator tekstur yang sederhana namun sangat efisien, yang memberikan label piksel dari suatu gambar dengan cara menetapkan batas lingkungan setiap piksel dan menganggap hasilnya sebagai angka biner. Metode LBP dapat dilihat sebagai metode yang menyatukan perbedaan statistik dan struktur model pada analisis tekstur secara tradisional. Mungkin hal paling penting dari operator LBP

dalam aplikasinya di dunia nyata adalah tidak adanya perubahan terhadap perubahan tingkat abu-abu (*gray level*) monotonik yang disebabkan, misalnya oleh variasi penerangan. Hal lain yang sama pentingnya ialah kesederhanaan komputasinya, yang memungkinan untuk menganalisa gambar secara langsung. (Pietikäinen, 2011:4)

Dasar dari operator LBP, diperkenalakan oleh Ojala dkk., didasarkan dengan asumsi bahwa tekstur memiliki dua aspek yang saling melengkapi, yaitu pola dan kekuatannya sendiri. LBP dimaksud sebagai versi tingkat dua tekstur untuk menggambarkan pola tekstur lokal. Versi asli dari operator LBP, berfungsi dalam 3x3 blok piksel gambar. Piksel dalam blok-blok tersebut dimulai dari nilai tengah nya, dikali dua kemudian dijumlahkan untuk menghasilkan label pada piksel tengah. Karena blok sekeliling terdiri dari 8 piksel, sehingga total terdapat 256 (hasil dari 2^8) label berbeda yang dapat diperoleh, tergantung pada nilai abu-abu relatif dari pusat dan sekeliling piksel tersebut. Ukuran kontras (C) diperoleh dengan mengurangi rata-rata nilai abu-abu yang nilainya dibawah piksel tengah dari nilai abu-abu yang nilainya di atas (atau sama dengan) piksel tengah.



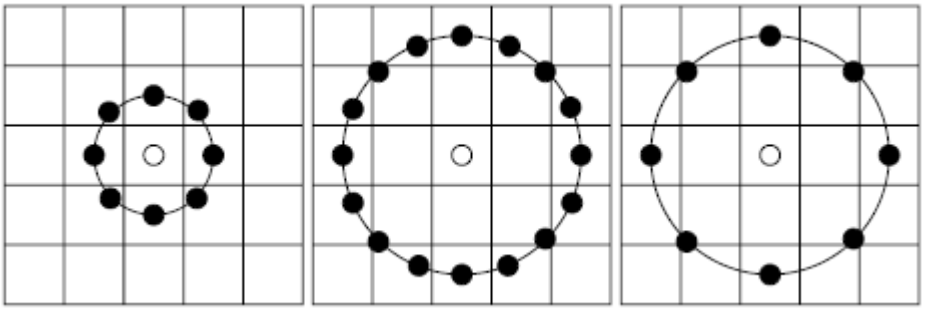
Gambar 2.9
Ilustrasi Dasar Operator LBP
(Sumber : Pietikäinen, 2011:5)

LBP menggunakan 8 piksel dalam blok 3x3 piksel, rumus umum dari operator ini tidak membatasi ukuran lingkungan atau jumlah poin sampel. Bayangkan sebuah gambar monokrom *I (x, y)* dan misalkan gc menunjukkan nilai abu-abu dari pixel yang berubah-ubah *(x, y)*, yaitu $g_c = I\ (x, y)$. Selain itu, misalkan g_p menunjukkan nilai abu-abu dari titik pengambilan sampel di lingkungan melingkar yang sama dari titik pengambilan sampel *P* dan jari-jari *R* di sekitar titik *(x, y)*:

$$g_p = I\,(x, y), \quad p = 0, \ldots, P-1,$$

$$x_p = x + R\cos(\frac{2\pi p}{P}), \tag{2.2}$$

$$y_p = y - R\cos\left(\frac{2\pi p}{P}\right).$$



Gambar 2.10
Lingkungan melingkar (8,1), (16,2) dan (8,2)
(Sumber : Pietikäinen, 2011:14)

Dengan asumsi bahwa tekstur lokal dari gambar I (x, y) ditandai oleh distribusi bersama nilai abu-abu dari P +1 (P> 0) piksel:

$$T = t(g_c,\ g_0, g_1, \ldots, g_{P-1}). \tag{2.3}$$

Tanpa kehilangan informasi, nilai piksel tengah dapat dikurangi dari nilai sekeliling:

$$T = t(g_c,\ g_0 - g_c, g_1 - g_c, \ldots, g_{P-1} - g_c). \tag{2.4}$$

Pada langkah selanjutnya distribusi bersama diperkirakan dengan mengasumsikan piksel tengah tidak bergantung secara statistik dari perbedaan, yang memungkinkan untuk faktorisasi distribusi:

$$T \approx t(g_c)\, t(g_0 - g_c, g_1 - g_c, \ldots, g_{P-1} - g_c). \tag{2.5}$$

Faktor pertama *t (gc)* adalah distribusi intensitas lebih dari *I (x, y)*. Dari sudut pandang analisa pola tekstur lokal, tidak ada informasi yang dapat digunakan. Sebaliknya distribusi perbedaan bersama

$$t(g_0 - g_c, g_1 - g_c, \ldots, g_{P-1} - g_c) \tag{2.6}$$

dapat digunakan untuk memodelkan tekstur lokal. Namun, estimasi yang dapat diandalkan dari distribusi multidimensi ini dari data gambar cukup sulit. Salah satu solusi untuk masalah ini, adalah dengan menerapkan kuantisasi vektor. Menggunakan pembelajaran kuantisasi vektor dengan *codebook* 384 kata sandi untuk mengurangi dimensi ruang fitur dimensi tinggi. Indeks 384 kata sandi sesuai dengan 384 tempat di histogram. Dengan demikian, operator yang kuat ini didasarkan pada perbedaan tingkat abu-abu yang telah ditandai yang dapat dianggap sebagai operator *texton*, menyerupai beberapa metode yang lebih baru.

Pendekatan pembelajaran kuantisasi vektor masih memiliki sifat tertentu yang membuat penggunaannya sulit. Pertama, perbedaan g_p - g_c tidak berbeda dengan perubahan nilai rata - rata abu - abu dari gambar tetapi tidak terhadap perubahan lain dalam tingkat abu - abu. Kedua, untuk menggunakannya dalam klasifikasi tekstur, *codebook* harus dilatih serupa dengan metode berbasis *texton* lainnya. Untuk mengatasi tantangan-tantangan ini, hanya tanda-tanda perbedaan yang dipertimbangkan:

$$t\big(s(g_0 - g_c), s(g_1 - g_c), \dots, s(g_{P-1} - g_c)\big), \tag{2.7}$$

di mana *s (z)* adalah fungsi *thresholding*

$$s(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{2.8}$$

Operator LBP yang umum berasal dari distribusi gabungan ini. Seperti dalam kasus LBP dasar, diperoleh dengan menjumlahkan perbedaan nilai batas yang dikali dua. Operator $LBP_{P,R}$ didefinisikan sebagai

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^P. \tag{2.9}$$

Persamaan diatas menunjukkan bahwa tanda-tanda perbedaan dalam lingkungan ditafsirkan sebagai angka biner *P*-bit, menghasilkan 2^P nilai yang berbeda untuk kode LBP. Distribusi skala abu-abu lokal, misalnya tekstur, dapat dijelaskan dengan distribusi diskrit LBP 2^P-*bin* :

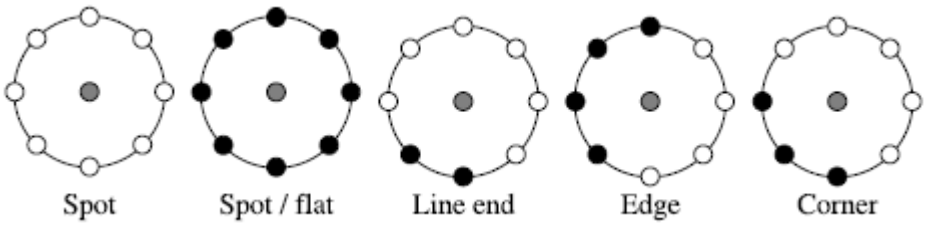
$$T \approx t\left(LBP_{P,R}(x_c, y_c)\right) \tag{2.10}$$

Dalam menghitung $LBP_{P,R}$ distribusi (vektor) untuk sampel gambar N × M yang diberikan ($x_c \in \{0, ..., N - 1\}$, $y_c \in \{0,.,., M - 1\}$), hanya bagian tengah yang dipertimbangkan karena lingkungan yang cukup besar tidak dapat digunakan di perbatasan. Kode LBP dihitung untuk setiap piksel di bagian yang dipotong dari gambar, dan distribusi kode digunakan sebagai vektor fitur, dilambangkan dengan *S* :

$$S = t\left(LBP_{P,R}(x, y)\right),$$
$$x \in \{[R], \ldots, N - 1 - [R]\}, y \in \{[R], \ldots, M - 1 - [R]\} \tag{2.11}$$

LBP asli (Gambar 2.10) sangat mirip dengan $LBP_{8,1}$, dengan dua perbedaan. Pertama, lingkungan dalam definisi umum diindeks secara melingkar, sehingga lebih mudah untuk mendapatkan deskriptor tekstur rotasi. Kedua, piksel diagonal dalam bentuk 3 × 3 diinterpolasi dalam $LBP_{8,1}$.

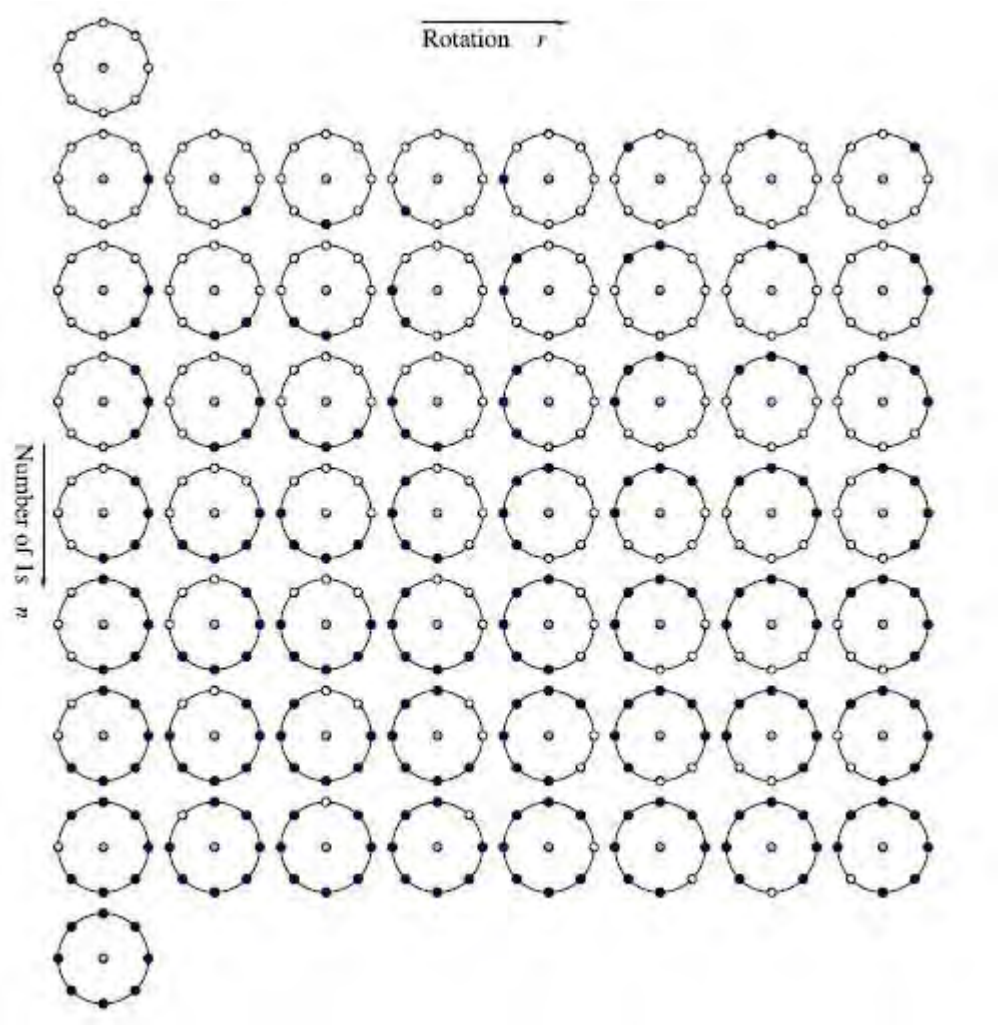
Perluasan dari operator LBP menggunakan pola yang disebut seragam (*uniform patterns*). Untuk ini, ukuran keseragaman pola digunakan: U (pola) adalah jumlah transisi *bitwise* dari 0 ke 1 atau sebaliknya ketika pola bit dianggap melingkar. Sebuah LBP disebut seragam jika ukuran keseragamannya paling banyak 2. Sebagai contoh, pola 00000000 (0 transisi), 01110000 (2 transisi) dan 11001111 (2 transisi) adalah seragam sedangkan pola 11001001 (4 transisi) dan 01010011 (6 transisi) bukan.



Gambar 2.11
Perbedaan Tekstur yang Terdeteksi oleh LBP
(Sumber : Pietikäinen, 2011:17)

Pola yang seragam memungkinkan untuk melihat metode LBP sebagai pendekatan pemersatu untuk model statistik dan struktural analisis tekstur tradisional

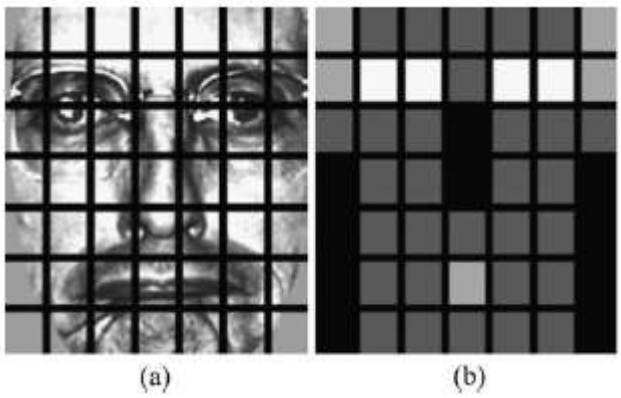
yang berbeda. Setiap piksel diberi label dengan kode tekstur primitif yang paling cocok dengan lingkungan setempat. Dengan demikian setiap kode LBP dapat dianggap sebagai *micro-texton*. Primitif lokal yang terdeteksi oleh LBP meliputi bintik-bintik, area datar, tepi, ujung tepi, kurva dan sebagainya. Beberapa contoh ditunjukkan pada Gambar. 2.11 dengan operator $LBP_{8,\ R}$. Dalam gambar, 1 direpresentasikan sebagai lingkaran hitam, dan 0 berwarna putih. Kombinasi pendekatan struktural dan statistik berasal dari fakta bahwa distribusi mikro-teks dapat dilihat sebagai aturan penempatan statistik. Oleh karena itu, distribusi LBP memiliki kedua sifat metode analisis struktural: primitif tekstur dan aturan penempatan. Di sisi lain, distribusi hanyalah statistik dari gambar yang disaring secara non-linier, jelas menjadikan metode ini statistik. Untuk alasan ini, distribusi LBP dapat berhasil digunakan dalam mengenali berbagai macam tekstur yang berbeda, dimana metode statistik dan struktural biasanya diterapkan secara terpisah.



Gambar 2.12
58 Perbedaan *Uniform Patterns* Dalam (8, *R*)
(Sumber : Pietikäinen, 2011:18)

Representasi wajah berbasis LBP diperkenalkan pada tahun 2004 oleh Ahonen dkk., pertama kali yang dinilai pada masalah pengenalan wajah. Sejak itu, metodologi ini telah menginspirasi banyak metode baru. Biasanya penggolongan *nearest neighbor* menggunakan LBP karena jumlah gambar pelatihan per subjek umumnya rendah, terkadang hanya satu. Gagasan peningkatan spasial histogram dapat dieksploitasi lebih lanjut ketika mendefinisikan ukuran jarak untuk pengklasifikasi. Properti asli dari metode deskripsi wajah yang diajukan adalah bahwa setiap elemen dalam histogram yang ditingkatkan, menyesuaikan dengan area kecil tertentu pada wajah.

Berdasarkan temuan psikofisik, yang menunjukkan bahwa beberapa fitur wajah (seperti mata) memainkan peran yang lebih penting dalam pengenalan wajah manusia daripada fitur lain, dapat diharapkan bahwa dalam metode ini beberapa daerah wajah berkontribusi lebih dari yang lain dalam hal perbedaan yang sangat pribadi. Dengan menggunakan asumsi ini, daerah dapat dipertimbangkan berdasarkan pentingnya informasi yang dikandungnya.



Gambar 2.13
(a) Contoh Gambar Wajah yang Dibagi Menjadi 7x7 Daerah
(b) Perbedaan Bobot yang Ditetapkan
(Sumber : Pietikäinen, 2011:160)

Gambar 2.14 menunjukkan contoh pembobotan daerah wajah yang berbeda. Dengan demikian, jarak *chi square* tertimbang dapat didefinisikan sebagai

$$\chi^2_\omega(X,\xi) = \sum_{j,i} \omega_j \frac{(x_{i,j} - \xi_{i,j})^2}{x_{i,j} - \xi_{i,j}}, \qquad (2.12)$$

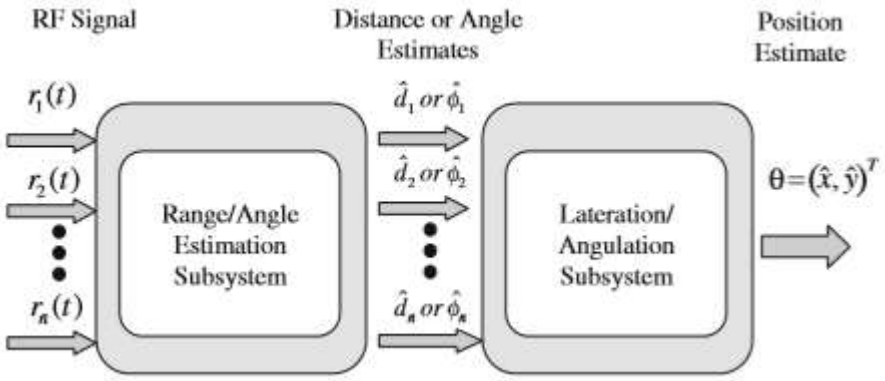
di mana x dan ξ adalah histogram tambahan yang dinormalisasi untuk dibandingkan, indeks i dan j merujuk ke *bin* ke-i dalam histogram yang sesuai dengan wilayah lokal ke-j dan ω_j adalah bobot untuk wilayah j.

2.4 Global Positioning System

Global Positioning System (GPS) merupakan sistem navigasi yang berbasis teknologi satelit. Teknik dasarnya meliputi pengukuran rentang antara penerima dan beberapa satelit yang diamati secara simultan, dan posisi satelit yang diperkirakan dan

disiarkan bersama dengan sinyal GPS kepada pengguna. Melalui beberapa posisi yang diketahui dan jarak yang diukur antar penerima dan satelit, maka posisi penerima dapat ditentukan. Perubahan posisi yang juga dapat ditentukan adalah kecepatan penerima. Aplikasi GPS yang paling penting adalah penentuan posisi dan navigasi (Xu, 2016:1).

Teknik geolokasi klasik (berbasis non-survei) bergantung pada hubungan geometris antara koordinat titik referensi (satelit dalam teknologi GPS) dan pengukuran rentang yang terkait. Biasanya, titik referensi adalah perangkat nirkabel dengan informasi lokasi yang diketahui (misalnya koordinat x dan y) baik yang sudah diatur atau yang diperoleh melalui GPS. (Gentile, 2013:17). Tingkat akurasi informasi lokasi dipengaruhi oleh tiga faktor utama: tingkat akurasi posisi referensi yang ditentukan, tingkat akurasi perkiraan rentang/sudut, dan konfigurasi geometris dari titik referensi dan posisi yang tidak diketahui. Teknik geolokasi non-survei menghitung perkiraan lokasi melalui dua



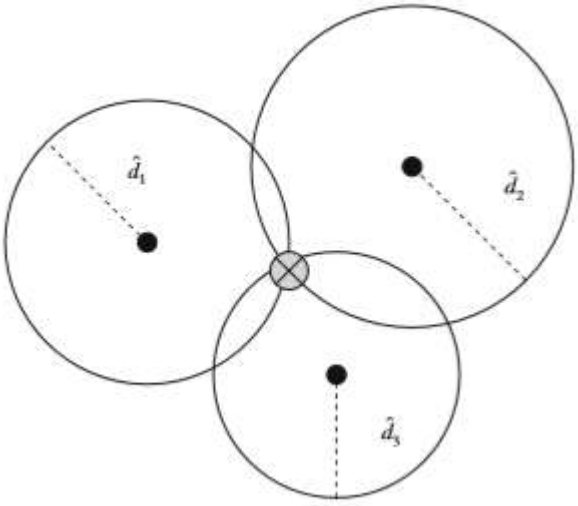
langkah : *range/angle estimation* dan *tri-lateration/angulation*.

Gambar 2.14
Perhitungan Perkiraan Lokasi
(Sumber : Gentile, 2013:18)

Terdapat beberapa teknik geolokasi yang umumnya digunakan, yaitu :

1. *TOA-Based Techniques*

   Metode ini membutuhkan setidaknya tiga titik referensi. Setelah pengukuran jarak terhadap titik-titik referensi tersedia, maka estimasi posisi dapat diperoleh. Pengukuran jarak dari titik referensi ke terminal seluler membentuk persamaan non-linier yang dapat diselesaikan untuk memperkirakan posisi.

