BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis merupakan penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponen dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, sehingga diperoleh solusi. Analisis merupakan tahapan yang paling penting, karena kesalahan dalam tahap ini akan menyebabkan kesalahan di tahap selanjutnya. Sistem yang dibuat merupakan program untuk mendeteksi sebuah *marker* sekaligus menampilkan objek tiga dimensi yang telah dibuat dengan menggunakan *software* tiga dimensi (3D Max). Objek yang dibuat merupakan sebuah rambut, seolah-olah pengguna berinteraksi langsung dengan objek virtual dalam dunia nyata.

3.1.1 Analisis Masalah

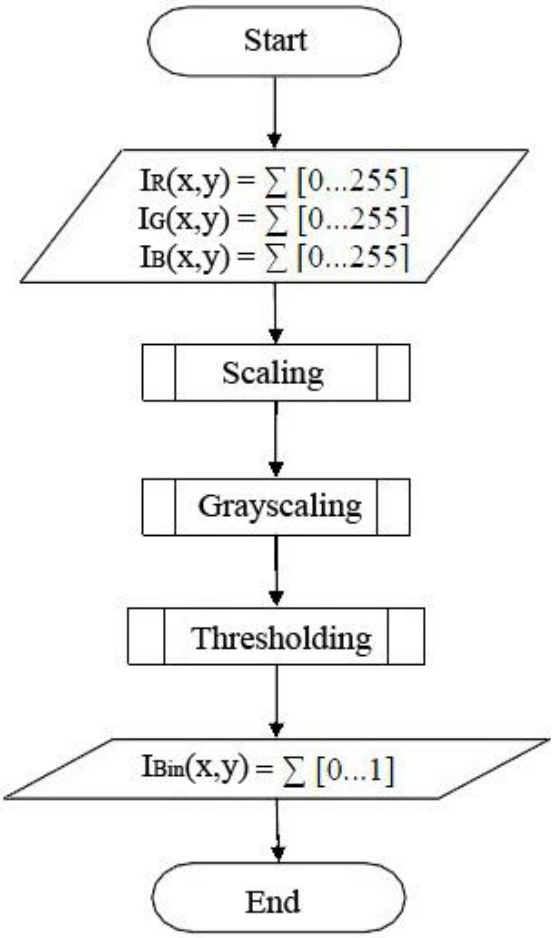
Akan dijelaskan berbagai masalah apa saja yang ada di HEADCORE BARBERSHOP sebelum dibangunnya aplikasi *Augmented Reality* ini. Berikut adalah beberapa masalah yang dihadapi :

1. Untuk membuat aplikasi *Augmented Reality* di BarberShop sebagai alat strategi pemasaran yang modern dan menarik.
2. Untuk memudahkan karyawan dalam mencukur rambut pelanggan dan meminimalisir kesalahan model yang diinginkan pelanggan.

3. Untuk memudahkan pelanggan agar mengetahui model rambut mana yang cocok dengannya.
4. Untuk membantu pelanggan agar dapat memilih model rambut yang sesuai degan kehendaknya.

3.1.2 Analisis *Pre-Processing Image*

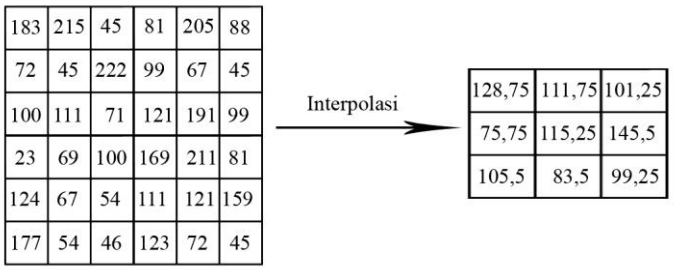
Dalam analisis pre processing image ini dibagi menjadi 3 tahap, tahap pertama yaitu penskalaan(*scaling*), tahap kedua proses *grayscaling*, dan tahap ketiga proses *tresholding*



Gambar 3.1 Alur proses *Pre-Processing Image*

3.1.2.1 Tahap *Scaling*

Scaling merupakan proses mengubah ukuran gambar digital, perubahan ukuran gambar sangat penting karena pada sistem pengenalan wajah membutuhkan data gambar yang seragam ukurannya. Tahap scaling dapat dilakukan dengan menggunakan properti *width* dan *height* atau properti skala seperti scalex dan scaley



Gambar 3.2 Metode Interpolasi Memperkecil Gambar

Nilai piksel pada koordinat pada citra hasil interpolasi diperoleh dengan menghitung nilai rata – rata dari 4 nilai piksel pada citra asli, yaitu :

Tabel 3.1 Perhitungan Nilai Piksel Hasil Interpolas

Nilai Piksel Citra Asli	**Nilai Piksel Citra Interpolasi**
(183+215+72+45) / 4	128,75
(100+111+23+69) / 4	75,75
(124+67+177+54) / 4	105,5
(45+81+222+99) / 4	111,75
(71+121+100+169) / 4	115,25

(54+111+46+123) / 4	83,5
(205+88+67+45) / 4	101,25
(191+99+211+81) / 4	145,5
(121+159+72+45) / 4	99,24

3.1.2.2 Tahap *Grayscaling*

Pada tahap *grayscaling* ini citra berwarna diubah menjadi *grayscale* dengan memperoleh informasi intensitas dari gambar tersebut, gambar dapat di sortir secara ekslusif mulai dari hitam untuk intensitas yang paling lemah sampai dengan putih untuk intensitas yang paling kuat. Untuk melakukan proses *grayscaling* dapat dilakukan dengan mengubah citra berwarna. Citra berwarna terdiri dari 3 layer matriks yaitu R-layer, G-layer, B-layer. Jadi dalam proses ini akan mengubah 3 layer matriks citra berwarna menjadi 1 layer matriks *grayscale*. Dalam citra *grayscale* ini tidak akan lagi ada warna yang ada hanya derajat keabuan. Untuk mengubah citra berwarna yang mempunya matriks R, G, B menjadi citra *grayscale* dengan nilai I. Misalkan suatu citra wajah memiliki nilai

R = 105,5 G = 75,75 B = 128,75

Berikut adalah perhitungan untuk mendapatkan *grayscale* :

$$I = (R \times 0.30) + (G \times 0.59) + (B \times 0.11)$$

Rumus 3.1

Dimana :

I = nilai piksel *grayscale*

R = nilai piksel *Red*

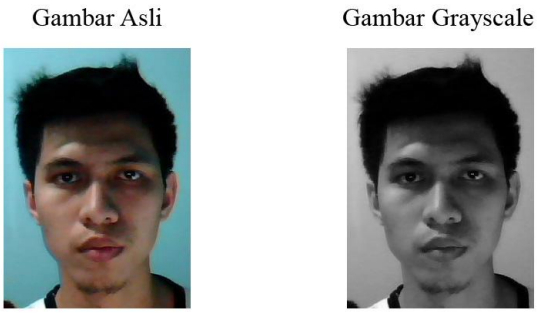
G = nilai piksel *Green*

B = nilai piksel *Blue*

Hasil dari perhitungan tersebut sebagai berikut

$$I = (105{,}5 \times 0.30) + (75{,}75 \times 0{,}59) + (128{,}75 \times 0{,}11) = 90{,}5$$

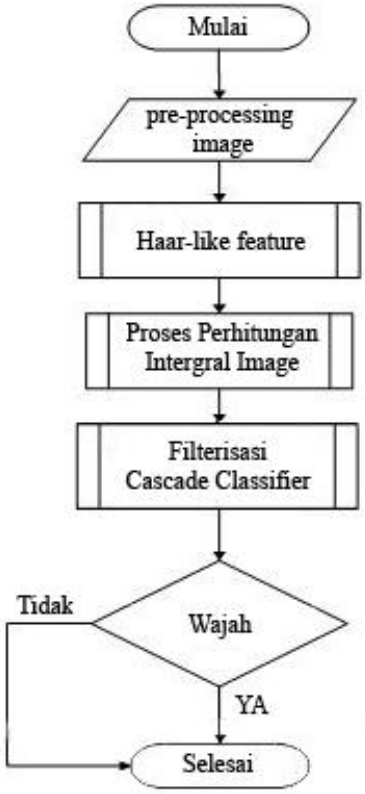
Berikut ini adalah citra hasil *grayscaling* :



Gambar 3.3 Proses Pengubahan Citra RGB Menjadi Citra *Grayscale*

3.1.3 *Library* Marilena dan Metode *Haar Cascade Classifier*

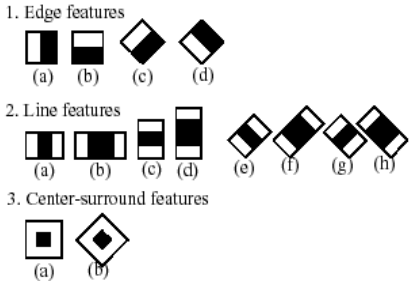
Marilena adalah sebuah library pendeteksi marker yang dapat mengenali pola wajah seseorang. Dan salah satu metode face recognition yang dapat bekerja sama sebagai pendeteksi pola wajah dengan marilena adalah *Haar Cascade Classifier*. Oleh Karena itu, penelitian ini menggunakan metode *Haar Cascade Classifier* sebagai metode untuk pengenalan pola wajah. Berikut adalah alur proses metode *Haar Cascade Classifier*



Gambar 3.4 Flowchart Metode *Haar Cascade Classifier*

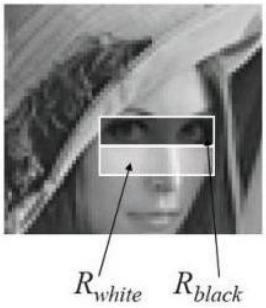
3.1.3.1 *Haar like Feature*

Haar-like feature memproses gambar dalam kotak-kotak, dimana dalam satu kotak terdapat beberapa piksel. Per kotak itu pun kemudian di-proses dan didapatkan perbedaan nilai (*threshold*) yang menandakan daerah gelap dan terang.



Gambar 3.5 Macam-Macam Variasi Feature Pada Haar [10]

Di dalam *library* Marilena hanya menggunakan *Edge Feature* dan *Line Feature*. Dimana *Edge Feature* adalah *feature haar* yang hanya memiliki dua variabel HaarRect, satu putih dan satu hitam. Sedangkan *Line Feature* adalah *feature haar* yang hanya memiliki tiga variabel HaarRect, satu putih, satu hitam, dan satu putih. Berikut adalah contoh di dalam sistem ini yang menggunakan *Edge Feature*. Area hitam dari fitur mendeteksi bagian mata dan area putih mendeteksi bagian kulit.



