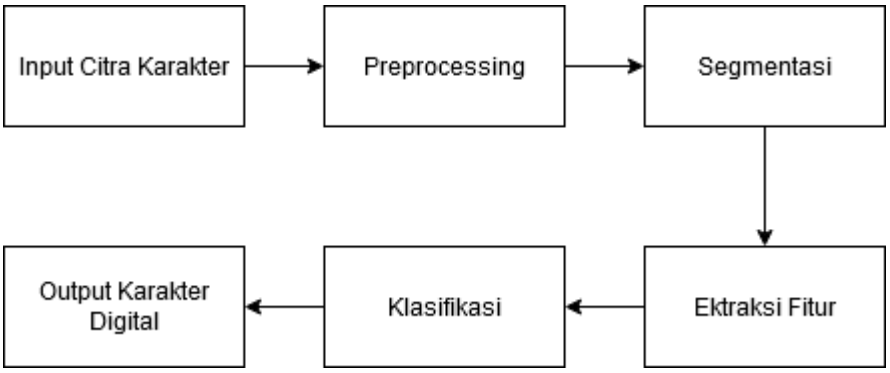
BAB 2
TINJAUAN PUSTAKA

2.1. Optical Character Recognition

Optical Character Recognition adalah teknik mengidentifikasi secara otomatis karakter yang berbeda dari gambar rekaman juga memberikan pengenalan alfanumerik lengkap dari karakter yang dicetak atau tulisan tangan, teks angka, huruf, dan simbol ke dalam tata letak yang dapat diproses oleh komputer termasuk ASCII[12]. OCR banyak digunakan untuk mengubah dokumen atau buku menjadi *file* elektronik, sehingga memungkinkan untuk mengedit teks, mencari kata atau frase, menyimpan, menampilkan atau mencetak salinan dari teks hasil scanning, dan sebagainya[13].

Secara umum tahap-tahap dari OCR terdiri dari *preprocessing*, segmentasi, ekstraksi ciri, klasifikasi. Diagram OCR ditunjukkan pada gambar 2.1 berikut ini:



Gambar 2.1 Diagram OCR

2.2. Preprocessing

Preprocessing merupakan tahap awal yang harus dilakukan dalam proses OCR. Inti dari *preprocessing* adalah tahap yang mengubah dokumen yang dicetak menjadi sekumpulan karakter yang selanjutnya akan masuk proses ekstraksi ciri sehingga siap untuk diklasifikasikan[14]. Tahap ini sangat menentukan hasil dari

proses pengenalan pola. Beberapa proses yang pada tahap *preprocessing* antara lain sebagai berikut:

2.2.1. Grayscale

Untuk memudahkan pekerjan-pekerjaan pada proses selanjutnya dalam pengolahan citra, pada tahap pertama dilakukan proses *Grayscaling*. Proses ini mengubah citra RGB menjadi citra 8 bit keabuan. Perubahan citra RGB menjadi citra keabuan dengan menggunakan Persamaan 1 sebagai berikut:

$$Gray = 0{,}2989*R + 0{,}5870*G + 0{,}1140*B \quad (1)$$

Dimana:

R : Nilai intensitas komponen warna merah
G : Nilai intensitas komponen warna hijau
B : Nilai intensitas komponen warna biru

Proses ini dilakukan dengan melakukan setiap pixel warna yang memliki tiga komponen warna (RGB). Dengan persamaan diatas maka didapatkan suatu warna baru yang memiliki suatu komponen warna dengan intensitas antara 0 sampai 225 (hal ini disebabkan karena citra bernilai 8 bit sehingga terdapat 28 warna atau 225)[15].

2.2.2. Threshold

Threshold merupakan proses untuk mengubah suatu citra menjadi citra yang hanya terdiri dari warna hitam dan putih. Pengkonversian citra *grayscale* menjadi citra biner dilakukan untuk alasan-alasan sebagai berikut[16]:

1. Untuk mengidentifikasi keberadaan objek, yang direpresentasikan sebagai daerah (*region*) di dalam citra. Misalnya ingin memisahkan objek dengan latar belakangnya. Piksel-piksel objek dinyatakan dengan nilai 1 sedangkan piksel lainnya dengan 0.
2. Untuk memfokuskan pada analisis bentuk morfologi, yang dalam hal ini piksel tidak terlalu penting dibandingkan bentuknya.

3. Untuk menampilkan citra pada piranti keluaran yang hanya mempunyai resolusi satu bit, yaitu piranti penampil biner seperti pencetak (*printer*).
4. Mengkonversi citra yang telah ditingkatkan kualitas tepinya ke penggambaran garis-garis tepi. *Threshold* dapat ditentukan melalui persamaan 2 berikut:

$$g(x,y) = \begin{cases} 1, & jika\ f(x,y) \leq T \\ 0, & jika\ f(x,y) > T \end{cases} \quad (2)$$

2.2.3. Segmentasi

Segmentasi citra merupakan suatu proses yang membagi citra ke dalam beberapa bagian, yaitu bagian yang diperlukan dan bagian yang tidak diperlukan oleh sistem. Segmentasi karakter perlu dilakukan sebelum proses pengenalan karakter. Biasanya citra yang akan disegmentasi sudah berupa citra biner (hitam-putih) untuk memudahkan pemisahan karakter[16].