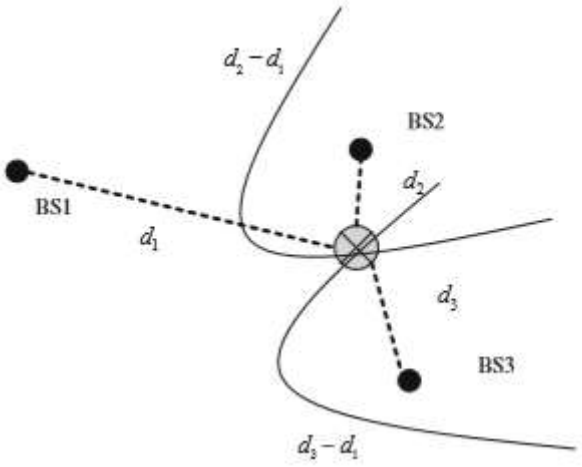


Gambar 2.15
TOA-Based Literation
(Sumber : Gentile, 2013:19)

Dalam Gambar 2.10, titik hitam merupakan stasiun dengan informasi posisi yang telah diketahui, sedangkan lingkaran merupakan perkiraan posisi perangkat seluler.

2. *TDOA Techniques*

Menurut Liu yang disertakan dalam buku "*Geolocation Techniques Principles and Applications*" teknik TDOA (Time Difference of Arrival) didasarkan pada gagasan bahwa posisi perangkat seluler dapat ditentukan dengan memeriksa perbedaan



waktu di mana sinyal tiba di beberapa titik referensi (Gentile, 2013:22).

Gambar 2.16
TDOA Localization
(Sumber : Gentile, 2013:23)

Setidaknya membutuhkan tiga stasiun (BS) untuk mencari posisi dua dimensi. Perbedaan waktu antara d2 – d1 dan d3 – d1 membentuk dua hiperboloid dimana persimpangan adalah posisi yang diperkirakan.

3. AOA-Based Techniques

Lokalisasi menggunakan *angle-of-arrival* lebih sederhana daripada teknik yang berbasis waktu karena hanya diperlukan dua pengukuran sudut untuk memperkirakan posisi dua dimensi.

4. Received Signal Strength Localization

Teknik ini hampir serupa dengan teknik berbasis TOA dalam hal jarak ke stasiun yang digunakan untuk memperkirakan posisi. Perbedaannya adala metode saat jarak diperkirakan.

2.5 Database MySQL

Dalam buku yang berjudul "*Database System A Pragmatic Approach*", sistem basis data didefinisikan sebagai :

"*A database system (DBS) is a computerized record keeping system with the overall purpose of maintaining information and making it available whenever required. The database typically stores related data in a computer system. A database management system (DBMS) is a set of programs that allow for the management of a database*" (Foster, 2016:3).

Berdasarkan definisi di atas, sistem basis data merupakan sistem pencatatan yang terkomputerisasi dengan tujuan memelihara informasi dan membuatnya tersedia kapanpun informasi tersebut dibutuhkan, biasanya menyimpan data yang terkait ke dalam sebuah sistem komputer. *Database Management System* merupakan satu set program yang disediakan untuk mengelola basis data.

Dalam buku lain yang berjudul "*Database Processing Fundamentals, Designn and Implementation Fourteenth Edition*", basis data didefinisikan sebagai :

"*The database is a collection of related tables and other structures. The database management system (DBMS) is a computer program used to create, process, and administer the database.*" (Kroenke, 2016, 43).

Berdasarkan definisi di atas, basis data merupakan sekumpulan struktur dan tabel yang saling terkait. *Database Management System* merupakan program komputer yang digunakan untuk membuat, mengeksekusi, dan mengelola basis data.

Dari beberapa definisi para ahli, maka dapat disimpulkan bahwa basis data merupakan struktur data yang saling terkait, dan disimpan ke dalam sistem komputer. Dan *Database Management System* merupakan set program yang digunakan untuk mengelola basis data. MySQL merupakan sistem manajemen basis data bersifat *open source*. Mencakup beberapa fitur yang lebih lanjut, alat pengelola dan dukungan teknis untuk mencapai tingkat skalabilitas, keamanan, keandalan, dan waktu operasional yang tinggi. Basis data SQL mendukung bermacam-macam tipe data : (Taylor, 2019:28)

1. *Numerics*

   Merupakan tipe data yang menyatakan nilai angka dengan tepat.

2. *Binary*

   Merupakan tipe data yang menyatakan nilai angka dan karakter yang dikonversi menjadi data biner. Tipe data *Binary Large Object* (BLOB) yang digunakan untuk string biner besar yang jumlahnya terlalu besar untuk tipe biner. Gambar grafis dan file musik merupakan contoh string biner besar.

3. *Strings*

   Merupakan tipe data yang menyatakan nilai angka dan karakter.

4. *Booleans*

   Merupakan tipe data yang menyatakan nilai *True*, *False* dan *Unknown*.

5. *Datetimes*

   Merupakan tipe data yang menyatakan nilai tanggal dan waktu.

6. *Intervals*

   Merupakan tipe data yang menyatakan jarak waktu antara dua nilai *datetime*.

7. *XML*

Merupakan singkatan dari *eXtensible Markup Language*, yang diartikan seperangkat aturan untuk menambahkan *markup* ke data. XML memungkinkan berbagi data antar *platform* yang berbeda.

2.6 UML

Dalam buku yang berjudul "*Software Engineering A Practitioner Approach Seventh Edition*" UML didefinisikan sebagai berikut

"*A standard language for writing software blueprints. UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system*"

(Pressman, 2010:841).

Dari definisi diatas, UML merupakan standar bahasa untuk menuliskan rancangan suatu perangkat lunak. UML dapat digunakan untuk menggambarkan, memperinci, membangun, dan mendokumentasikan sebuah sistem perangkat lunak.

Terdapat beberapa diagram UML yang digunakan dalam pemodelan perangkat lunak, antara lain :

1. *Use Case Diagram*

   Use case diagram menentukan fitur dan fungsionalitas perangkat lunak dari sudut pandang pengguna. *Use case* menjelaskan bagaimana interaksi pengguna dengan sistem dengan menetapkan langkah – langkah dengan suatu tujuan tertentu.

2. *Class Diagram*

   Class diagram menyajikan gambaran sistem secara statis atau berstruktur. Elemen utama dari *class diagram* adalah kotak, yang merupakan ikon untuk merepresentasikan *class* dan *interface*. Kotak tersebut dibagi menjadi tiga bagian secara vertikal, bagian atas berisi nama *class*, bagian tengah berisi daftar atribut yang digunakan sebuah *class*, sedangkan bagian bawah berisi method yang digunakan sebuah *class*.

3. *Activity Diagram*

   Activity diagram menggambarkan tingkah laku yang dinamis sebuah atau sebagian dari sistem melalui arus kontrol antar tindakan yang dilakukan oleh sistem. Komponen utama dari class diagram adalah *action node*, yang berisi tugas yang

dilakukan oleh sebuah sistem perangkat lunak. Arah panah dari satu *action node* ke *action node* lainnya menunjukkan arus kontrol. Jika terdapat arah panah diantara dua *action node*, berarti setelah *action* pertama selesai dilakukan, *action* kedua dimulai. Sebuah titik hitam *solid* menunjukkan titik awal *activity*, sedangkan titik hitam yang dikelilingi lingkaran hitam nenunjukkan akhir dari *activity*.

2.7 Visual Studio

Visual Studio merupakan *integrated development environment* (IDE) dari Microsoft. Digunakan untuk mengembangkan aplikasi komputer, *mobile, gaming*, maupun *web*. Mendukung beberapa bahasa pemrograman seperti C++, .NET, Javascript, Python, dan masih banyak lagi. (Sumber : https://visualstudio.microsoft.com/vs/)

Visual Studio memiliki beberapa fitur yang mendukung penulisan program dengan lebih baik. Terdapat fitur yang dapat menandai baris, sehingga mempermudah dalam navigasi. Dapat mengatur rangkaian pengujian dengan mudah sehingga dapat menganalisis berapa banyak kode yang diuji dan melihat hasilnya secara langsung. Juga terdapat fitur Collaborate, sehingga penulisan program dapat dilakukan secara bersamaan apapun bahasa dan platform yang dipakai.

BAB III

ANALISIS DAN PERANCANGAN

3.1 Gambaran Umum Perangkat Lunak

Secara garis besar, perangkat lunak yang dirancang adalah sistem pencatatan kehadiran dengan menggunakan teknik pengenalan wajah dan pengambilan lokasi. Dengan memindai wajah seseorang, maka kehadiran akan dicatat berdasarkan posisi dimana absensi dilakukan. Jika absensi menggunakan pemindaian wajah gagal, maka tersedia absensi manual untuk dilakukan.

Berikut proses bagaimana perangkat lunak sistem absensi bekerja :

1. Proses Kelola Data Pegawai:
    a. Administrator akan mengisikan data pegawai ke dalam perangkat lunak.
    b. Pengguna diminta untuk registrasi wajahnya ke dalam perangkat lunak dengan cara memindai wajah melalui kamera yang tersedia.
    c. Setelah proses registrasi selesai, sistem akan menyimpan data pengguna ke dalam basis data.
2. Proses Kelola Data *Outlet*:
    a. Administrator akan mengisikan data *outlet* ke dalam perangkat lunak.
    b. Setelah proses registrasi selesai, sistem akan menyimpan data *outlet* ke dalam basis data.
3. Proses Absensi :
    a. Pada proses ini, terdapat *panel* yang akan merekam wajah pengguna.
    b. Proses pemindaian wajah akan selalu aktif, sehingga saat wajah terdeteksi, sistem akan menentukan, apakah wajah yang dipindai dikenali atau tidak.
    c. Jika wajah dikenal, maka akan tampil pemberitahuan bahwa absensi berhasil dilakukan. Kemudian sistem akan mengambil posisi dimana pengguna melakukan absensi.
    d. Jika wajah tidak dikenal, maka akan tampil pemberitahuan bahwa sistem tidak mengenali wajah tersebut, kemudian pengguna disarankan untuk

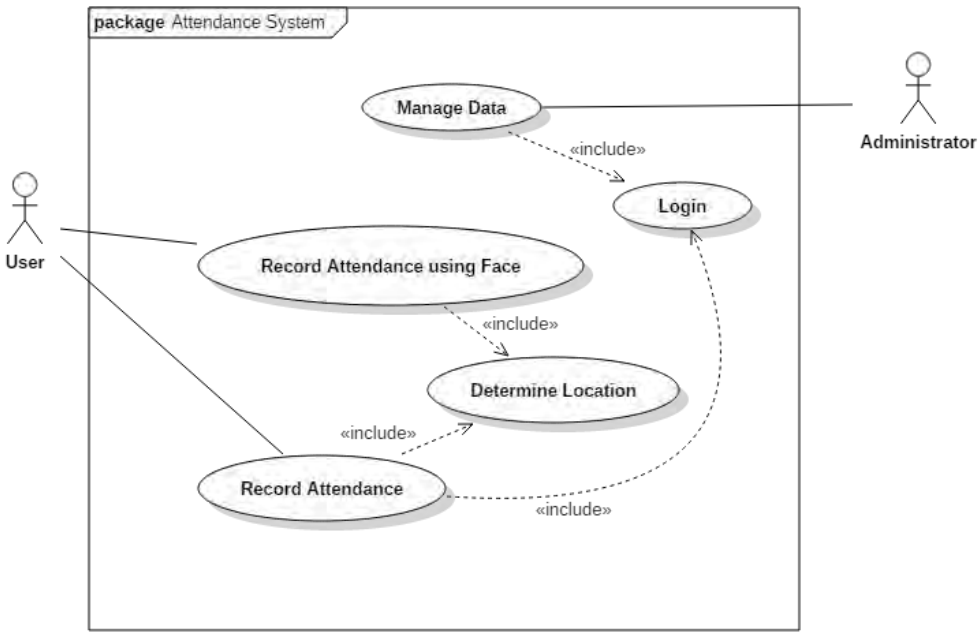
menggunakan absensi secara manual, yaitu dengan memasukkan *id* dan *password* ke dalam *textbox* yang telah tersedia, kemudian menekan tombol Absen.

3.2 Spesifikasi Kebutuhan Perangkat Lunak

Perangkat lunak yang dirancang memiliki fitur dan spesifikasi sebagai berikut :

1. Pengguna dapat menyimpan data diri dan wajah ke dalam basis data.
2. Sistem dapat mengambil posisi dimana pengguna melakukan absensi.

3.3 Use Case Diagram



Gambar 3.1
Use Case Diagram

3.4 Skenario Diagram

Pada sub-bab ini menjelaskan tentang diagram skenario dalam sistem absensi yang akan dirancang. Diagram ini menjelaskan kemungkinan skenario yang akan dilalui dalam sistem. Terdapat lima skenario diagram, yaitu untuk proses *Login*, *Manage Data*, *Record Attendance Using Face, Record Attendance* dan *Determine Location.*

3.4.1 Skenario *Use Case Login*

Aktor : Administrator

Pre – Condition : Sistem belum menampilkan *form* Menu

Post – Condition : Sistem menampilkan *form* Menu

Deskripsi : Skenario Administrator melakukan login untuk mengelola data

Tabel 3.1
Skenario *Use Case Login*

Aksi (Aktor)	Reaksi (Sistem)
1. Isi *username* dan *password*	
2. Klik tombol Login	
	3. Verifikasi *username* dan *password*
	4. *Username* dan *password* valid
	5. Tampilkan form Menu

Alternatif 1 : *Username* atau *password* tidak valid

Aktor : Administrator

Pre – Condition : Sistem belum menampilkan *form* Menu

Post – Condition : Sistem menampilkan *form* Menu

Deskripsi : Skenario Administrator melakukan login untuk mengelola data.

Tabel 3.2
Skenario *Use Case Login* Alternatif 1

Aksi (Aktor)	Reaksi (Sistem)
1. Isi *username* dan *password*	
2. Klik tombol Login	
	3. Verifikasi *username* dan *password*
	4. *Username* dan *password* tidak valid
	5. Tampilkan pesan bahwa *username* atau *password* salah

3.4.2 Skenario *Use Case Manage Data*

Aktor : Administrator

Pre – Condition : Sistem belum memiliki data pengguna

Post – Condition : Sistem menyimpan data pengguna

Deskripsi : Skenario untuk menyimpan data pengguna ke dalam sistem

Tabel 3.3
Skenario *Use Case Manage Data*

Aksi (Aktor)	Reaksi (Sistem)
	1. Tampilkan form Menu
2. Klik menu Data, klik sub-menu User	
	3. Tampilkan form User
4. Klik tombol bertanda "+"	
	5. Membuat dan menampilkan id user
6. Isi data pengguna	
7. Isi data wajah pengguna	
8. Klik tombol *Submit*	
	9. Sistem menyimpan data pengguna
	10. Sistem menyimpan wajah pengguna
	11. Tampilkan pesan bahwa data telah disimpan

Alternatif 1 : Administrator mengubah data pengguna

Aktor : Administrator

Pre – Condition : Sistem sudah memiliki data pengguna

Post – Condition : Sistem menyimpan perubahan data pengguna

Deskripsi : Skenario penyimpanan data pengguna yang diubah.

Tabel 3.4
Skenario *Use Case Manage Data* Alternatif 1

Aksi (Aktor)	Reaksi (Sistem)
1. Isi *username* dan *password*	
2. Klik tombol Login	
	3. Tampilkan form Admin
4. Klik menu Data, klik sub-menu User	
	5. Tampilkan form User
6. Klik salah satu data pengguna	
	7. Tampilkan data pengguna
8. Ubah data pengguna	
9. Klik tombol *Submit*	

	10. Sistem menyimpan perubahan data pengguna
	11. Tampilkan pesan bahwa perubahan data telah disimpan

Alternatif 2 : Administrator menghapus data pengguna

Aktor : Administrator

Pre – Condition : Sistem sudah memiliki data pengguna

Post – Condition : Sistem menghapus data pengguna

Deskripsi : Skenario menghapus data pengguna

Tabel 3.5
Skenario *Use Case Register Data* Alternatif 2

Aksi (Aktor)	Reaksi (Sistem)
1. Isi *username* dan *password*	
2. Klik tombol Login	
	3. Tampilkan form Admin
4. Klik menu Data, klik sub-menu User	
	5. Tampilkan form User
6. Klik salah satu data pengguna	
7. Klik tombol bertanda “-“	
	8. Tampilkan form konfirmasi penghapusan data
9. Klik tombol Yes	
	10. Sistem menghapus data pengguna
	11. Tampilkan pesan bahwa data pengguna yang dipilih sudah dihapus

3.4.3 Skenario *Use Case Record Attendance Using Face*

Aktor : Sistem

Pre – Condition : Sistem memiliki data pengguna

Post – Condition : Sistem menyimpan data absensi pengguna

Deskripsi : Skenario untuk menyimpan data absensi ke dalam sistem

Tabel 3.6
Skenario *Use Case Record Attendance Using Face*

Aksi (Aktor)	Reaksi (Sistem)
	1. Tampilkan form *Login*

	2. Aktifkan kamera
	3. Deteksi wajah di kamera
4. Wajah terdeteksi	
	5. Ambil data wajah yang terdeteksi
	6. Lakukan verifikasi wajah
7. Wajah dikenal	
	8. Ambil lokasi absensi
	9. Simpan data absensi
	10. Tampilkan pesan bahwa absensi berhasil dilakukan

Alternatif 1 : Wajah pengguna tidak dikenal

Aktor : Pengguna

Pre – Condition : Sistem memiliki data pengguna

Post – Condition : Sistem tidak menyimpan data absensi pengguna

Deskripsi : Skenario sistem gagal menyimpan data absensi

Tabel 3.7
Skenario *Use Case Record Attendance Using Face* Alternatif 1

Aksi (Aktor)	**Reaksi (Sistem)**
	1. Tampilkan form *Login*
	2. Aktifkan kamera
	3. Deteksi wajah di kamera
4. Wajah terdeteksi	
	5. Ambil data wajah yang terdeteksi
	6. Lakukan verifikasi wajah
7. Wajah tidak dikenal	
	8. Tampilkan pesan bahwa wajah tidak dikenal dan menyarankan untuk melakukan absensi secara manual atau menghubungi admin

3.4.4 Skenario *Use Case Record Attendance*

Aktor : Pengguna

Pre – Condition : Sistem memiliki data pengguna

Post – Condition : Sistem menyimpan data absensi pengguna

Deskripsi : Skenario untuk menyimpan data absensi ke dalam sistem

Tabel 3.8
Skenario *Use Case Record Attendance*

Aksi (Aktor)	**Reaksi (Sistem)**
	1. Tampilkan form *Login*
2. Isi *id* dan *password*	
3. Klik tombol Absen	

	4. Verifikasi *id* dan *password*
5. *Id* dan *password* valid	
	6. Ambil lokasi absensi
	7. Simpan data absensi
	8. Tampilkan pesan bahwa absensi berhasil dilakukan

Alternatif 1 : *Id* atau *password* tidak valid

Aktor : Pengguna

Pre – Condition : Sistem memiliki data pengguna

Post – Condition : Sistem tidak menyimpan data absensi pengguna

Deskripsi : Skenario sistem gagal menyimpan data absensi

Tabel 3.9
Skenario *Use Case Record Attendance* Alternatif 1

Aksi (Aktor)	Reaksi (Sistem)
	1. Tampilkan form *Login*
2. Isi *id* dan *password*	
3. Klik tombol Absen	
	4. Verifikasi *id* dan *password*
5. *Id* dan *password* tidak valid	
	6. Tampilkan pesan bahwa *id* atau *password* yang diisi salah

3.4.5 Skenario *Use Case Determine Location*

Aktor : Sistem

Pre – Condition : Sistem belum mengetahui posisi pengguna

Post – Condition : Sistem mendapat posisi pengguna

Deskripsi : Skenario untuk mendapatkan lokasi pengguna sehingga sistem dapat menentukan posisi pengguna saat absensi dilakukan

Tabel 3.10
Skenario *Use Case Determine Location*

Aksi (Aktor)	Reaksi (Sistem)
	1. *Request* koordinat posisi
	2. Sistem menyimpan koordinat

	3. Verifikasi koordinat
	4. Koordinat valid
	5. Menentukan posisi pengguna
	6. Tampilkan posisi pengguna

Alternatif 1 : Koordinat tidak valid

Aktor : Sistem

Pre – Condition : Sistem belum mengetahui posisi pengguna

Post – Condition : Sistem tidak mendapat posisi pengguna

Deskripsi : Skenario sistem gagal untuk mendapatkan lokasi pengguna

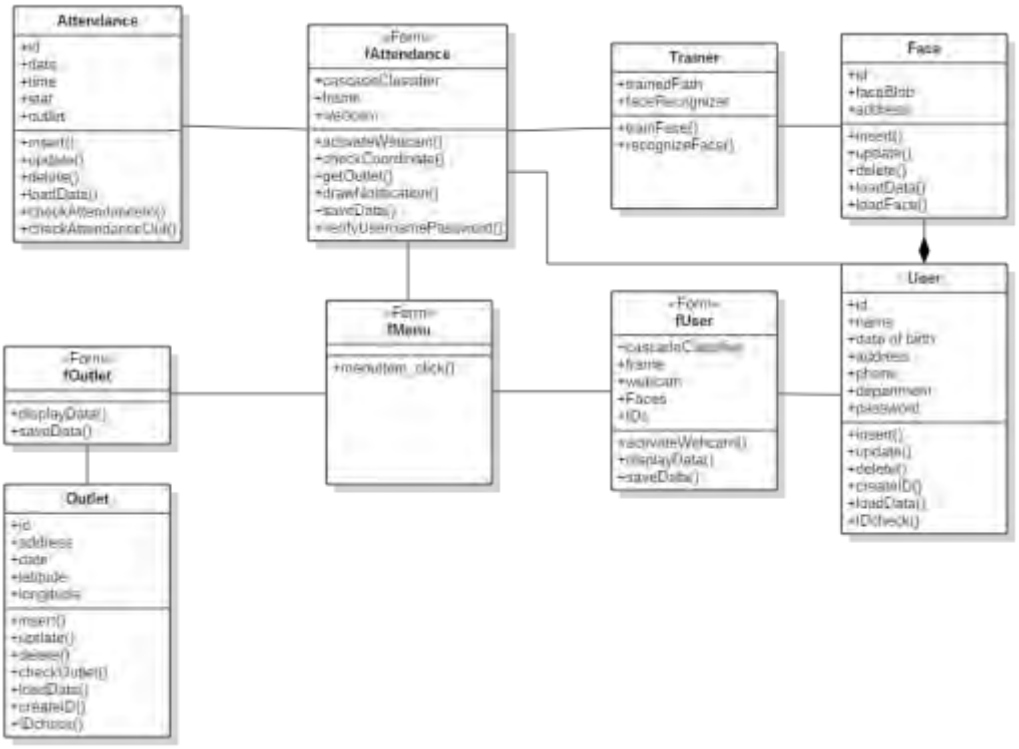
Tabel 3.11
Skenario *Use Case Determine Location* Alternatif 1

Aksi (Aktor)	Reaksi (Sistem)
	1. *Request* koordinat posisi
	2. Sistem menyimpan koordinat
	3. Verifikasi koordinat
	4. Koordinat tidak valid
	5. Tampilkan pesan bahwa koordinat berada di luar jangkauan

3.5 Class Diagram

Pada sub-bab ini menjelaskan mengenai *class diagram* pada sistem absensi yang akan dirancang. *Class* utama dalam sistem ini adalah *class* Attendance. Juga terdapat *class* Face, dimana setiap gambar (wajah) yang diambil, akan diproses dalam *class* tersebut. Kemudian *class* Trainer yang berfungsi untuk melatih sistem dalam mengenali wajah, dari gambar yang telah diproses. *Class* User digunakan untuk proses penyimpanan data pengguna. *Class* Face memiliki relasi komposisi terhadap *class* User, dikarenakan jika *class* Face tidak memiliki data, maka data pengguna tidak dapat tersimpan. Metode LBP diterapkan pada *class* Attendance, pada saat metode *verifyFace*. Sistem akan melakukan pengenalan wajah menggunakan metode tersebut, jika wajah dikenali, maka sistem akan menentukan posisi (melalui titik koordinat) dimana absensi dilakukan, sehingga akan muncul pemberitahuan bahwa absensi berhasil dilakukan. Proses pengenalan wajah dan pengambilan posisi diproses dalam *class Attendance.*

Class Outlet memproses titik koordinat yang telah diambil, sehingga sistem dapat menentukan dimana *outlet* tempat pengguna melakukan absensi.



