BAB 3
ANALISIS DAN PERANCANGAN

3.1. Analisis Masalah

Analisis masalah yang dikembangkan dalam penelitian ini bertujuan untuk mengetahui masalah yang terjadi pada promosi helm di toko Zona Helm, yang diantaranya adalah :

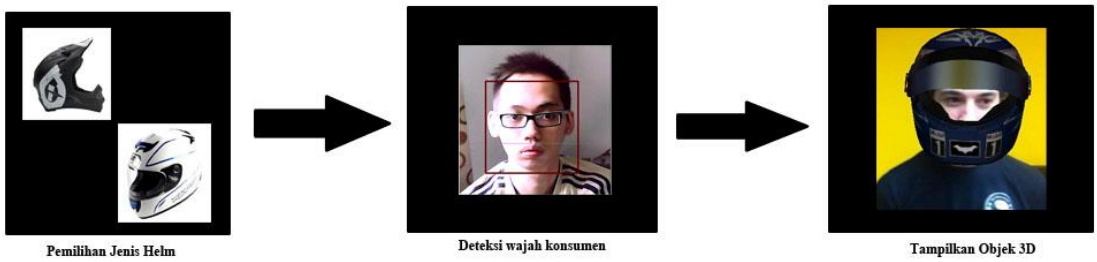
1. Promosi yang dilakukan oleh pihak toko dinilai oleh pemilik toko masih kurang efektif, dimana konsumen masih kesulitan menentukan model maupun warna yang cocok untuk dibeli maupun dipesan. Katalog maupun poster yang digunakan masih memiliki keterbatasan karena bersifat statis dalam memberikan informasi mengenai model helm.
2. Konsumen tidak dapat menentukan model dan warna helm mana yang tepat dan cocok untuk dibeli. Seiring dengan perkembangan teknologi, konsumen cenderung tertarik pada sesuatu yang lebih mudah, menarik dan memberikan dampak langsung terhadap konsumen.
3. Pemilik toko dalam menjual helm ditokonya dapat dilakukan pemesanan helm sesuai corak warna yang diinginkan konsumen. Hal ini menyebabkan pemilik toko harus memesan lebih banyak helm dengan corak maupun warna yang berbeda ke distributor. Keterbatasan tempat yang ada mengakibatkan model-model helm ini tidak dapat dipajang di toko. Model-model helm seperti helm *full-face* dan helm *cross* juga kurang diminati oleh konsumen, sehingga pemilik toko ingin memperkenalkan model-model ini.

3.2. Deskripsi Aplikasi yang Akan Dibangun

Aplikasi *Virtual Mirror* ZonaHelm yang akan dibangun adalah sebuah aplikasi *desktop* menggunakan platform *Windows* dengan teknologi *Augmented Reality*. Pemilihan dengan aplikasi *desktop* membuat konsumen dapat mengaksesnya hanya saat berada di dalam toko, sehingga memudahkan konsumen dan pemilik toko berdiskusi mengenai pemilihan model helm yang dipilih konsumen. Teknologi *augmented reality* yang menampilkan suatu informasi secara

realtime menjadi keunggulan dalam membangun aplikasi dan memberikan solusi pada permasalahan yang ada.

Aplikasi ini merupakan aplikasi yang menggunakan wajah sebagai marker untuk menyesuaikan helm dengan wajah konsumen. Adapun proses *augmented reality* dilakukan ketika konsumen selesai memilih jenis helm yang akan dibeli. Aplikasi akan meminta konsumen untuk mengarahkan wajah ke kamera sebagai marker untuk menampilkan objek 3D helm yang akan ditampilkan. Rancangan dari aplikasi *augmented reality* yang akan dibangun dapat dilihat pada Gambar 3.1.



Gambar 3.1. Usulan Aplikasi yang Dibangun

3.3. Analisis Kebutuhan Data

Aplikasi yang dibangun adalah aplikasi *Zona Helm Virtual Mirror* dengan memanfaatkan teknologi *augmented reality*. Data masukan yang dibutuhkan antara lain berupa marker berupa wajah dan objek 3D sebagai input utama pada aplikasi. Pada proses *augmented reality*, aplikasi akan memunculkan objek tiga dimensi jika aplikasi mendeteksi wajah yang diarahkan pada kamera, sehingga aplikasi ini membutuhkan *input* berupa citra wajah yang diambil secara *real-time* menggunakan alat *input* berupa kamera dimana citra yang diambil tersebut merupakan kumpulan matriks yang nantinya akan diolah lebih lanjut dalam sistem untuk proses pendeteksian marker wajah untuk memunculkan objek tiga dimensi yang memiliki posisi yang sama dengan posisi pada wajah yang dideteksi. Objek 3 dimensi yang dapat dipilih oleh user adalah objek 3 dimensi berformat FBX.

