BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis

Analisis bertujuan untuk mengidentifikasi permasalahan-permasalahan yang terdapat pada sistem serta menentukan kebutuhan-kebutuhan dari sistem yang dibangun. Analisis tersebut meliputi analisis masalah, analisis kebutuhan non-fungsional, dan analisis kebutuhan fungsional.

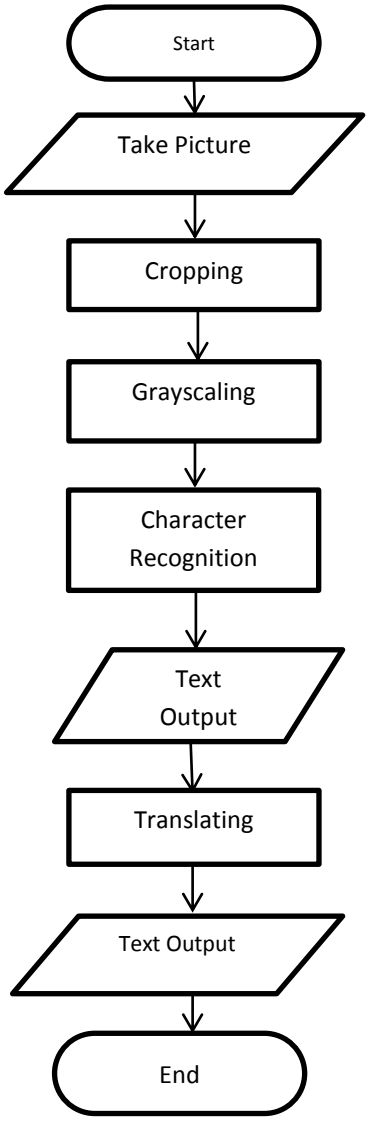
3.2 Analisis Masalah

Berdasarkan penelitian yang dilakukan, permasalahan yang terjadi dapat dirumuskan sebagai berikut:

1. Keyboard standar yang tersedia pada ponsel android tidak menyediakan huruf Jerman. Sehingga pengguna tidak bisa menginputkan teks non-digital dengan bahasa Jerman.
2. Penggunaan kamus tidak efisien jika menterjemahkan satu baris kalimat.

3.3 Analisis Metode

Berikut adalah tahapan-tahapan beserta metode yang digunakan dalam aplikasi yang akan dibangun, mulai dari masukan hingga keluaran.



Gambar 3.1 Tahapan Pada Aplikasi

1. Take Picture

Tahap ini merupakan tahap awal yang digunakan untuk mengambil gambar melalui kamera yang terpasang pada perangkat android, sebagai inputan yang akan diproses oleh tahap berikutnya.

2. Cropping

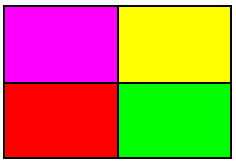
Tahap ini berfungsi untuk memotong gambar yang telah diambil oleh pengguna. Tahap *cropping* perlu dilakukan untuk memfokuskan gambar pada area teks saja agar tidak ada objek selain teks yang akan dikenali oleh

OCR. Pengguna menentukan sendiri area teks pada gambar untuk dipotong, agar ukuran gambar lebih kecil sehingga mempercepat proses penengenalan karakter. Proses pemotongan gambar menggunakan fungsi yang sudah ada pada perangkat android. Berikut adalah kode untuk memanggil fungsi *crop*.

3. Grayscaling

Citra berwarna yang sudah terpotong, selanjutnya akan dibuat menjadi citra *grayscale*. Metode yang digunakan untuk proses *grayscaling* adalah *luminosity*. Formula untuk metode *luminosity* adalah 0.21 R + 0.72 G + 0.07 B. Berikut adalah langkah-langkah pada metode *luminosity*.

Metode luminosity melakukan perulangan pada setiap piksel untuk mencari nilai piksel tersebut. Warna pada setiap piksel akan direpresetasikan ke dalam bentuk desimal yaitu 0-255. Berikut adalah contoh citra berwarna.



Gambar 3.2 Contoh Citra Warna

Metode luminosity melakukan perulangan pada setiap piksel untuk mencari nilai piksel tersebut. Warna merah murni (*pure red*) pada gambar di atas memiliki nilai R=255, G=0, B=0, warna hijau murni (*pure green*) memiliki nilai R=0, G=255, B=0, sedangkan warna ungu memiliki nilai R=255, B=255, G=0, dan piksel terakhir berwarna kuning, memiliki nilai R=255,

B=0, G=255, maka nilai-nilai tersebut akan dikalikan dengan formula yang sudah ditentukan sebagai berikut:

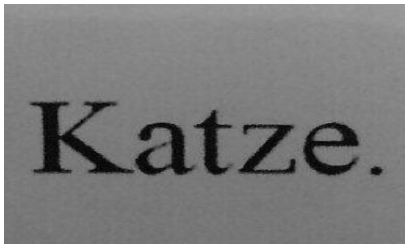
1. Merah = 0.21*255 + 0.72*0 + 0.07*0 = 54 + 0 + 0 = 54
2. Hijau = 0.21*0 + 0.72*255 + 0.07*0 = 0 + 184 + 0 = 184
3. Ungu = 0.21*255 + 0.72*0 + 0.07*255 = 54 + 0 + 18 = 72
4. Kuning = 0.21*255 + 0.72*255 + 0.07*0 = 54 + 184 + 0 = 238

Citra yang berwarna akan digantikan dengan citra *grayscale* sesuai dengan nilai yang sudah didapatkan melalui perhitungan di atas. Berikut adalah citra *grayscale* yang dihasilkan.