Gambar 3.2
Class Diagram

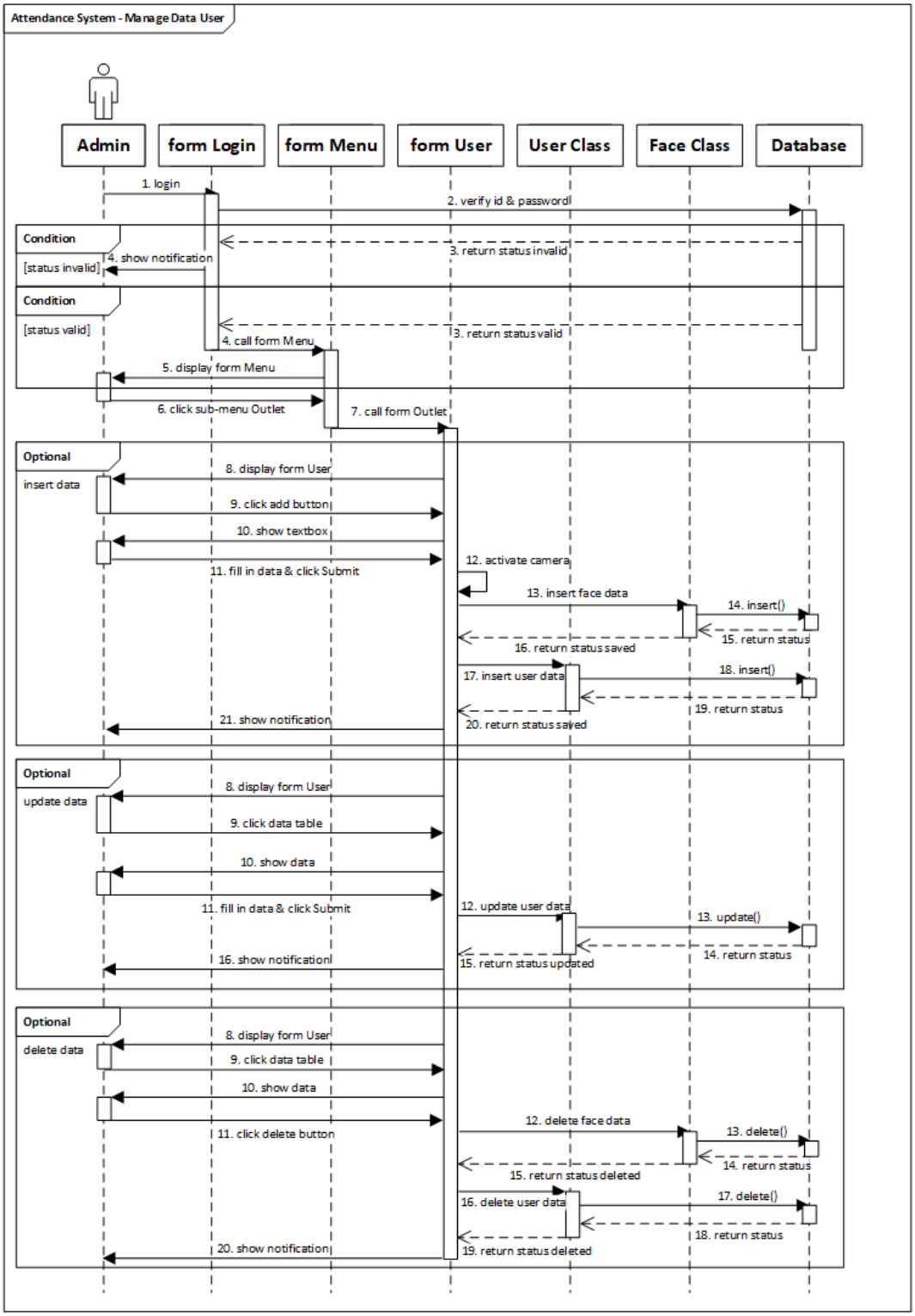
3.6 Sequence Diagram

Pada sub-bab ini menjelaskan bagaimana *sequence diagram* dalam sistem absensi yang akan dirancang. Dalam sistem ini, terdapat lima *sequence diagram*, *Manage Data Outlet*, *Manage Data User*, *Record Attendance Using Face*, *Record Attendance* dan *Determine Location*.

3.6.1 Sequence Diagram Manage Data User

Gambar 3.3 merupakan *sequence diagram* dalam proses manage data. Untuk dapat mengelola data, Admin diharuskan untuk melakukan login, demi keamanan data. Setelah login, maka *form* Menu akan muncul. Untuk melihat data User, maka Admin harus memilih dan menekan tombol Data User, maka *form* User akan muncul. Dalam *form Outlet* terdapat tabel yang berisi data yang sudah pernah disimpan. Untuk menambahkan data User, terdapat tombol "+" yang akan menampilkan *textbox* yang

harus diisi, kemudian setelah menekan tombol Submit, maka data yang terisi akan disimpan dalam *class* User dan *class* Face, yang kemudian akan disimpan dalam basis data. Untuk mengubah data, dapat menekan salah satu data yang ada di tabel, kemudian data tersebut akan muncul dalam *textbox* yang kemudian dapat diubah dan disimpan, setelah menekan tombol Submit, maka data akan disimpan dalam *class* User, kemudian akan disimpan ke dalam basis data. Dalam aplikasi ini hanya data pengguna yang dapat diubah, wajah pengguna tidak dapat diubah. Untuk menghapus data, dapat menekan salah satu data yang ada di tabel, kemudian menekan tombol “-”, , kemudian akan muncul pertanyaan konfirmasi, jika setuju maka data yang dipilih akan terhapus.

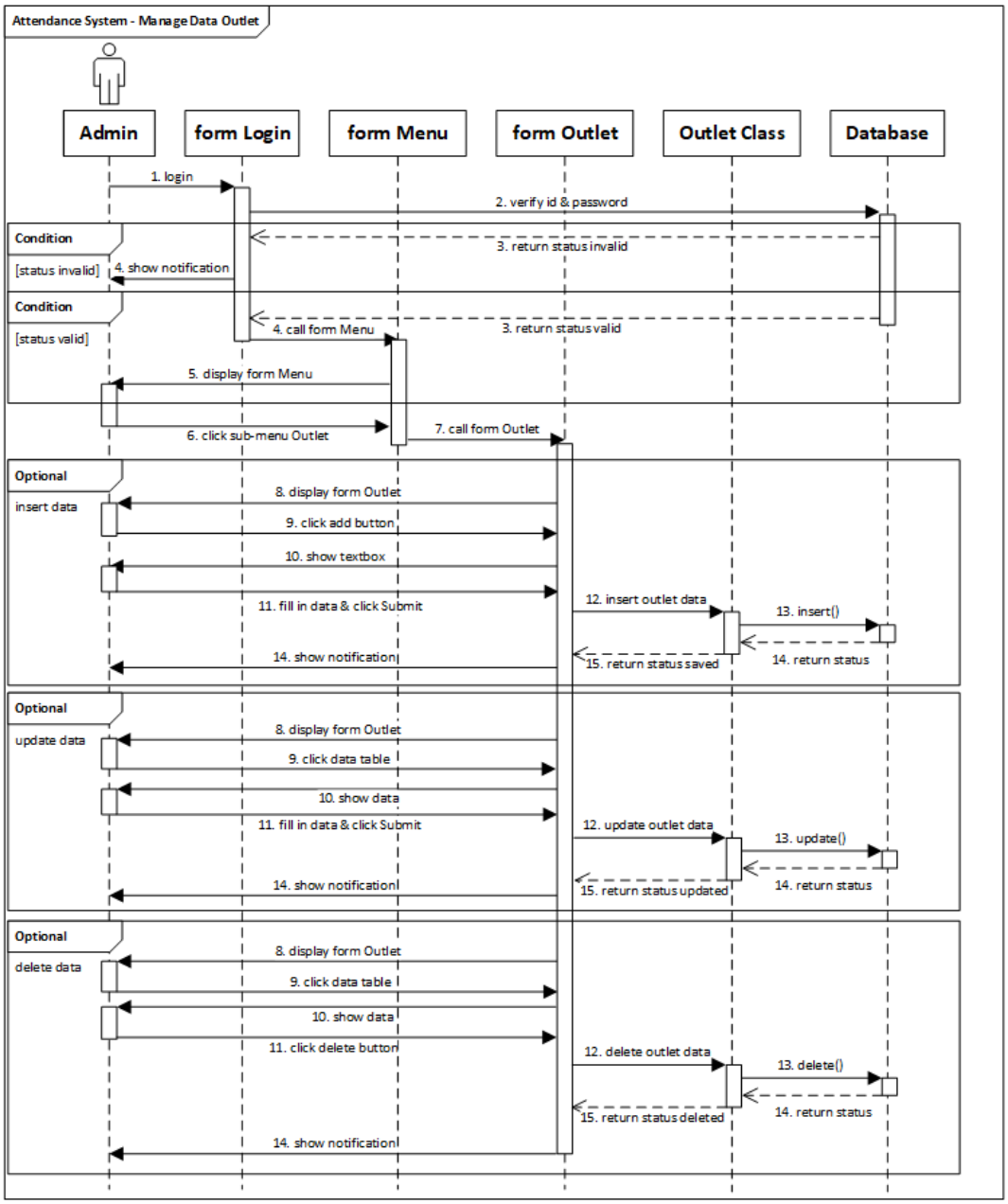


Gambar 3.3
Sequence Diagram Manage Data User

3.6.2 Sequence Diagram Manage Data Outlet

Gambar 3.4 merupakan *sequence diagram* dalam proses manage data Outlet. Prosesnya serupa dengan Manage Data User, melakukan login dan menekan tombol

Data Outlet. Proses penyimpanan dan perubahan data juga memiliki proses yang serupa dengan Manage Data User.

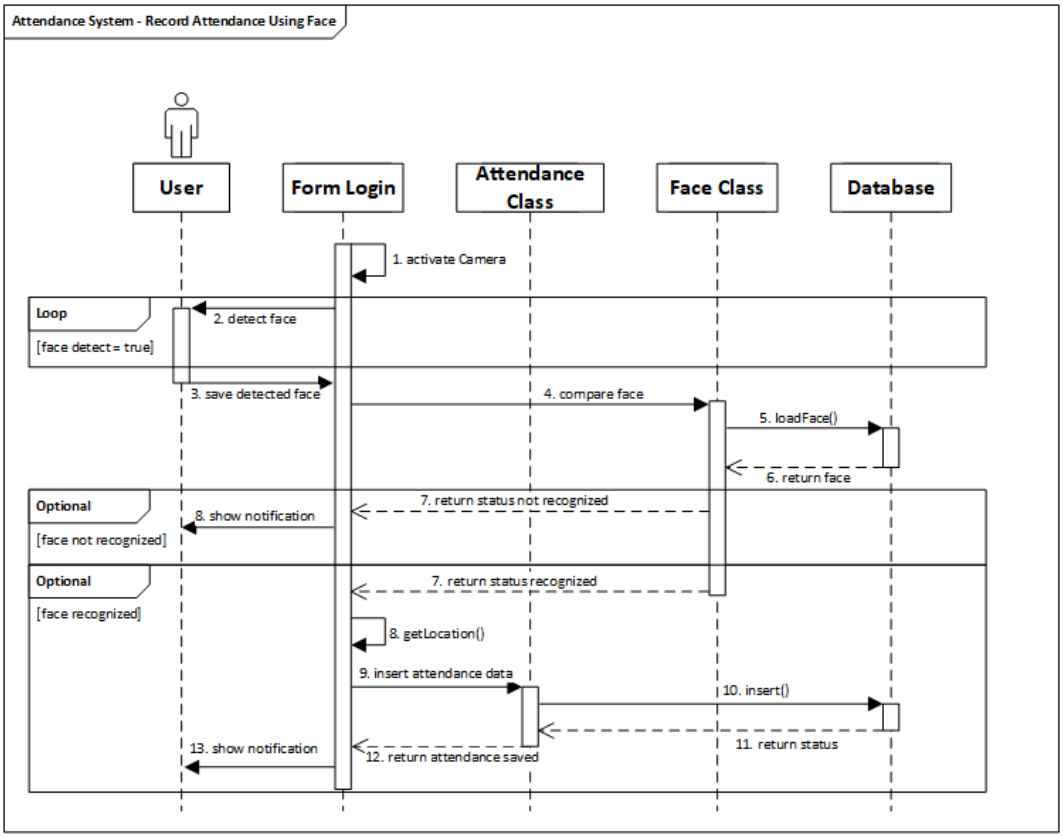


Gambar 3.4
Sequence Diagram Manage Data Outlet

3.6.3 Sequence Diagram Record Attendance Using Face

Gambar 3.5 merupakan *sequence diagram* dalam proses absensi berbasis pengenalan wajah. *Form* Login akan muncul, dan mengaktifkan kamera secara otomatis. Proses *scan* akan selalu dilakukan sampai wajah terdeteksi. Setelah wajah terdeteksi, maka gambar wajah tersebut akan diambil, kemudian sistem akan membandingkan

wajah yang diambil dengan data wajah yang ada dalam basis data. Jika wajah dikenali, maka sistem akan mengambil koordinat tempat absensi dilakukan, setelah itu menentukan dimana posisi pengguna, kemudian data absensi akan disimpan ke dalam basis data. Jika wajah tidak dikenali, maka pengguna disarankan untuk menggunakan metode *manual login* atau menghubungi Administrator.

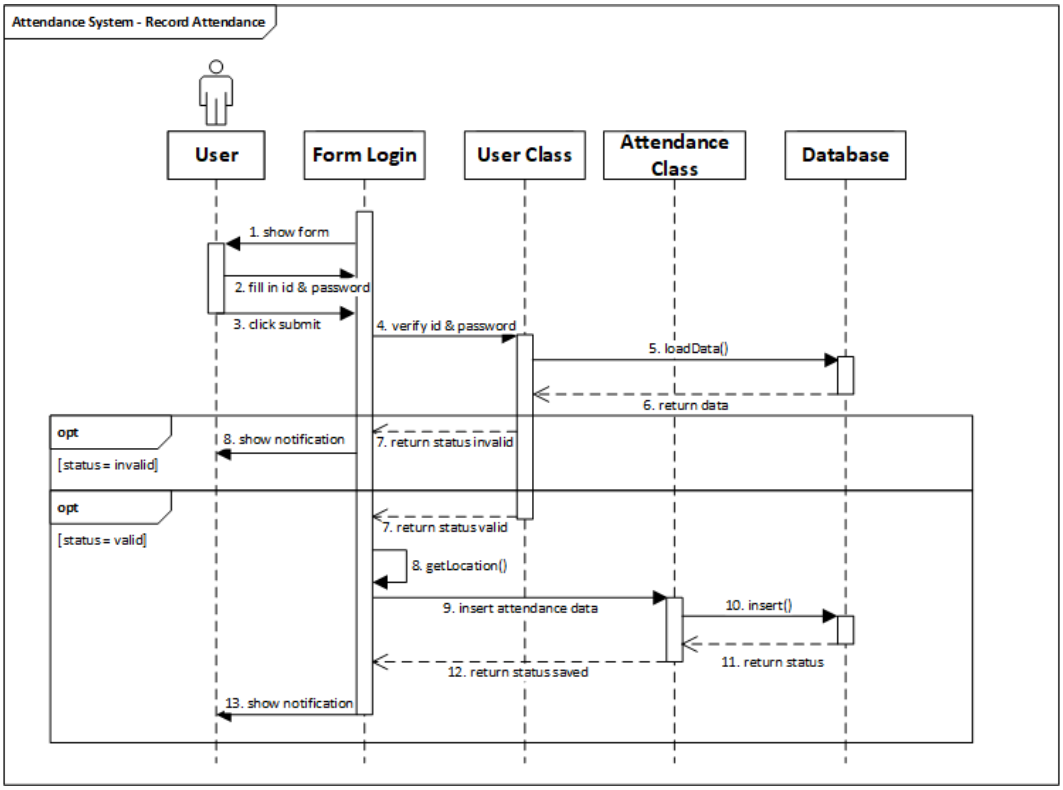


Gambar 3.5
Sequence Diagram Record Attendance Using Face

3.6.4 Sequence Diagram Record Attendance

Gambar 3.6 merupakan *sequence diagram* dalam proses absensi yang tidak berbasis pengenalan wajah. Metode ini merupakan cadangan jika terjadi kerusakan pada kamera atau terjadi kesalah pada absensi berbasis pengenalan wajah. Pengguna diminta untuk mengisi *id* dan *password* yang sudah ditentukan sebelumnya. Setelah menekan tombol Submit, maka sistem akan melakukan verifikasi. Jika *id* dan *password* valid, maka sistem akan mengambil koordinat tempat absensi dilakukan, setelah itu menentukan dimana posisi pengguna, kemudian data absensi akan disimpan ke dalam basis data. Jika *id* dan *password* tidak valid, absensi tidak bisa dilakukan dan pengguna akan

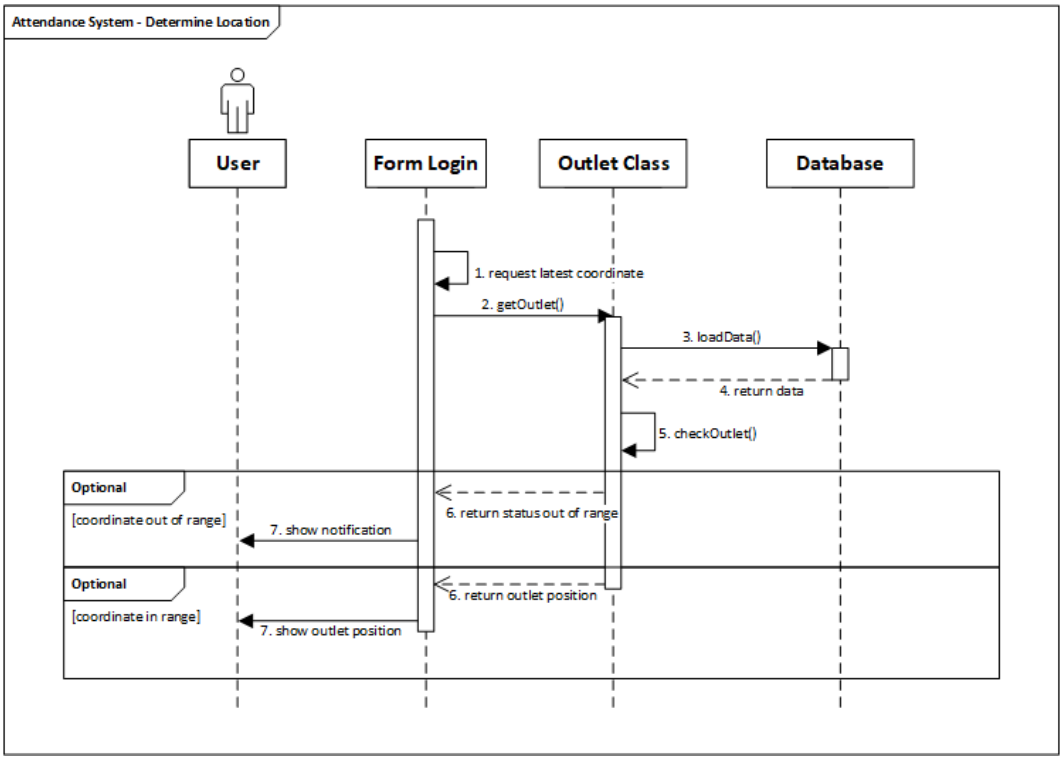
diminta untuk memeriksa apakah benar *id* dan *password* yang diketik atau diminta untuk menghubungi Administrator.