3.4. Tahap Pembangunan Aplikasi

Secara garis besar, *augmented reality* (AR) adalah penggabungan antara objek virtual dengan objek nyata. Objek virtual pada aplikasi yang akan dibangun merupakan model 3D dari helm yang telah dipilih konsumen. Objek nyata dalam

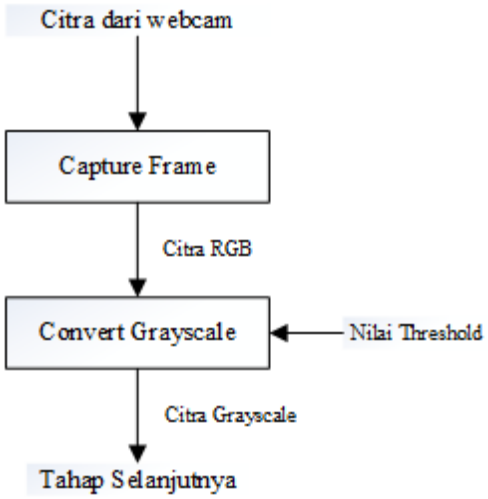
aplikasi ini berupa marker yaitu pola wajah yang ditangkap oleh webcam. Di aplikasi ini, webcam berperan penting sebagai media visi bagi perangkat lunak untuk mendapatkan video masukan. Kamera mengambil frame-frame video untuk dapat diterima oleh komputer. Komputer memproses citra digital yang diambil oleh webcam, frame demi frame, mendeteksi pola yang mirip dengan wajah dari setiap frame video tersebut, dan mencari lokasi wajah. Dengan informasi tersebut, objek helm 3D digabungkan dengan video dari webcam dan menempatkannya sesuai dengan informasi posisi yang diperoleh dari *face detection*. Proses tersebut berlangsung secara *real-time* sehingga model helm akan tampil di layar dan mengikuti posisi wajah. Gambaran umum proses aplikasi ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 Proses Algoritma

3.4.1. Tahapan Kamera

Tahapan ini adalah tahap dimana proses berhubungan langsung dengan kamera *webcam*. Citra yang diambil dari *webcam* akan diproses menjadi citra *grayscale*. Citra *grayscale* ini dibutuhkan pada tahap selanjutnya, dimana citra hasil konversi tersebut akan digunakan dalam mendeteksi wajah konsumen. Proses yang terjadi dalam tahap kamera ini dijelaskan dalam Gambar 3.3.



Gambar 3.3 Proses pada Tahap Camera

3.4.1.1. Capture Frame

Citra yang ditangkap dari webcam adalah berupa frame-frame video, sehingga perlu diubah menjadi citra statis yang dapat dikonversi. Frame-frame ini disimpan didalah sebuah array image. Fungsi capture frame pada library EmguCV berfungsi untuk menangkap citra yang berasal dari frame-frame video webcam. Berikut adalah fungsi capture dari frame-frame video webcam :

```
Image <Bgr, byte> frame;
Capture capture = new Capture(Device, Index);
frame = capture.QueryFrame();
```

Fungsi QueryFrame akan memberikan nilai balikan berupa matriks citra. Bgr merupakan format awal citra RGB, sedangkan byte merupakan nilai kedalaman citra dimana nilai byte adalah 0-255. Device adalah perangkat keras webcam yang digunakan, sedangkan Index adalah index dari perangkat webcam.

3.4.1.2. Convert Grayscale

Sebelum citra yang ditangkap diproses pada tahap-tahap berikutnya citra perlu dikonversi menjadi citra *grayscale.* Citra yang akan diproses dengan metode *HaarCascade* harus merupakan citra *grayscale*. Dalam penelitian ini pengkonversian citra dilakukan dengan menggunakan fungsi *Convert* pada library *EmguCV*, berikut merupakan fungsi *Convert* :

```
Image<Gray,byte> imagegray;
imagegray = frame.Convert<Gray,byte>();
```

Fungsi convert akan memberikan nilai balikan berupa matriks citra. Byte merupakan nilai kedalaman citra dimana nilai byte adalah 0-255. Gray merupakan format citra *grayscale* yang telah dideklarasi pada library EmguCV dengan satuan threshold umum dari rumus *grayscaling*. Rumus *grayscale* secara umum adalah sebagai berikut :

$$w = (0{,}2989 \times R) + (0{,}5870 \times G) + (0{,}1140 \times B) \qquad (6)$$