Gambar 3.6 Contoh Pencarian *Edge Feature*

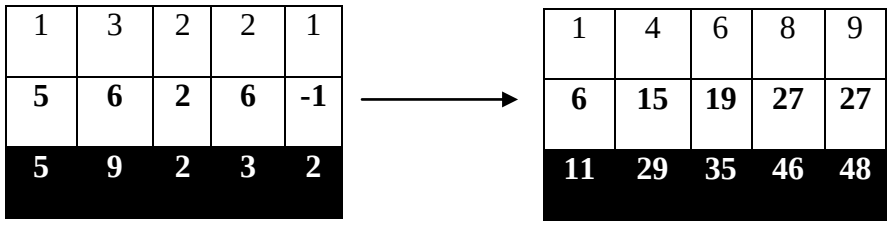
Adanya fitur Haar ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang. Jika nilai perbedaannya itu diatas nilai ambang atau *treshold*, maka dapat dikatakan bahwa fitur tersebut ada. Nilai dari *Haar-like feature* adalah perbedaan antara jumlah nilai-nilai piksel *gray level* dalam daerah kotak hitam dan daerah kotak putih seperti dilihat pada persamaan berikut :

$$f(x) = \sum_B (grey) - \sum_W (grey)$$

Rumus 3.2

3.1.3.2 *Intergral Image*

Integral Image digunakan untuk menentukan ada atau tidaknya dari ratusan fitur Haar pada sebuah gambar dan pada skala yang berbeda secara efisien. Kotak Haar feature dapat dihitung menggunakan *integral image*. Umumnya integral menambahkan unit kecil secara bersamaan. Dalam hal ini unit kecil ini disebut dengan nilai dari pixel. Nilai dari integral / *integral value* pada masing-masing pixel merupakan penjumlahan dari semua pixel di atasnya dan di sebelah kirinya. Dimulai dari kiri atas sampai kanan bawah. Salah satu contoh nilai *Haar like Features* yang ada di dalam library Marilena adalah sebagai berikut :



maka diperolah nilai integral fitur putih= { 6,15,19,27,27 }dan nilai integral fitur hitam = { 11,29,35,46,48 }. Maka nilai *haar feature* tersebut adalah :

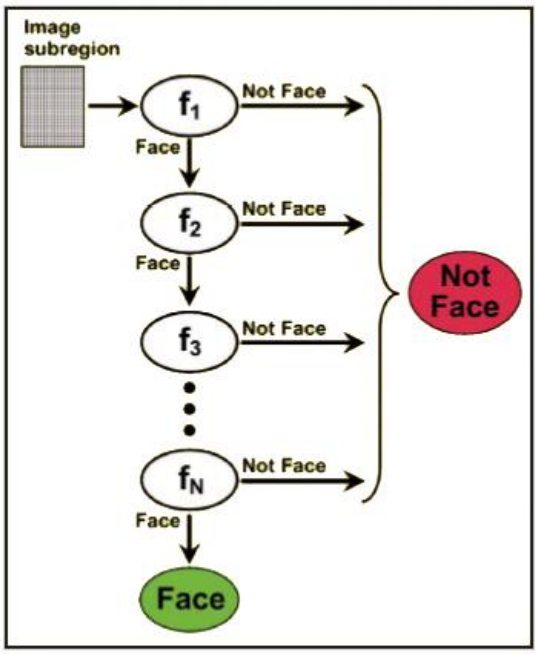
f(x) = (11 + 29 + 35 + 46 + 48) - (6 + 15 + 19 +27 + 27) = 75

Untuk perhitungan integral image di dalam sistem ini terdapat di class TargetImage.

3.1.3.3 Proses Filterisasi *Cascade Classifier*

Pada proses ini menggunakan klasifikasi bertingkat. Filter pada masing-masing level mengklasifikasikan gambar yang sebelumnya telah difilter. Selama penggunaannya, jika satu dari filter-filter tersebut gagal, image region / daerah

pada gambar diklasifikasikan sebagai “Bukan Wajah”. Saat filter berhasil melewatkan *image region*, *image region* kemudian masuk pada filter yang selanjutnya. *Image region* yang telah melalui semua filter akan dianggap sebagai “Wajah”. Berikut ini adlah rantai filter Cascade Classifier



Gambar 3.7 Rantai Filter Cascade Classifier

Untuk dapat menentukan wajah atau bukan wajah, Haar menggunakan sebuah pelatihan. Proses pelatihan ini dikenal dengan algoritma *haartraining* yang pada akhir pelatihan akan menghasilkan parameter model statistik. Pelatihan dengan model statistik haar menggunakan beberapa sampel, yaitu :

1. *Positive samples*

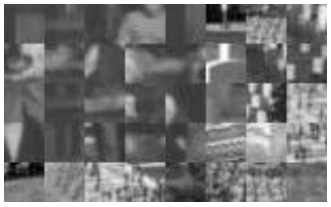
   Berisi gambar objek yang ingin di deteksi, apabila ingin mendeteksi wajah maka *positive samples* ini berisi gambar wajah.



Gambar 3.8 Contoh *Positive Samples*

2. *Negative samples*

   Berisi gambar objek selain objek yang ingin dikenali, umumnya berupa gambar *background* seperti tembok, pemandangan, lantai, dan gambar lainnya



Gambar 3.9 Contoh *Negative Samples*

Berikut adalah pengujian sampel dan membuat sampel dengan *createsamples utility* :

```
Usage: ./createsamples
  [-info <description_file_name>]
  [-img <image_file_name>]
  [-vec <vec_file_name>]
  [-bg <background_file_name>]
  [-num <number_of_samples = 1000>]
  [-bgcolor <background_color = 0>]
  [-inv] [-randinv] [-bgthresh <background_color_threshold = 80>]
  [-maxidev <max_intensity_deviation = 40>]
  [-maxxangle <max_x_rotation_angle = 1.100000>]
  [-maxyangle <max_y_rotation_angle = 1.100000>]
  [-maxzangle <max_z_rotation_angle = 0.500000>]
  [-show [<scale = 4.000000>]]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
```

kedua jenis sampel dilatihkan bersamaan dan perbedaannya digunakan sebagai parameter klasifikasi objek terdeteksi sebagai wajah atau tidak.

Sampel data yang telah dibuat dilatih menggunakan *haartraining utility*. Berikut ini adalah pelatihan data dengan *haartraining utility* :

```
Usage: ./haartraining
  -data <dir_name>
  -vec <vec_file_name>
  -bg <background_file_name>
  [-npos <number_of_positive_samples = 2000>]
  [-nneg <number_of_negative_samples = 2000>]
  [-nstages <number_of_stages = 14>]
  [-nsplits <number_of_splits = 1>]
  [-mem <memory_in_MB = 200>]
  [-sym (default)] [-nonsym]
  [-minhitrate <min_hit_rate = 0.995000>]
  [-maxfalsealarm <max_false_alarm_rate = 0.500000>]
  [-weighttrimming <weight_trimming = 0.950000>]
  [-eqw]
  [-mode <BASIC (default) | CORE | ALL>]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
  [-bt <DAB | RAB | LB | GAB (default)>]
  [-err <misclass (default) | gini | entropy>]
  [-maxtreesplits <max_number_of_splits_in_tree_cascade = 0>]
  [-minpos <min_number_of_positive_samples_per_cluster = 500>]
```

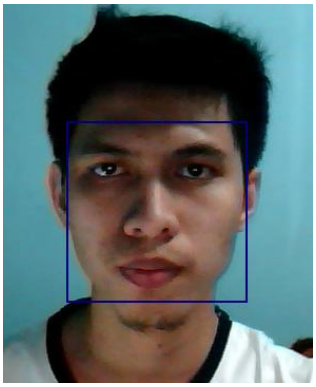
Ketika proses training sudah selesai makan haartraining akan menghasilkan file .xml dan file tersebut adalah *haarcascade_frontalface_alt.xml*. *haarcascade_frontalface_alt* adalah *training* yang dikhususkan untuk pelacakan wajah dengan posisi geometris lurus ke depan. Berikut ini adalah file .xml hasil *training* :

```
<opencv_storage>
<haarcascade_frontalface_alt type_id="opencv-haar-classifier">
  <size>20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>3 7 14 4 -1.</_>
```

```
                <_>3 9 14 2 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>4.0141958743333817e-003</threshold>
            <left_val>0.0337941907346249</left_val>
            <right_val>0.8378106951713562</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>1 2 18 4 -1.</_>
                <_>7 2 6 4 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>0.0151513395830989</threshold>
            <left_val>0.1514132022857666</left_val>
            <right_val>0.7488812208175659</right_val></_></_>
        <_>
           <!-- tree 2 -->
          ...
<stage_threshold>0.8226894140243530</stage_threshold>
<parent>-1</parent>
<next>-1</next></_>
<_>
  <!-- stage 1 -->
  ...
  <!-- stage 21 -->
       <trees>
       <_>
             <!-- tree 0 -->
             …
             <!-- tree 212 -->
  <stage_threshold>105.7611007690429700</stage_threshold>
  <parent>20</parent>
  <next>-1</next></_></stages></haarcascade_frontalface_alt>
</opencv_storage>
```

Training memiliki isi yang berbeda baik dari segi jumlah *stage*, jumlah *tree*, model segi empat dari fitur (*rectangle*) maupun nilai *treshold* nya. *Stage* melambangkan banyaknya tingkatan dalam *cascade of classifier*, dalam *training* ini digunakan 22 tingkatan (stage 0 sampai stage 21). Tingkatan ini digunakan untuk mengurangi jumlah *sub window* citra yang perlu diperiksa. Di tingkatan pertama dilakukan pengklasifikasian terhadap seluruh *sub window* citra, lalu di tingkatan kedua dilakukan pengklasifikasian terhadap *sub window* yang berasal dari hasil pengklarifikasian tingkatan pertama. Maka semakin tinggi tingkatanya semakin sedikiit jumlah *sub window* yang harus diperiksa. Di tiap tingkatan terdiri dari beberapa *tree*, biasanya semakin tinggi tingkatan maka *tree* yang terdapat di

dalamnya pun semakin banyak. Pada *stage* 0 terdapat 2 *tree* dan pada *stage* 21 terdapat 212 *tree*. Setelah proses pendeteksian wajah berhasil dilakukan, maka sistem akan menetukan wajah dan tidak. Pada citra yang mengandung wajah dilakukan proses pemberian kotak di sekitar wajah. Hal ini untuk mengecek apakah proses pendeteksian wajah ini telah berhasil mengenali wajah dengan tepat.



Gambar 3.10 Contoh Wajah Terdeteksi Oleh *Haar Cascade Classifier*

3.1.4 Analisis Spesifikasi Kebutuhan Perangkat Lunak

Analisis spesifikasi kebutuhan perangkat lunak dibagi menjadi dua bagian yaitu SKPL-F (Spesifikasi kebutuhan perangkat lunak fungsional) dan SKPL-NF (Spesifikasi kebutuhan perangkat lunak non-fungsional). Analisis tersebut berisi deskripsi dari kebutuhan perangkat lunak yang akan dibangun baik kebutuhan fungsional maupun kebutuhan non fungsional.

3.1.4.1 Analisis Spesifikasi Kebutuhan Perangkat Lunak Fungsional

Analisis spesifikasi kebutuhan perangkat lunak fungsional berisi pernyataan dari layanan sistem yang harus disediakan, bagaimana sistem harus bereaksi terhadap input tertentu dan bagaimana sistem harus berperilaku dalam

situasi tertentu. Intinya menjelaskan fungsionalitas dari sistem yang akan dibangun secara detail, berikut ini table spesifikasi kebutuhan perangkat lunak fungsional :

Tabel 3.2 Spesifikasi kebutuhan perangkat lunak fungsional

Nomor	Spesifikasi Kebutuhan Perangkat Lunak Fungsional
SKPL-F-001	Sistem harus bisa terhubung dengan kamera yang terpasang di komputer.
SKPL-F-002	Sistem dapat mendeteksi posisi wajah di depan kamera
SKPL-F-003	Sistem dapat menyediakan berbagai pilihan model rambut
SKPL-F-004	Sistem dapat menampilkan objek rambut ke posisi wajah (di kepala atas)
SKPL-F-005	Sistem dapat menyediakan ruang 3D untuk user memilih model rambut

3.1.4.2 Analisis Spesifikasi Kebutuhan Perangkat Lunak Non Fungsional

Analisis spesifikasi kebutuhan perangkat lunak non fungsional adalah deskripsi mengenai batasan dari layanan dan fungsi-fungsi dari sistem yang akan dibangun. Berikut ialah table spesifikasi kebutuhan perangkat lunak non fungsional peran yang akan dibangun :

Tabel 3.3 Spesifikasi kebutuhan perangkat lunak non fungsional

Nomor	Spesifikasi Kebutuhan Perangkat Lunak Non Fungsional
SKPL-NF-001	Sistem membutuhkan minimum spesifikasi kamera 2 megapiksel.
SKPL-NF-002	Sistem dapat mendeteksi wajah dimana posisi wajah menghadap tegak ke kamera.
SKPL-NF-003	Wajah dapat terdeteksi oleh sistem apabila memiliki kelengkapan wajah pada umumnya. Seperti , memiliki dua mata, hiung dan mulut.