Gambar 3.6
Sequence Diagram Record Attendance

3.6.5 Sequence Diagram Determine Location

Gambar 3.7 merupakan *sequence diagram* dalam proses penentuan posisi saat absensi dilakukan. Saat pengguna melakukan absensi, maka sistem akan meminta Windows API untuk mengambil koordinat saat absensi dilakukan. Kemudian koordinat tersebut akan dibandingkan dengan koordinat outlet yang ada di basis data. Jika koordinat tersebut sesuai atau berada dalam cakupan wilayah salah satu *outlet*, maka sistem akan mengambil data *outlet* tersebut sebagai posisi dimana pengguna melakukan absensi. Jika koordinat berada diluar jangkauan, maka akan muncul pesan bahwa koordinat tidak sesuai dengan *outlet* yang ada.



Gambar 3.7
Sequence Diagram Determine Location

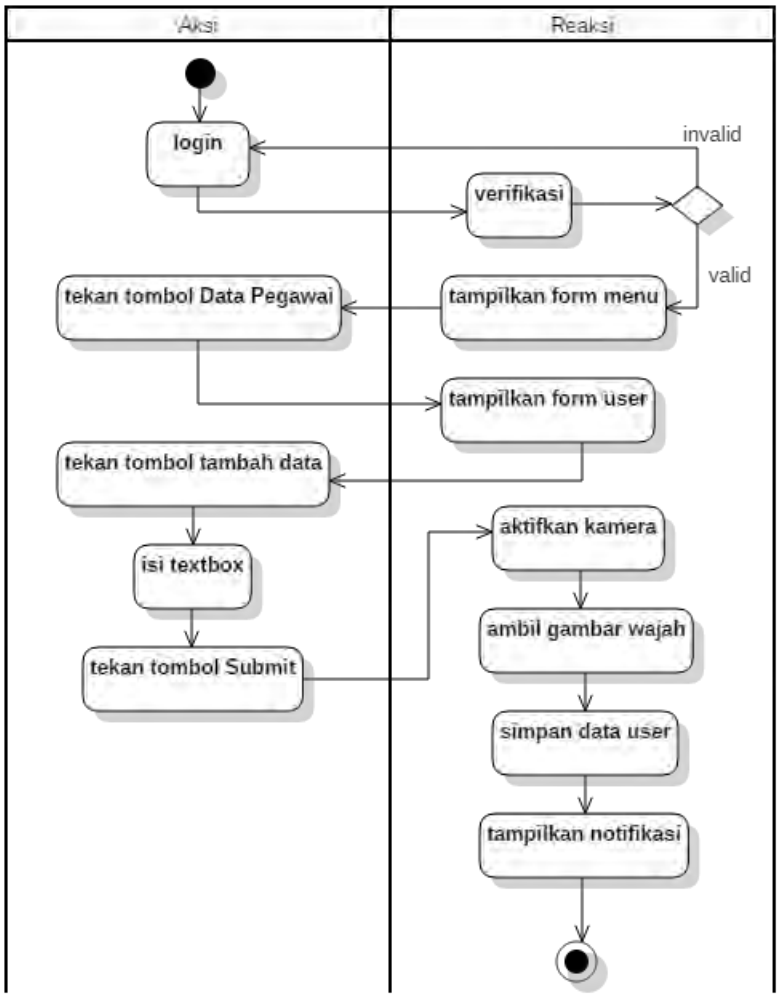
3.7 Activity Diagram

Pada sub-bab ini menjelaskan tentang *activity diagram* untuk sistem absensi yang akan dirancang. Dalam sistem ini, terdapat empat *activity diagram*, yaitu untuk proses *Manage Data*, *Scan Face Login, Manual Login* dan *Determine Location*.

3.7.1 Activity Diagram Manage Data

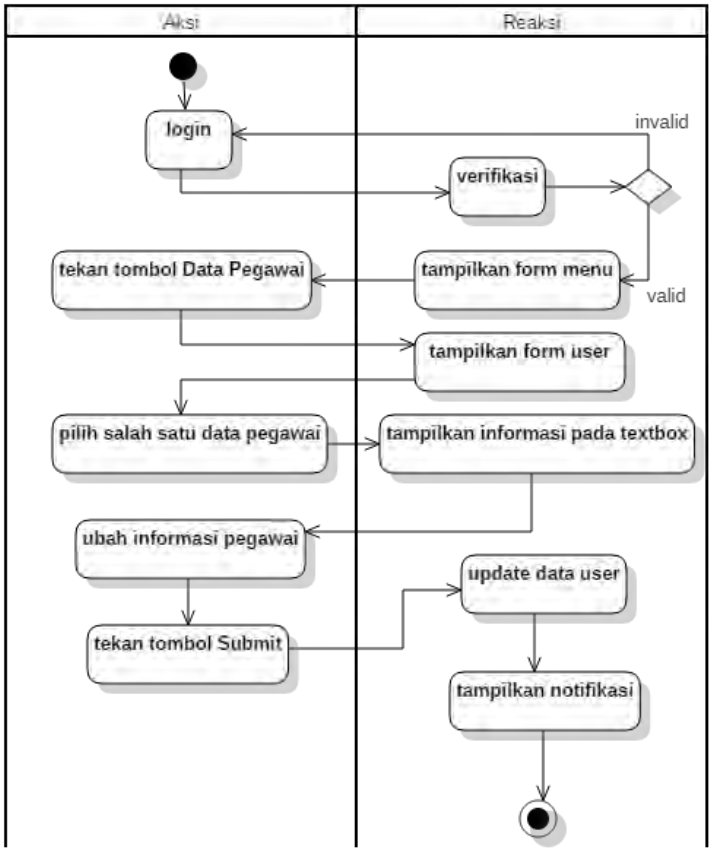
Terdapat dua data yang dapat dikelola, data User dan Outlet. Dalam kedua data tersebut terdapat proses registrasi, pengubahan dan dan penghapusan data. Gambar 3.8 merupakan proses registrasi untuk data *user*. Setelah melakukan login, sistem akan melakukan verifikasi. Setelah *valid*, sistem akan manampilkan form Menu, kemudian Administrator menekan tombol Data Pegawai, maka *form User* akan tampil. Untuk registrasi data pengguna, tekan tombol bertanda “+”, kemudian *textbox* akan aktif untuk mengisi data pengguna. Setelah terisi, tekan tombol Submit, maka kamera akan aktif

untuk mengambil gambar wajah pengguna yang didaftarkan, setelah itu data akan disimpan kemudian akan muncul pesan bahwa data telah tersimpan.



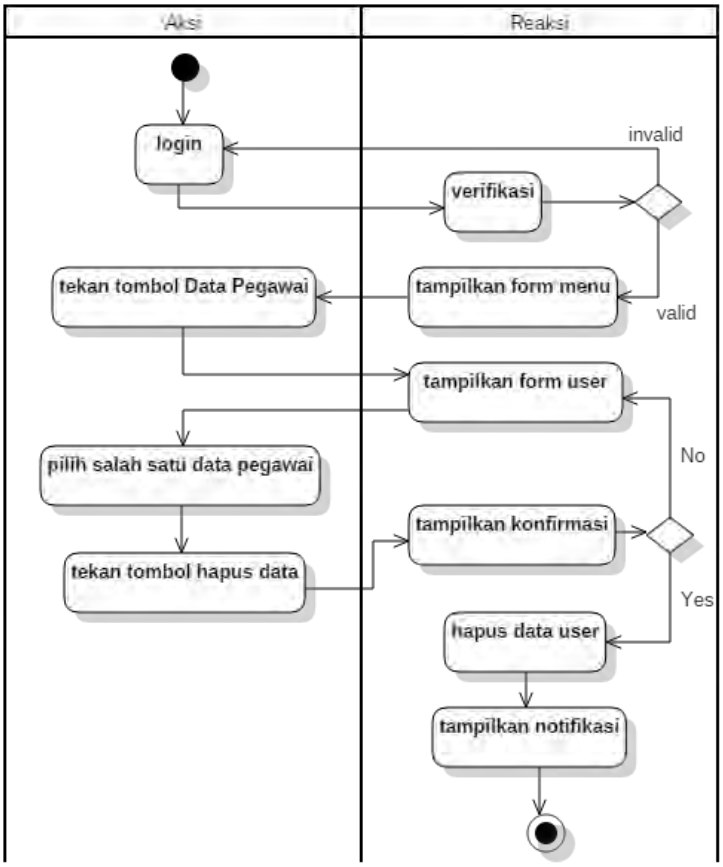
Gambar 3.8
Activity Diagram Manage Data User - Register

Kemudian terdapat proses pengubahan data. Gambar 3.9 merupakan *Activity Diagram* proses pengubahan data untuk data pengguna. Untuk mengubah data, proses awal sama dengan proses registrasi data, dari proses *login* sampai dengan munculnya *form User*. Kemudian untuk mengubah data, pilih salah satu data pengguna yang muncul dalam tabel, kemudian informasi mengenai pengguna tersebut akan muncul dalam *textbox*. Setelah mengubah data dalam *textbox*, tekan tombol Submit, maka sistem akan memperbarui basis data, lalu sistem akan menampilkan pesan bahwa data yang diubah telah disimpan.



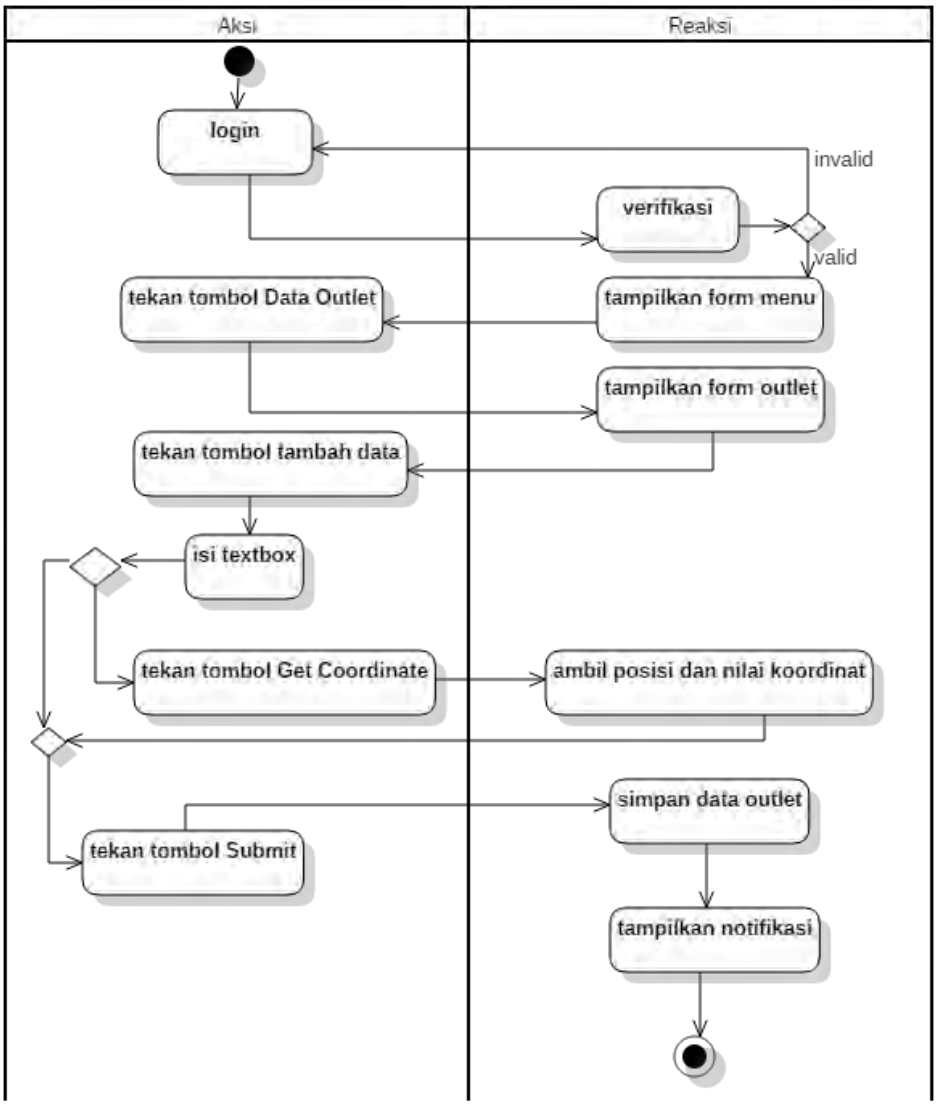
Gambar 3.9
Activity Diagram Manage Data User - Update

Lalu terdapat proses penghapusan data. Gambar 3.10 merupakan *Activity Diagram* proses penghapusan data untuk data pengguna. Untuk mengubah data, proses awal sama dengan proses registrasi data, dari proses *login* sampai dengan munculnya *form User*. Kemudian untuk menghapus data, pilih salah satu data pengguna yang akan dihapus, setelah itu tekan tombol bertanda "-" untuk menghapus data. Sistem akan memunculkan pertanyaan konfirmasi, jika menekan tombol "Ya", maka data pengguna yang terpilih akan dihapus dan pesan akan muncul bahwa data telah dihapus.



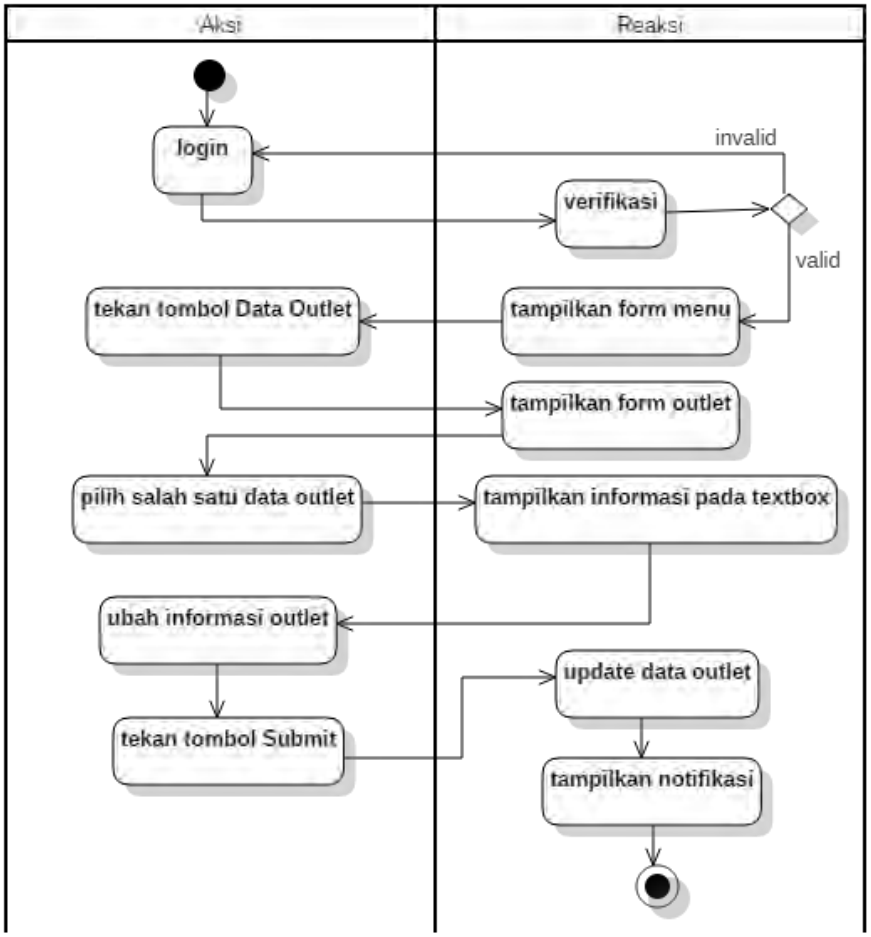
Gambar 3.10
Activity Diagram Manage Data User - Delete

Lalu terdapat proses Manage Data untuk data *outlet*. Gambar 3.11 merupakan *Activity Diagram* proses registrasi untuk data *outlet*. Untuk registrasi data outlet, Administrator perlu melakukan login, kemudia sistem akan melakukan verifikasi. Setelah *valid*, *form* menu akan muncul. Setelah itu tekan tombol Data Outlet, maka *form* Outlet akan muncul. Untuk menambahkan data outlet, tekan tombol bertanda “+”, kemudian *textbox* akan aktif untuk diisikan data. Terdapat tombol Get Coordinate jika ingin mengambil data kordinat secara langsung. Setelah data diisi, kemudian tekan tombol Submit, maka sistem akan menyimpan data kemudian pesan akan muncul bahwa data *outlet* telah disimpan.



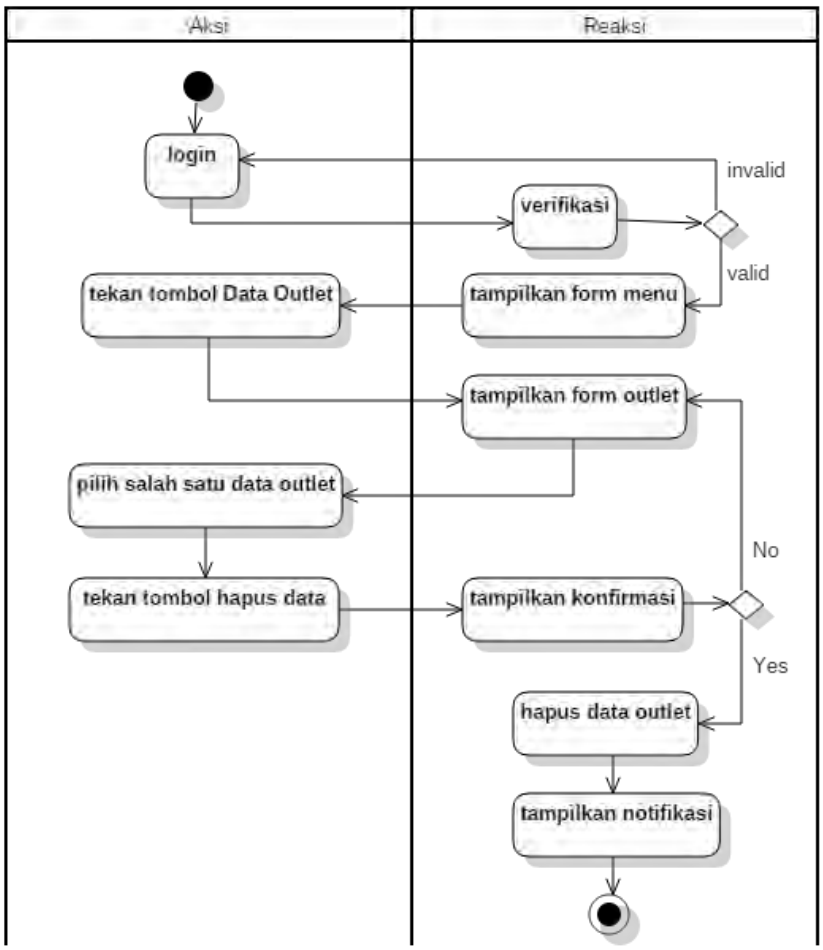
Gambar 3.11
Activity Diagram Manage Data Outlet - Register

Lalu terdapat proses proses pengubahan data. Gambar 3.12 merupakan *Activity Diagram* proses pengubahan data untuk data *outlet*. Sama seperti proses sebelumnya, Administrator perlu melakukan login, setelah *form* menu muncul kemudian menekan tombol Data Outlet, maka *form outlet* akan muncul. Untuk mengubah data *outlet*, pilih salah satu data yang akan diubah yang terdapat dalam tabel, kemudian informasi akan muncul dalam *textbox*. Setelah data diuubah, tekan tombol Submit, maka sistem akan menyimpan perubahan tersebut kemudian pesan akan muncul bahwa data yang diubah telah disimpan.