Gambar 3.3 Citra Hasil Grayscaling

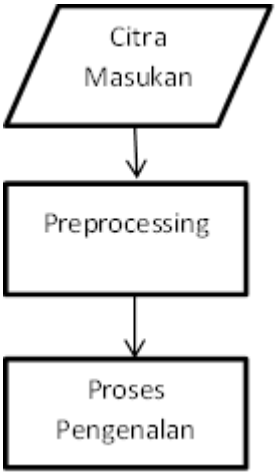
Berikut adalah contoh citra *grayscale*



Gambar 3.4 Contoh Citra Grayscale

4. Character Recognition

Proses pengenalan karakter menggunakan layanan yang sudah ada, yaitu *Tesseract*. Proses pengenalan karakter pada citra dengan *Tesseract* terdapat beberapa langkah yaitu:



Gambar 3.5 Proses Pengenalan Karakter

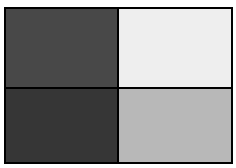
A. Preprocessing

Berikut adalah beberapa proses yang terdapat pada tahap preprocessing:

1. Segmentasi Citra

   Segmentasi citra dilakukan utnuk memisahkan *background* dengan *foreground* dengan merubah citra *grayscale* menjadi citra biner menggunakan metode otsu. Berikut adalah penjelasan mengenai metode otsu

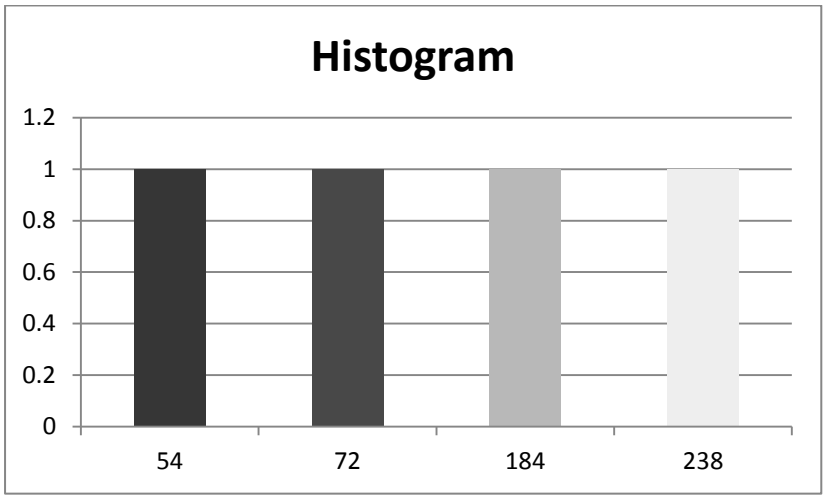
   Contoh citra masukan adalah citra grayscale dengan matriks 2x2.



Gambar 3.6 Citra Masukan

Langkah pertama adalah mengurutkan nilai derajat keabuan dari yang terkecil hingga yang terbesar, berdasarkan perhitungan pada pembahasa sebelumnya maka nilai derajat keabuan yang didapat

adalah 54, 72, 184, 238. Di bawah ini merupakan histogram dari citra masukan.



Gambar 3.7 Histogram Citra Masukan

Selanjutnya mencari nilai weight, mean dan variance background dengan rumus sebagai berikut:

Wb = Jumlah piksel background / total piksel pada citra
= (1+1) / 4 = 0.5

Mb = (derajat keabuan n * jumlah piksel derajat keabuan n) / total piksel background
= (54 * 1) + (72 * 1) / 2
= 63

Vb = ((derajat keabuan n – Mb)^2 * jumlah piksel derajat keabuan n) / jumlah piksel background
= ((54 – 63)^2 * 1) + ((72 – 63)^2 * 1) / 2
= 81

Selanjutnya mencari nilai weight, mean dan variance foreground dengan rumus sebagai berikut:

Wf = Jumlah piksel foreground / total piksel pada citra
= (1+1) / 4 = 0.5

Mf = (derajat keabuan n * jumlah piksel derajat keabuan n) / total piksel foreground

= (184 * 1) + (238 * 1) / 2
= 211

Vf = ((derajat keabuan n – Mf)^2 * jumlah piksel derajat keabuan n) /
jumlah piksel foreground
= ((184 – 211)^2 * 1) + ((238 – 211)^2 * 1) / 2
= 729

Langkah terakhir adalah mencari nilai Within Class Variance dengan rumus sebagai berikut:

Wb*Vb + Wf*Vf = 0.5*81 + 0.5*729 = 405

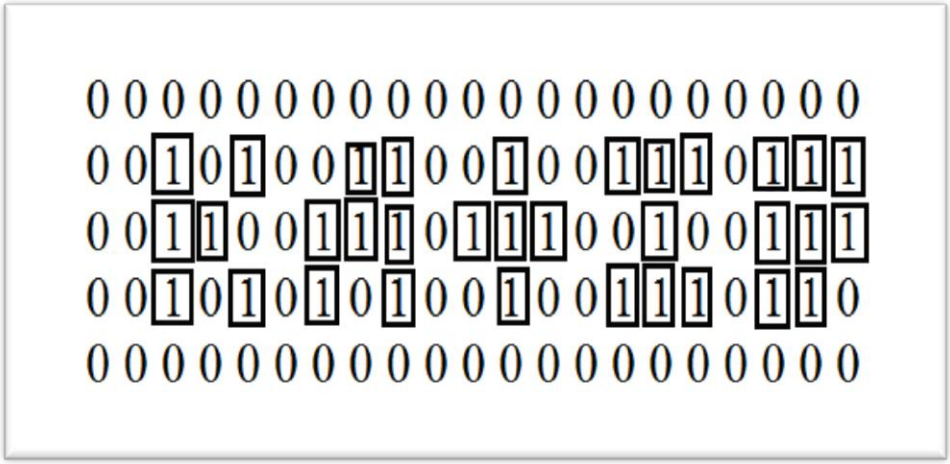
Berikut adalah contoh dari citra biner



Gambar 3.8 Contoh Citra Biner

2. Segmentasi Karakter

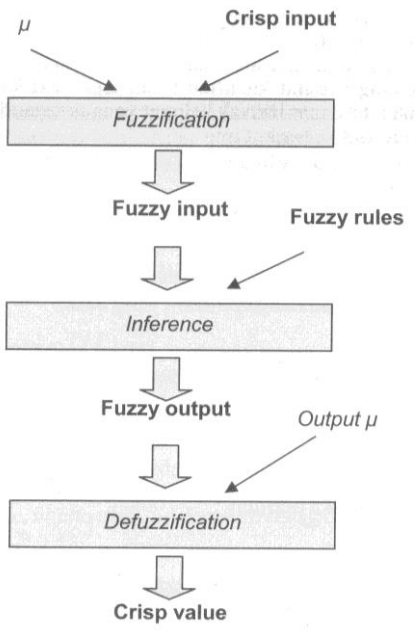
   Segmentasi karakter dilakukan untuk menemukan teks pada gambar dan memotongnya menjandi karakter. Segmentasi karakter dilakukan dengan menggunakan metode *Connected component analysis*. *Connected component analysis* bekerja dengan cara memeriksa intensitas pixel di sekitar pixel yang sedang dianalisis. Berikut adalah ilustrasi dari *connected component analysis*.



Gambar 3.9 Connected Compenent Analysis

B. Proses Pengenalan

Hasil dari segmentasi karakter akan diproses untuk dicocokan dengan data yang sudah ada pada *database* tesseract. Algoritma *fuzzy* menilai tingkat kebenaran dari setiap karakter. Berikut adalah ilustrasi dari algoritma *fuzzy logic*.



Gambar 3.10 Ilustrasi Fuzzy Logic

Fuzzification: mengubah masukan-masukan yang nilai kebenarannya bersifat pasti (crisp input) ke dalam bentuk fuzzy input.

Inference: melakukan penalaran menggunakan fuzzy input dan fuzzy rules yang telah ditentukan sehingga menghasilkan fuzzy output. Secara sintaks, suatu fuzzy rule (aturan fuzzy) dituliskan sebagai
IF antecendent THEN consequent

Defuzzification: mengubah fuzzy output menjadi *crisp value* berdasarkan fungsi keanggotaan yang telah ditentukan.

5. Text Output

Hasil pengenalan karakter oleh OCR akan dikirim kembali ke aplikasi berupa *file* teks dengan bahasa Jerman yang selanjutnya akan diterjemahkan ke dalam bahasa Indonesia.

6. Translating

Proses penterjemahan teks menggunakan layanan dari *Microsoft Translator*. *Microsoft Translator* secara otomatis menterjemahkan hasil pengenalan karakter tanpa menunggu perintah dari pengguna.

7. Text Output

Selanjutnya Microsoft Translator akan mengembalikan *output file* berupa teks yang sudah diterjemahkan.

3.4 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional terdiri dari analisis perangkat keras, perangkat lunak dan pengguna yang menggunakan aplikasi yang akan dibangun.

3.4.1 Analisis Kebutuhan Perangkat Keras

Kebutuhan perangkat keras untuk menjalankan aplikasi ini adalah *smartphone* Android yang memiliki beberapa spesifikasi utama sebagai berikut:

1. Sistem operasi Android v4.0 (*Ice Cream Sandwich*).
2. Kamera 3 MP, untuk keperluan pengambilan gambar yang dilakukan secara langsung.
3. Tampilan layar sentuh, untuk keperluan pemotongan gambar yang dilakukan secara manual.
4. Media penyimpanan eksternal, untuk keperluan menyimpan gambar yang dihasilkan oleh program aplikasi.
5. Koneksi internet.

3.4.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dipakai dalam pengembangan aplikasi, diantaranya adalah sebagai berikut:

1. Sistem Operasi Windows 7 Home Edition.
2. Java Development Kit versi 1.8.0, untuk keperluan kompilasi kode-kode program.
3. Eclipse IDE 3.7, digunakan sebagai perangkat lunak untuk membangun aplikasi android.
4. Android SDK, untuk pengembangan aplikasi Android

5. Android Development Tool, sebagai plugin untuk mengintegrasikan Android SDK pada eclipse.
6. Tesseract digunakan untuk pengenalan teks pada citra dan proses ekstraksi teks.
7. Microsoft translator java API, digunakan untuk mengintegrasikan aplikasi dengan layanan *Microsoft Translator*, sehingga aplikasi dapat menerjemahkan teks dengan bahasa Jerman ke bahasa Indonesia.

3.4.3 Analisis Pengguna

Aplikasi yang akan dibangun diperuntukan bagi pengguna *smartphone* Android di Indonesia, khususnya bagi pelajar Indonesia yang sedang melanjutkan pendidikan di negara Jerman dan kurang begitu memahami bahasa Jerman. Aplikasi ini juga dapat difungsikan sebagai kamus untuk mengenal berbagai macam kata yang ditulis menggunakan bahasa Jerman, sehingga tidak menutup kemungkinan dapat dipakai oleh berbagai kalangan.