SKPL-NF-004	Sistem membutuhkan cahaya yang berada di depan kamera.
SKPL-NF-005	Sistem hanya dibangun dalam bentuk *dekstop*
SKPL-NF-006	Objek yang dapat ditampilkan hanya yang berekstenti *.DAE

3.1.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional adalah sebuah langkah untuk menganalisis sumber daya manusia yang akan menggunakan perangkat lunak yang dibangun, perangkat keras dan perangkat lunak yang dimiliki sesuai dengan kebutuhan atau permintaan, sehingga dapat ditentukan kompabilitas perangkat lunak yang dibangun terhadap sumber daya yang ada. Karna program ini berbasis *Augmented Reality* yang menggunakan *Markerless Face Recognition* sebagai tracking *marker* maka dibutuhkan komputer atau laptop dan perangkat lunak sebagai pengolah citra dan beberapa set *marker*.

3.1.5.1 Analisis Perangkat Keras

Analisis perangkat keras dimaksudkan untuk mengetahui spesifikasi perangkat keras yang akan digunakan oleh HEADCORE BARBERSHOP. Perangkat keras yang dianjurkan untuk menggunakan aplikasi *Augmented Reality* adalah sebagai berikut :

1. *Processor* Pentium 4 > 1 GHz atau setara
2. RAM minimal 512 MB
3. Ruang sisa *hardisk* minimal 10 GB
4. *Motherboard* dengan *chipset* yang kompatibel dengan VGA *card* yang dipakai
5. Memiliki opsi konektifitas USB 2.0 dan nirkabel

6. Optimum menggunakan VGA *card* dengan kemampuan me-*render* grafis 3D. seperti GeForce 6xxx series atau ATI 1xxx series

3.1.5.2 Analisis Perangkat Lunak

Perangkat lunak yang dibutuhkan dalam pembangunan aplikasi *Augmented Reality* di HEADCORE BARBERSHOP ini adalah sebagai berikut :

1. Sistem operasi XP
2. IDE FlashDevelop 4.1
3. Flex Builder 4.0
4. Library Marilena
5. Adobe Flash Player 11.1
6. Adobe Photoshop CS3
7. Macromedia Director

3.1.5.3 Analisis Pengguna

Pada umumnya pengunjung BarberShop berasal dari berbagai kalangan dan latar belakang yang berbeda. Untuk dapat menggunakan aplikasi ini dibutuhkan karateristik pengguna seperti berikut :

1. Setidaknya pengguna pernah tahu apa itu *Augmented Reality* dan pernah menggunakan komputer dan tahu fungsi tombol keyboard secara umum.
2. Pengguna tidak memiliki cacat pada bagian wajah, seperti tidak memiliki mata.

Sedangkan kemampuan yang harus dimiliki karyawan BarberShop yang bertugas untuk mengelola perangkat lunak ini yaitu sebagai berikut :

1. Karyawan yang memiliki keterampilan menggunakan *software* 3ds Max untuk membuat model rambut dan Macromedia Directore untuk membuat aplikasi.
2. Karyawan dapat mengerti bagaimana mengkonfigurasi dan memasukan objek ke dalam aplikasi.

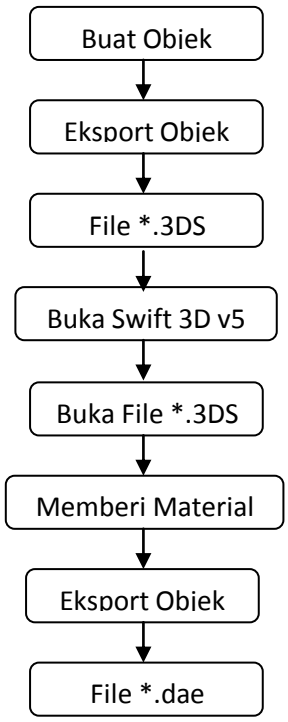
Karyawan ini dapat dijadikan karyawan tetap atau karyawan yang sewaktu-waktu hanya dibutuhkan untuk membuat mengembangkan aplikasi ini.

3.1.5.4 Analisis Proses Pembuatan Objek

Objek yang dibuat menggunakan *software* 3D MAX 10. Dalam proses pemodelan terdiri dari 5 langkah.

1. Membuat objek 3D dengan menggunakan 3D MAX 10.
2. Mengeksport objek yang sudah di buat menjadi file *.3DS.
3. Membuka aplikasi Swift 3D v5 dan membuka file *.3DS yang sudah di buat.
4. Menambahkan material pada rambut.
5. Mengeksport objek ke Papervision3D agar mendapatkan file *.dae.

Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 3.11 Tahapan Pemodelan

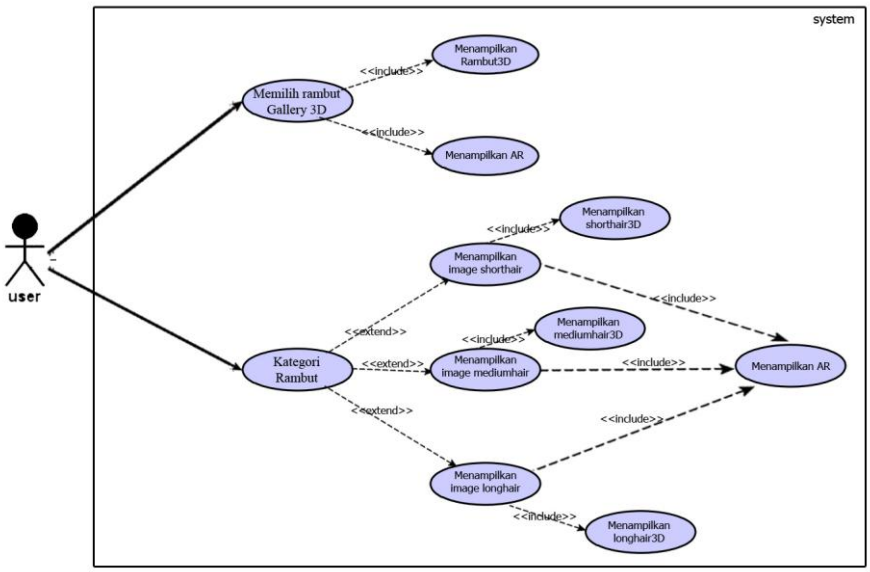
3.1.6 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional diperlukan untuk memodelkan sistem. Pemodelan yang digunakan untuk memodelkan aplikasi *Augmented Reality* untuk BarberShop ini adalah pemodelan berorientasi objek. Perangkat lunak ini dimodelkan menggunakan UML (*Unified Modeling Language*).

3.1.6.1 Diagram *Use Case*

Usecase Diagram merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada aplikasi. Sasaran permodelan usecase diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan sistem yang akan dibangun. Dari hasil analisis aplikasi yang ada

maka usecase diagram untuk aplikasi *Augmented Reaity* di BarberShop dapat di lihat pada gambar berikut :



Gambar 3.12 *Usecase Diagram*

Bagian ini menjelaskan masing-masing deskripsi dari *Use Case* diagram yang dilakukan oleh pengguna. Mulai dari proses memiih rambut *Gallery3D*, menampilkan rambut3D , menampilkan ARRambut, dan katagori rambut.

1. Memilih Rambut *Gallery* 3D

   Interaksi antara pelanggan dengan *usecase* memilih rambut *Gallery* 3D dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.4 *Usecase Skenario* Memilih Rambut *Gallery* 3D

Nama	Memilih Rambut *Gallery* 3D
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman utama
Aksi Aktor	**Reaksi Sistem**

1. Melihat dan memilih salah satu model rambut yang diinginkan	
	2. Menampilkan menu untuk melihat rambut 3D dan menu ARRambut
Kondisi Akhir	Halaman model rambut

2. Menampilkan Rambut3D

Interaksi antara pelanggan dengan *usecase* menampilkan rambut3D dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.5 *Usecase Skenario* Menampilkan Rambut3D

Nama	Menampilkan Rambut3D
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman model rambut
Aksi Aktor	**Reaksi Sistem**
1. Menekan *button* lihat rambut 3D.	
	2. Membuka *form* Rambut3D
	3. Merender rambut yang dipilih
	4. Menampilkan rambut
5. Pengguna dapat memutar untuk melihat rambut dari berbagai sisi	
	6. Rambut berputar sesuai yang diinginkan pengguna
Kondisi Akhir	Halaman rambut3D.

3. Menampilkan ARRambut

Interaksi antara pelanggan dengan *usecase* menampilkan ARRambut dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.6 *Usecase Skenario* Menampilkan ARRambut

Nama	Menampilkan ARRambut	
Aktor	Pelanggan	
Skenario Utama		
Kondisi Awal	Halaman model rambut	
Aksi Aktor		**Reaksi Sistem**
1. Menekan *button* ARRambut		
		2. Membuka *form* ARRambut
		3. Merender rambut yang dipilih
		4. Menampilkan model rambut
5. Menghadap wajah ke kamera		
		6. Membaca bentuk wajah
		7. Rambut berada di wajah sesuai koordinat yang ditentukan.
8. Melakukan pergerakan untuk mencocokan rambut dengan wajah		
Kondisi Akhir	Halaman ARRambut.	