Gambar 3.12
Activity Diagram Manage Data Outlet - Update

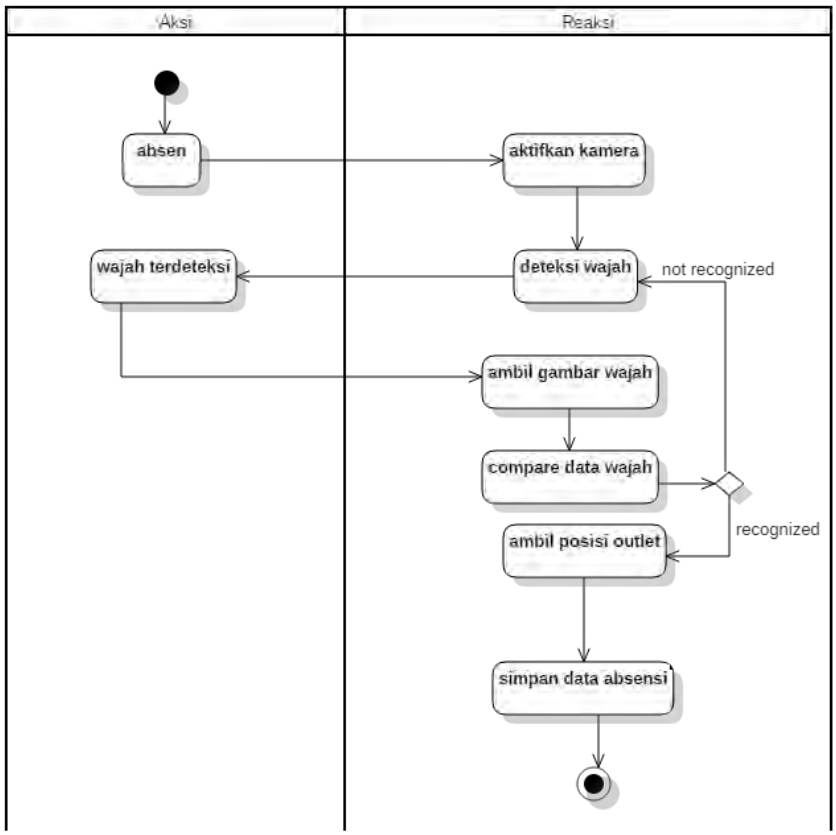
Lalu terdapat proses penghapusan data. Gambar 3.13 merupakan *Activity Diagram* proses penghapusan data untuk data *outlet*. Sama seperti proses sebelumnya, Administrator perlu melakukan login, setelah *form* menu muncul kemudian menekan tombol Data Outlet, maka *form outlet* akan muncul. Untuk menghapus data outlet, pilih salah satu data yang tertera pada tabel, kemudian tekan tombol bertanda "-". Pertanyaan konfirmasi akan muncul, jika menekan tombol "Ya", maka data *outlet* yang terpilih akan dihapus dan pesan akan muncul bahwa data telah dihapus.



Gambar 3.13
Activity Diagram Manage Data Outlet - Delete

3.7.2 Activity Diagram Record Attendance Using Face

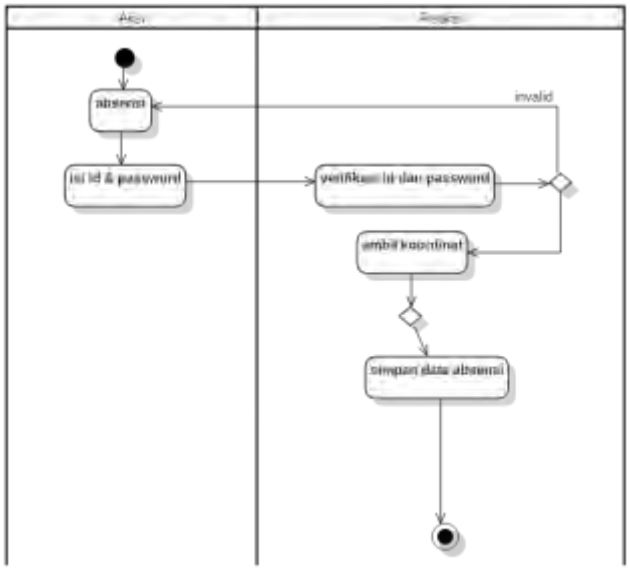
Gambar 3.9 merupakan *activity diagram* dalam proses *Scan Face Login*. Salah satu fungsi utama dari sistem absensi yang dirancang ada dalam proses ini. *Form Login* akan muncul, terdapat *layout* kamera yang otomatis aktif ketika *form Login* dibuka. Kamera akan selalu aktif untuk mendeteksi wajah. Saat wajah terdeteksi, maka gambar wajah akan diambil, kemudian dibandingkan dengan data wajah yang ada dalam basis data. Jika wajah tidak dikenali, maka sistem akan menyarankan untuk absensi dengan metode *Manual Login* atau diminta untuk menghubungi Administrator. Jika wajah dikenali, maka sistem akan menentukan posisi dimana pengguna melakukan absensi. Setelah itu data absensi akan disimpan ke dalam basis data.



Gambar 3.14
Activity Diagram Record Attendance Using Face

3.7.3 Activity Diagram Record Attendance

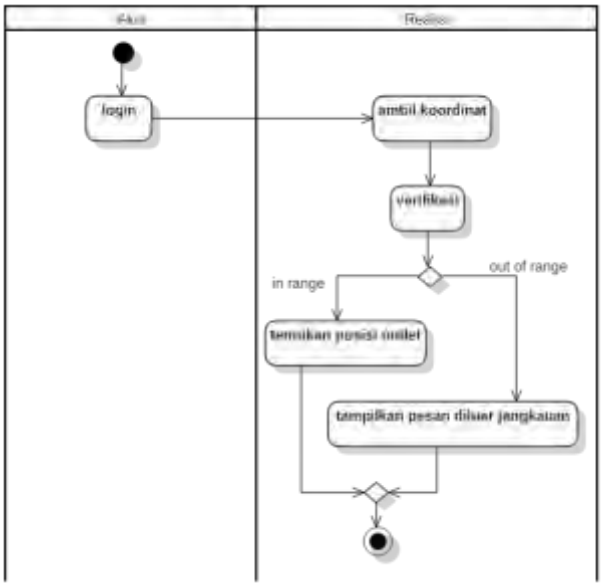
Gambar 3.10 merupakan *activity diagram* dalam proses *Record Attendance*. Metode Manual Login merupakan cadangan jika terjadi kesalahan dari kamera yang digunakan atau terdapat kesalahan sistem dalam mengenali wajah. Saat *form* Login muncul, terdapat *textbox* untuk mengisi *id* dan *password*. Setelah diisi, kemudian klik Submit, maka sistem akan memverifikasi *id* dan *password* dengan basis data yang ada, Jika valid, sistem akan menentukan posisi dimana absensi dilakukan, kemudian data absensi akan disimpan. Jika tidak valid, maka absensi tidak dapat dilakukan.



Gambar 3.15
Activity Diagram Record Attendance

3.7.4 Activity Diagram Determine Location

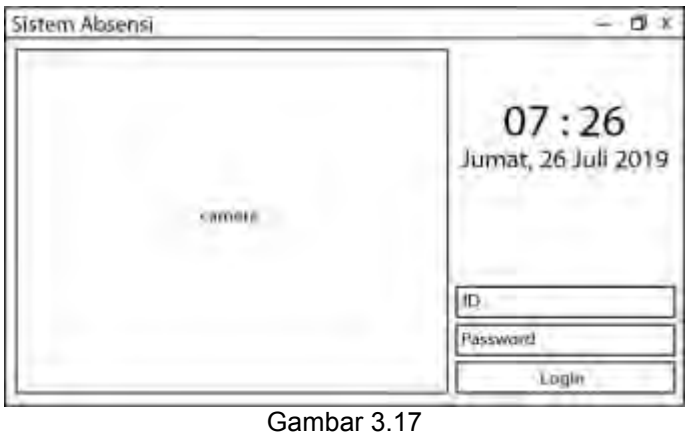
Gambar 3.11 merupakan *activity diagram* dalam proses penentuan lokasi absensi. Saat melakukan *login*, sistem akan meminta koordinat posisi dimana absensi dilakukan, kemudian koordinat tersebut akan dibandingkan dengan koordinat yang ada dalam basis data. Jika berada dalam jangkauan salah satu *outlet*, maka sistem akan menentukan *outlet* tersebut sebagai posisi dimana absensi dilakukan. Jika berada diluar jangkauan, maka terdapat pesan bahwa posisi berada diluar jangkauan.



Gambar 3.16
Activity Diagram Determine Location

3.8 Rancangan Antarmuka

Gambar 3.12 merupakan rancangan antarmuka untuk bagian absensi.

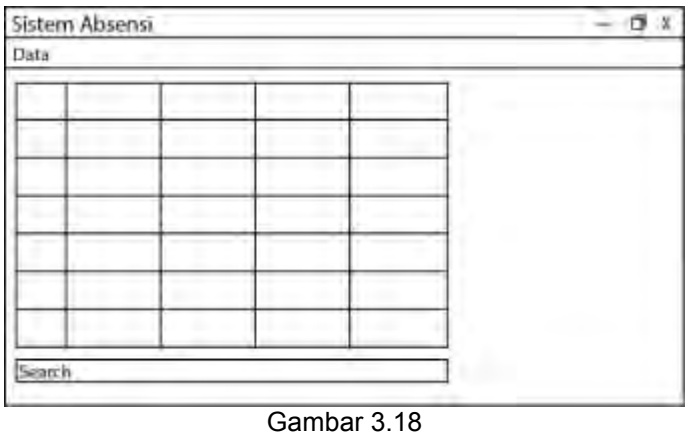


Gambar 3.17
Rancangan Antarmuka Login

Absensi dilakukan pada *form Attendance*. Terdapat panel kamera, yang langsung aktif saat *form* dibuka. Saat kamera mendeteksi wajah, maka proses absensi dilakukan. Jika terjadi kesalahan pada kamera dan tidak dapat digunakan, terdapat *textbox* untuk mengisi *id* dan *password.* Setelah itu tekan tombol Login untuk melakukan absensi. Terdapat info waktu dan tanggal.

3.8.1 Rancangan Antarmuka Menu

Gambar 3.13 merupakan rancangan antarmuka Menu

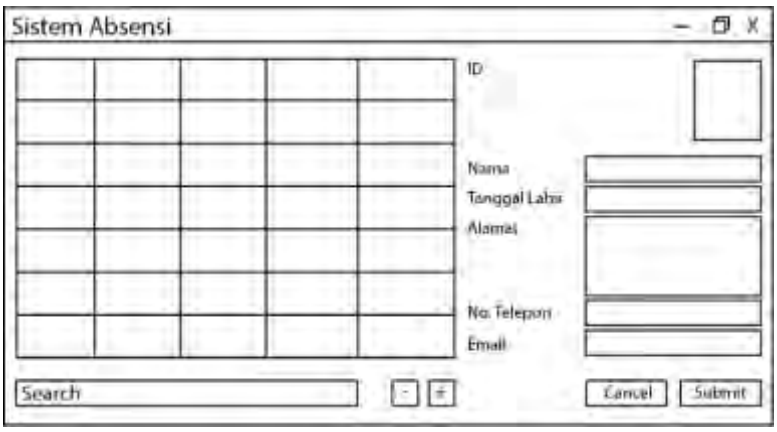


Gambar 3.18
Rancangan Antarmuka Menu

Untuk mengelola data, Administrator diharuskan untuk melakukan *login* pada *form Login*. Kemudian *form* Menu akan muncul. Pada *form* Menu terdapat tabel data absensi seluruh pengguna. Di bagian bawah terdapat *textbox* untuk mencari data dalam tabel. Di bagian atas terdapat *toolbar* dengan menu Data. Saat menu Data ditekan, terdapat dua pilihan yaitu *User* dan *Outlet.*

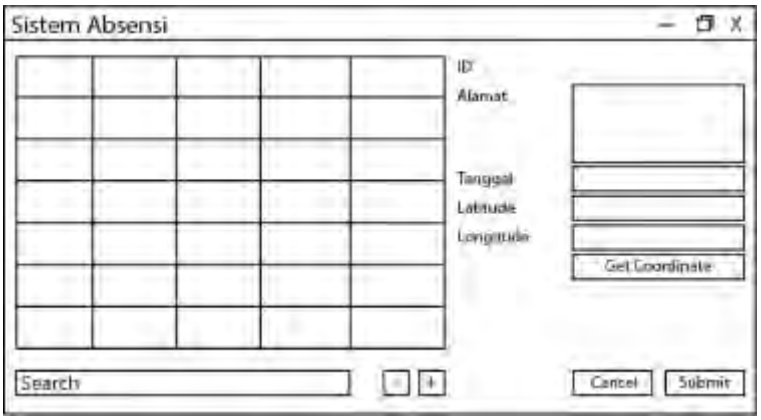
3.8.2 Rancangan Antarmuka Manage Data

Gambar 3.10 merupakan rancangan antarmuka *Manage Data User*



Gambar 3.19
Rancangan Antarmuka *Manage Data User*

Pada *form User*, terdapat tabel data pengguna yang sudah pernah disimpan dalam basis data. Untuk menambah data pengguna, tekan tombol “+”, sehingga sistem akan membuat dan menampilkan *id* baru, kemudian akan mengaktifkan *textbox*. Setelah semua data pengguna baru diisi dalam *textbox*, tekan Submit, kemudian terdapat panel kamera pada bagian atas kanan *form* yang akan aktif dan mendeteksi wajah. Setelah selesai makan data baru akan disimpan ke dalam basis data. Untuk mengubah data, tekan salah satu data tabel pengguna yang akan diubah. Kemudian data yang dipilih akan tertera ke dalam *textbox* sehingga dapat diubah. Tekan Submit jika sudah mengubah data. Untuk menghapus data, tekan salah satu data tabel pengguna yang akan dihapus, kemudia tekan tombol “-”. Terdapat pertanyaan konfirmasi, jika setuju maka data akan dihapus. Terdapat *textbox* di bagian bawah tabel untuk mencari data dalam tabel.



Gambar 3.20
Rancangan Antarmuka *Manage Data Outlet*

Pada *form Outlet*, terdapat data tabel *outlet* yang sudah pernah disimpan dalam basis data. Untuk menambahkan data, tekan tombol “+”, maka sistem akan membuat dan menampilkan *id* baru untuk data *outlet* yang akan disimpan. Sistem membutuhkan data *latitude* dan *longitude* untuk menyimpan data *outlet*, untuk mempermudah, terdapat tombol Get Coordinate yang berfungsi untuk mendapatkan data *latitude* dan *longitude* secara otomatis dengan syarat Administrator harus berada di tempat dimana *outlet* akan ditempatkan. Setelah itu tekan tombol Submit untuk menyimpan data. Untuk mengubah data, tekan salah satu data *outlet* yang akan diubah pada tabel, maka data akan tertera pada *textbox* sehingga dapat diubah. Tekan tombol Submit untuk menyimpan perubahan data. Untuk menghapus data, tekan salah satu data *outlet* yang akan dihapus pada tabel, kemudian tekan tombol “-“. Akan muncul pertanyaan konfirmasi, jika setuju maka data akan dihapus. Juga terdapat *textbox* di bagian bawah tabel untuk mencari data dalam tabel.

BAB IV

IMPLEMENTASI DAN PENGUJIAN PERANGKAT LUNAK

4.1 Spesifikasi Kebutuhan Perangkat Keras

Pada sub-bab ini menjelaskan tentang spesifikasi perangkat keras yang digunakan selama aplikasi ini dikembangkan. Pengembangan aplikasi ini menggunakan *notebook* dengan spesifikasi sebagai berikut :

Model	: Asus A455L
Processor	: Intel Core i3-4005U 1.7 GHz
RAM	: 6GB DDR3L
Kartu Grafis	: NVIDIA GeForce 930M 2GB
Layar	: 14” dengan resolusi 1366x768
Media Penyimpanan	: 500GB
Sistem Operasi	: Windows 10 64-bit

4.2 Pengujian Antarmuka

Sub-bab ini akan menjelaskan pengujian dan kesesuaian antar muka yang telah dirancang dalam bab 3. Terdapat empat antar muka utama dalam aplikasi sistem absensi yang dibuat, *form Attendance*, *form Menu*, *form User* dan *form Outlet*.

4.2.1 Pengujian Antarmuka *Form Attendance*

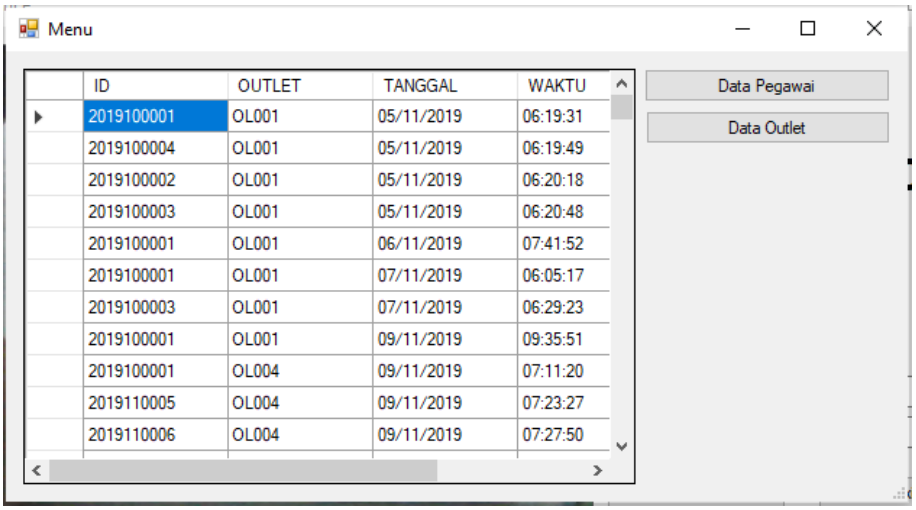
Gambar 4.1 merupakan antarmuka *form Attendance*. Bagian kiri *form* terdapat tempat untuk menampilkan hasil *capture* kamera. Kemudian terdapat tanggal dan waktu sebagai salah satu informasi absensi. Untuk melakukan absensi, pengguna hanya perlu menempatkan posisi wajah ke depan kamera, kemudian sistem akan mendeteksi wajah. Setelah itu sistem akan membandingkan wajah yang di depan kamera dengan data wajah yang ada di basis data, setelah mendapat status pengenalan wajah, sistem akan mengambil posisi dimana absensi dilakukan, kemudian akan muncul notifikasi bahwa absensi telah dilakukan. Jika terjadi kesalahan pada kamera, pengguna dapat melakukan

absensi secara manual. Pengguna harus mengisi *username* dan *password* ke *textbox* yang telah disediakan. Setelah itu tekan tombol Absen, maka sistem akan melakukan verifikasi *username* dan *password* terhadap basis data. Jika *username* dan *password* valid, maka sistem akan mengambil posisi dimana absensi dilakukan dan menampilkan notifikasi bahwa absensi telah dilakukan. Untuk mengakses menu, Administrator perlu melakukan absensi terlebih dahulu menggunakan absensi manual, kemudian terdapat pertanyaan untuk mengakses menu, jika menjawab "*Yes*", maka *form* menu akan muncul.

Gambar 4.1
Antarmuka *form Attendance*

4.2.2 Pengujian Antarmuka *Form Menu*

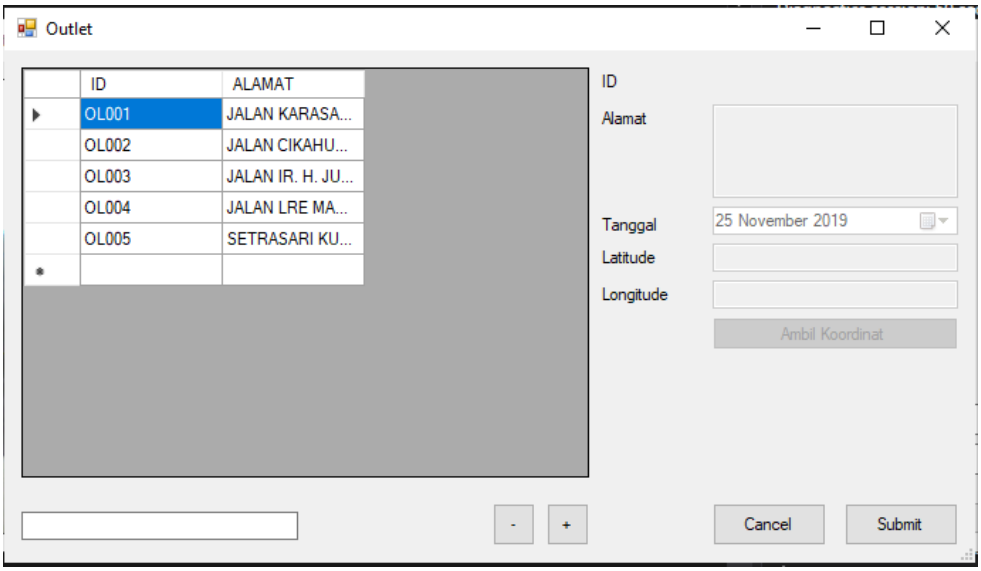
Gambar 4.2 merupakan antarmuka *form Menu*. Dalam *form* ini terdapat informasi data absensi melalui tabel. Bagian kiri atas terdapat *menu*, jika di tekan, maka akan muncul *sub-menu Outlet* dan *User*.



Gambar 4.2
Antarmuka *form Menu*

4.2.3 Pengujian Antarmuka *Form Outlet*

Gambar 4.3 merupakan antarmuka *form Outlet*. Bagian kiri merupakan data *outlet* yang sudah ada dalam basis data yang ditampilkan dalam bentuk tabel. Untuk menghapus data yang sudah ada, tekan salah satu ID *outlet* yg akan dihapus, kemudian tekan tombol " - ", maka akan muncul pertanyaan konfirmasi untuk menghapus data tersebut, jika memilih Yes, maka data akan dihapus. Jika ingin menambah data, tekan tombol " + ", maka *textbox* akan aktif, kemudian isi data yg sesuai ke dalam *textbox*, terdapat tombol Ambil Koordinat jika administrator ingin mendapatkan koordinat lokasi pada saat itu. Kemudian tekan tombol Submit, maka data akan tersimpan ke dalam basis data. Untuk mengubah data, tekan salah satu ID *outlet* yang akan diubah, kemudian data akan tertera ke dalam *textbox* yang telah tersedia. Setelah itu ubah data yang tertera dalam *textbox*, kemudian tekan Submit, maka sistem akan memperbarui data yang ada dalam basis data.