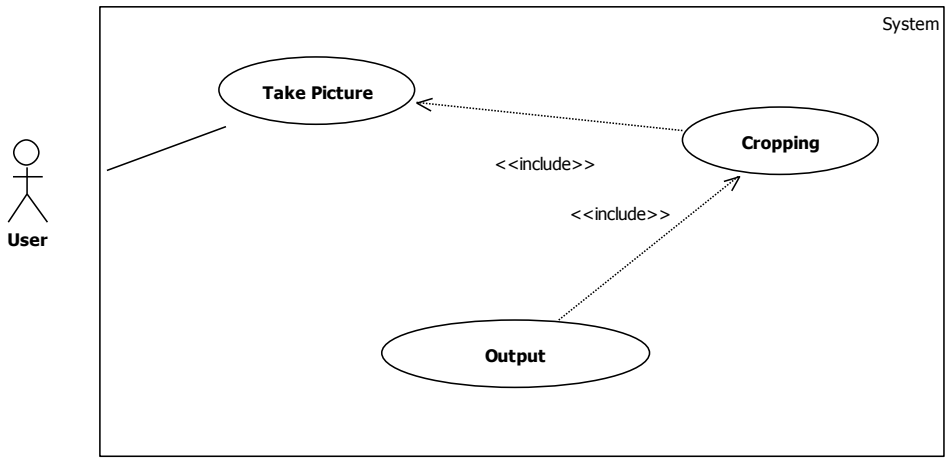
3.5 Analisis Kebutuhan Fungsional

Kebutuhan fungsional meliputi beberapa diagram UML, diantaranya: diagram *use case*, diagram *activity*, diagram *class* dan diagram *sequence*. Diagram tersebut dipergunakan sebagai desain rancangan untuk mempermudah proses pengembangan aplikasi khususnya untuk aplikasi yang dikembangkan menggunakan pemrograman berorientasi objek.

3.5.1 Diagram Use case

Diagram Use case

Pada sistem aplikasi yang dirancang terdapat tiga buah *use case* dan satu *user* sebagai aktor tunggal yang berperan dalam sistem.



Gambar 3.11 Diagram Use case

Deskripsi Use case

Tabel 3.1 Use case Take Picture

Reaksi Aktor	Reaksi Sistem
Skenario Normal	
1. Mengklik button Take Picture	
	2. Menampilkan aplikasi kamera yang terpasang di perangkat
3. Memilih aplikasi kamera	
	4. Menjalankan aplikasi kamera yang dipilih
5. Mengambil gambar	
	6. Menampilkan gambar yang diambil
7. Mengklik tombol yes/ok	
	8. Menyimpan gambar yang diambil
Skenario Alternatif	
1. Mengklik tombol retake	
	2. Mengulangi pengambilan gambar
3. Mengklik tombol cancel	
	4. Mengembalikan ke menu utama aplikasi

Tabel 3.2 Use case Cropping

Reaksi Aktor	Reaksi Sistem
Skenario Normal	
	1. Menampilkan hasil gambar yang diambil
2. Menentukan area yang akan dipotong	
3. Mengklik tombol crop	
	4. Melakukan proses *cropping*
	5. Menyimpan gambar
Skenario Alternatif	
1. Mengklik tombol cancel	
	2. Mengembalikan ke menu utama aplikasi

Tabel 3.3 Use case Output

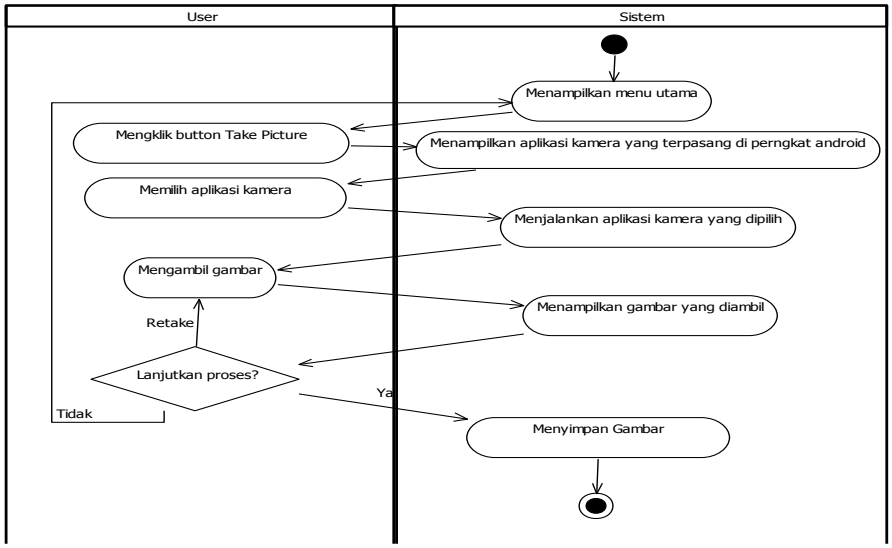
Reaksi Aktor	Reaksi Sistem
	1. Menampilkan hasil pengenalan teks
	2. Mengirimkan hasil pengenalan ke layanan Microsoft Translator
	3. Menampilkan teks hasil penerjemahan

3.5.3 Activity Diagram

Activity diagram digunakan untuk menggambarkan alur rangkaian proses yang terjadi di dalam sistem aplikasi serta menjelaskan bagaimana sebuah *use case* terjadi atau dipicu dari *use case* yang lain