4. Kategori rambut

Interaksi antara pelanggan dengan *usecase* kategori rambut dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.7 *Usecase Skenario* Kategori Rambut

Nama	Kategori Rambut	
Aktor	Pelanggan	
Skenario Utama		
Kondisi Awal	Halaman Utama	
Aksi Aktor		**Reaksi Sistem**
1. Melihat dan memilih salah satu model rambut berdasarkan jenis		
		2. Membuka *form* jenis rambut yang dipilih pengguna
Kondisi Akhir	Halaman rambut berdasarkan jenis	

5. Menampilkan *image short hair*

Interaksi antara pelanggan dengan *usecase* menampilkan *image short hair* dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.8 *Usecase Skenario* Menampilkan *Image Short Hair*

Nama	Menampilkan *Image Short Hair*	
Aktor	Pelanggan	
Skenario Utama		
Kondisi Awal	Halaman Utama	
Aksi Aktor		**Reaksi Sistem**
1. Menekan *button Short Hair*		
		2. Membuka *form Short Hair*
Kondisi Akhir	Halaman *image short hair*	

6. Menampilkan *image medium hair*

   Interaksi antara pelanggan dengan *usecase* menampilkan *image medium hair* dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.9 *Usecase Skenario* Menampilkan *Image Medium Hair*

Nama	Menampilkan *Image Medium Hair*
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman Utama
Aksi Aktor	**Reaksi Sistem**
1. Menekan *button Medium Hair*	
	2. Membuka *form Medium Hair*
Kondisi Akhir	Halaman *image medium hair*

7. Menampilkan *image long hair*

   Interaksi antara pelanggan dengan *usecase* menampilkan *image long hair* dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.10 *Usecase Skenario* Menampilkan *Image Long Hair*

Nama	Menampilkan *Image Long Hair*
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman Utama
Aksi Aktor	**Reaksi Sistem**
3. Menekan *button Long Hair*	
	4. Membuka *form Long Hair*
Kondisi Akhir	Halaman *image long hair*

8. Menampilkan *short hair* 3D

Interaksi antara pelanggan dengan *usecase* menampilkan *short hair* 3D dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.11 *Usecase Skenario* Menampilkan *Short Hair* 3D

Nama	Menampilkan *Short Hair* 3D
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman *image short hair*
Aksi Aktor	**Reaksi Sistem**
1. Memilih salah satu *image*.	
2. Menekan *button* lihat rambut 3D.	
	3. Membuka *form* Rambut3D
	4. Merender rambut yang dipilih
	5. Menampilkan rambut
6. Pengguna dapat memutar untuk melihat rambut dari berbagai sisi	
	7. Rambut berputar sesuai yang diinginkan pengguna
Kondisi Akhir	Halaman rambut3D.

9. Menampilkan *medium hair* 3D

Interaksi antara pelanggan dengan *usecase* menampilkan *medium hair* 3D dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.12 *Usecase Skenario* Menampilkan *Medium Hair* 3D

Nama	Menampilkan *Medium Hair* 3D
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman *image medium hair*
Aksi Aktor	**Reaksi Sistem**
1. Memilih salah satu *image*.	
2. Menekan *button* lihat rambut 3D.	
	3. Membuka *form* Rambut3D
	4. Merender rambut yang dipilih
	5. Menampilkan rambut
6. Pengguna dapat memutar untuk melihat rambut dari berbagai sisi	
	7. Rambut berputar sesuai yang diinginkan pengguna
Kondisi Akhir	Halaman rambut3D.

10. Menampilkan *long hair* 3D

Interaksi antara pelanggan dengan *usecase* menampilkan *long hair* 3D dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.13 *Usecase Skenario* Menampilkan *Long Hair* 3D

Nama	Menampilkan *Long Hair* 3D
Aktor	Pelanggan
Skenario Utama	
Kondisi Awal	Halaman *image long hair*
Aksi Aktor	**Reaksi Sistem**
1. Memilih salah satu *image*.	
2. Menekan *button* lihat rambut 3D.	
	3. Membuka *form* Rambut3D
	4. Merender rambut yang dipilih
	5. Menampilkan rambut
6. Pengguna dapat memutar untuk melihat rambut dari berbagai sisi	
	7. Rambut berputar sesuai yang diinginkan pengguna
Kondisi Akhir	Halaman rambut3D.

11. Menampilkan ARRambut

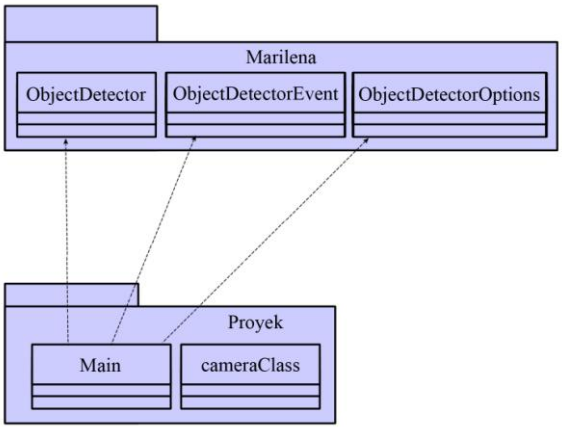
Interaksi antara pelanggan dengan *usecase* menampilkan ARRambut dijelaskan dalam *usecase* skenario sebagai berikut :

Tabel 3.14 *Usecase Skenario* Menampilkan ARRambut

Nama	Menampilkan ARRambut	
Aktor	Pelanggan	
Skenario Utama		
Kondisi Awal	Halaman model rambut	
Aksi Aktor		**Reaksi Sistem**
1. Menekan *button* ARRambut		
		2. Membuka *form* ARRambut
		3. Merender rambut yang dipilih
		4. Menampilkan model rambut
5. Menghadap wajah ke kamera		
		6. Membaca bentuk wajah
		7. Rambut berada di wajah sesuai koordinat yang ditentukan.
8. Melakukan pergerakan untuk mencocokan rambut dengan wajah		
Kondisi Akhir	Halaman ARRambut.	

3.1.6.2 *Package Diagram*

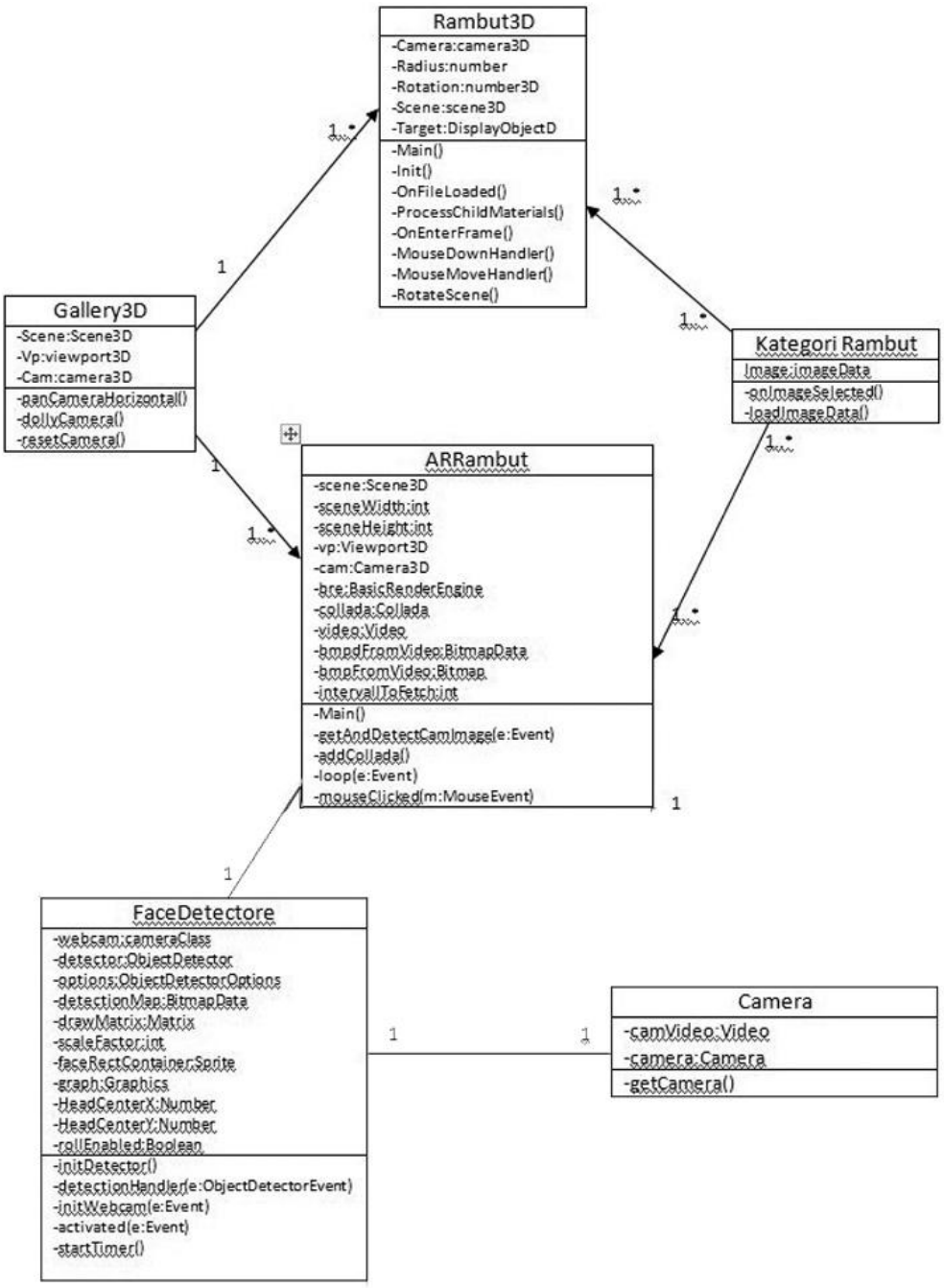
Package diagram digunakan untuk menjelaskan hubungan modularitas, enkapsulasi, dan abstraksi pada pengembangan sistem yang kompleks. *Package diagram* juga digunakan untuk menjelaskan hubungan *dependency* dan *visibility* antar *class* yang memiliki asosiai secara modular dengan *class* lain dalam sistem. *Package diagram* memungkinkan perancangan sistem bisa di mengerti dengan tingkat pemahaman yang lebih tinggi pada kolaborasi antar elemen model. Dengan menggunakan *package diagram* perancangan *class diagram* bisa di modularisasi secara lebih terstruktur. *Package diagram* sangat dibutuhkan ketika *class diagram* yang dibangun sudah semakin kompleks, sehingga penggambaran *class diagram* bisa dibangun secara terpisah dengan menggunakan *Package diagram* untuk mewakili hubungan yang antar class pada *package* yang berbeda. Pada halaman berikutnya merupakan *package diagram* dari *virtual accessories room*.



Gambar 3.13 *Package Diagram*

3.1.6.3 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. *Class* dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.



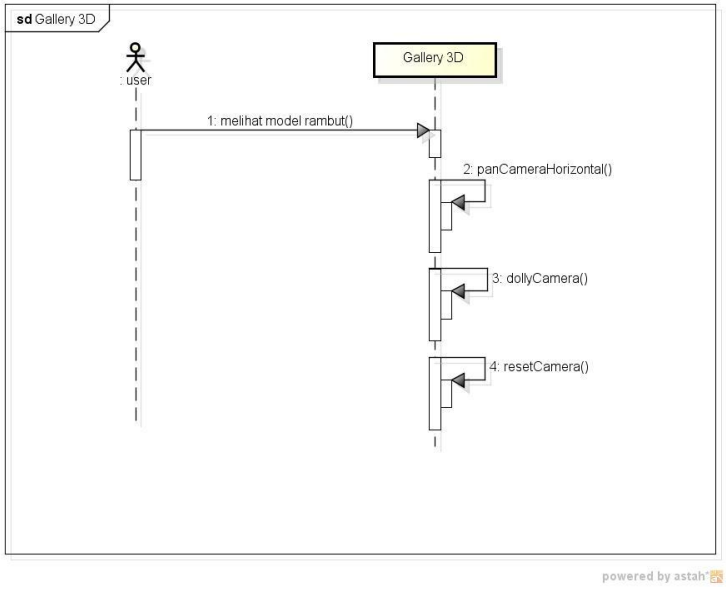
Gambar 3.14 ***Class Diagram***

3.1.6.4 *Sequence Diagram*

Sequence diagram menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. *Sequence diagram* menggambarkan interaksi antara objek di dalam dan disekitar sistem berupa pesan yang digambarkan terhadap waktu.