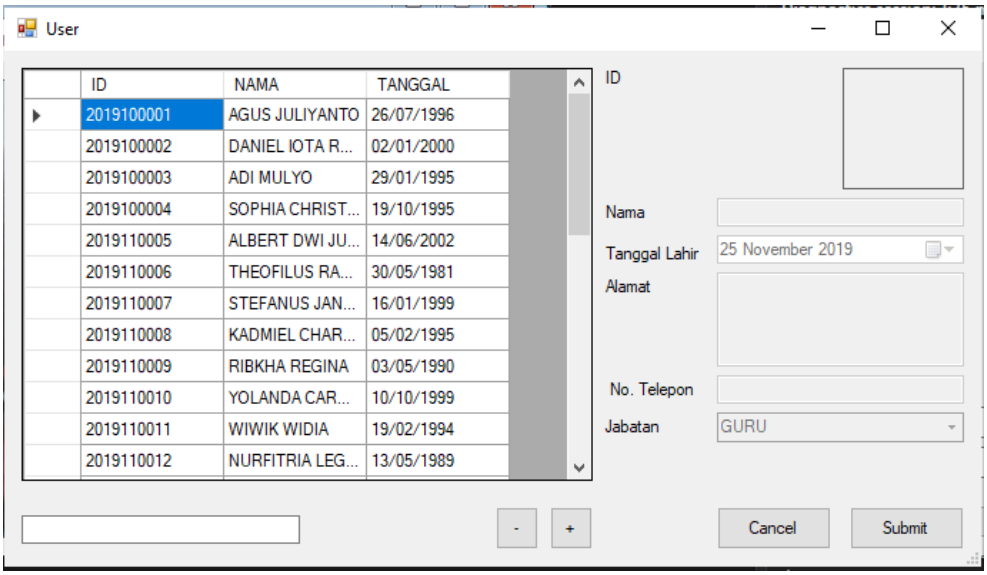


Gambar 4.3
Antarmuka *form Outlet*

4.2.4 Pengujian Antarmuka *Form User*

Gambar 4.4 merupakan antarmuka *form Menu*. Bagian kiri terdapat data pengguna yang ditampilkan dalam bentuk tabel. Untuk menghapus data yang sudah ada, pilih salah satu ID pengguna, kemudian tekan tombol " - ", maka akan muncul pertanyaan

konfirmasi, jika memilih Yes, maka data yang dipilih akan dihapus dari basis data. Untuk mengubah data, maka data akan tertera ke *textbox* dan *combobox* yang tersedia. Ubah data, kemudian tekan tombol Submit, maka data yang dipilih akan diperbarui. Untuk menambahkan data pengguna, tekan tombol " + ", maka *textbox* dan *combobox* yang tersedia akan aktif untuk mengisikan data pengguna. Setelah selesai mengisi, tekan Submit, maka kamera akan aktif untuk mengambil data wajah pengguna. Akan muncul pemberitahuan bahwa data telah disimpan.



Gambar 4.4
Antarmuka *form User*

4.3 Pengujian Fungsi

Sub-bab ini akan menjelaskan pengujian fungsi dari perangkat lunak sistem absensi yang telah dibuat. Terdapat beberapa fungsi yang akan diuji, diantaranya Pengujian *White Box* dan Pengujian *Face Recognition*.

4.3.1 Pengujian *White Box*

Pengujian *White Box* merupakan pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. *Cyclomatic Complexity* adalah matriks perangkat lunak yang memberikan pengukuran kuantitatif

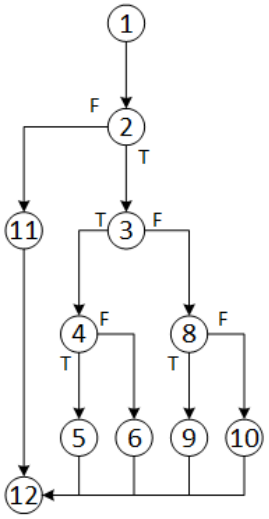
terhadap kompleksitas logis suatu program. Nilai yang didapat akan menentukan jumlah alur independen dalam himpunan *path*, serta akan memberi nilai batas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua pernyataan telah dilakukan paling sedikit satu kali.

4.3.1.1 Pengujian White Box Absensi

Tabel 4.1
Tabel *Source Code* Evaluasi Absensi

Node	Source Code
1	Input username and password from user
2	if (username == usr.id && password == usr.password) {
3	if (DateTime.Now > DateTime.Parse("06:00:00 AM") && DateTime.Now < DateTime.Parse("05:00:00 PM")) {
4	stat = "CHECK-IN"; if(att.checkAttendanceIn(username, dtnow) == "checked") {
5	MessageBox.Show(usr.name + " sudah melakukan absensi masuk!"); }
6	Else { saveData(username); }
7	Else if (DateTime.Now >= DateTime.Parse("05:00:00 PM") && DateTime.Now < DateTime.Parse("10:00:00 PM")) {
8	stat = "CHECK-OUT"; if(att.checkAttendanceOut(username, dtnow) == "checked") {
9	MessageBox.Show(usr.name + " sudah melakukan absensi masuk!"); }
10	Else { saveData(username); }}
11	} else { MessageBox.Show("Username/Password salah!"); }

Dari tabel 4.1 dapat ditentukan *Flow Graph Notation*



Gambar 4.5
Flow Graph Notation Absensi

Dari gambar 4.5 maka dapat ditentukan *Cyclomatic Complexity* sebagai berikut :

$$V(G) = E - N + 2 = 14 - 11 + 2 = 5 \quad (4.1)$$

Jadi, jalur bebas pada *flow graph notation* yang akan diuji adalah sebanyak 5 jalur.

Berdasarkan *flow graph notation* di atas, maka dapat ditentukan *Independent path*

Tabel 4.2
Tabel *Independent Path* Absensi

Basis Flow	Independent Path (Jalur Bebas)
Jalur 1	1 – 2 – 11 – 12
Jalur 2	1 – 2 – 3 – 4 – 5 – 12
Jalur 3	1 – 2 – 3 – 4 – 6 – 12
Jalur 4	1 – 2 – 3 – 8 – 9 – 12
Jalur 5	1 – 2 – 3 – 8 – 10 – 12

4.3.1.2 Pengujian White Box Penentuan Lokasi

Tabel 4.3
Tabel *Source Code* Evaluasi Penentuan Lokasi

Node	Source Code

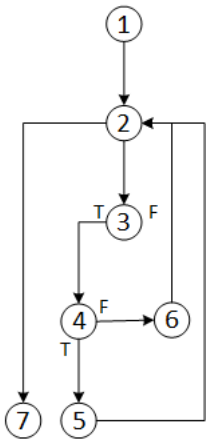
1	string otlID = "";
	listOutlet = getList("otl_id");
	listLatitude = getCoordinate("otl_lat");
	listLongitude = getCoordinate("otl_long");

2	for (int i = 0; i < listOutlet.Count; i++) {
3	double dataLat = Math.Abs(listLatitude[i]);

	double dataLong = Math.Abs(listLongitude[i]);
	
	double latitudeMax = Math.Abs(dataLat + 0.0005);
	double latitudeMin = Math.Abs(dataLat - 0.0005);
	double longitudeMax = Math.Abs(dataLat + 0.0005);
	double longitudeMin = Math.Abs(dataLat - 0.0005);
	
	if (Math.Abs(Math.Round(posLat, 6)) >= latitudeMin &&
	Math.Abs(Math.Round(posLong, 6)) >= longitudeMin) {

4	if(Math.Abs(Math.Round(posLat, 6)) <= latitudeMax && Math.Abs(Math.Round(posLat, 6)) <= longitudeMax) {
5	otlID = listOutlet[i]; } }
6	Else { otlID = "outofrange!"; } }
7	return otlID; } }

Dari tabel 4.3 dapat ditentukan *Flow Graph Notation*



Gambar 4.6
Flow Graph Notation Penentuan Lokasi

Dari gambar 4.6 maka dapat ditentukan *Cyclomatic Complexity*

$$V(G) = E - N + 2 = 8 - 7 + 2 = 3 \quad (4.2)$$

Jadi, jalur bebas pada *flow graph notation* yang akan diuji adalah sebanyak 3 jalur.

Berdasarkan *flow graph notation* di atas, maka dapat ditentukan *Independent path*

Tabel 4.4
Tabel *Independent Path* Penentuan Lokasi

Basis Flow	**Independent Path (Jalur Bebas)**
Jalur 1	1 – 2 – 7

Jalur 2	1 – 2 – 3 – 4 – 5 – 2 – 7
Jalur 3	1 – 2 – 3 – 4 – 6 – 2 – 7

4.3.1.3 Pengujian White Box Pengenalan Wajah

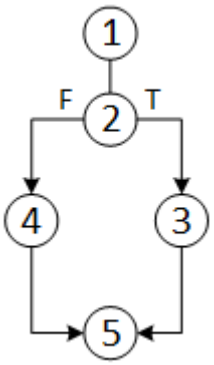
Tabel 4.5
Tabel *Source Code* Evaluasi Pengenalan Wajah

Node	Source Code

1	imageFrame.Draw(face, new Bgr(Color.AntiqueWhite), 2);
	var grayFace = imageFrame.Copy(face).Convert<Gray, byte>().Resize(120, 120, Emgu.CV.CvEnum.Inter.Cubic);
	
	Recognizer rec = new Recognizer();
	string sid = rec.recognizeFace(grayFace).ToString();
	usr.loadData(sid);

2	if (sid == "-1" || sid == "0") {
3	var w = new Form() { Size = new Size(0, 0) }; Task.Delay(TimeSpan.FromSeconds(1.5)).ContinueWith((t) => w.Close(), TaskScheduler.FromCurrentSynchronizationContext()); MessageBox.Show(w, "Wajah tidak dikenal!", "Attendance");
4	Else { drawNotification(sid, usr.name, outlet); }

Dari tabel 4.5 dapat ditentukan *Flow Graph Notation*



Gambar 4.7
Flow Graph Notation Pengenalan Wajah

Dari gambar 4.7 maka dapat ditentukan *Cyclomatic Complexity*

$$V(G) = E - N + 2 = 5 - 5 + 2 = 2 \quad (4.3)$$

Jadi, jalur bebas pada *flow graph notation* yang akan diuji adalah sebanyak 2 jalur.

Berdasarkan *flow graph notation* di atas, maka dapat ditentukan *Independent path*

Tabel 4.6
Tabel *Independent Path* Pengenalan Wajah

Basis Flow	Independent Path (Jalur Bebas)
Jalur 1	1 – 2 – 3 – 5
Jalur 2	1 – 2 – 4 – 5

4.3.2 Pengujian Metode *Local Binary Patterns*

Pengujian ini dilakukan untuk mengetahui kinerja sistem dalam mengenali wajah seseorang dengan menggunakan metode *Local Binary Patterns*. Terdapat beberapa parameter yang menjadi faktor penentu kinerja sistem pengenalan wajah yaitu FAR (*False Acceptance Rate*), FRR (*False Rejection Rate*) dan tingkat akurasi. *False Acceptance Rate* merupakan kesalahan sistem dalam mengenali identitas gambar masukan, baik kesalahan dalam mengenali identitas gambar masukan dari individu di luar basis data yang terdeteksi sebagai individu di dalam basis data, maupun kesalahan dalam mengenali identitas gambar masukan dari individu di dalam basis data yang dikenali sebagai individu lain.

$$FAR = \frac{jumlah\ FAR}{jumlah\ percobaan} \times 100\% \quad (4.4)$$

False Rejection Rate merupakan kesalahan sistem dalam menolak gambar masukan. Sebuah gambar masukan yang seharusnya dapat dikenali (identitasnya terdapat di dalam basis data) berubah menjadi tidak dikenali.

$$FRR = \frac{jumlah\ FRR}{jumlah\ percobaan} \times 100\% \quad (4.5)$$

Pengujian ini menggunakan 31 data wajah yang disimpan ke dalam basis data. Dengan merubah parameter dalam metode LBP yaitu nilai *threshold* untuk mengevaluasi kinerja sistem dalam mengenali wajah seseorang.

Tabel 4.7
Tabel Pengujian Metode LBP

		Uji ke-1	Uji ke-2	Uji ke-3	Uji ke-4	Uji ke-5
Threshold 50	**FAR**	0	0	0	0	0
	FRR	29	30	29	30	28
	Berhasil	2	1	2	1	3
Threshold 75	**FAR**	0	0	0	0	0
	FRR	16	13	15	12	10

	Berhasil	15	18	16	19	21
Threshold 100	**FAR**	4	3	3	3	2
	FRR	2	2	0	0	0
	Berhasil	25	26	28	28	29
Threshold 150	**FAR**	3	2	2	3	1
	FRR	0	0	0	0	0
	Berhasil	28	29	29	28	30
Threshold 200	**FAR**	3	3	4	4	2
	FRR	0	0	0	0	0
	Berhasil	28	28	27	27	29

Dari tabel 4.7 pengujian dapat disederhanakan menggunakan metode *Confusion Matrix*

Tabel 4.8
Tabel *Confusion Matrix* Metode LBP

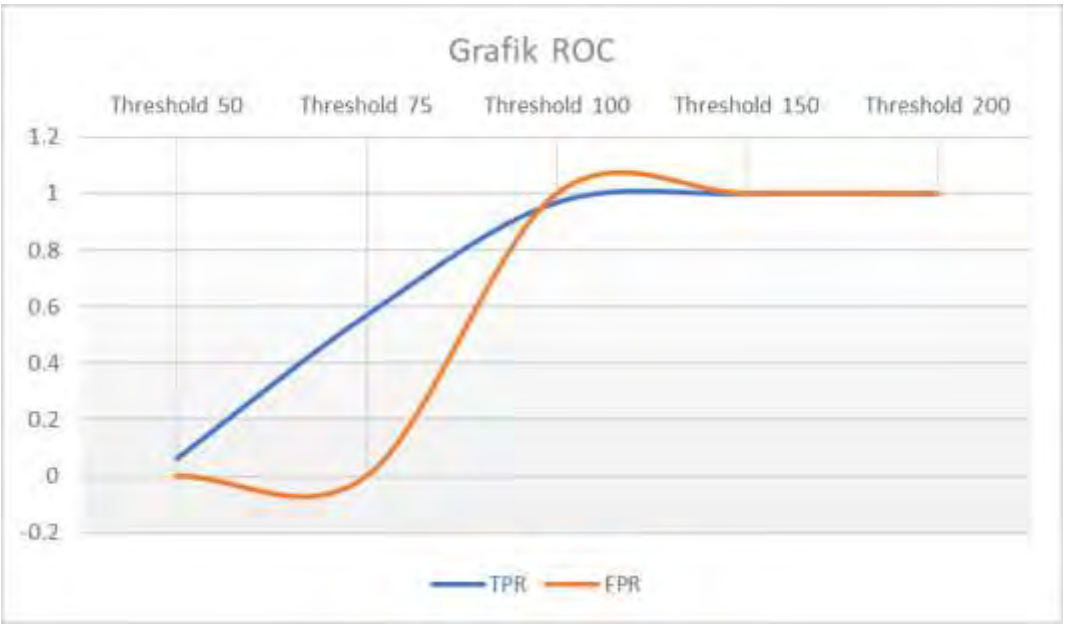
	Threshold 50	**Threshold 75**	**Threshold 100**	**Threshold 150**	**Threshold 200**
True Positive (Success)	9	89	136	144	139
False Positive (FAR)	0	0	15	11	16
False Negative (FRR)	146	66	4	0	0

Dari Tabel 4.8 maka dapat dihitung nilai *Accuracy*, *Precision*, *Recall* dan *F1 Score*.

Tabel 4.9
Tabel Perhitungan Akurasi

	Threshold 50	**Threshold 75**	**Threshold 100**	**Threshold 150**	**Threshold 200**
Accuracy	5.81 %	57.42 %	87.74 %	92.90 %	89.68 %
Precision	100 %	100 %	90.07 %	92.90 %	89.68 %
Recall	5.81 %	57.42 %	97.14 %	100 %	100 %
F1 Score	10.98 %	72.95 %	93.47 %	96.32 %	94.65%

Dari Tabel 4.9 maka dapat digambarkan kurva ROC (*Receiver Operating Characteristics*)



Gambar 4.8
Grafik ROC

Dari seluruh pengujian yang dilakukan, maka dapat disimpulkan bahwa perangkat lunak pencatatan kehadiran berbasis *face recognition* dengan menggunakan metode *Local Binary Patterns* (LBP) memiliki tingkat akurasi sebesar 92.90% dan tingkat presisi sebesar 92.90% dengan menggunakan nilai *threshold* 150.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah menyelesaikan rancangan perangkat lunak sistem absensi yang menggunakan pengenalan wajah dengan metode LBP (*Local Binary Patterns*) dan *geolocation* serta melakukan beberapa pengujian dan menganalisis hasil dari pengujian, maka dapat diperoleh beberapa kesimpulan :

1. Sistem absensi ini dirancang menggunakan metode *Local Binary Patterns* (LBP). Saat melakukan registrasi pengguna, sistem akan menyimpan gambar wajah yang diambil oleh kamera ke dalam basis data kemudian sistem akan belajar untuk mengenali wajah yang diambil. Secara detail, gambar akan diubah menjadi *grayscale*, kemudian akan dibagi menjadi beberapa *grid*. Lalu dari setiap *grid*, ditetapkan nilai tengah yang diambil dari nilai biner piksel disekelilingnya. Jadi, setiap *grid* memiliki satu nilai yang mewakili. Kemudian nilai-nilai tersebut digabung menjadi satu histogram yang akan menjadi nilai perbandingan. Kemudian saat pencatatan kehadiran dilakukan sistem akan mengambil gambar wajah melalui kamera, kemudian sistem akan membandingkan gambar wajah yang diambil dengan data wajah yang tersimpan dalam basis data. Jika wajah dikenali, maka sistem akan mencatat kehadiran dan menentukan posisi dimana absensi dilakukan. Jika wajah tidak dikenali, pengguna diminta untuk melakukan absensi dengan cara *login username* dan *password* atau menghubungi Administrator.
2. Nilai akurasi yang dapat diambil adalah ketika nilai TPR (*True Positive Rate*) dan nilai FPR (*False Positive Rate*) memiliki nilai yang sama yaitu pada Threshold 150 dengan nilai akurasi 92.90% dan nilai presisi 92.90%.

5.2 Saran

Perancangan sistem absensi ini sangat sederhana dan tidak sempurna karena keterbatasan pada alat dan sumber daya dalam pengembangan. Oleh sebab itu, terdapat beberapa saran yang dapat diberikan untuk meningkatkan kinerja perangkat lunak ini :

1. Menggunakan kamera (atau penggunaan kamera tambahan) dengan resolusi yang lebih besar untuk mempertajam hasil gambar yang akan diambil.
2. Menambahkan atau menggabungkan metode lain dengan metode LBP untuk meningkatkan nilai akurasi.