Dimana *w* adalah nilai hasil citra *grayscale, R* adalah nilai matriks *red* pada citra, G adalah nilai *green*, sedangkan *B* adalah nilai *blue.*

Contoh perhitungan dari proses *grayscale* ini adalah sebagai berikut :

a. Sebuah citra memiliki nilai R = 203, G = 145, dan B = 64
b. Perhitungan nilai *grayscale*-nya adalah $(203 \times 0{,}2989) + (145 \times 0{,}5870) + (64 \times 0{,}1140)$
c. Hasil dari perhitungan tersebut adalah 153, 0877

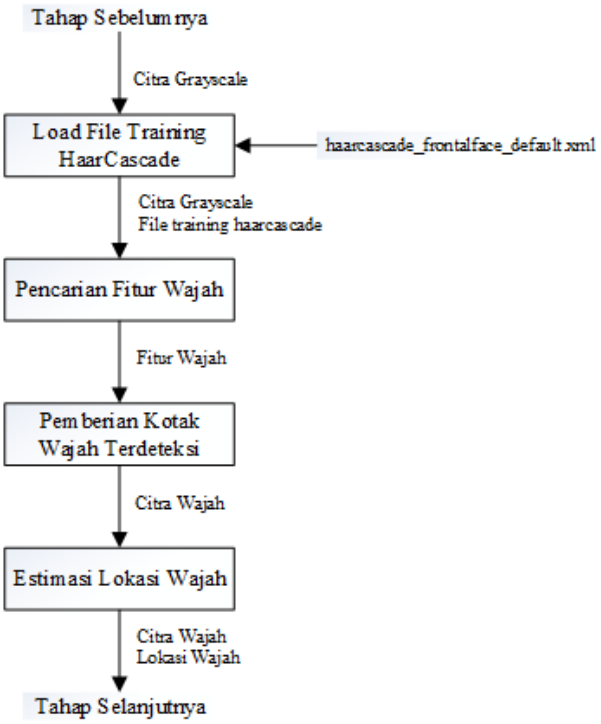
Hasil konversi citra RGB menjadi citra grayscale dapat dilihat pada Gambar 3.4.



Gambar 3.4 Hasil citra *grayscale*

3.4.2. Tahapan Deteksi

Pada tahap ini, aplikasi akan mendeteksi lokasi wajah dari hasil output tahap pertama yang berupa citra *grayscale*. Tahapan ini akan menghasilkan citra wajah dan nilai-nilai dari lokasi wajah yang telah dideteksi, yang dibutuhkan untuk keperluan tahap augmented. Proses yang terjadi pada tahap deteksi dapat dilihat pada Gambar 3.5.



Gambar 3.5 Proses pada Tahap Deteksi

3.4.2.1. Load File Training Haar Cascade

Output dari tahap sebelumnya yang merupakan citra *grayscale* yang diambil dari *webcam.* Citra tersebut masih berisi informasi lingkungan dan wajah, sehingga diperlukan sebuah metode untuk mengambil citra wajah dari citra *webcam* yang telah diproses sebelumnya. Metode yang digunakan pada aplikasi ini adalah metode *haar cascade classifier* sebagai metode untuk pengenalan pola wajah. Library EmguCV telah menyediakan sebuah class untuk mendeteksi objek melalui citra. Class ini bernama HaarCascade, dimana class ini berfungsi untuk mendeteksi objek, baik itu wajah maupun benda lainnya. Class Haar Cascade memerlukan sebuah file data training untuk deteksi wajah. File ini telah disediakan oleh library EmguCV bernama *haarcascade.frontalface_default.xml* digunakan sebagai media training *haar cascade classifer* dimana pengembang aplikasi tidak perlu melakukan kembali training untuk mendeteksi wajah manusia. File training *haarcascade.frontalface_default.xml* adalah file training bawaan (*default*). Proses pemanggilan file pada Library Emgu CV ini adalah sebagai berikut :

```
      Private HaarCascade haarCascade = new HaarCascade
("haarcascade_frontalface_default.xml");
```