1) Sequence diagram memilih rambut *Gallery 3D*

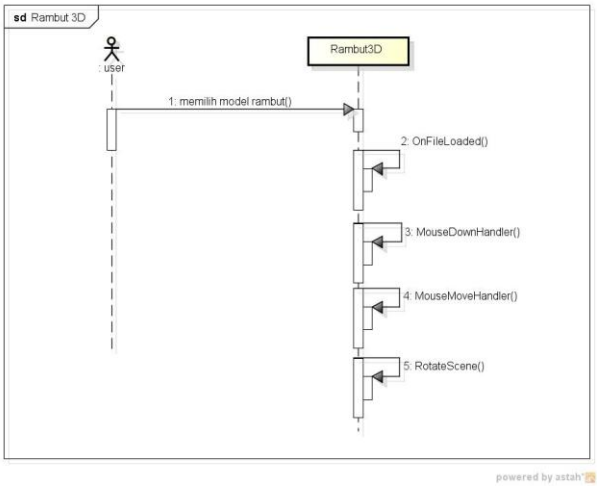
Interaksi antara user dengan *use case* memilih rambut *Gallery 3D* dijelaskan dalam *sequence diagram* sebagai berikut :



Gambar 3.15 *Sequence Diagram* Memilih Rambut *Gallery 3D*

2) Sequence diagram menampilkan Rambut3D

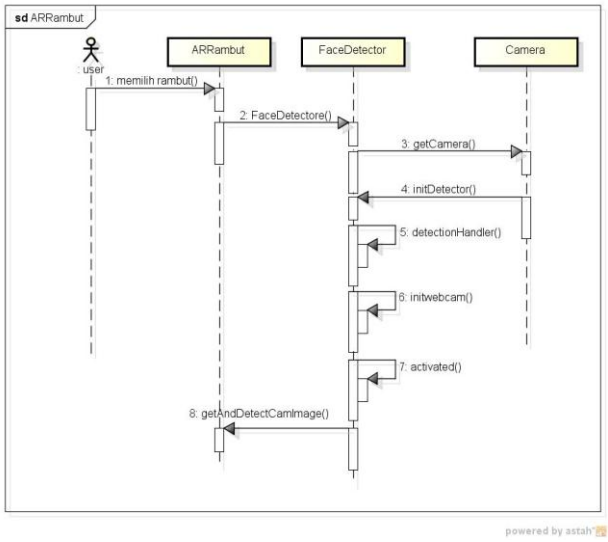
Interaksi antara user dengan *use case* menampilkan Rambut3D dijelaskan dalam *sequence diagram* sebagai berikut :



Gambar 3.16 *Sequence Diagram* Menampilkan Rambut3D

3) Sequence diagram menampilkan ARRambut

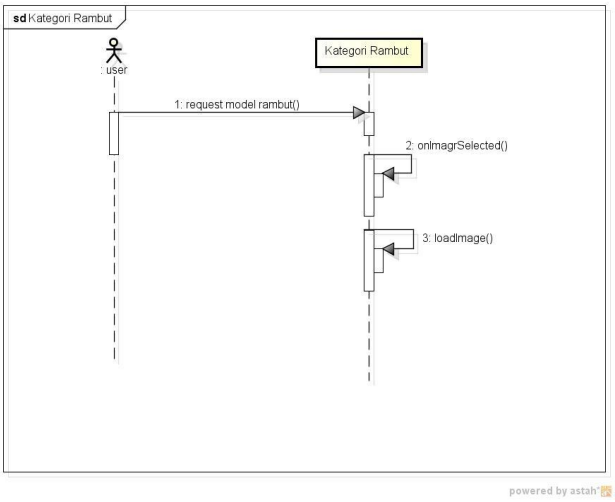
Interaksi antara user dengan *use case* menampilkan ARRambut dijelaskan dalam *sequence diagram* sebagai berikut.



Gambar 3.17 *Sequence Diagram* Menampilkan ARRambut

4) Sequence diagram kategori rambut

Interaksi antara user dengan *use case* kategori rambut dijelaskan dalam *sequence diagram* sebagai berikut



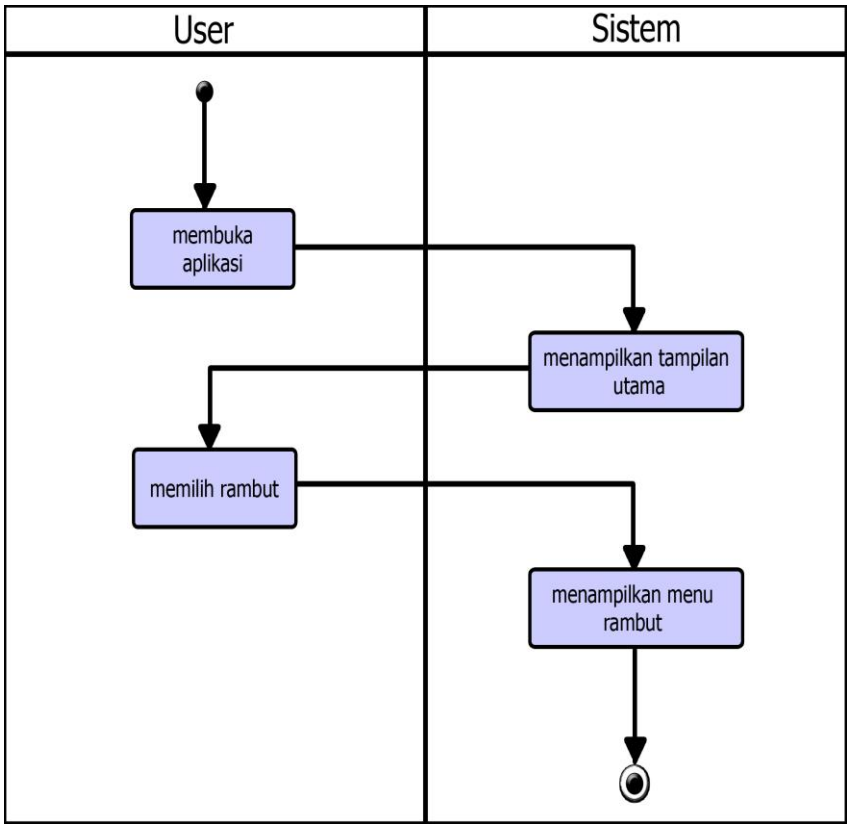
Gambar 3.18 *Sequence Diagram* Kategori Rambut

3.1.6.5 *Activity Diagram*

Activity diagram merupakan *state* diagram khusus, dimana sebagian besar state adalah *action*. Menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi. *Activity diagram* juga dapat menggambarkan proses parallel yang mungkin terjadi pada beberpa eksekusi.

1. *Activity Diagram* Memilih Rambut *Gallery* 3D

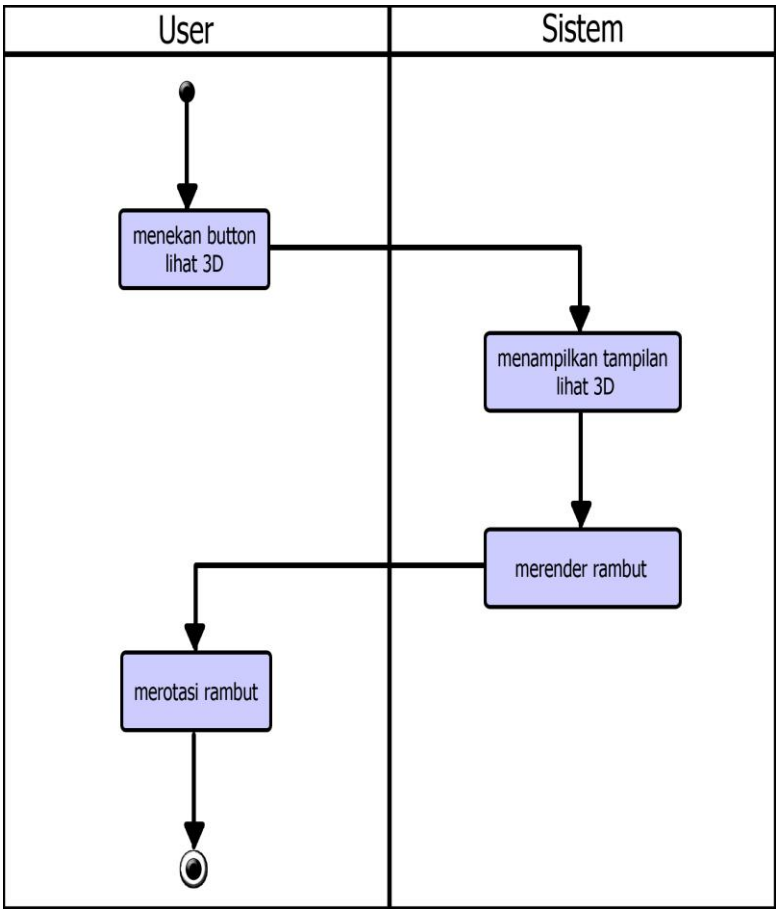
   Activity diagram Memilih Rambut *Gallery* 3D menjelaskan aliran kerja user pada saat memilih rambut.



Gambar 3.19 *Activity Diagram* Memilih Rambut *Gallery* 3D

2. *Activity Diagram* Menampilkan Rambut3D

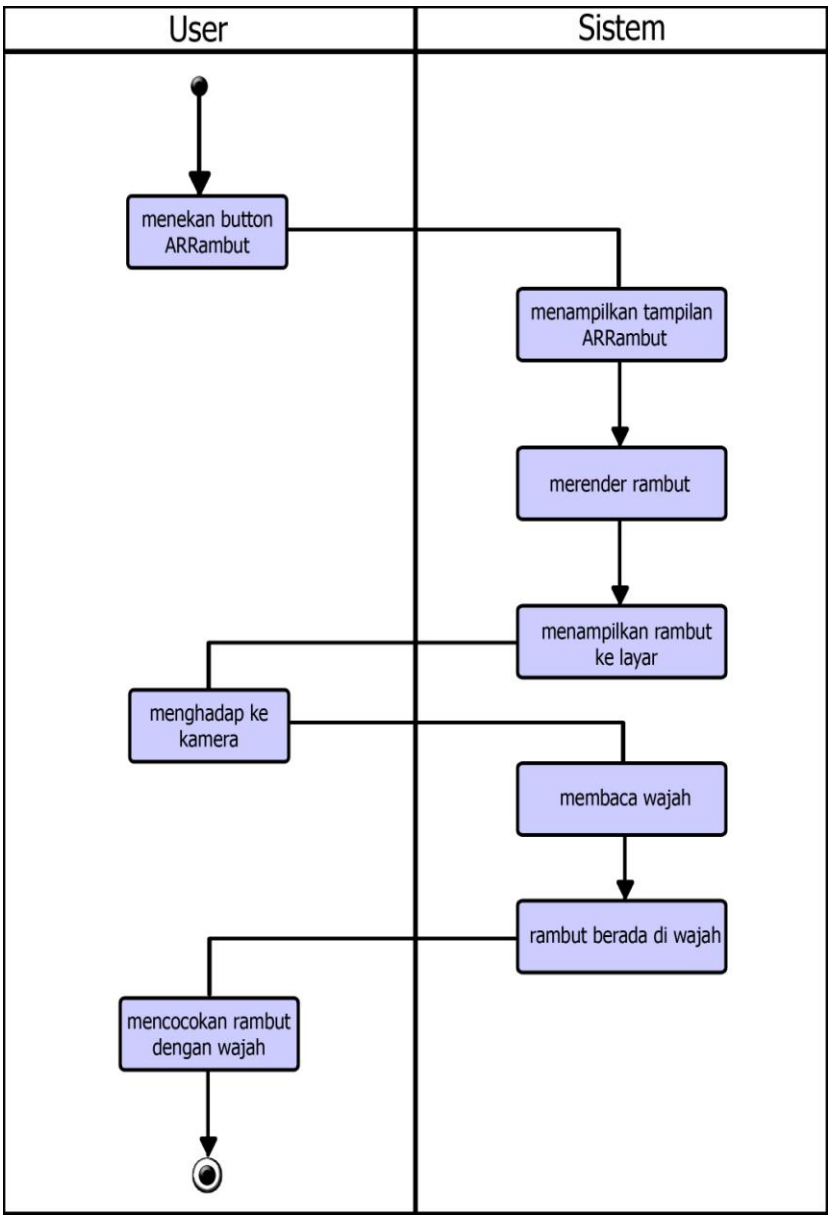
Activity diagram Menampilkan Rambut3D menjelaskan aliran kerja user pada saat melihat rambut3D.