2.2.3.1. *Profile Projection*

Profile projection merupakan salah satu teknik untuk segmentasi karakter pada citra. *Profile Projection* terdiri atas 2 bagian yakni horizontal dan vertikal. Horizontal proyeksi merupakan segmentasi untuk memisahkan setiap baris pada sebuah citra dengan menghitung dan mendapatkan area baris karakter, sedangkan vertikal proyeksi merupakan segmentasi untuk memisahkan setiap kolom dari segmentasi baris (horizontal) yang telah didapatkan sehingga menghasilkan masing masing karakter yeng telah disegmentasi. Horizontal proyeksi dapat didapatkan dengan menjumlahkan nilai piksel sepanjang garis horizontal (sumbu-x) untuk setiap baris (sumbu-y).[17]

2.2.3.2. *Maximally Stable Extremal Regions*

Maximally Stable Extremal Regions (MSER) adalah metode untuk mendeteksi gumpalan (*blob detection*) dalam sebuah citra. MSER didasarkan pada

ide dalam mengambil daerah yang hampir sama melalui berbagai nilai ambang (*threshold*)[18], [19]. Semua piksel yang berada di bawah nilai ambang (*threshold*) yang diberikan adalah berwarna putih, selain itu berwarna hitam. Pada urutan citra *threshold* tersebut dengan *frame/t,* dapat dilihat terlebih dulu gambar hitam, kemudian bintik-bintik putih yang sesuai dengan intensitas lokal minima akan muncul, kemudian berkembang lebih besar. Bintik-bintik putih ini pada akhirnya akan bergabung, sampai seluruh gambar putih. Himpunan semua komponen terhubung secara berurutan adalah himpunan semua daerah *extremal*[18], [19]. Sebagai pilihan, dapat dilakukan pembentukan elips dengan proses *fitting*. *Frameelips* yang melekat pada MSER dengan proses *fitting* dilakukan pembentukan elips ke daerah citra yang dituju. Kemudian daerah-daerah tersebut akan disimpan sebagai fitur. Kata *extremal* mengacu pada sifat bahwa semua pixel dalam MSER memiliki intensitas baik yang lebih tinggi (daerah *extremal* yang terang) atau lebih rendah (daerah *extremal* yang gelap) dari semua pixel pada batas luarnya[19]. Tahapan yang akan dilakukan seperti *flowchart* dibawah ini:

Segmentasi MSER dimulai dengan memasukan citra keabuan kemudian menentukan parameter yang akan di gunakan pada proses ini seperti *MinArea*, *MaxArea*, *MaxVariation*, dan *Delta*[20].

1. *MinArea*, nilai parameter ini digunakan untuk menentukan batas ukuran area terkecil *Connected Component*. Nilai default dari *MinArea* adalah 30
2. *MaxArea*, nilai parameter ini digunakan untuk menentukan batas ukuran area terbesar *Connected Component*. Nilai default dari *MaxArea* adalah 14000.
3. *Delta*, nilai parameter ini digunakan untuk menentukan akan ada berapa banyak iterasi proses yang dilakukan. Nilai default dari *delta* adalah 2.
4. *MaxVariation*, nilai parameter ini digunakan untuk menentukan variasi maksimum yang dapat diterima atau nilai variasi yang hampir dekat dengan *MaxVariation* dan di anggap stabil. Nilai default dari *MaxVariation* adalah 0,25.

Proses selanjutnya adalah melakukan proses *threshold* yaitu proses mengkonveksikan citra *grayscale* menjadi citra biner. Citra biner hanya memiliki 2 nilai yaitu 0 dan 1. Nilai 0 sebagai warna hitam dan Nilai 1 sebagai warna putih. Proses *threshold* dapat dilakukan dengan menggunakan persamaan 2.

Sesudah melakukan *threshold* lakukan proses *connected component* dengan mencari objek karakter pada citra sertifikat. Proses selanjutnya membuat komponen *tree* untuk mengurutkan piksel sesuai nilai *threshold* dari awal hingga akhir. Cek apakah komponen *tree* setiap *threshold* satu hirarki. Jika komponen *tree* dari setiap *threshold* satu hirarki, hitung jumlah variasi dengan persamaan 3 sebagai berikut[21]:

$$Variasi_i = \frac{Area^{i+\Delta} - Area^i}{Area^i} \quad (3)$$

Dari hasil jumlah setiap variasi terdapat nilai paling minimum pada variasi ke-n, maka area tersebut dikatakan MSER.