DAFTAR PUSTAKA

Caponetti, Giovanna, Giovanna Castellano, 2017, "*Fuzzy Logic for Image Processing A Gentle Introduction Using Java*", Springer

Das, Ravindra, 2014, "*Biometric Technology Authentication, Biocryptography, and Cloud-Based Architecture*", CRC Press

Datta, Madhura, et al., 2015, "*Face Detection and Recognition Theory and Practice*", CRC Press

Foster, C. Elvis, 2014, "*Software Engineering A Methodical Approach*", Apress

Foster, Shripad, Shripad Godbole, 2016, "*Database System A Pragmatic Approach*", Apress

Gentile, Nayef et al., 2013, "*Geolocation Technique - Principles and Applications*", Springer

Gregory, Michael, Michael A. Simon, 2008, "*Biometric For Dummies*", For Dummies

Jain, Arun, et al., 2011, "*Introduction to Biometrics*", Springer

Karande, Sanjay, Sanjay Talbar, 2014, "*Independent Component Analysis of Edge Information for Face Recognition*", Springer

Kroenke, David J., David J. Auer, 2016, "*Database Processing Fundamentals, Design, and Implementation Fourteenth Edition*", Pearson

Pal, Sankar, Sankar K. Pal, 2017, "*Pattern Recognition and Big Data*", World Scientific

Pietikäinen, Abdenour, et al., 2011, "*Computer Vision Using Local Binary Patterns", Springer*

Pressman, Bruce, Bruce Maxim, 2015, "*Software Engineering A Practitioner's Approach Eight Edition*", McGraw-Hill

Pressman, Roger s., 2010, "*Software Engineering A Practitioner's Approach Seventh Edition*", McGraw-Hill

Puntambekar, Anuradha A., 2009, "*Software Engineering Third Revised Edition*", Technical Publications Pune

Xu, Yan, Yan Xu, 2016, "*GPS Theory, Algorithms and Applications Third Edition*", Springer

https://visualstudio.microsoft.com/vs/ , diakses tanggal 25 April 2019, jam 09:30

http://www.sqlmanager.net/en/products/mysql/manager , diakses tanggal 20 November 2017, jam 07:59

Class Attendance

```
using MySql.Data.MySqlClient;
using System;
using System.Data;

namespace Thesis
{
   class Attendance
   {
      public string emp_id { get; set; }
      public string otl_id { get; set; }
      public string date { get; set; }
      public string time { get; set; }
      public string stat { get; set; }

      public MySqlConnection connection = new MySqlConnection
         ("datasource=localhost;port=3306;Initial
Catalog='thesis';username=root;password=");
      MySqlCommand command;
      MySqlDataReader mdr;

      public void openConnection()
      {
         if (connection.State == ConnectionState.Closed)
         {
            connection.Open();
         }
      }

      public void closeConnection()
      {
         if (connection.State == ConnectionState.Open)
         {
            connection.Close();
         }
      }

      public string insert()
      {
         string status = "";
         try
         {
            string query = "insert into attendance(emp_id, otl_id, att_date, att_time,
att_stat) values('" + emp_id + "','" + otl_id + "','" + date + "','" + time + "','" + stat + "')";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
               return status = "saved";
            }
            else
            {
               return status = "failed";
            }
```

```
            }
            catch (Exception ex)
            {
                return status = "Error : " + ex.Message;
            }
            finally
            {
                closeConnection();
            }
        }

        public string update(string sid)
        {
            string status = "";
            try
            {
                string query = "update attendance set emp_id = '" + emp_id + "', otl_id = '" +
otl_id + "', att_date = '" + date + "', att_time = '" + time + "', att_stat = '" + stat + "' where
att_id = '" + sid + "'";
                command = new MySqlCommand(query, connection);

                openConnection();
                if (command.ExecuteNonQuery() == 1)
                {
                    return status = "updated";
                }
                else
                {
                    return status = "failed";
                }

            }
            catch (Exception ex)
            {
                return status = "Error : " + ex.Message;
            }
            finally
            {
                closeConnection();
            }
        }

        public string delete(string sid)
        {
            string status = "";
            try
            {
                string query = "delete from attendance where att_id = '" + sid + "'";
                command = new MySqlCommand(query, connection);

                openConnection();
                if (command.ExecuteNonQuery() == 1)
                {
                    return status = "deleted";
                }
                else
                {
                    return status = "failed";
```

```
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public void loadData(string sid)
    {
        try
        {
            string query = "select emp_id, otl_id, att_date, att_time, att_stat where att_id =
'" + sid + "'";
            MySqlCommand command = new MySqlCommand(query, connection);

            openConnection();
            mdr = command.ExecuteReader();

            while (mdr.Read())
            {
                emp_id = mdr.GetValue(0).ToString();
                otl_id = mdr.GetValue(1).ToString();
                date = mdr.GetValue(2).ToString();
                time = mdr.GetValue(3).ToString();
                stat = mdr.GetValue(4).ToString();
            }
        }
        catch (Exception ex)
        {
            return;
        }
        finally
        {
            closeConnection();
        }
    }

    public string checkAttendanceIn(string sid, string dt)
    {
        string status = "";

        openConnection();

        command = new MySqlCommand("select count("+ sid +") from attendance where
emp_id = '" + sid + "' and att_date = '"+ dt +"' and att_stat LIKE '%CHECK-IN%'",
connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            int count = Convert.ToInt32(mdr[0].ToString());
            if (count == 1)
            {
```

```
                status = "checked";
            }
            else
            {
                status = "failed";
            }
        }
        closeConnection();
        return status;
    }

    public string checkAttendanceOut(string sid, string dt)
    {
        string status = "";

        openConnection();

        command = new MySqlCommand("select count(" + sid + ") from attendance
where emp_id = '" + sid + "' and att_date ='" + dt + "' and att_stat ='CHECK-OUT'",
connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            int count = Convert.ToInt32(mdr[0].ToString());
            if (count == 1)
            {
                status = "checked";
            }
            else
            {
                status = "failed";
            }
        }
        closeConnection();
        return status;
    }
  }
}
```

Class Face

```
using Emgu.CV;
using Emgu.CV.Structure;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;

namespace Thesis
{
  class Face
  {
    public string id { get; set; }
    public Byte[] faceBlob { get; set; }
    public string address { get; set; }

    public MySqlConnection connection = new MySqlConnection
```

```
        ("datasource=localhost;port=3306;Initial
Catalog='thesis';username=root;password=");
    MySqlCommand command;
    MySqlDataReader mdr;

    public List<Image<Gray, byte>> faces { get; set; }
    public List<int> lists { get; set; }

    public void openConnection()
    {
        if (connection.State == ConnectionState.Closed)
        {
            connection.Open();
        }
    }

    public void closeConnection()
    {
        if (connection.State == ConnectionState.Open)
        {
            connection.Close();
        }
    }

    public string insert()
    {
        string status = "";
        try
        {
            string query = "insert into face(emp_id, fac_sample, fac_source) values('" + id +
"','" + faceBlob + "','" + address + "')";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
                return status = "saved";
            }
            else
            {
                return status = "failed";
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public string update(string sid)
    {
        string status = "";
        try
```

```
        {
            string query = "update face set face_blob = '" + faceBlob + "', fac_source = '" +
faceBlob + "' where emp_id = '" + sid + "'";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
                return status = "updated";
            }
            else
            {
                return status = "failed";
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public string delete(string sid)
    {
        string status = "";
        try
        {
            openConnection();
            string query = "delete from face where emp_id = '" + sid + "'";
            command = new MySqlCommand(query, connection);


            if (command.ExecuteNonQuery() == 1)
            {
                return status = "delete";
            }
            else
            {
                return status = "failed";
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public void loadData(string sid)
    {
```

```
            try
            {
                openConnection();
                string query = "select emp_id, face_blob, fac_source from face where emp_id =
'" + sid + "'";
                MySqlCommand command = new MySqlCommand(query, connection);


                mdr = command.ExecuteReader();

                while (mdr.Read())
                {
                    id = mdr.GetValue(0).ToString();
                    faceBlob = (byte[])mdr.GetValue(1);
                    address = mdr.GetValue(2).ToString();
                }
            }
            catch (Exception ex)
            {
                return;
            }
            finally
            {
                closeConnection();
            }
        }

        public void loadFaces()
        {
            faces = new List<Image<Gray, byte>>();
            lists = new List<int>();

            try
            {
                openConnection();
                string query = "select emp_id, fac_source from face order by emp_id";
                MySqlCommand command = new MySqlCommand(query, connection);


                mdr = command.ExecuteReader();

                while (mdr.Read())
                {
                    var sid = Convert.ToInt32(mdr.GetValue(0).ToString());
                    lists.Add(sid);

                    Bitmap bmf = new Bitmap(mdr.GetValue(1).ToString());
                    var face = new Image<Gray, byte>(bmf);
                    faces.Add(face);
                }
            }
            catch(Exception ex)
            {

            }
            finally
            {
                closeConnection();
```

```
        }
    }

    public string dataCheck()
    {
        string status = "";
        openConnection();

        command = new MySqlCommand("select count(emp_id) from face", connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            int count = Convert.ToInt32(mdr.GetValue(0));
            if (count < 1)
            {
                status = "insert";
            }
            else
            {
                status = "update";
            }
        }
        closeConnection();
        return status;
    }
  }
}
```

Class Outlet

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Thesis
{
  class Outlet
  {
    public string id { get; set; }
    public string address { get; set; }
    public string dob { get; set; }
    public double latitude { get; set; }
    public double longitude { get; set; }

    public MySqlConnection connection = new
MySqlConnection("datasource=localhost;port=3306;Initial
Catalog='thesis';username=root;password=");
    MySqlCommand command;
    MySqlDataReader mdr;

    public List<string> listOutlet = new List<string>();
    public List<double> listLatitude = new List<double>();
    public List<double> listLongitude = new List<double>();
```

```
        public void openConnection()
        {
            if (connection.State == ConnectionState.Closed)
            {
                connection.Open();
            }
        }

        public void closeConnection()
        {
            if (connection.State == ConnectionState.Open)
            {
                connection.Close();
            }
        }

        public string insert()
        {
            string status = "";
            try
            {
                string query = "insert into outlet(otl_id, otl_address, otl_dob, otl_lat, otl_long)
values('" + id + "','" + address + "','" + dob + "','" + latitude + "','" + longitude + "')";
                command = new MySqlCommand(query, connection);

                openConnection();
                if(command.ExecuteNonQuery() == 1)
                {
                    return status = "saved";
                }
                else
                {
                    return status = "failed";
                }

            }
            catch (Exception ex)
            {
                return status = "Error : " + ex.Message;
            }
            finally
            {
                closeConnection();
            }
        }

        public string update(string sid)
        {
            string status = "";
            try
            {
                string query = "update outlet set otl_address = '" + address + "', otl_dob = '" +
dob + "', otl_lat = '" + latitude + "', otl_long = '" + longitude + "' where otl_id = '" + sid + "'";
                command = new MySqlCommand(query, connection);

                openConnection();
                if (command.ExecuteNonQuery() == 1)
                {
```

```
                return status = "update";
            }
            else
            {
                return status = "failed";
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public string delete(string sid)
    {
        string status = "";
        try
        {
            string query = "delete from outlet where otl_id = '"+ sid +"'";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
                return status = "delete";
            }
            else
            {
                return status = "failed";
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public void loadData(string sid)
    {
        try
        {
            string query = "select otl_id, otl_address, otl_dob, otl_lat, otl_long from outlet
where otl_id = '" + sid + "'";
            MySqlCommand command = new MySqlCommand(query, connection);

            openConnection();
            mdr = command.ExecuteReader();
```

```
            while (mdr.Read())
            {
                id = mdr.GetValue(0).ToString();
                address = mdr.GetValue(1).ToString();
                dob = mdr.GetValue(2).ToString();
                latitude = double.Parse(mdr.GetValue(3).ToString(),
System.Globalization.CultureInfo.InvariantCulture);
                longitude = double.Parse(mdr.GetValue(4).ToString(),
System.Globalization.CultureInfo.InvariantCulture);
            }
        }
        catch (Exception ex)
        {
            return;
        }
        finally
        {
            closeConnection();
        }
    }

    public string IDcheck(string sid)
    {
        string status = "";
        openConnection();

        command = new MySqlCommand("select count(otl_id) from outlet where otl_id='"
+ sid + "'", connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            int count = Convert.ToInt32(mdr[0].ToString());
            if(count == 0)
            {
                status = "insert";
            }
            else
            {
                status = "update";
            }
        }
        closeConnection();
        return status;
    }

    public string createID()
    {
        string tmpID = "";
        openConnection();

        command = new MySqlCommand("select max(otl_id) from outlet", connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            if(mdr.IsDBNull(0))
            {
                tmpID = "OL001";
            }
```

```
            else
            {
                int max = Convert.ToInt32(mdr[0].ToString().Substring(2, 3));
                int idn = max + 1;
                if (idn > 99)
                {
                    tmpID = "OL" + idn;
                }
                else if (idn > 9)
                {
                    tmpID = "OL0" + idn;
                }
                else
                {
                    tmpID = "OL00" + idn;
                }
            }
        }
        closeConnection();
        return tmpID;
    }

    private List<string> getList(string atr)
    {
        List<string> lst = new List<string>();
        openConnection();

        command = new MySqlCommand("select " + atr + " from outlet order by otl_id",
connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            lst.Add(mdr.GetValue(0).ToString());
        }

        closeConnection();
        return lst;
    }

    private List<double> getCoordinate(string atr)
    {
        List<double> lst = new List<double>();
        openConnection();

        command = new MySqlCommand("select " + atr + " from outlet order by otl_id",
connection);
        mdr = command.ExecuteReader();
        while (mdr.Read())
        {
            lst.Add(double.Parse(mdr.GetValue(0).ToString(),
System.Globalization.CultureInfo.InvariantCulture));
        }
        closeConnection();
        return lst;
    }

    public string checkOutlet(double posLat, double posLong)
    {
```

```
            string otlID = "outofrange!";
            listOutlet = getList("otl_id");
            listLatitude = getCoordinate("otl_lat");
            listLongitude = getCoordinate("otl_long");

            for (int i = 0; i < listOutlet.Count; i++)
            {
               double dataLat = Math.Abs(listLatitude[i]);
               double dataLong = Math.Abs(listLongitude[i]);

               double latitudeMax = Math.Abs(dataLat + 0.0005);
               double latitudeMin = Math.Abs(dataLat - 0.0005);
               double longitudeMax = Math.Abs(dataLat + 0.0005);
               double longitudeMin = Math.Abs(dataLat - 0.0005);

               if (Math.Abs(Math.Round(posLat, 6)) >= latitudeMin &&
Math.Abs(Math.Round(posLong, 6)) >= longitudeMin)
               {
                  if(Math.Abs(Math.Round(posLat, 6)) <= latitudeMax &&
Math.Abs(Math.Round(posLat, 6)) <= longitudeMax)
                  {
                     otlID = listOutlet[i];
                  }
               }
               else
               {
                  return otlID;
               }

            }
            return otlID;
         }
      }
   }
```

Class Program

```
using System;
using System.Windows.Forms;

namespace Thesis
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new fAttendance());
      }
   }
}
```

Class Recognizer

```
using Emgu.CV;
using Emgu.CV.Face;
using Emgu.CV.Structure;
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Thesis
{
   class Recognizer
   {
      public LBPHFaceRecognizer faceRecognizer = new LBPHFaceRecognizer(1, 8, 8,
8, 150);
      /*
       * PERUBAHAN PARAMETER :
       * RADIUS 1, NEIGHBOR 8, THRESHOLD 50
       * RADIUS 1, NEIGHBOR 8, THRESHOLD 75
       * RADIUS 1, NEIGHBOR 8, THRESHOLD 100
       * RADIUS 1, NEIGHBOR 8, THRESHOLD 150
       * RADIUS 1, NEIGHBOR 8, THRESHOLD 200
      */
      public String YMLPath { get; set; } = "D:/Project/Visual
Studio/Thesis/Thesis/bin/Debug/trainingData.yml";


      public void trainFace(List<Image<Gray, byte>> face, List<int> id)
      {
         try
         {
            faceRecognizer.Train(face.ToArray(), id.ToArray());
            faceRecognizer.Save(YMLPath);
         }
         catch(Exception ex)
         {
            MessageBox.Show("error : " + ex);
         }
      }

      public void updateFace(List<Image<Gray, byte>> face, List<int> id)
      {
         try
         {
            faceRecognizer.Update(face.ToArray(), id.ToArray());
            faceRecognizer.Save(YMLPath);
         }
         catch (Exception ex)
         {
            MessageBox.Show("error : " + ex);
         }
      }

      public int recognizeFace(Image<Gray, byte> face)
      {
         int res = 0;
         try
         {
            faceRecognizer.Load(YMLPath);
            var result = faceRecognizer.Predict(face);
```

```
            res = Convert.ToInt32(result.Label.ToString());
        }
        catch(Exception ex)
        {
            MessageBox.Show("Error : " + ex);
        }
        return res;
    }
  }
}
```

Class User

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Thesis
{
   class User
   {
      public string id { get; set; }
      public string name { get; set; }
      public string dob { get; set; }
      public string address { get; set; }
      public string phone { get; set; }
      public string department { get; set; }
      public string password { get; set; }

      public MySqlConnection connection = new MySqlConnection
         ("datasource=localhost;port=3306;Initial
Catalog='thesis';username=root;password=");


      public void openConnection()
      {
         if (connection.State == ConnectionState.Closed)
         {
            connection.Open();
         }
      }

      public void closeConnection()
      {
         if (connection.State == ConnectionState.Open)
         {
            connection.Close();
         }
      }

      public string insert()
      {
         string status = "";
         try
         {
```

```
            MySqlCommand command;
            string query = "insert into employee(emp_id, emp_name, emp_dob, emp_address, emp_phone, emp_department, emp_password) values('" + id + "','" + name + "','" + dob + "','" + address + "','" + phone + "','" + department + "','" + password + "')";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
               return status = "saved";
            }
            else
            {
               return status = "failed";
            }

         }
         catch (Exception ex)
         {
            return status = "Error : " + ex.Message;
         }
         finally
         {
            closeConnection();
         }
      }

      public string update(string sid)
      {
         string status = "";
         try
         {
            MySqlCommand command;
            string query = "update employee set emp_name = '" + name + "', emp_dob = '" + dob + "', emp_address = '" + address + "', emp_phone = '" + phone + "', emp_department = '" + department + "', emp_password = '" + password + "' where emp_id = '" + sid + "'";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
               return status = "updated";
            }
            else
            {
               return status = "failed";
            }

         }
         catch (Exception ex)
         {
            return status = "Error : " + ex.Message;
         }
         finally
         {
            closeConnection();
```

```
        }
    }

    public string delete(string sid)
    {
        string status = "";
        try
        {
            MySqlCommand command;
            string query = "delete from employee where emp_id = '" + sid + "'";
            command = new MySqlCommand(query, connection);

            openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
                return status = "deleted";
            }
            else
            {
                return status = "failed";
            }

        }
        catch (Exception ex)
        {
            return status = "Error : " + ex.Message;
        }
        finally
        {
            closeConnection();
        }
    }

    public void loadData(string sid)
    {
        try
        {
            MySqlDataReader mdr;
            string query = "select emp_id, emp_name, emp_dob, emp_address,
emp_phone, emp_department, emp_password from employee where emp_id = '" + sid +
"'";
            MySqlCommand command = new MySqlCommand(query, connection);

            openConnection();
            mdr = command.ExecuteReader();

            while (mdr.Read())
            {
                id = mdr[0].ToString();
                name = mdr[1].ToString();
                dob = mdr[2].ToString();
                address = mdr[3].ToString();
                phone = mdr[4].ToString();
                department = mdr[5].ToString();
                password = mdr[6].ToString();
            }
        }
        catch (Exception ex)
```

```
            {
                return;
            }
            finally
            {
                closeConnection();
            }
        }

        public string IDcheck(string sid)
        {
            string status = "";
            openConnection();

            MySqlCommand command;
            MySqlDataReader mdr;
            command = new MySqlCommand("select count(emp_id) from employee where
emp_id='" + sid + "'", connection);
            mdr = command.ExecuteReader();
            while (mdr.Read())
            {
                int count = Convert.ToInt32(mdr[0].ToString());
                if (count == 0)
                {
                    status = "insert";
                }
                else
                {
                    status = "update";
                }
            }
            closeConnection();
            return status;
        }

        public string createID()
        {
            string tmpID = "", tmpDate = "";
            openConnection();
            MySqlCommand command;
            MySqlDataReader mdr;
            tmpDate = DateTime.Now.ToString("yyyy") + DateTime.Now.ToString("MM");

            command = new MySqlCommand("select max(emp_id) from employee",
connection);
            mdr = command.ExecuteReader();
            while (mdr.Read())
            {
                if (mdr.IsDBNull(0))
                {
                    tmpID = tmpDate + "0001";
                }
                else
                {
                    int max = Convert.ToInt32(mdr[0].ToString().Substring(6, 4));
                    int idn = max + 1;
                    if(idn > 999)
                    {
```

```
                tmpID = tmpDate + idn;
            }
            else if (idn > 99)
            {
                tmpID = tmpDate + "0" + idn;
            }
            else if (idn > 9)
            {
                tmpID = tmpDate + "00" + idn;
            }
            else
            {
                tmpID = tmpDate + "000" + idn;
            }
        }
    }
    closeConnection();
    return tmpID;
  }
 }
}
```