Gambar 3.20 *Activity Diagram* Menampilkan Rambut3D

3. *Activity Diagram* Menampilkan ARRambut

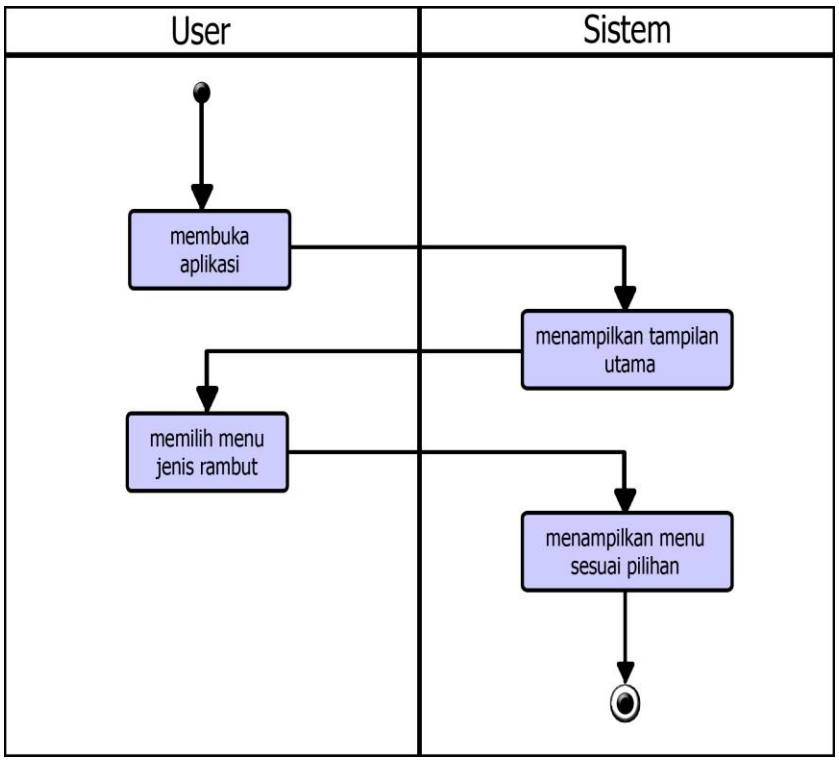
Activity diagram Menampilkan ARRambut menjelaskan aliran kerja user pada saat mencoba *Augmented Reality.*



Gambar 3.21 *Activity Diagram* Menampilkan ARRambut

4. *Activity Diagram* kategori rambut

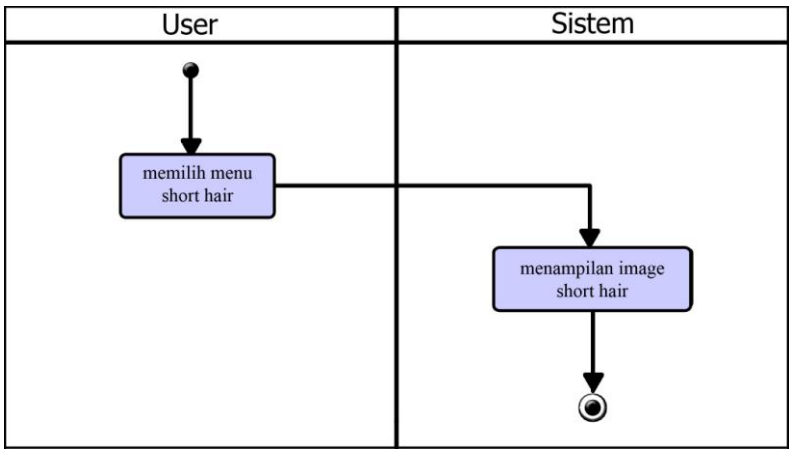
Activity diagram kategori rambut menjelaskan aliran kerja user pada saat melihat jenis rambut yang tersedia.



Gambar 3.22 *Activity Diagram* Kategori Rambut

5. *Activity Diagram* Menampilkan *image short hair*

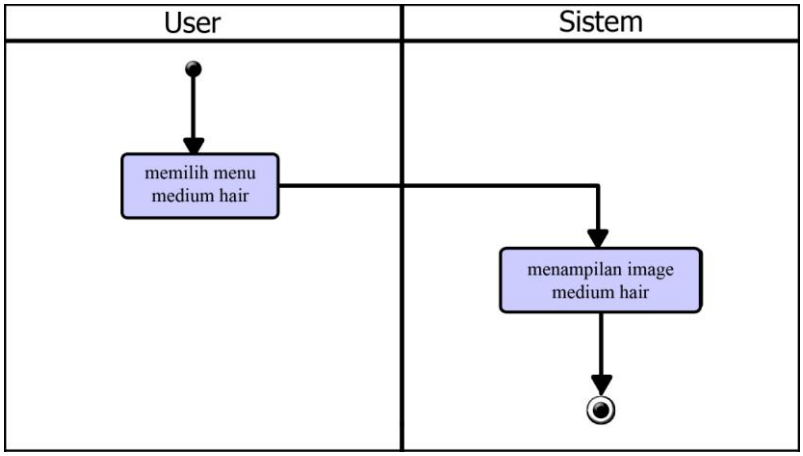
   Activity diagram Menampilkan *image short hair* menjelaskan aliran kerja user pada saat memilih menu *short hair*.



Gambar 3.23 *Activity Diagram* Menampilkan *Image Short Hair*

6. *Activity Diagram* Menampilkan *image medium hair*

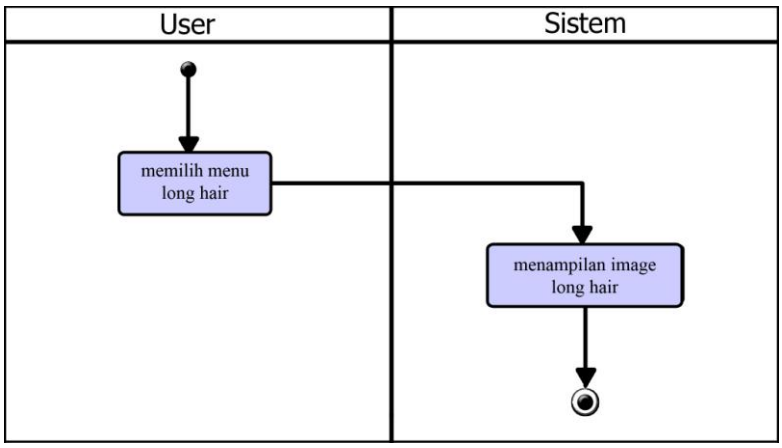
   Activity diagram Menampilkan *image medium hair* menjelaskan aliran kerja user pada saat memilih menu *medium hair*.



Gambar 3.24 *Activity Diagram* Menampilkan *Image Medium Hair*

7. *Activity Diagram* Menampilkan *image long hair*

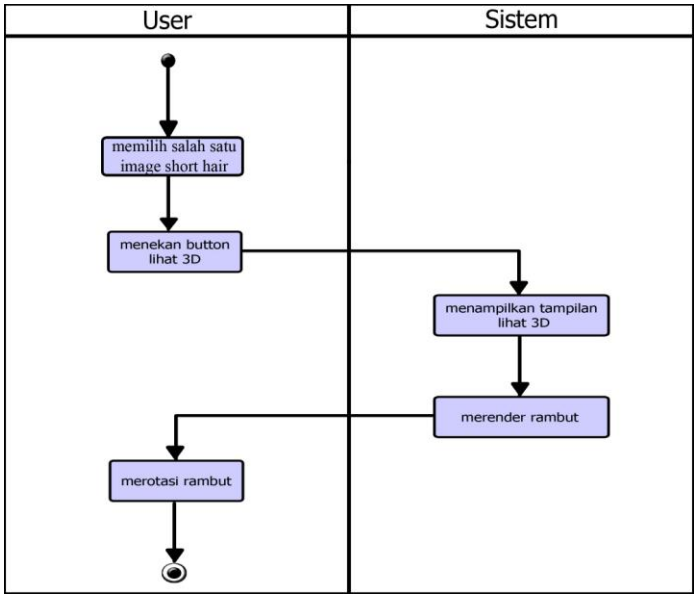
   Activity diagram Menampilkan *image long hair* menjelaskan aliran kerja user pada saat memilih menu *long hair*.



Gambar 3.25 *Activity Diagram* Menampilkan *Image Long Hair*

8. Menampilkan *short hair* 3D

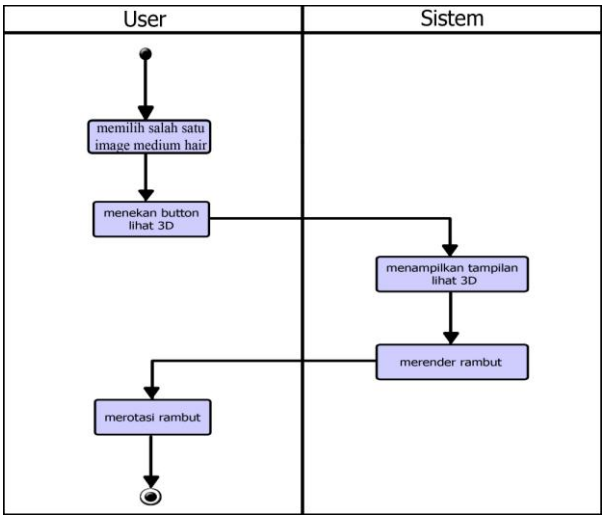
   Activity diagram Menampilkan *short hair* 3D menjelaskan aliran kerja user pada saat ingin menampilkan *short hair* 3D.



Gambar 3.26 *Activity Diagram* Menampilkan *Short Hair* 3D

9. Menampilkan *medium hair* 3D

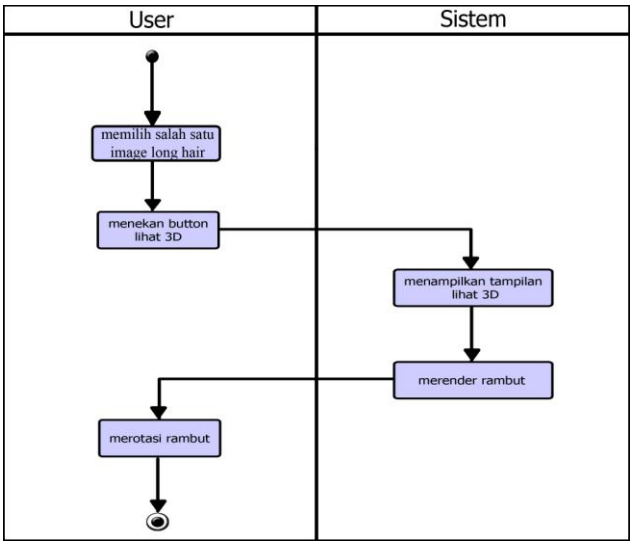
   Activity diagram Menampilkan *medium hair* 3D menjelaskan aliran kerja user pada saat ingin menampilkan *medium hair* 3D.