2.2.4. Scaling

Hasil karakter setelah proses segmentasi memiliki ukuran yang tidak sama untuk masing-masing karakter sehingga diperlukan proses *scaling*. *Scaling* adalah proses mengubah ukuran citra, baik menambah atau mengurangi, menjadi ukuran yang ditentukan tanpa menghilangkan informasi penting dari citra tersebut. Dengan adanya proses normalisasi maka ukuran semua citra yang akan diproses menjadi seragam[22]. Metode yang digunakan adalah bukan metode khusus melainkan menggunakan perbandingan ukuran dari setiap citra hasil segmentasi dengan ukuran citra yang diinginkan. Rumus yang digunakan dalam proses *scaling* dapat dilihat pada persamaan 4 dan 5 berikut:

Menghitung nilai koordinat baris:

$$x = \frac{pb * pp}{pa} \quad (4)$$

Keterangan:

x = Nilai piksel baris baru

pb = Ukuran panjang matriks baru

pp = Posisi piksel baris

pa = Ukuran panjang matriks lama

Menghitung nilai koordinat kolom:

$$y = \frac{lb * lp}{la} \quad (5)$$

Keterangan:

y = Nilai piksel kolom baru

lb = Ukuran lebar matriks baru

lp = Posisi piksel kolom

la = Ukuran lebar matriks lama

2.3. Moment Invariant

Moment Invariant merupakan sebuah metode pengambilan ciri dari sebuah objek. Ciri yang diambil dapat berupa posisi, area, orientasi dan ciri lainnya. Metode ini dikenalkan oleh *Hu* pada tahun 1962. Persamaan dasar dari Momen suatu objek didefinisikan pada persamaan 6 berikut:

$$m_{ij} = \sum_{x} \sum_{y} x^i y^j a_{xy} \quad (6)$$

Keterangan:

x = panjang citra

y = lebar citra

a_{xy} = nilai intensitas citra pada titik x dan y

i, j = order momen

Selanjutnya momen pusat (central moment) μ adalah momen yang bersesuaian dengan pusat area. Moment pusat didefinisik an pada persamaan 7 sampai dengan 11 berikut[23]:

$$\mu_{ij} = \sum_{x}\sum_{y}(x - x')^{i}(y - y')^{j}a_{xy} \quad (7)$$

$$x' = \frac{m_{10}}{m_{00}} \; dan \; y' = \frac{m_{01}}{m_{00}} \quad (8)$$

$$m_{00} = \sum_{x}\sum_{y} a_{xy} \quad (9)$$

$$m_{10} = \sum_{x}\sum_{y} x.a_{xy} \quad (10)$$

$$m_{01} = \sum_{x}\sum_{y} y.a_{xy} \quad (11)$$

Keterangan:

x = panjang citra

y = lebar citra

a_{xy} = nilai intensitas citra pada titik x dan y

(x',y') = *centroid* citra dengan koordinat x' dan y'

μ_{ij} = *central moment*

Kemudian momen pusat yang telah dinormalisasi memiliki persamaan seperti pada persamaan 12 berikut:

$$\eta_{ij} = \frac{\mu_{ij}}{(\mu_{00})^{\lambda}} \; ; \; \lambda = \frac{(i+j)}{2} + 1 \quad (12)$$

Keterangan :

μ_{ij} = *central moment*

Hasil dari momen merupakan 7 Hasil dari momen yang sudah dinomalisasi, maka dihasilkan 7 *Hu moment invariants* yang dapat dilihat pada persamaan 13 sampai dengan 19 berikut:

$$\Phi_1 = \eta_{20} + \eta_{02} \tag{13}$$

$$\Phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}{}^2 \tag{14}$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{15}$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{16}$$

$$\Phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\{(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\} + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\{3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} \tag{17}$$

$$\Phi_6 = (\eta_{20} + \eta_{02})\{(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{18}$$

$$\Phi_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})\{(\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{30})^2\} + (3\eta_{21} - \eta_{03})(3\eta_{21} + \eta_{03})\{3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} \tag{19}$$

2.4. K-Support Vector Nearest Neighbor

K-SVNN merupakan metode hasil evolusi dari K-NN dengan cara mereduksi terlebih dahulu data latih sebelum melakukan klasifikasi yang bertujuan untuk mengurangi waktu komputasi. Dalam K-SVNN, data hasil ekstraksi ciri dibagi kedalam data uji dan data latih, yang untuk selanjutnya, data latih akan direduksi.

Data yang telah direduksi dengan KSVNN digunakan untuk pengklasifikasian data uji dengan algoritma KNN[3].

K-SVNN berusaha mengurangi data latih berdasarkan properti skor dan properti relevansi/derajat signifikansi pada setiap data latih berdasarkan prinsip K tetangga terdekat. Setiap data latih mempunyai kedua properti tersebut. Properti skor untuk setiap data latih ada 2, yaitu nilai kiri (LV) dan nilai kanan (RV). Nilai kiri untuk kelas yang sama sedangkan nilai kanan untuk kelas yang berbeda. Jumlah LV dan RV dari semua data latih sama dengan nilai NxK. Seperti dinyatakan pada persamaan 20 berikut:

$$\sum_{i=1}^{n} LV_1 + \sum_{i=1}^{n} RV_1 = N \; x \; K \tag{20}$$

Properti relevansi/derajat signifikansi adalah nilai yang menyatakan tingkat signifikansi data latih tersebut pada fungsi tujuan (daerah *hyperplane*). Nilainya dalam rentang 0 sampai 1. Semakin tinggi nilainya maka relevansinya untuk menjadi *support vector* juga semakin tinggi.