Berikut ini adalah isi dari file training *haarcascade.frontalface_default.xml* :

```
<opencv_storage>
<haarcascade_frontalface_default 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_default>
</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 sedikit jumlah *sub window* yang harus diperiksa. Di tiap tingkatan terdiri dari beberapa *tree,* biasanya semakin tinggi tingkatan maka *tree* yang terdapat didalamnya pun semakin banyak. Pada *stage* 0 terdapat 2 *tree* dan pada *stage* 21 terdapat 212 *tree*.

3.4.2.2. Pencarian Fitur Wajah

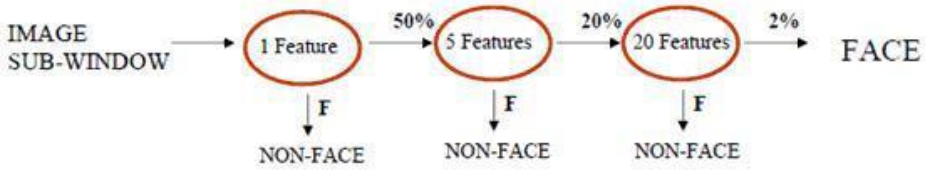
Setelah file *training* dipanggil, *training* dilakukan terlebih dahulu untuk mendapatkan suatu pohon keputusan dengan nama ***cascade classifier*** sebagai penentu apakah ada obyek atau tidak dalam tiap frame yang di proses. Pencarian posisi wajah dilakukan dengan cara mencari fitur-fitur yang memiliki tingkat pembeda yang tinggi. Hal ini dilakukan dengan mengevaluasi setiap fitur terhadap data latih dengan menggunakan nilai dari fitur tersebut. Fitur yang memiliki batas terbesar antara wajah dan bukan wajah dianggap sebagai fitur terbaik. Contoh dari pencarian fitur wajah dapat dilihat pada Gambar 3.6.



Gambar 3.6 Contoh pencarian fitur wajah dengan *haar cascade*

Haar-like feature mempunyai sifat *learner* dan *classifier* yang lemah. Jika ingin mendapatkan hasil yang lebih akurat maka harus dilakukan proses *haar-like feature* secara massal, semakin banyak proses *haar-like feature* yang dilakukan maka akan semakin akurat hasil yang dicapai. Oleh karena itu pemrosesan *haar-like feature* yang banyak terebut diorganisir di dalam *cascade classifier*.

Pada Gambar 3.7 adalah alur kerja dari klasifikasi bertingkat.



Gambar 3.7 Proses *Cascade Classifier*

Pada klasifikasi filter pertama, tiap subcitra akan diklasifikasi menggunakan satu fitur. Jika hasil nilai fitur dari filter tidak memenuhi kriteria yang diinginkan, hasil ditolak. Algoritma kemudian bergerak ke sub window selanjutnya dan menghitung nilai fitur kembali. Jika didapat hasil sesuai dengan *threshold* yang diinginkan, maka dilanjutkan ke tahap filter selanjutnya. Hingga jumlah sub window yang lolos klasifikasi pun akan berkurang hingga mendekati citra yang ada pada sampel.

Seperti pada Gambar 3.7, dapat dijelaskan secara keseluruhan proses *cascade classifier*-nya sebagai berikut :

1. Filter pertama (*first classifier*)
   - 1 fitur
   - *Detection rate* : 100%, *false positive rate* : 50%
2. Filter kedua (*second classifier*)
   - 5 fitur
   - *Detection rate* : 100%, *false positive rate* : 80%
3. Filter ketiga dan keempat (*third and fourth classifiers*)
   - 20 fitur

Pencarian fitur wajah sendiri dalam library EmguCV telah dideklarasi dalam class HaarCascade. Fungsi pencarian fitur wajah dalam Class HaarCascade dari library EmguCV adalah sebagai berikut:

```
public DetectHaarCascade(
imagegray, haarCascade, scaleFactor, minNeighbors,
HAAR_DETECTION_TYPE, minSize, maxSize )
```

Fungsi *DetectHaarCascade* akan memberikan nilai balikan berupa array citra. yaitu array khusus pada library EmguCV untuk menyimpan array hasil deteksi

objek. *imagegray* merupakan citra *grayscale* yang akan dideteksi, dimana citra ini adalah citra hasil dari proses tahap kamera. *haarCascade* adalah file *training* yang telah dipanggil pada tahap sebelumnya untuk menguji deteksi wajah. *ScaleFactor* adalah faktor skala ketika pencarian fitur wajah, dimana jika slaka diisi 1.1 maka akan membuat skala *windows* lebih besar sebesar 10%. *minNeighbors* adalah jumlah minimum (minus 1) dari persegi tetangga yang membentuk sebuah objek. *minSize* dan *maxSize* adalah ukuran minimum dan maksimum dari ukuran aplikasi *windows.*

HAAR_DETECTION_TYPE adalah tipe-tipe dari pendeteksi *Haar Cascade*, dimana tipe-tipe tersebut adalah sebagai berikut :

1. *DEFAULT*

   Jenis default di mana tidak ada optimasi dilakukan. Biasanya digunakan ketika mendeteksi objek.

2. *DO_CANNY_PRUNING*

   Jika menggunakan tipe ini, fungsi deteksi akan menggunakan detektor tepi *Canny* untuk menolak beberapa daerah gambar yang mengandung terlalu sedikit atau terlalu banyak tepi atau objek yang bukan dicari. Nilai ambang batas tertentu akan disetel untuk deteksi wajah dan dalam hal ini mempercepat pemangkasan sampai pengolahan citra.

3. *SCALE_IMAGE*

   Untuk setiap faktor skala yang digunakan pada fungsi akan diturunkan skala gambarnya daripada melakukan zoom fitur koordinat pada *cascade classifier*.

4. *FIND_BIGGEST_OBJECT*

   Fungsi ini untuk mencari objek terbesar dalam sebuah citra yang dideteksi.

5. *DO_ROUGH_SEARCH*

   Fungsi ini akan aktif ketika fungsi *FIND_BIGGEST_OBJECT* dipanggil, sehingga fungsi tidak akan mencari hasil deteksi dari ukuran yang lebih kecil segera setelah menemukan objek yang lebih besar pada skala tertentu.

DO_CANNY_PRUNING merupakan salah satu tipe pendeteksi *Haar Cascade,* dimana tipe ini menggunakan algoritma deteksi tepi *Canny.* Detektor tepi *Canny* adalah operator deteksi tepi yang menggunakan algoritma multi-tahap untuk mendeteksi berbagai macam tepi dalam gambar. Ini dikembangkan oleh John F. Canny pada tahun 1986. Operator *Canny* merupakan pendeteksi tepi hasil pengembangan pendekatan gradien. Dalam algoritma *Canny*, deteksi tepi ditemukan dengan melihat maksimum lokal pada gradien citra. Gradien dihitung dengan menggunakan turunan dari filter *Gaussian*. Dalam algoritma ini digunakan dua nilai *threshold* (ambang), untuk mendeteksi tepi yang kuat dan tepi yang lemah dan memasukkan tepi yang lemah dalam *output* hanya jika terhubung pada tepi yang kuat.

Penjelasan algoritma deteksi tepi *Canny* adalah sebagai berikut :

1. Penghalusan citra dengan filter *Gaussian*
2. Menghitung besar dan arah gradien untuk turunan parsial
3. Memakai *nonmakisma suppression* untuk besar gradien
4. Menggunakan *double thresholding* untuk mendeteksi dan menghubungkan sisi dari deteksi tepi
5. Melacak tepi dengan *hysteresis*: Finalisasi deteksi tepi dengan menekan semua tepi lain yang lemah dan tidak terhubung ke tepi yang kuat

Algoritma Canny bertujuan untuk memenuhi tiga kriteria utama:

1. *Low error rate*: deteksi yang baik hanya wujud tepi yang ada
2. *Good localization* : jarak antara tepi piksel terdeteksi dan tepi piksel nyata harus diminimalkan
3. *Minimal response*: satu detektor respons per tepi

Pada library EmguCV, algortima deteksi tepi *Canny* ini digunakan sebagai tipe deteksi *Haar Cascade* untuk membuang sejumlah tepi gambar yang bukan merupakan tepi dari lingkar wajah. Ini membuat pencarian fitur wajah menjadi jauh lebih cepat sehingga wajah akan terdeteksi tanpa harus melakukan pencarian secara menyeluruh terhadap gambar hasil *capture* dari *webcam.* Algortima *Canny* menggunakan citra *grayscale* sebagai citra masukan untuk mendeteksi tepi dari citra. Algoritma *Canny* yang digunakan pada EmguCV merupakan salah satu

algortima yang diturunkan dari OpenCV. Pada library OpenCV, algoritma *Canny* yang digunakan disebut *Canny Edge Detection.*

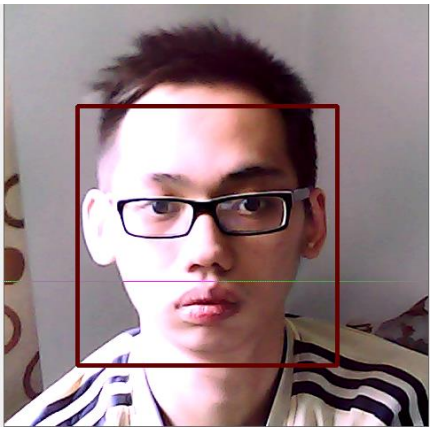
3.4.2.3. Pemberian Kotak Wajah Terdeteksi

Pada tahapan ini, class Haar Cascade pada library EmguCV yang telah menghasilkan pola wajah diposisikan pada citra asli secara real-time melalui webcam. Hasil deteksi wajah yang telah dilakukan pada proses sebelumnya akan ditampilkan pada citra real-time dengan menampilkan sebuah kotak dimana posisi kotak tersebut adalah hasil deteksi lokasi wajah setelah dilakukan pencarian citra wajah pada tahap sebelumnya. Fungsi *DetectHaarCascade* yang pada proses sebelumnya menghasilkan citra array citra akan disimpan ke dalam citra array lain, dimana array citra ini selanjutnya akan dibuat sebuah kotak ke frame-frame hasil citra dari *webcam.* Fungsi untuk membuat kotak pada wajah yang telah dideteksi adalah sebagai berikut :