Gambar 3.27 *Activity Diagram* Menampilkan *Mediumt Hair* 3D

10. Menampilkan *long hair* 3D

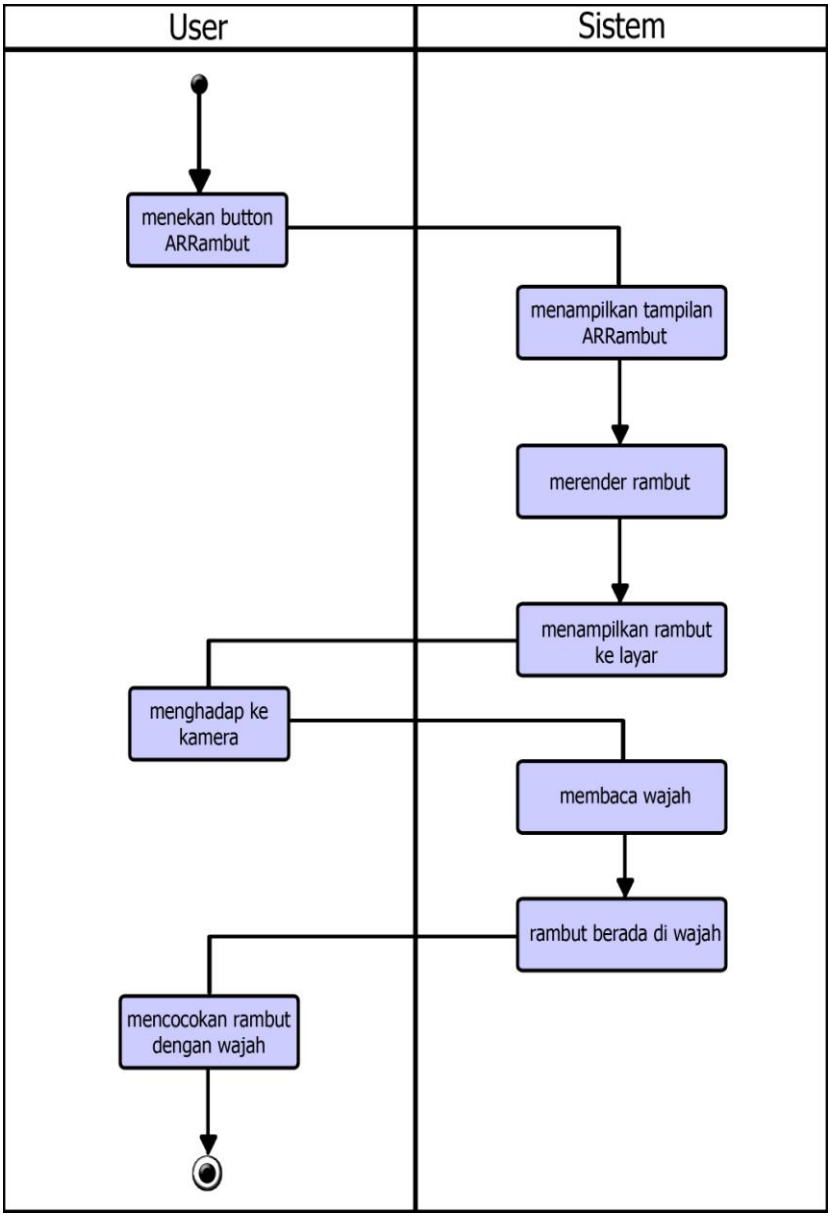
Activity diagram Menampilkan *long hair* 3D menjelaskan aliran kerja user pada saat ingin menampilkan *long hair* 3D.



Gambar 3.28 *Activity Diagram* Menampilkan *Long Hair* 3D

11. *Activity Diagram* Menampilkan ARRambut

Activity diagram Menampilkan ARRambut menjelaskan aliran kerja user pada saat mencoba *Augmented Reality.*



Gambar 3.29 *Activity Diagram* Menampilkan ARRambut

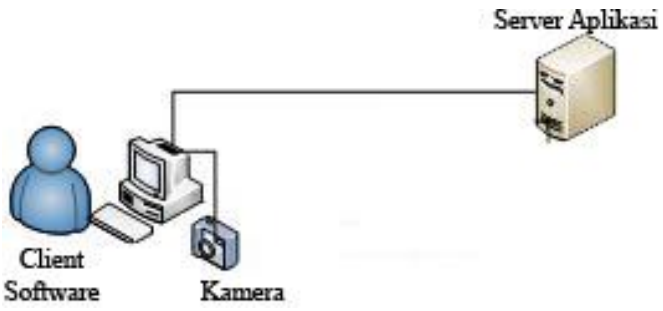
3.2 Perancangan Sistem

Tahap perancangan sistem dalam pembuatan suatu program merupakan suatu tahapan yang sangat penting, karena tahapan ini akan menentukan apakah

suatu program akan memiliki konstruksi yang baik sesuai dengan yang diinginkan, proses pengolahan data yang tepat dan akurat serta memiliki dasar-dasar untuk pengembangan selanjutnya. Desain yang dibuat berupa aplikasi yang akan digunakan oleh pelanggan, serta informasi yang dihasilkan program tersebut dapat dimengerti oleh pelanggan. Perancangan sistem bertujuan untuk memberikan gambaran sistem yang dibuat.

3.2.1 Perancangan Arsitektur Perangkat Lunak

Perancangan arsitektur merupakan perancangan yang dibuat sebelum perangkat lunak dibangun. Dalam aplikasi ini dibangun atas dua komponen yakni komponen basis data, komponen perangkat tengah. Berikut ini gambar dari perancangan arsitektur perangkat lunak :

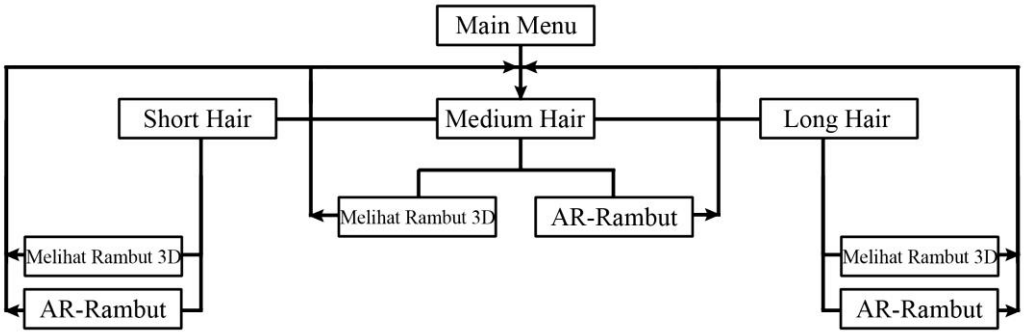


Gambar 3.30 Arsitektural Perangkat Lunak

3.2.2 Struktur Menu

Sturktur menu alah bentuk umum dari suatu rancangan program untukmemudahkan pemakaian dalam menjalanjan program komputer sehingga

pada saat menjalankan program komputer, user tidak mengalami kesulitan dalam memilih menu-menu yang diinginkan. Pada perancangan ini di buat menu yang dpat mengintegrasikan seuruh data dalam suatu sistem dan disertai dengan instruksi yang ada pada pilihan menu tersebut. Adapun menu tersebut dapat dilihat pada gambar berikut :

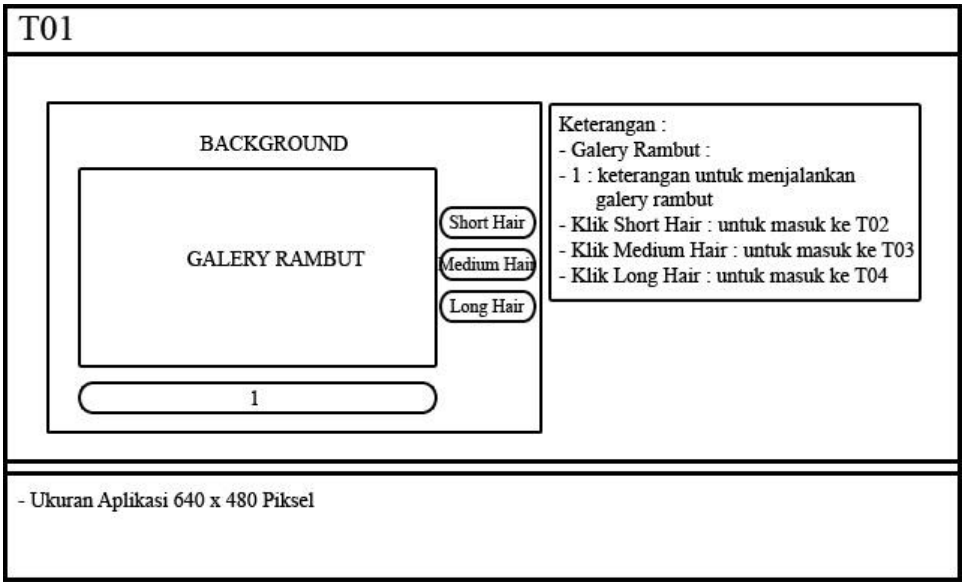


Gambar 3.31 Struktur Menu

3.2.3 Perancangan Antarmuka Perangkat Lunak

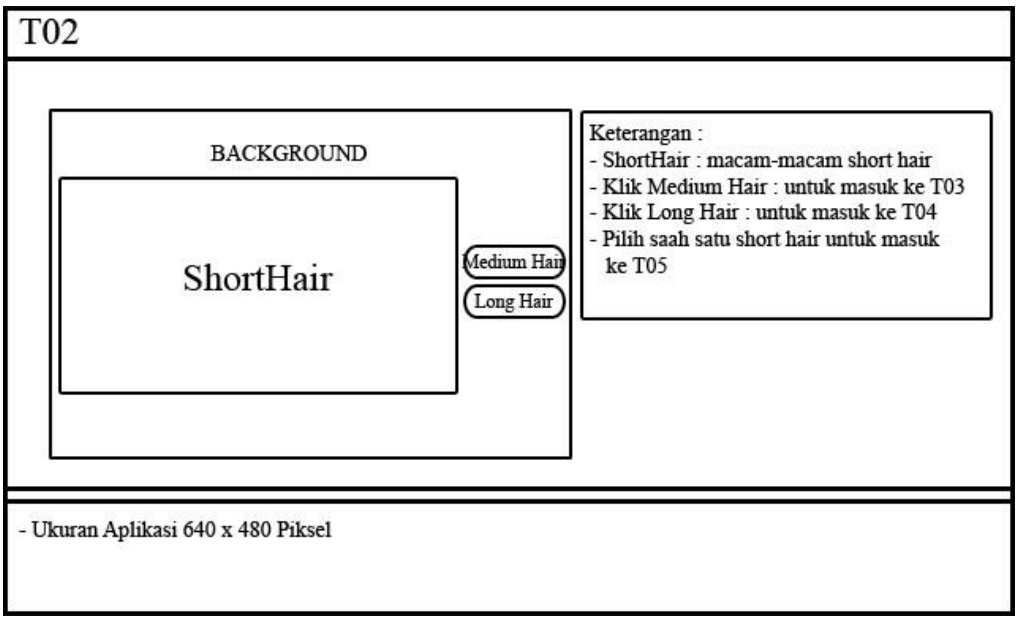
Interface atau antarmuka merupakan tampilan dari suatu perangkat lunak yang berperan sebagai media komunikasi yang digunakan sebagai sarana berdialog antara perangkat lunak dengan *user.*