Class fAttendance : Form

```
using Emgu.CV;
using Emgu.CV.Structure;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Thesis
{
   public partial class fAttendance : Form
   {
      public Capture webcam { get; set; }
      public CascadeClassifier cascadeClassifier { get; set; }
      public Mat frame { get; set; }
      string outlet = "", stat = "";
      double latitude = 0.0, longitude = 0.0;
      Timer tmr = null;

      public List<string> listOutlet = new List<string>();
      public List<double> listLatitude = new List<double>();
      public List<double> listLongitude = new List<double>();
      public List<Image<Gray, byte>> Faces { get; set; }
      public List<int> IDs { get; set; }

      User usr = new User();
      Attendance att = new Attendance();
      Face fac = new Face();
      Recognizer rec = new Recognizer();

      public fAttendance()
      {
         InitializeComponent();
      }
```

```
private void preStart()
{
   StartTimer();
   preLearnFace();
   checkCoordinate();
   activateWebcam();

   txtPassword.PasswordChar = '*';
   imgCamera.Enabled = false;
}

private void preLearnFace()
{
   Faces = new List<Image<Gray, byte>>();
   IDs = new List<int>();

   fac.loadFaces();
   if (fac.faces.Count != 0)
   {
      Faces.AddRange(fac.faces);
      IDs.AddRange(fac.lists);
   }
   rec.trainFace(Faces, IDs);
}

private void btnLogin_Click(object sender, EventArgs e)
{
   string username = txtUsername.Text;
   string password = txtPassword.Text;

   usr.loadData(username);

   if (password == usr.password && usr.department == "HRD")
   {
      clearForm();
      deactivateWebcam();
      fMenu adm = new fMenu();
      adm.ShowDialog();
   }
   else if (password == usr.password && usr.department!= "HRD")
   {
      MessageBox.Show("Anda tidak mempunyai akses!");
   }
   else
   {
      MessageBox.Show("Username/password salah!");
   }
}

private void btnExit_Click(object sender, EventArgs e)
{
   clearForm();
   this.Close();
}

private void txtPassword_KeyDown(object sender, KeyEventArgs e)
{
   string username = txtUsername.Text;
```

```
        string password = txtPassword.Text;
        string dtnow = Convert.ToDateTime(DateTime.Now).ToString("yyyy-MM-dd");

        getOutlet(username);
        usr.loadData(username);

        if (e.KeyCode == Keys.Enter)
        {
           if (password == usr.password)
           {
              if (DateTime.Now > DateTime.Parse("06:00:00 AM") && DateTime.Now <
DateTime.Parse("05:00:00 PM"))
              {
                 stat = "CHECK-IN";
                 if (att.checkAttendanceIn(username, dtnow) == "checked")
                 {
                    MessageBox.Show(usr.name + " sudah melakukan absensi masuk!");
                 }
                 else
                 {
                    saveData(username);
                 }

                 if (usr.department == "HRD")
                 {
                    if (MessageBox.Show("", "Apakah Anda ingin mengakses menu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
                    {
                       clearForm();
                       deactivateWebcam();
                       fMenu adm = new fMenu();
                       adm.ShowDialog();
                    }
                    else
                    {
                       clearForm();
                       return;
                    }
                 }

              }
              else if (DateTime.Now >= DateTime.Parse("05:00:00 PM") &&
DateTime.Now < DateTime.Parse("10:00:00 PM"))
              {
                 stat = "CHECK-OUT";
                 if (att.checkAttendanceOut(username, dtnow) == "failed")
                 {
                    saveData(username);
                 }
                 else
                 {
                    MessageBox.Show(usr.name + " sudah melakukan absensi keluar!");
                 }

                 if (usr.department == "HRD")
                 {
                    if (MessageBox.Show("", "Apakah Anda ingin mengakses menu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
```

```
                    {
                        clearForm();
                        deactivateWebcam();
                        fMenu adm = new fMenu();
                        adm.ShowDialog();
                    }
                    else
                    {
                        clearForm();
                        return;
                    }
                }
            }
        }
        else
        {
            MessageBox.Show("Username/Password salah!");
        }
    }
}

private void btnAbsen_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text;
    string password = txtPassword.Text;

    verifyUsernamePassword(username, password);
}

private void verifyUsernamePassword(string username, string password)
{

    string dtnow = Convert.ToDateTime(DateTime.Now).ToString("yyyy-MM-dd");
    getOutlet(username);
    usr.loadData(username);

    if (username == usr.id && password == usr.password)
    {
        if (DateTime.Now > DateTime.Parse("05:00:00 AM") && DateTime.Now <
DateTime.Parse("05:00:00 PM"))
        {
            stat = "CHECK-IN";
            if (att.checkAttendanceIn(username, dtnow) == "checked")
            {
                MessageBox.Show(usr.name + " sudah melakukan absensi masuk!");
            }
            else
            {
                saveData(username);
            }
        }
        else if (DateTime.Now >= DateTime.Parse("05:00:00 PM") && DateTime.Now <
DateTime.Parse("10:00:00 PM"))
        {
            stat = "CHECK-OUT";
            if (att.checkAttendanceOut(username, dtnow) == "checked")
            {
                MessageBox.Show(usr.name + " sudah melakukan absensi keluar!");
```

```
                }
                else
                {
                    saveData(username);
                }
            }
            if (usr.department == "HRD")
            {
                if (MessageBox.Show("", "Apakah Anda ingin mengakses menu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    clearForm();
                    deactivateWebcam();
                    fMenu adm = new fMenu();
                    adm.ShowDialog();
                }
                else
                {
                    clearForm();
                    return;
                }
            }
        }
        else
        {
            MessageBox.Show("Username/Password salah!");
        }
    }

    private void deactivateWebcam()
    {
        webcam.Dispose();
        imgCamera.Image.Dispose();
        imgCamera.Update();
    }

    private void activateWebcam()
    {
        cascadeClassifier = new CascadeClassifier("D:/Project/Visual
Studio/Thesis/Thesis/haarcascade_frontalface_alt2.xml");
        frame = new Mat();

        if (webcam == null)
        {
            webcam = new Capture();
        }

        webcam.ImageGrabbed += Webcam_ImageGrabbed;
        webcam.Start();
    }

    private void Webcam_ImageGrabbed(object sender, EventArgs e)
    {
        webcam.Retrieve(frame);
        var imageFrame = frame.ToImage<Bgr,
byte>().Flip(Emgu.CV.CvEnum.FlipType.Horizontal);

        if (imageFrame != null)
```

```
        {
            Rectangle[] faces = cascadeClassifier.DetectMultiScale(imageFrame, 1.2, 10);

            foreach (var face in faces)
            {
                imageFrame.Draw(face, new Bgr(Color.AntiqueWhite), 2);
                var grayFace = imageFrame.Copy(face).Convert<Gray, byte>().Resize(120,
120, Emgu.CV.CvEnum.Inter.Cubic);

                string sid = rec.recognizeFace(grayFace).ToString();
                usr.loadData(sid);

                if (sid == "-1" || sid == "0")
                {
                    var w = new Form() { Size = new Size(0, 0) };
                    Task.Delay(TimeSpan.FromSeconds(2.0)).ContinueWith((t) => w.Close(),
TaskScheduler.FromCurrentSynchronizationContext());
                    MessageBox.Show(w, "Wajah tidak dikenal!", "Attendance");
                }
                else
                {
                    drawNotification(sid, usr.name, outlet);
                }

            }
            imgCamera.Image = imageFrame;
        }
    }

    private void drawNotification(string sid, string name, string otlId)
    {
        getOutlet(sid);
        string dtnow = Convert.ToDateTime(DateTime.Now).ToString("yyyy-MM-dd");

        if (DateTime.Now > DateTime.Parse("05:00:00 AM") && DateTime.Now <
DateTime.Parse("05:00:00 PM"))
        {
            stat = "CHECK-IN";
            if (att.checkAttendanceIn(sid, dtnow) == "checked")
            {
                var w = new Form() { Size = new Size(0, 0) };
                Task.Delay(TimeSpan.FromSeconds(2.0)).ContinueWith((t) => w.Close(),
TaskScheduler.FromCurrentSynchronizationContext());
                MessageBox.Show(w, name + " sudah melakukan absensi masuk!",
"Attendance");
            }
            else
            {
                saveData(sid);

            }
        }
        else if (DateTime.Now >= DateTime.Parse("05:00:00 PM") && DateTime.Now <
DateTime.Parse("10:00:00 PM"))
        {
            stat = "CHECK-OUT";
            if (att.checkAttendanceOut(sid, dtnow) == "failed")
            {
```

```
                saveData(sid);
            }
            else
            {
                var w = new Form() { Size = new Size(0, 0) };
                Task.Delay(TimeSpan.FromSeconds(2.0)).ContinueWith((t) => w.Close(),
TaskScheduler.FromCurrentSynchronizationContext());
                MessageBox.Show(w, name + " sudah melakukan absensi keluar!",
"Attendance");
            }
        }
    }

    private void saveData(string sid)
    {
        att.emp_id = sid;
        att.otl_id = outlet;
        att.date = Convert.ToDateTime(DateTime.Now).ToString("yyyy-MM-dd");
        att.time = Convert.ToDateTime(DateTime.Now).ToString("hh:mm:ss");
        att.stat = stat;

        usr.loadData(sid);

        if (outlet != "outofrange")
        {
            if (att.insert() == "saved")
            {
                var w = new Form() { Size = new Size(0, 0) };
                Task.Delay(TimeSpan.FromSeconds(2.0)).ContinueWith((t) => w.Close(),
TaskScheduler.FromCurrentSynchronizationContext());
                MessageBox.Show(w, "Result : " + sid + Environment.NewLine +
                    "Face : " + usr.name + Environment.NewLine +
                    "Outlet : " + outlet + Environment.NewLine +
                    "Time : " + DateTime.Now.ToString("HH:mm:ss") + Environment.NewLine
+
                    "Date : " + DateTime.Now.ToString("dd/MM/yyyy") +
Environment.NewLine +
                    "ABSENSI BERHASIL", "Attendance");
            }
            else
            {
                MessageBox.Show("ABSENSI GAGAL : " + att.insert());
            }
        }
        else
        {
            MessageBox.Show("Anda berada di luar jangkauan!");
        }
    }

    private void StartTimer()
    {
        tmr = new System.Windows.Forms.Timer();
        tmr.Interval = 1000;
        tmr.Tick += new EventHandler(tmr_Tick);
        tmr.Enabled = true;
    }
```

```
        void tmr_Tick(object sender, EventArgs e)
        {
            lblTime.Text = DateTime.Now.ToString("HH:mm:ss") + Environment.NewLine +
DateTime.Now.ToString("dd MMM yyyy");
        }

        private string getOutlet(string id)
        {
            Outlet otl = new Outlet();
            outlet = otl.checkOutlet(latitude, longitude);

            return outlet;
        }

        private void fAttendance_Load(object sender, EventArgs e)
        {
            preStart();
        }

        private void lblTime_Click(object sender, EventArgs e)
        {
            deactivateWebcam();
            Application.Restart();
            preStart();
        }

        private async void checkCoordinate()
        {
            try
            {
                var locator = new Windows.Devices.Geolocation.Geolocator();
                var location = await locator.GetGeopositionAsync();
                var position = location.Coordinate.Point.Position;

                latitude = position.Latitude;
                longitude = position.Longitude;
            }
            catch(Exception ex)
            {
                MessageBox.Show("Please TURN ON your GPS/LOCATION!");
                Close();
            }
        }

        private void clearForm()
        {
            txtUsername.Clear();
            txtPassword.Clear();
        }
    }
}
```

Class fMenu : Form

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Windows.Forms;
```

```
namespace Thesis
{
   public partial class fMenu : Form
   {
      public fMenu()
      {
         InitializeComponent();
         displayData();
      }

      private void displayData()
      {
         Attendance att = new Attendance();
         MySqlDataAdapter data = new MySqlDataAdapter("SELECT emp_id as ID, otl_id
as OUTLET, att_date as TANGGAL, att_time as WAKTU, att_stat as STATUS from
attendance", att.connection);
         DataTable tbl = new DataTable();
         data.Fill(tbl);
         tblAttendance.DataSource = tbl;
         tblAttendance.ReadOnly = true;
      }

      private void pegawaiToolStripMenuItem_Click(object sender, EventArgs e)
      {
         fUser emp = new fUser();
         emp.ShowDialog();
      }

      private void outletToolStripMenuItem1_Click(object sender, EventArgs e)
      {
         fOutlet otl = new fOutlet();
         otl.ShowDialog();
      }

      private void btnBack_Click(object sender, EventArgs e)
      {
         Close();
      }
   }
}
```

Class fOutlet : Form

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Globalization;
using System.Windows.Forms;

namespace Thesis
{
   public partial class fOutlet : Form
   {
      Outlet otl = new Outlet();
      public fOutlet()
      {
         InitializeComponent();
         preStart();
      }
```

```
        private void preStart()
        {
            displayData();
            txtAddress.Enabled = false;
            txtLat.Enabled = false;
            txtLong.Enabled = false;
            dtpDob.Enabled = false;
            btnCoordinate.Enabled = false;
        }

        private void displayData()
        {
            MySqlDataAdapter data = new MySqlDataAdapter("SELECT otl_id as ID,
otl_address as ALAMAT from outlet", otl.connection);
            DataTable tbl = new DataTable();
            data.Fill(tbl);
            tblOutlet.DataSource = tbl;
            tblOutlet.ReadOnly = true;
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            if(otl.delete(lblId.Text) == "delete")
            {
                MessageBox.Show("Outlet berhasil dihapus");
            }
            else
            {
                MessageBox.Show("Gagal menghapus outlet!");
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            clearContent();
            lblId.Text = otl.createID();
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            clearContent();
            this.Close();
        }

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            otl.id = lblId.Text;
            otl.address = txtAddress.Text.ToUpper();
            otl.dob = Convert.ToDateTime(dtpDob.Value).ToString("yyyy-MM-dd");
            otl.latitude = Convert.ToDouble(txtLat.Text);
            otl.longitude = Convert.ToDouble(txtLong.Text);

            string status = otl.IDcheck(lblId.Text);
            if (status == "insert")
            {
                if (lblId.Text != "")
                {
```

```
                if (otl.insert() == "saved")
                {
                    MessageBox.Show("Outlet berhasil ditambahkan!");
                }
                else
                {
                    MessageBox.Show("Gagal menambahkan outlet! Kesalahan : " +
otl.insert());
                }
            }
            else
            {
                MessageBox.Show("Isi semua data!");
            }
        }
        else if(status == "update")
        {
            if (otl.update(lblId.Text) == "update")
            {
                MessageBox.Show("Outlet berhasil diperbaharui!");
            }
            else
            {
                MessageBox.Show("Gagal memperbaharui outlet! Kesalahan : " +
otl.insert());
            }
        }
        displayData();
    }

    private async void btnCoordinate_Click(object sender, EventArgs e)
    {
        var locator = new Windows.Devices.Geolocation.Geolocator();
        var location = await locator.GetGeopositionAsync();
        var position = location.Coordinate.Point.Position;
        double latitude = position.Latitude;
        double longitude = position.Longitude;

        txtLat.Text = latitude.ToString();
        txtLong.Text = longitude.ToString();
    }

    private void tblOutlet_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        string id = "";

        try
        {
            clearContent();
            id = tblOutlet.SelectedCells[0].Value.ToString();
            otl.loadData(id.ToString());

            lblId.Text = otl.id;
            txtAddress.Text = otl.address;
            dtpDob.Value = DateTime.ParseExact(otl.dob, "yyyy-MM-dd",
CultureInfo.InvariantCulture);
            txtLat.Text = otl.latitude.ToString();
            txtLong.Text = otl.longitude.ToString();
```

```
            }
            catch(Exception ex)
            {
                return;
            }
        }

        private void clearContent()
        {
            lblId.Text = "ID";
            txtAddress.Clear();
            dtpDob.Value = DateTime.Today;
            txtLat.Clear();
            txtLong.Clear();

            txtAddress.Enabled = true;
            txtLat.Enabled = true;
            txtLong.Enabled = true;
            dtpDob.Enabled = true;
            btnCoordinate.Enabled = true;
        }
    }
}
```

Class fUser : Form

```
using Emgu.CV;
using Emgu.CV.Structure;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Windows.Forms;

namespace Thesis
{
    public partial class fUser : Form
    {
        User usr = new User();
        Recognizer rec = new Recognizer();
        Face fac = new Face();

        public Capture webcam { get; set; }
        public CascadeClassifier cascadeClassifier { get; set; }
        public Mat frame { get; set; }
        public List<Image<Gray, byte>> Faces { get; set; }
        public List<int> IDs { get; set; }
        public Timer timer { get; set; }

        public int TimerCounter { get; set; } = 0;
        public int TimeLimit { get; set; } = 20;

        public fUser()
        {
            InitializeComponent();
```

```
            preStart();
        }

        private void preStart()
        {
            displayData();
            ComboBox.CheckForIllegalCrossThreadCalls = false;
            cmbDepartment.FlatStyle = FlatStyle.Popup;
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            Faces = new List<Image<Gray, byte>>();
            IDs = new List<int>();

            fac.loadFaces();
            if(fac.faces.Count != 0)
            {
                Faces.AddRange(fac.faces);
                IDs.AddRange(fac.lists);
            }

            clearContent();
            lblID.Text = usr.createID();
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            DialogResult dialogResult = MessageBox.Show("Hapus Data ?", "Hapus Data",
MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                if (usr.delete(lblID.Text) == "deleted")
                {
                    fac.delete(lblID.Text);
                    //delete photo file
                    string faceFolder = @"D:/Project/Visual
Studio/Thesis/Thesis/bin/Debug/TrainedFaces/";
                    string faceFile = lblID.Text + ".bmp";
                    if (File.Exists(Path.Combine(faceFolder, faceFile)))
                    {
                        File.Delete(Path.Combine(faceFolder, faceFile));
                    }

                    MessageBox.Show("Data pegawai berhasil dihapus.");
                }
                else
                {
                    MessageBox.Show("Gagal menghapus data pegawai!");
                }
            }
            else if (dialogResult == DialogResult.No)
            {
                //do something else
            }
        }

        private void btnCancel_Click(object sender, EventArgs e)
```

```
        {
            clearContent();
            this.Close();
        }

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            string status = usr.IDcheck(lblID.Text);
            if(status == "insert")
            {
                activateWebcam();
            }
            else if(status == "update")
            {
                updateUserOnly();
            }
            else
            {
                MessageBox.Show("Error when checking!");
            }
        }

        private void tblEmployee_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            string id = "";
            try
            {
                clearContent();
                id = tblEmployee.SelectedCells[0].Value.ToString();
                usr.loadData(id.ToString());

                lblID.Text = usr.id;
                txtName.Text = usr.name;
                string dob = usr.dob;
                dtpDob.Value = DateTime.ParseExact(dob, "yyyy-MM-dd",
CultureInfo.InvariantCulture);
                txtAddress.Text = usr.address;
                txtPhone.Text = usr.phone;
                cmbDepartment.SelectedIndex =
cmbDepartment.Items.IndexOf(usr.department);
            }
            catch (Exception ex)
            {
                return;
            }
        }

        private void activateWebcam()
        {
            cascadeClassifier = new CascadeClassifier("D:/Project/Visual
Studio/Thesis/Thesis/haarcascade_frontalface_alt2.xml");
            frame = new Mat();

            if (webcam == null)
            {
                webcam = new Capture();
                webcam.ImageGrabbed += Webcam_ImageGrabbed;
                webcam.Start();
```

```
        }
    }

    private void Webcam_ImageGrabbed(object sender, EventArgs e)
    {
        webcam.Retrieve(frame);
        var imageFrame = frame.ToImage<Bgr,
byte>().Flip(Emgu.CV.CvEnum.FlipType.Horizontal);

        if (imageFrame != null)
        {
            Rectangle[] faces = cascadeClassifier.DetectMultiScale(imageFrame, 1.2, 10);

            if (faces.Count() > 0)
            {
                imageFrame.Draw(faces[0], new Bgr(Color.AntiqueWhite), 2);
                var grayFace = imageFrame.Copy(faces[0]).Convert<Gray,
byte>().Resize(120, 120, Emgu.CV.CvEnum.Inter.Cubic);
                imgFace.Image = grayFace;

                Faces.Add(grayFace);
                var tmpID = Convert.ToInt32(lblID.Text);
                IDs.Add(tmpID);

                rec.trainFace(Faces, IDs);
                saveData();
            }
        }
    }

    private void updateUserOnly()
    {
        usr.id = lblID.Text;
        usr.name = txtName.Text.ToUpper();
        usr.dob = Convert.ToDateTime(dtpDob.Value).ToString("yyyy-MM-dd");
        usr.address = txtAddress.Text.ToUpper();
        usr.phone = txtPhone.Text;
        usr.department = cmbDepartment.SelectedItem.ToString().ToUpper();
        usr.password = dtpDob.Value.ToString("ddMMyyyy");

        string status = usr.IDcheck(lblID.Text);
        if (status == "update")
        {
            if (usr.update(lblID.Text) == "updated")
            {
                MessageBox.Show("Data pegawai berhasil diperbaharui!");
                this.Close();
            }
            else
            {
                MessageBox.Show("Gagal memperbaharui data pegawai! Kesalahan : " +
usr.update(lblID.Text));
            }
        }
    }

    private void saveData()
    {
```

```
        Byte[] file;
        var faceToSave = new Image<Gray, byte>(imgFace.Image.Bitmap);

        //save employee data
        usr.id = lblID.Text;
        usr.name = txtName.Text.ToUpper();
        usr.dob = Convert.ToDateTime(dtpDob.Value).ToString("yyyy-MM-dd");
        usr.address = txtAddress.Text.ToUpper();
        usr.phone = txtPhone.Text;
        usr.department = cmbDepartment.SelectedItem.ToString().ToUpper();
        usr.password = dtpDob.Value.ToString("ddMMyyyy");

        //save face data
        var filePath = String.Format("D:/Project/Visual
Studio/Thesis/Thesis/bin/Debug/TrainedFaces/{0}.bmp", lblID.Text);
        faceToSave.ToBitmap().Save(filePath);
        using (var stream = new FileStream(filePath, FileMode.Open, FileAccess.Read))
        {
            using (var reader = new BinaryReader(stream))
            {
                file = reader.ReadBytes((int)stream.Length);
            }
        }
        fac.id = lblID.Text;
        fac.faceBlob = file;
        fac.address = filePath;

        string status = usr.IDcheck(lblID.Text);
        if (status == "insert" && fac.insert() == "saved")
        {
            if (lblID.Text != "")
            {
                if (usr.insert() == "saved")
                {
                    MessageBox.Show("Data pegawai berhasil ditambahkan!");
                    webcam.Stop();
                    webcam.Dispose();
                    this.Close();
                }
                else
                {
                    MessageBox.Show("Gagal menambahkan data pegawai! Kesalahan : " +
usr.insert());
                }
            }
            else
            {
                MessageBox.Show("Isi semua data!");
            }
        }
        else if (status == "update")
        {
            if (usr.update(lblID.Text) == "updated")
            {
                MessageBox.Show("Data pegawai berhasil diperbaharui!");
                this.Close();
            }
            else
```

```
            {
                MessageBox.Show("Gagal memperbaharui data pegawai! Kesalahan : " +
usr.update(lblID.Text));
            }
        }
    }

    private void displayData()
    {
        MySqlDataAdapter data = new MySqlDataAdapter("SELECT emp_id as ID,
emp_name as NAMA, emp_dob as TANGGAL from employee", usr.connection);
        DataTable tbl = new DataTable();
        data.Fill(tbl);
        tblEmployee.DataSource = tbl;
        tblEmployee.ReadOnly = true;

        clearContent();

        txtName.Enabled = false;
        dtpDob.Enabled = false;
        txtAddress.Enabled = false;
        txtPhone.Enabled = false;
        cmbDepartment.Enabled = false;
    }

    private void clearContent()
    {
        lblID.Text = "ID";
        txtName.Text = "";
        dtpDob.Value = DateTime.Now;
        txtAddress.Text = "";
        txtPhone.Text = "";
        cmbDepartment.SelectedIndex = cmbDepartment.Items.IndexOf("GURU");

        txtName.Enabled = true;
        dtpDob.Enabled = true;
        txtAddress.Enabled = true;
        txtPhone.Enabled = true;
        cmbDepartment.Enabled = true;
    }

    private void txtPhone_KeyPress(object sender, KeyPressEventArgs e)
    {
        e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);
    }
  }
}
```