```
foreach (array face in DetectHaarCascade)
{
      frame.Draw(face.rect, Color, thickness);
}
```

Array *face* adalah array citra khusus pada library EmguCV untuk menyimpan array hasil deteksi objek. *DetectHaarCascade* adalah fungsi pencarian wajah dari tahap sebelumnya. Fungsi *Draw* adalah fungsi turunan dari *frame* yang telah dipanggil pada tahap kamera, fungsi ini akan menggambar kotak pada citra dari *webcam.* Fungsi Draw sendiri terdiri dari *face.rect,* dimana array citra *face* menghasilkan fungsi *rectangle* untuk menggambar sebuah kotak untuk objek yang berhasil terdeteksi, *Color* sebagai warna kotak, dan *thickness* sebagai ketebalan garis kotak.

Pada fungsi tersebut, setiap kali frame video dari webcam dideteksi sebagai wajah, maka kotak hasil deteksi akan muncul. Hasil pada proses ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 Hasil deteksi wajah

3.4.2.4. Estimasi Lokasi dan Posisi Wajah

Pada proses ini aplikasi akan mengestimasi lokasi dan juga posisi dari wajah berdasarkan lokasi hasil deteksi wajah dari proses sebelumnya. Lokasi wajah ditentukan dari nilai vector x, vector y, panjang dan lebar dari kotak deteksi wajah. Berikut fungsi dari penentuan lokasi kotak deteksi wajah :

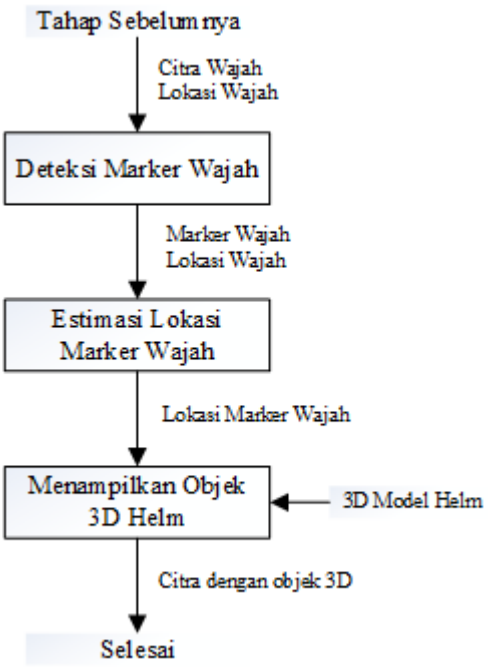
```
foreach (array face in DetectHaarCascade)
{
        frame.Draw(face.rect, White, 2);
        int X = face.rect.X;
        int Y = face.rect.Y;
        int height = face.rect.Height;
        int width = face.rect.Width;
}
```

Nilai dari setiap vector tersebut dikalkulasi menjadi nilai tengah dimana nilai tengah tersebut akan menjadi posisi dari objek 3 dimensi yang akan dimunculkan. Berikut adalah fungsi nilai tengah dari vector-vector tersebut :

```
Point center = new Point((X + (height / 2)),(Y + (width / 2)));
```

3.4.3. Tahap Augmented

Tahapan ini merupakan tahapan terakhir dalam algoritma dimana pada tahapan ini aplikasi akan melihat apakah marker wajah berhasil ditemukan atau tidak, apabila wajah berhasil dideteksi maka aplikasi akan memunculkan objek 3 dimensi berdasarkan lokasi wajah. Proses-proses yang ada dalam tahap ini dapat dilihat pada gambar 3.9.



Gambar 3.9 Proses dalam Tahapan Augmented

3.4.3.1. Deteksi Marker Wajah

Proses deteksi marker wajah adalah proses untuk menentukan apakah wajah sebagai marker telah ditemukan dalam proses sebelumnya. Jika terdeteksi, maka proses selanjutnya akan dijalankan, namun jika wajah tidak terdeteksi maka proses *augmented reality* tidak akan dijalankan. Berikut adalah fungsi deteksi marker wajah:

```
if (array face != null)
{
bool marker = true;
array markerAR = face;
}
else
{
bool marker = false;
}
```

3.4.3.2. Estimasi Lokasi Marker Wajah

Pada proses ini aplikasi akan mengestimasi lokasi dari marker berdasarkan lokasi wajah dari proses sebelumnya. Pada tahapan ini lokasi wajah yang merupakan bidang 2 dimensi akan diestimasi ke bidang 3 dimensi dengan acuan

parameter intrinsic dari kamera yang telah didapatkan dari proses kalibrasi kamera. Hasil estimasi bidang 3 dimensi ini akan dibandingkan dengan objek 3 dimensi berupa objek helm yang telah diinisialisasi untuk menghasilkan vector skala dan juga translasi. Matriks skala dan matrik translasi ini merupakan matriks yang akan dipakai pada tahap berikutnya yaitu tahap memunculkan objek 3 dimensi. Pada tahap ini untuk mendapatkan matriks skala dan translasi digunakan fugnsi *solvePNP* pada *library* EmguCV, berikut merupakan fungsi *solvePNP* :

```
bool SolvePnP(
      IInputArray objectPoints,
      IInputArray imagePoints,
      IOutputArray scaleVector,
      IOutputArray translationVector,
      bool useExtrinsicGuess = false,
      SolvePnpMethod flags = SolvePnpMethod.Iterative
)
```

Fungsi SolvePnP adalah fungsi untuk mencari vector translasi dan skala suatu objek 3 dimensi. Parameter objectPoints merupakan parameter point-point pada bidang 3 dimensi pada kasus ini adalah point-point objek 3D Helm pada bidang 3 dimensi. Parameter imagePoints merupakan parameter point-point dari marker dalam bidang 2 dimensi pada kasus ini adalah lokasi wajah yang didapatkan dari proses sebelumnya. Parameter intrinsicMatrix merupakan matriks intrinsic dari kamera yang didapatkan dari proses kalibrasi kamera. Parameter distortion adalah parameter distorsi kamera yang didapatkan dari proses kalibrasi juga. Parameter scaleVector merupakan parameter keluaran parameter ini akan mengeluarkan nilai dari vector skala yang akan digunakan dalam proses memunculkan objek 3 dimensi pada tahap berikutnya. Parameter translationVector merupakan parameter keluaran yang mengularkan nilai dari vector translasi.

3.4.3.3. Memunculkan Objek 3D

Tahapan ini merupakan tahapan terakhir pada tahapan augmented dan juga pada algoritma. Pada tahapan ini objek 3 dimensi yang dimaksud yaitu objek 3 dimensi berupa helm akan munculkan di wajah. Pada tahap ini aplikasi akan

memanggil objek 3 dimensi yang telah didefinisikan ke layar dengan lokasi berdasarkan letak dari posisi dan lokasi yang telah dideteksi sehingga objek 3 dimensi akan mengikuti pergerakan wajah.

3.5. Spesifikasi Kebutuhan Perangkat Lunak

Tahap ini menjelaskan analisis spesifikasi dari kebutuhan perangkat lunak yang akan dibangun yaitu meliputi spesifikasi kebutuhan perangkat lunak fungsional (SKPL-F) dan spesifikasi kebutuhan perangkat lunak non-fungsional (SKPL-NF).

1. Spesifikasi Kebutuhan Fungsional

Spesifikasi kebutuhan funsional berisi mengenai fungsi atau layanan apa yang harus disediakan atau bisa dilakukan oleh aplikasi terhadap suatu kondisi tertentu, dan apa yang harus dilakukan oleh apliasi jika terdapat suatu *input* tertentu. Detail dari spesifikasi kebutuhan fungsional dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi Kebutuhan Perangkat Lunak Fungsional

Nomor	Spesifikasi Kebutuhan Perangkat Lunak Fungsional
SKPL-F-001	Aplikasi harus dapat menambah foto helm oleh admin
SKPL-F-002	Aplikasi harus dapat menghapus foto helm oleh admin
SKPL-F-003	Aplikasi harus dapat melakukan login oleh admin
SKPL-F-004	Aplikasi harus dapat menampilkan foto-foto helm
SKPL-F-005	Aplikasi harus dapat menampilkan model 3D helm
SKPL-F-006	Aplikasi harus dapat menampilkan model 3D helm ke posisi wajah
SKPL-F-007	Aplikasi harus dapat mendeteksi posisi wajah di depan kamera

2. Spesifikasi Kebutuhan Non Fungsional

Spesifikasi kebutuhan non-fungsional merupakan penjelasan mengenai batasan-batasan fungsi dari aplikasi yang akan dibangun. Detail mengenai spesifikasi kebutuhan non-fungsional dapat dilihat pada Tabel 3.2.