1. Activity Diagram Take Picture

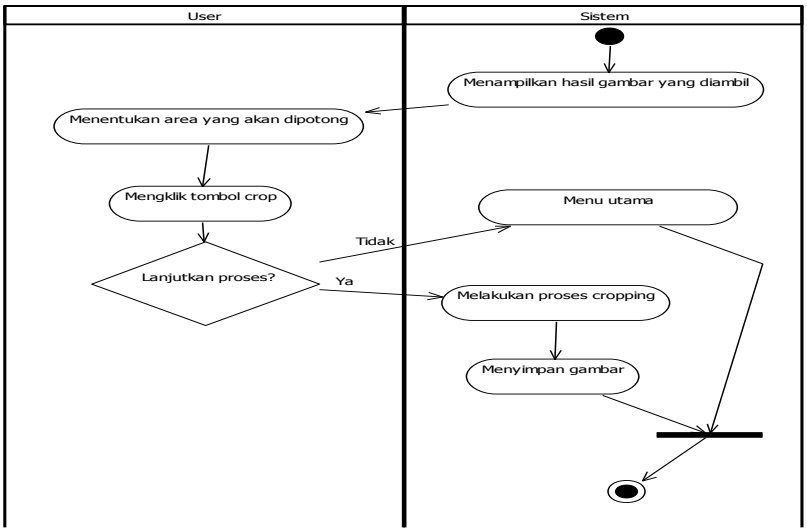
   Berikut adalah proses pengambilan gambar yang digambarkan dengan diagram aktivitas.



Gambar 3.12 Activity Diagram Take Picture

2. Activity Diagram Cropping

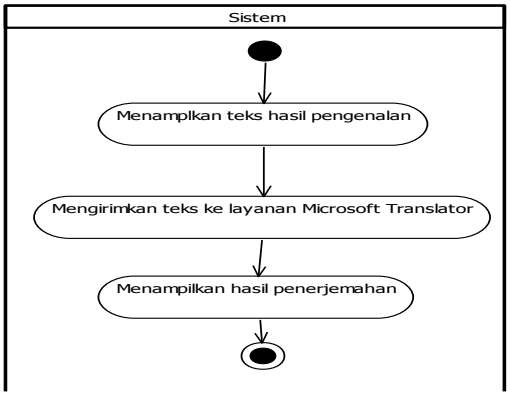
Berikut adalah proses pemotongan gambar yang digambarkan dengan diagram aktivitas.



Gambar 3.13 Activity Diagram Cropping

3. Activity Diagram Output

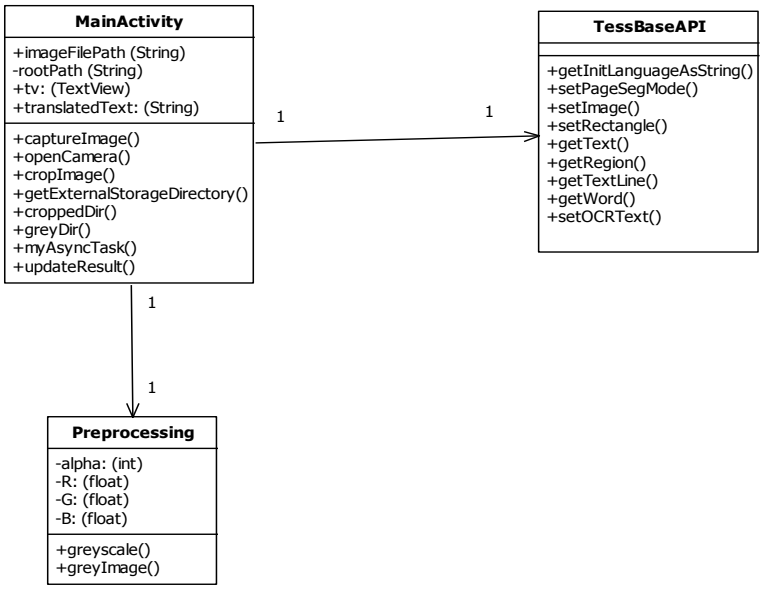
   Berikut adalah aktifitas dari hasil pengenalan teks menggunakan OCR yang kemudian hasil pengenalan tersebut dikirimkan kelayanan *Microsoft Translator* untuk diterjemahkan.