Data yang dijadikan support vector diseleksi oleh batas T yang ditentukan. Nilai T dalam rentang [0,1]. Semakin tinggi nilai T maka semakin sedikit *support vector* yang didapatkan karena hanya yang memiliki relevansi yang tinggi sajalah yang berhasil lolos. Nilai T = 0 berarti sekecil apapun nilai relevansi (diatas 0) akan tetap digunakan sebagai *support vector* (SV). Nilai 0 pada derajat signifikansi sebuah data latih berarti data latih tersebut harus dibuang (tidak digunakan sebagai SV). Nilai derajat signifikansi (SD) didapatkan dengan membagi LV terhadap RV atau RV terhadap LV sesuai syarat yang terpenuhi. Nilai SD dapat didapatkan menggunakan pumus pada persamaan 21 berikut[24]:

$$SD_i = \begin{cases} 0 & ,SV_i = RV_i = 0 \\ \dfrac{SV_i}{RV_i} & ,SV_i < RV_i \\ \dfrac{RV_i}{SV_i} & ,SV_i > RV_i \\ 1 & ,SV_i = RV_i \end{cases} \tag{21}$$

Data sampel terbagi menjadi data latih dan data uji. Selanjutnya, data latih direduksi dengan K-SVNN dimana nilai K=3. Nilai k dapat menentukan kinerja dari metode K-SVNN, semakin kecil nilai k maka *support vector* diperoleh semakin sedikit dan akurasi yang diperoleh semakin kecil, tetapi besarnya nilai k tidak dapat menjamin akurasi semakin besar[3]. Berikut merupakan algoritma dari K-SVNN:

1. Melakukan inisialisasi D merupakan set data latih, K sebagai jumlah tetangga terdekat, T merupakan *threshold Significant Degree*(SD), *Left Value*(LV) dan *Right Value*(RV) untuk semua data latih = 0.
2. Untuk setiap data latih $d_i \in$ D, lakukan langkah 3 hingga 5.
3. Menghitung jarak dari d_i ke data latih lain. untuk mendapatkan jarak digunakan dengan rumus Euclidian Distance menggunakan persamaan 22 berikut:

$$D_{x,y} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (22)$$

   Keterangan:
   x = data uji
   y = data sample
   n = jumlah dimensi data
4. Pilih d_t sebagai K tetangga terdekat.
5. Untuk setiap data dalam d_t, jika memiliki kelas yang sama, berikan nilai 1 pada LV_i, jika tidak berikan nilai 1 pada RV_i menggunakan persamaan 20 diatas.
6. Untuk setiap data d_i, hitung SD_i mengunakan persamaan 21 diatas.
7. Pilih data dengan $SD_i > T$, simpan dalam variabel.

2.5. *K-Nearest Neighbor* (K-NN)

K-Nearest Neighbor merupakan salah satu teknik klasifikasi yang popular digunakan dan terbukti menjadi algoritma yang sederhana dan efektif. Klasifikasi merupakan proses pengelompokkan objek yang memiliki karakteristik atau ciri yang sama ke dalam beberapa kelas[25]. Tujuan dari algoritma ini adalah mengklasifikasikan data baru berdasarkan jarak dari data latih yang sudah ada. Prinsip kerja *K-Nearest Neighbor* adalah mencari jarak terdekat antara data baru dengan K data terdekatnya dalam data pelatihan. Dekat atau jauhnya jarak data yang akan dievaluasi dengan K tetangga dapat dihitung berdasarkan rumus *Euclidian Distance* pada persamaan 22 diatas.

Kemudian dihitung total untuk setiap kelas pada K data terdekat. Kelas yang mempunyai jarak terdekat menang dan merupakan kelas untuk data yang diuji. Sehingga data uji akan dikategorikan sebagai kelas tersebut.

2.6. Sertifikat

Menurut Kamus Besar Bahasa Indonesia (KBBI) sertifikat merupakan tanda atau surat keterangan (pernyataan) tertulis atau tercetak dari orang yang berwenang yang dapat digunakan sebagai bukti pemilikan atau suatu kejadian. Sertifikat berfungsi sebagai bukti telah mengikuti suatu kegiatan seperti kegiatan pelatihan, seminar, dan sebagainya. Bagian-bagian pada sertifikat adalah sebagai berikut:

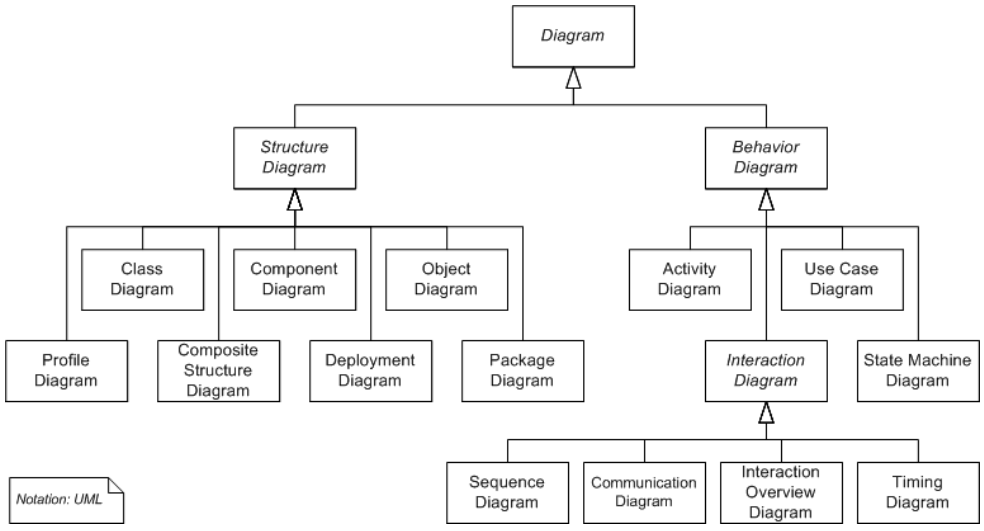
1. Logo sertifikat
2. Nama sertifikat
3. Nomor surat
4. Nama peserta
5. Waktu dan tempat
6. Pengesahan

2.7. UML

UML adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti

lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C[26].