3.2.3.1 Perancangan Antarmuka Tampilan Utama



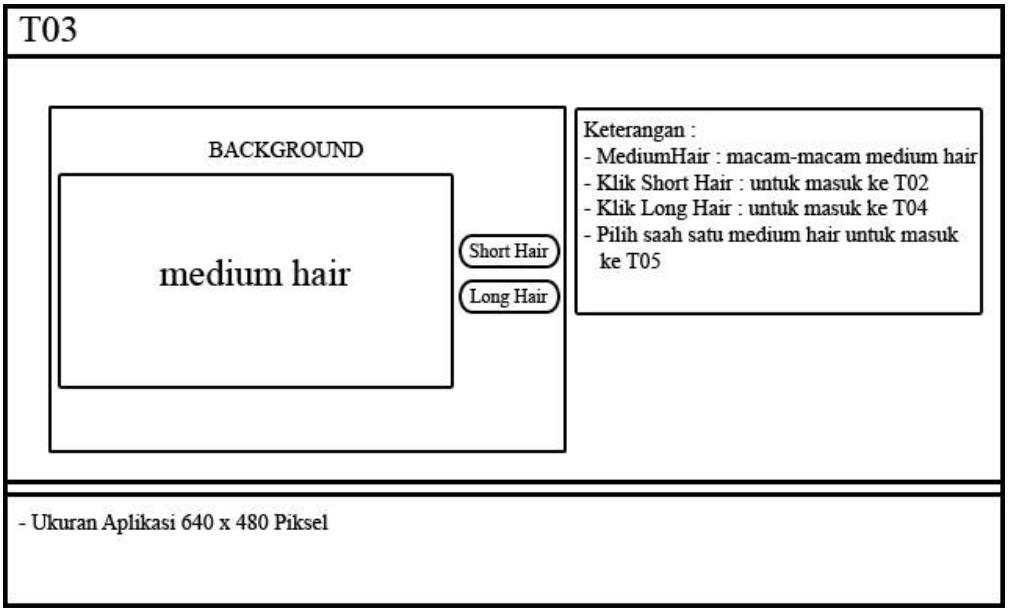
Gambar 3.32 Perancangan Antarmuka Tampilan Utama

3.2.3.2 Perancangan Antarmuka Short Hair



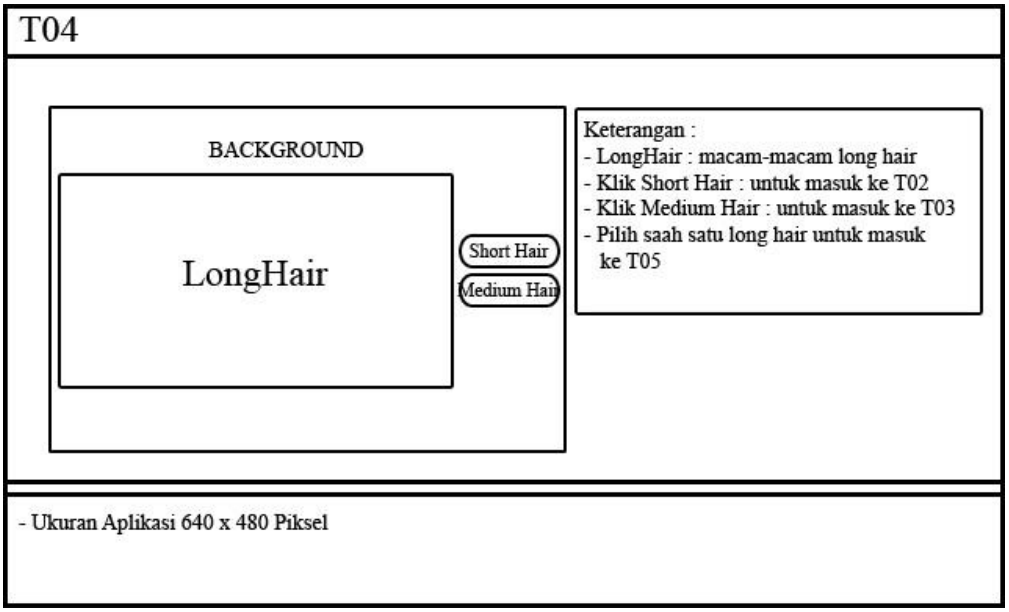
Gambar 3.33 Perancangan Antarmuka Short Hair

3.2.3.3 Perancangan Antarmuka Medium Hair



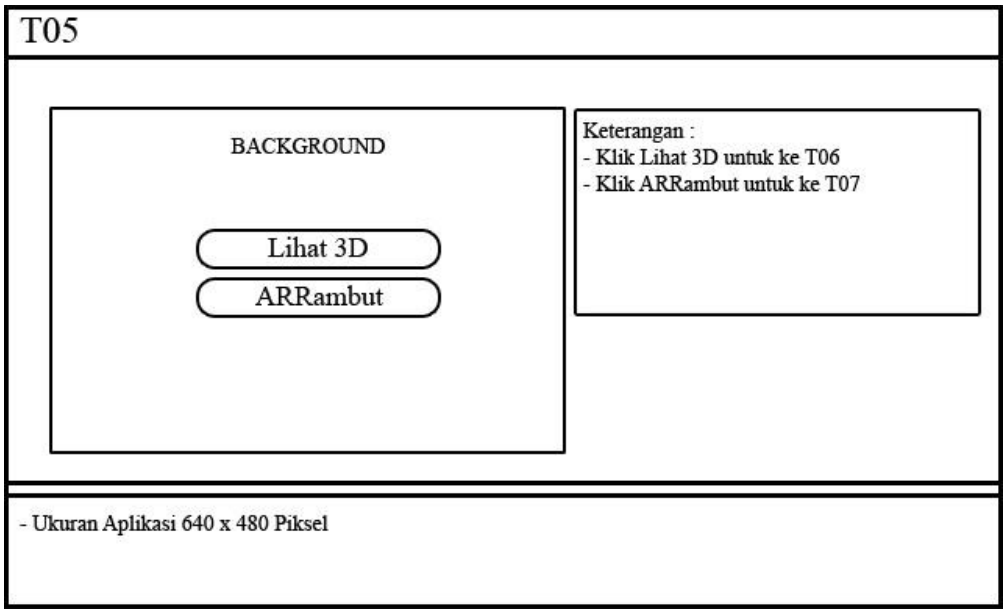
Gambar 3.34 Perancangan Antarmuka Medium Hair

3.2.3.4 Perancangan Antarmuka Long Hair



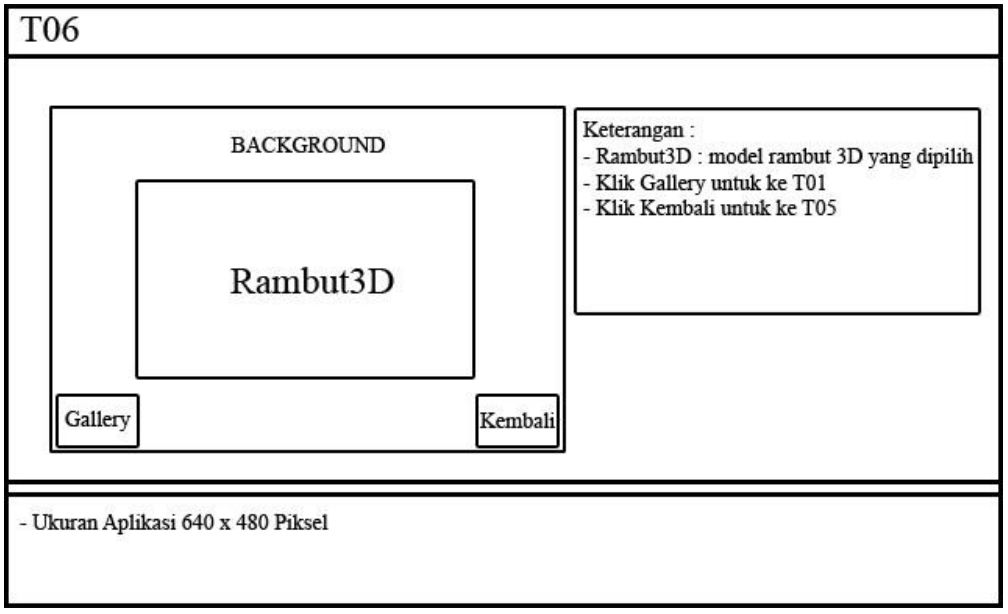
Gambar 3.35 Perancangan Antarmuka Long Hair

3.2.3.5 Perancangan Antarmuka Menu Rambut



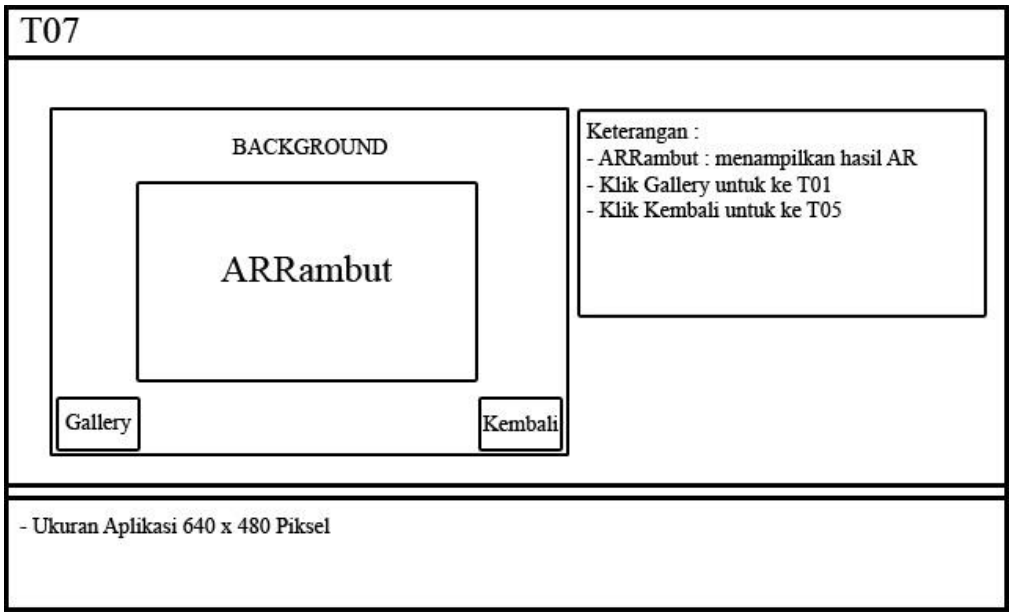
Gambar 3.36 Perancangan Antarmuka Menu Rambut

3.2.3.6 Perancangan Antarmuka Rambut3D



Gambar 3.37 Perancangan Antarmuka Rambut3D

3.2.3.7 Perancangan Antarmuka ARRambut



Gambar 3.38 Perancangan Antarmuka ARRambut