Gambar 3.14 Activity Diagram Output

3.5.4 Diagram *Class*

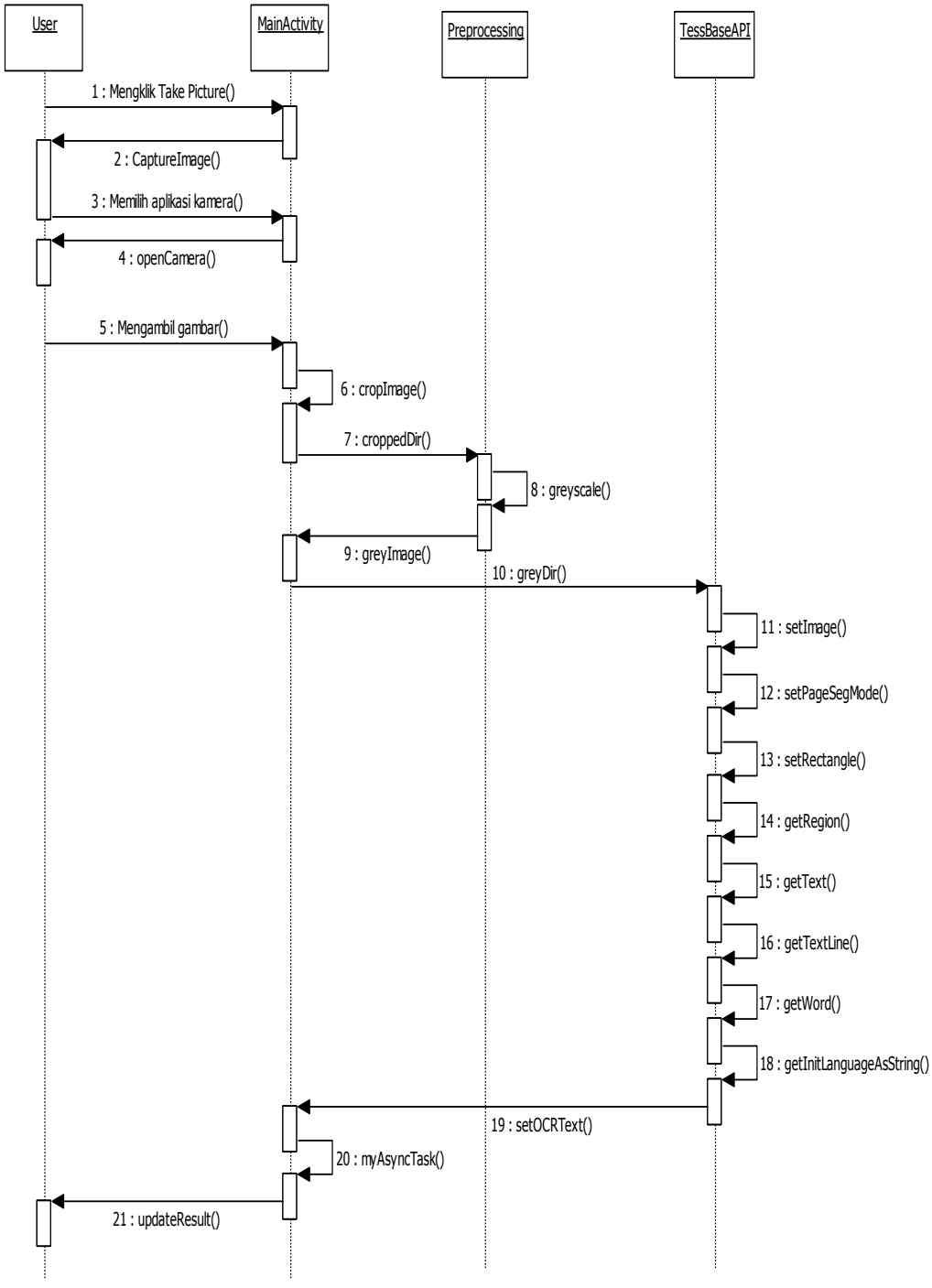
Diagram *Class* digunakan untuk menggambarkan secara abstrak struktur dari aplikasi yang akan dibangun, *class-class* yang terlibat, serta hubungan antar *class* untuk saling berkomunikasi satu sama lain.



Gambar 3.15 Diagram Class

3.5.5 Diagram *Sequence*

Diagram *sequence* menggambarkan urutan aktifitas yang terjadi di dalam sistem. Diagram ini menunjukkan pengiriman pesan-pesan yang melewati objek yang terlibat di dalam sistem serta tahapan langkah yang harus dilalui untuk mencapai *use case* tertentu.



Gambar 3.16 Diagram Sequance

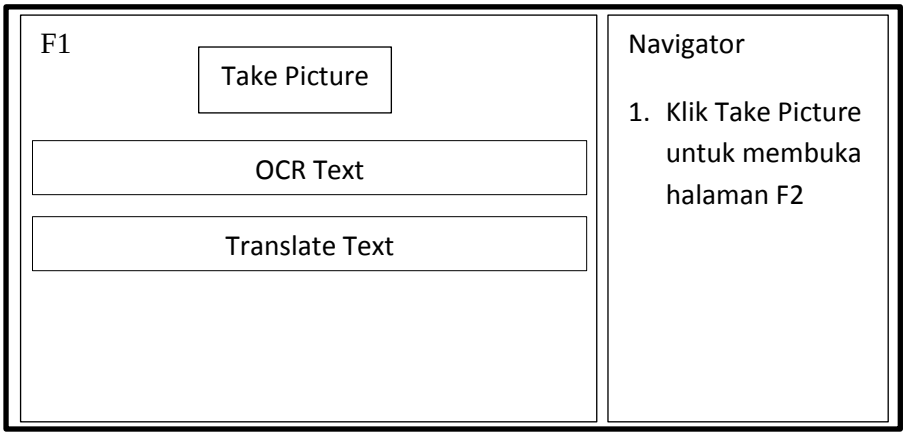
3.6 Perancangan Sistem

Sistem yang akan dibangun dari aplikasi berbasis *mobile android* ini memiliki fungsi utama untuk melakukan proses pengolahan citra. Sedangkan untuk proses pengenalan teks dan penerjemahan akan memanfaatkan dua layanan yang sudah ada, yaitu Tesseract dan *Microsoft Translator*. Citra yang dapat dikenali oleh layanan Tesseract berupa citra yang berwarna hitam dengan latar belakang berwarna putih. Oleh karena itu, tujuan dari pengolahan citra yang dikerjakan oleh program aplikasi ialah untuk menghasilkan citra yang bisa dikenali oleh layanan OCR tersebut.