Secara fundamental, UML berkaitan dengan penangkapan dan komunikasi pengetahuan. Konsep-konsep yang diterapkan di UML adalah satu model berisi informasi mengenai sistem (atau domain), model-model berisi elemen-elemen model seperti kelas-kelas, simpul-simpul, paket-paket, dan sebagainya. Satu diagram menunjukkan satu pandangan tertentu dari model. UML adalah meta model, yaitu UML mendefinisikan jenis-jenis elemen yang dapat digunakan pengembang di model-model UML-nya dan konstrain-konstrain dari penggunaannya. UML menyediakan mekanisme perluasan dan mengakomodasikan konsep-konsep baru dengan meta model yang ditawarkannya[27]. Diagram UML dapat dilihat pada gambar berikut:



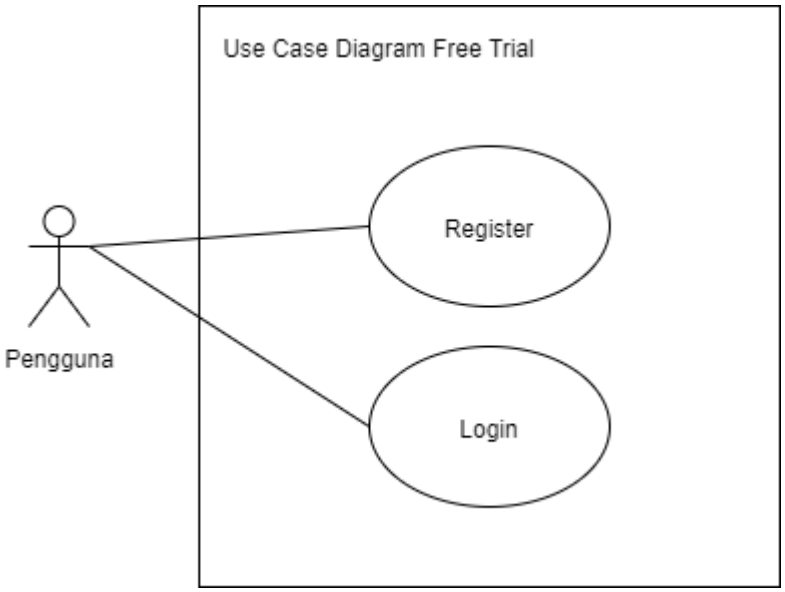
Gambar 2.2 Diagram UML

2.7.1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya[26].

Diagram *usecase* digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Diagram *usecase* menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem dan interaksi-interaksinya dengan dunia luar. Dengan cara ini, diagram *usecase* menggantikan diagram konteks pada pendekatan konvensional. Pemodelan ini biasa dilakukan lewat proses berulang interaksi antara pengembang dan pemakai untuk memperoleh spesifikasi kebutuhan yang sama-sama disepakati. Pemodelan *usecase* diciptakan oleh Ivar Jacobson. Tujuan utama pemodelan *usecase* adalah[27]:

1. Memutuskan dan mendeskripsikan kebutuhan-kebutuhan fungsional sistem
2. Memberikan deskripsi jelas dan konsisten dari apa yang seharusnya dilakukan, sehingga model *usecase* digunakan di seluruh proses pengembangan untuk komunikasi dan menyediakan basis pemodelan berikutnya yang mengacu sistem harus memberikan fungsionalitas yang dimodelkan pada *usecase*
3. Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem. Menguji apakah sistem telah memberikan fungsionalitas yang diminta
4. Menyediakan kemampuan melacak kebutuhan fungsionalitas menjadi kelas-kelas dan operasi-operasi aktual di sistem. Untuk menyederhanakan perubahan dan ekstensi ke sistem dengan mengubah model *usecase* dan kemudian melacak *usecase* yang dipengaruhi ke perancangan dan implementasi sistem. Contoh diagram *use case* dapat dilihat pada gambar berikut:

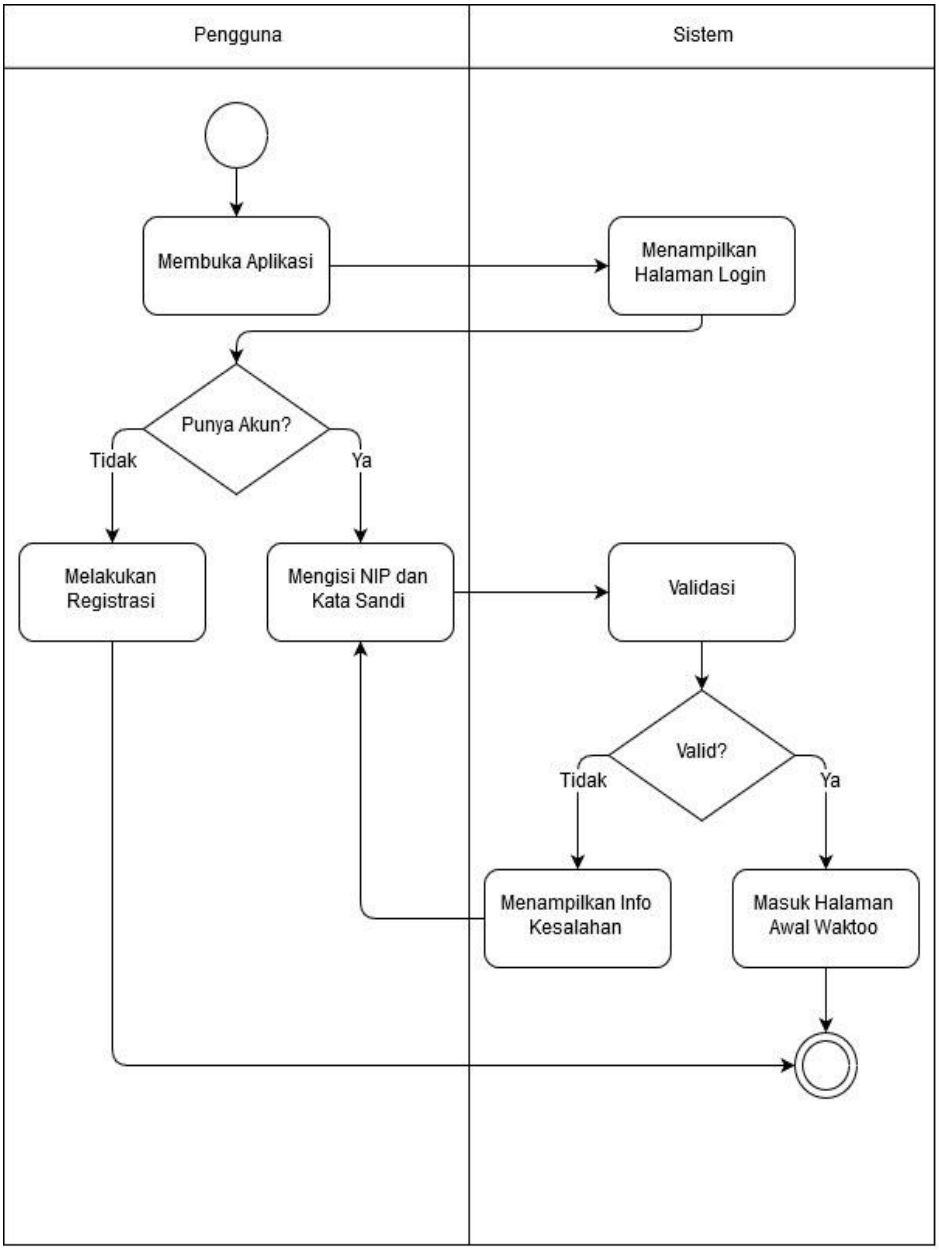


Gambar 2.3 Contoh *Use Case Diagram*

2.7.2. Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi[26].

Diagram aktivitas merupakan jenis khusus dan diagram *statechart*. *State* adalah aksi-aksi yang menuju state berikutnya setelah selesai aksi itu. Diagram aktivitas berfokus pada aktivitas-aktivitas, potongan-potongan dari proses yang boleh jadi (mungkin) berkorespondensi dengan metode-metode atau fungsi-fungsi anggota dan pengurutan dari aktivitas-aktivitas ini. Hal ini serupa dengan *flowchart*. Namun, diagram aktivitas berbeda dari *flowchart* terutama karena diagram aktivitas secara eksplisit mendukung aktivitas-aktivitas paralel dan sinkronasi aktivitas-aktivitas itu[27].



Gambar 2.4 Contoh *Activity Diagram*

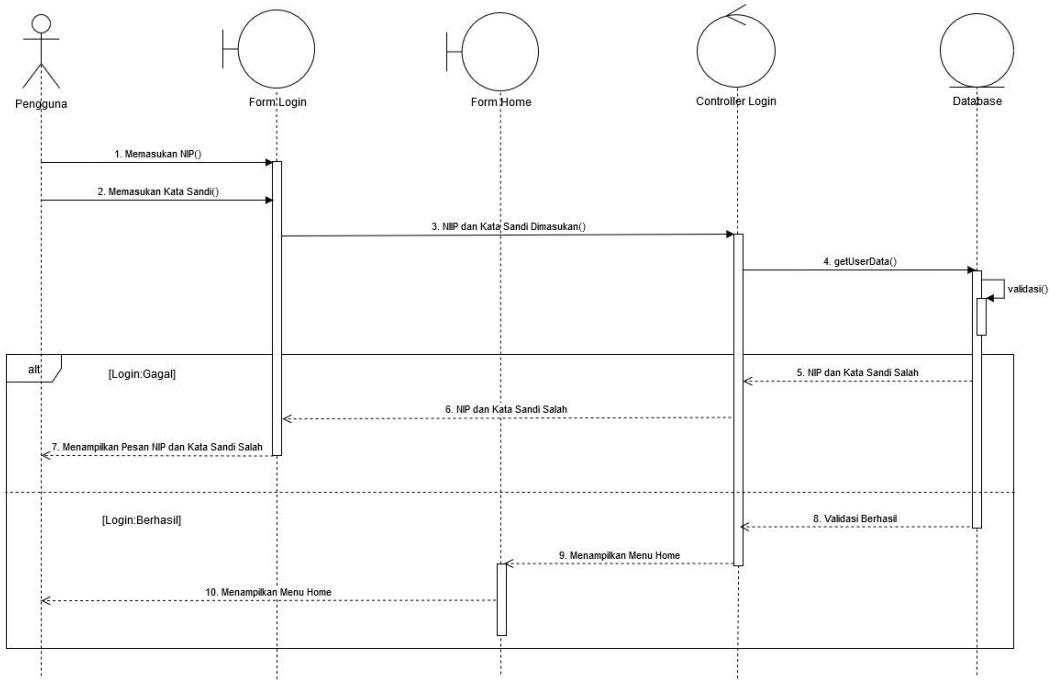
2.7.3. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi *horizontal* (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau

rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan[26].

Sequence diagram digunakan untuk memodelkan skenario penggunaan. Skenario penggunaan adalah barisan kejadian yang terjadi selama satu eksekusi sistem. Cakupan skenario dapat beragam, dari mulai semua kejadian di sistem atau hanya kejadian pada objek-objek tertentu.skenario menjadi rekaman historis eksekusi sistem atau gagasan eksperimen eksekusi sistem yang diusulkan.

Diagram sekuen menunjukkan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukkan barisan kejadian, bukan pewaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan kejadian[27].



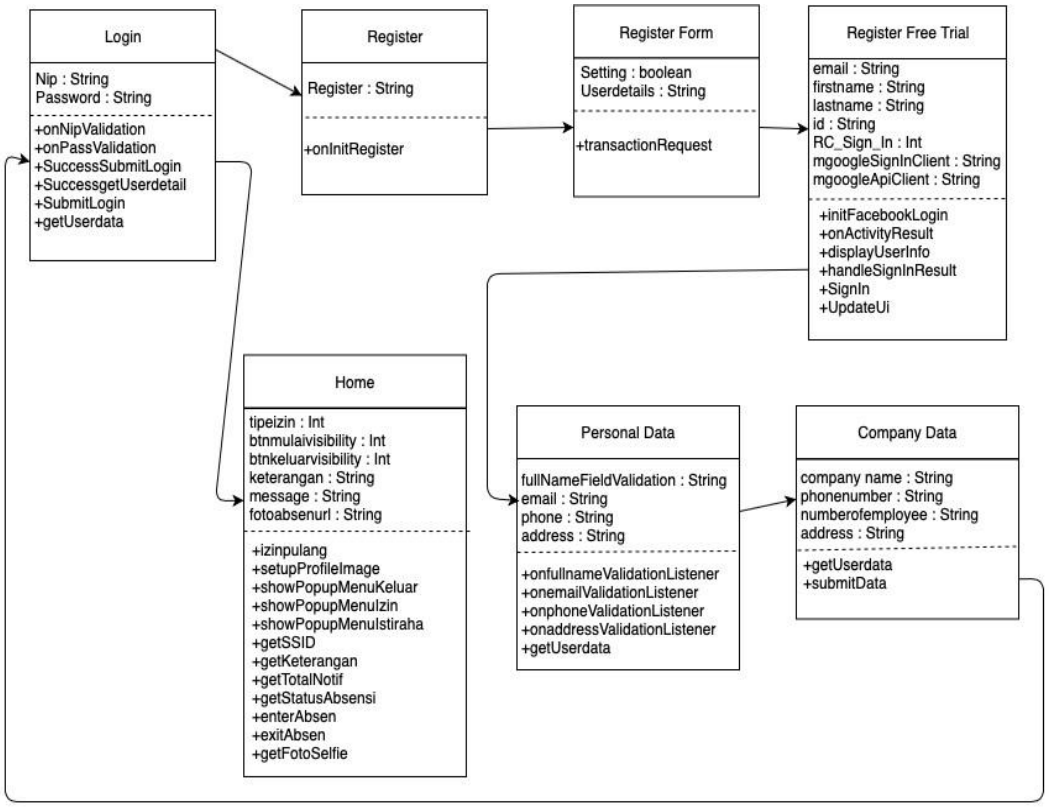
Gambar 2.5 Contoh *Sequence Diagram*

2.7.4. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class* diagram menggambarkan struktur dan deskripsi *class*, package dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu nama, atribut, dan metoda. Atribut dan metoda dapat memiliki salah satu sifat berikut[26]:

1. *Private*, tidak dapat dipanggil dari luar class yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

Diagram kelas menunjukkan aspek statik sistem terutama untuk mendukung kebutuhan fungsional sistem. Kebutuhan fungsional berarti layanan-layanan yang harus disediakan sistem ke pemakai. Meskipun diagram kelas serupa dengan model data, namun kelas-kelas tidak hanya menunjukkan struktur informasi tapi juga mendeskripsikan perilaku. Salah satu maksud diagram kelas adalah untuk mendefinisikan pondasi bagi diagram-diagram lain dimana aspek-aspek lain dari sistem ditunjukkan[27].