3.6.1 Rancangan Antarmuka

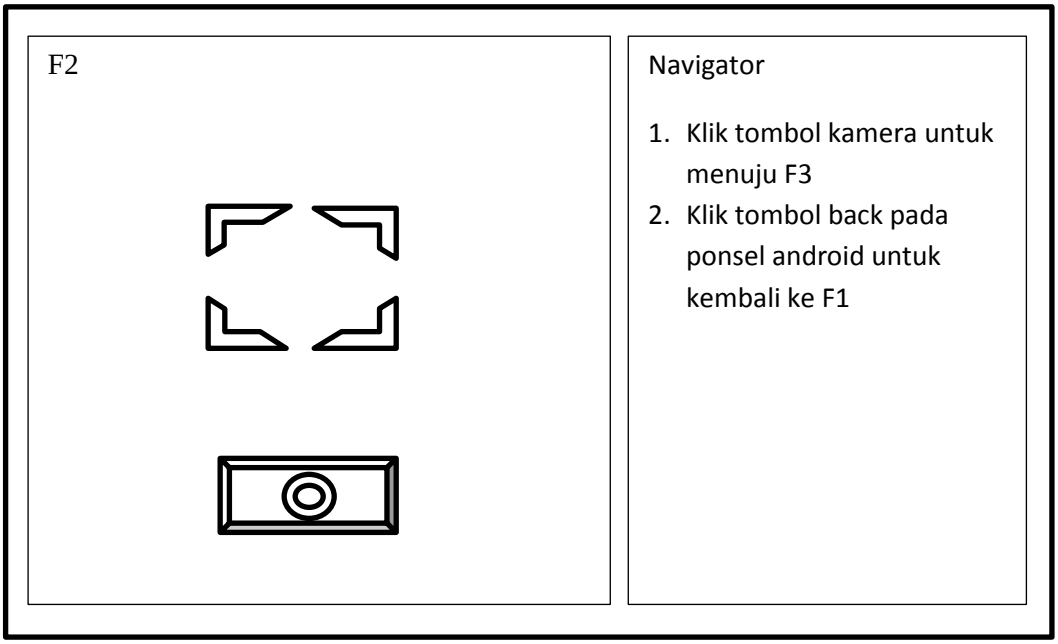
Sistem dari aplikasi yang akan dibangun memiliki satu buah tampilan utama, yaitu tampilan Menu Utama. Tampilan Menu Utama digunakan untuk mengambil sumber foto yang ingin diolah, hasil pengenalan karakter dan hasil terjemahan teks.

1. Tampilan Utama Aplikasi



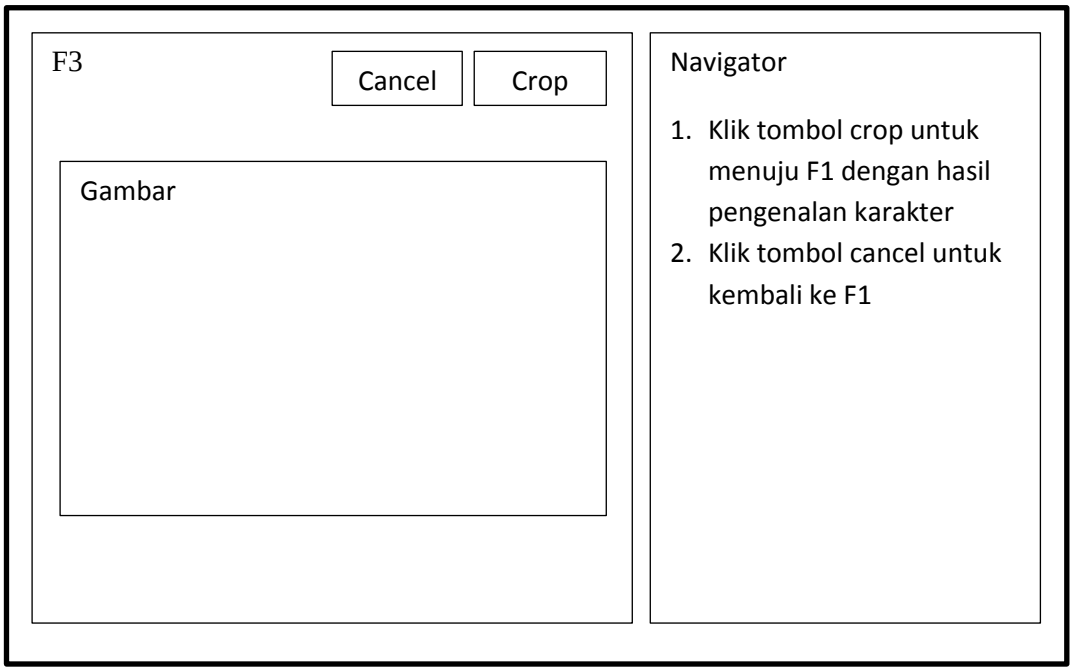
Gambar 3.17 Tampilan Utama Aplikasi

2. Tampilan Antarmuka Pengambilan Gambar



Gambar 3.18 Tampilan Antarmuka Pengambilan Gambar

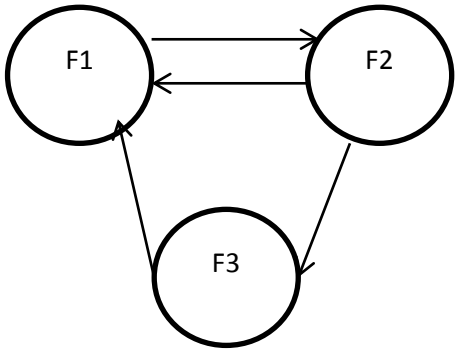
3. Tampilan Antarmuka Pemotongan Gambar



Gambar 3.19 Tampilan Antarmuka Pemotongan Gambar

3.6.2 Perancangan Jaringan Semantik

Jaringan semantik menggambarkan hubungan navigasi menu dari satu halaman ke halaman lainnya (dari form satu ke form lainnya). Berikut adalah gambaran jaringan semantik yang ada pada aplikasi yang akan dibangun.