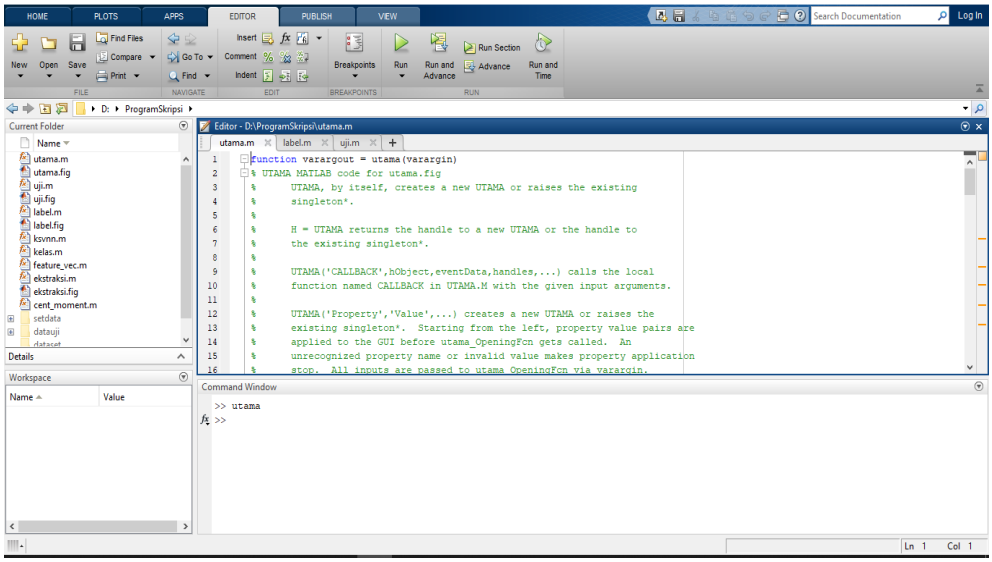
Gambar 2.6 Contoh *Class Diagram*

2.8. MATLAB

Matlab merupakan bahasa pemrograman yang hadir dengan fungsi dan karakteristik yang berbeda dengan bahasa pemrograman lain yang sudah ada lebih dahulu seperti Delphi, Basic maupun C++. Matlab merupakan bahasa pemrograman level tinggi yang dikhususkan untuk kebutuhan komputasi teknis, visualisasi dan pemrograman seperti komputasi matematik, analisis data, pengembangan algoritma, simulasi dan pemodelan dan grafik-grafik perhitungan.

Matlab hadir dengan membawa warna yang berbeda. Hal ini karena matlab membawa keistimewaan dalam fungsi-fungsi matematika, fisika, statistik, dan visualisasi. Matlab dikembangkan oleh MathWorks, yang pada awalnya dibuat untuk memberikan kemudahan mengakses data matrik pada proyek LINPACK dan EISPACK. Saat ini matlab memiliki ratusan fungsi yang dapat digunakan sebagai problem solver mulai dari simple sampai

masalah-masalah yang kompleks dari berbagai disiplin ilmu[28]. Tampilan aplikasi matlab dapat dilihat pada gambar berikut:



Gambar 2.7 Tampilan Matlab

2.9. Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan sebuah sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan antara hasil yang diterapkan dengan hasil yang terjadi[27]. Pengujian seharusnya meliputi tiga konsep berikut.

1. Demonstrasi validitas perangkat lunak pada masing-masing tahap di siklus pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3. Pemeriksa perilaku sistem dengan mengeksekusi sistem pada data sampel pengujian.

Pada dasarnya pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah pengkodean (kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dapat dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan. Evaluasi terhadap spesifikasi dan

perancangan juga merupakan Teknik pengujian. Kategori pengujian dapat dikategorikan menjadi dua [20], yaitu:

1. Berdasarkan ketersediaan *logic* sistem, terdiri dari *Black box testing* dan *White box testing*.
2. Berdasarkan arah pengujian, terdiri dari Pengujian *top down* dan Pengujian *bottom up.*

2.9.1. Pengujian *Black Box*

Konsep *Black box* digunakan untuk merepresentasikan sistem yang cara kerja didalamnya tidak tersedia untuk diinspeksi. Di dalam *black box*, item-item yang diuji dianggap "gelap" karena logikanya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari *black box*[27].

Pada pengujian *black box*, kasus-kasus pengujian berdasarkan pada spesifikasi sistem. Rencana pengujian dapat dimulai sedini mungkin di proses pengembangan perangkat lunak.

Pada pengujian *black box*, kita mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Kita dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis scenario, dimana isi dalam sistem mungkin tidak tersedia diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain.

2.9.2. Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* atau *reference value*). Tingkat akurasi diperoleh dengan persamaan sebagai berikut.

$$Akurasi = \frac{Jumlah\ Karakter\ Sama}{Jumlah\ Semua\ Karakter} \times 100\% \qquad (22)$$