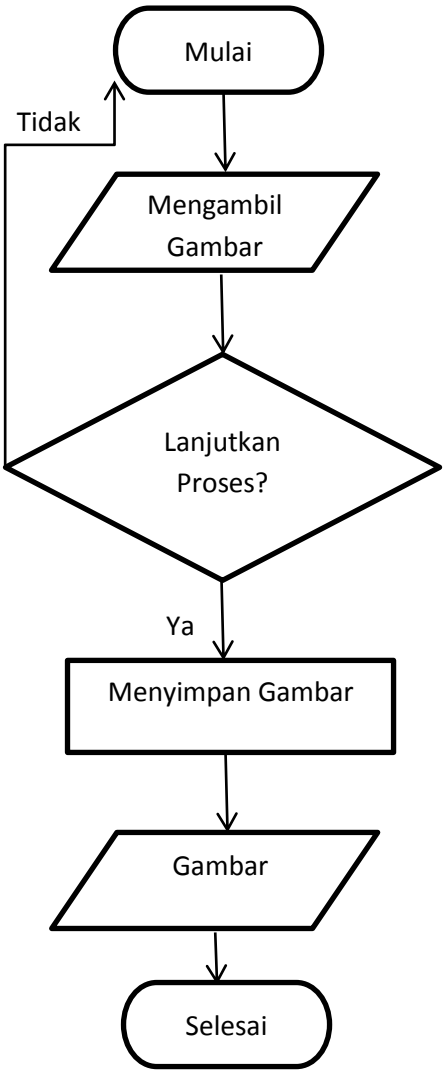


Gambar 3.20 Jaringan Semantik

3.6.3 Perancangan Prosedural

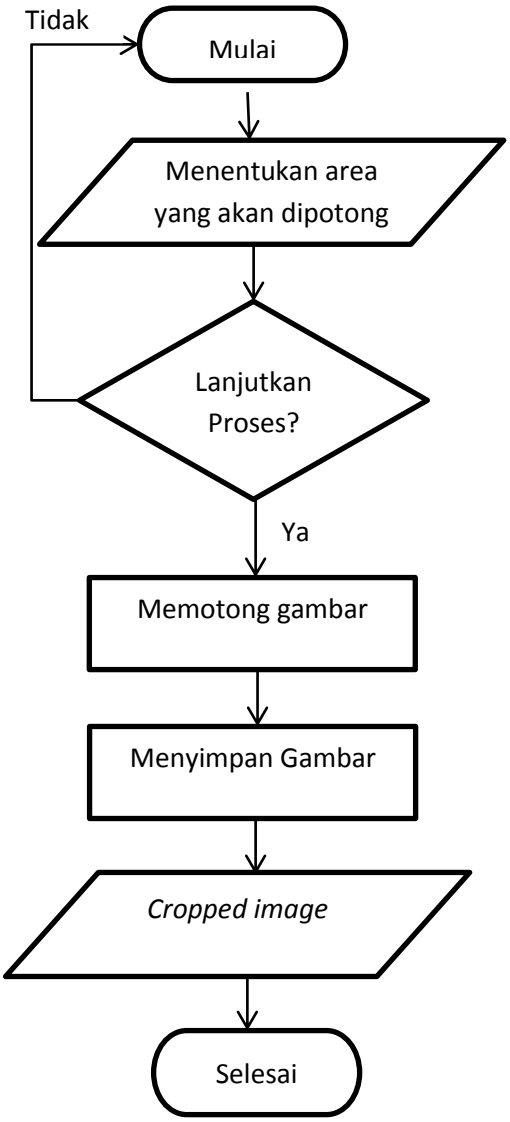
Perancangan prosedural digunakan untuk menggambarkan langkah-langkah atau alur dari setiap proses yang ada pada sistem. Berikut adalah rancangan prosedural dari aplikasi yang akan dibangun.

1. Proses Pengambilan Gambar



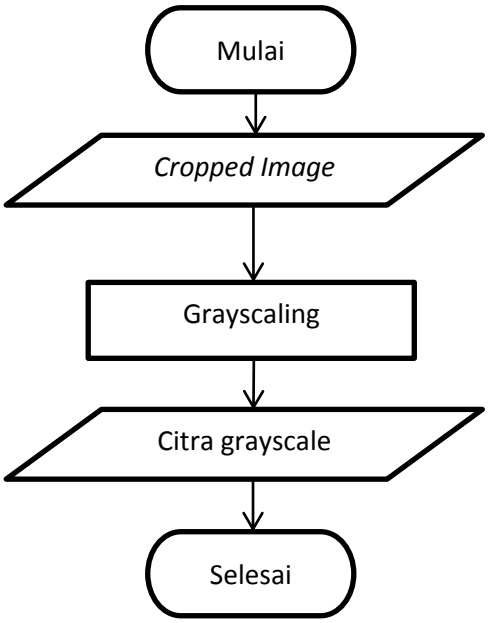
Gambar 3.21 Flowchart Proses Pengambilan Gambar

2. Cropping



Gambar 3.22 Proses Pemotongan Gambar

3. *Grayscaling*



Gambar 3.23 Flowchart Grayscaling