Tabel 3.2 Spesifikasi Kebutuhan Perangkat Lunak Non Fungsional

Nomor	Spesifikasi Kebutuhan Perangkat Lunak Fungsional
SKPL-NF-001	Aplikasi memerlukan kamera dengan resolusi yang cukup untuk mengenali wajah
SKPL-NF-002	Aplikasi memerlukan cahaya yang cukup agar kamera dapat mendeteksi wajah
SKPL-NF-003	Aplikasi memelukan sudut yang cukup untuk membaca wajah dari arah kamera
SKPL-NF-004	Posisi wajah harus diluruskan (*Frontal Face*) terhadap *webcam*.
SKPL-NF-005	Aplikasi hanya dibangun dalam bentuk *dekstop*

3.6. Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional menggambarkan kebutuhan luar sistem yang diperlukan untuk menjalankan aplikasi yang dibangun. Adapun kebutuhan non fungsional pada aplikasi *Zona Helm Virtual Mirror* dengan teknologi *augmented reality* ini meliputi kebutuhan perangkat keras, kebutuhan perangkat lunak dan pengguna sistem yang akan memakai aplikasi.

3.6.1. Analisis Kebutuhan Perangkat Keras

Analisis Perangkat keras atau *hardware* merupakan salah satu hal yang penting karena tanpa *hardware* yang memenuhi syarat, aplikasi yang akan dibuat tidak akan dapat berjalan. Kebutuhan perangkat keras yang digunakan untuk membangun aplikasi *Zona Helm Virtual Mirror* dengan teknologi *Augmented Reality* dapat dilihat pada Tabel 3.3.

Tabel 3.3 Kebutuhan perangkat keras pembangunan aplikasi

No.	Perangkat Keras
1	Processor 2.3GHz atau lebih
2	RAM minimal 4GB
3	Harddisk Minimal 8GB untuk instalasi software
4	Kamera webcam untuk Service tester minimal 5MP
5	Kartu Grafis (VGA Card) minimal 512MB

3.6.2. Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak atau *software* merupakah hal yang terpenting dalam mendukung kinerja sebuah sistem. Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar dapat saling berinteraksi diantara keduanya. Kebutuhan perangkat lunak yang digunakan untuk membangun aplikasi *Zona Helm Virtual Mirror* dengan teknologi *Augmented Reality* dapat dilihat pada Tabel 3.4.

Tabel 3.4 Kebutuhan perangkat lunak pembangunan aplikasi

No.	Perangkat Lunak
1	Windows 7 Profesional 64-bit atau lebih
2	Microsoft Visual Studio 2010 atau lebih
3	Emgu CV 3.0 for Windows
4	Blender 2.7.4
5	WAMP Server versi 2.5
6	VGA Driver support for OpenGL rendering
7	Microsoft XNA Game Studio 4.0.4
8	SQLyog 10.4.2.0

3.6.3. Analisis Kebutuhan Pengguna

Analisis pengguna dimaksudkan untuk mengetahui siapa saja pengguna aplikasi yang akan di bangun ini, maka dari itu analisis terhadap pengguna perlu dilakukan analisis. Pengguna utama aplikasi *Zona Helm Virtual Mirror* dengan teknologi *Augmented Reality* ini adalah admin selaku pemilik toko Zona Helm dan konsumen pengguna motor di usia sekitar 16-60 tahun. Aplikasi ini digunakan dengan pengarahan dari pemilik toko. Karakteristik kebutuhan pengguna aplikasi ini dapat dilihat seperti pada Tabel 3.5.

Tabel 3.5 Analisis kebutuhan pengguna

Pengguna	Hak Akses	Tingkat Pendidikan	Pengalaman
Pemilik Toko (Admin)	Menambah info helm, menambah foto helm, menghapus foto helm.	Tidak dibatasi	Dapat menggunakan perangkat komputer.
Pengguna motor, usia 16-60 tahun	Menjalankan fitur-fitur yang ada pada aplikasi	Tidak dibatasi	Dapat menggunakan perangkat komputer

3.7. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional ini dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Dimana tahap-tahap perancangan yang dilakukan dalam membangun aplikasi *magic coloring book* dengan teknologi *augmented reality* antara lain *Use Case Diagram*, *Class Diagram*, *Sequence Diagram*, dan *Activity Diagram.*

3.7.1. *Use Case Diagram*

Use Case Diagram merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada aplikasi. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional aplikasi dengan mendefinisikan skenario penggunaan aplikasi yang akan dibangun.

Berikut ini adalah perancangan proses-proses yang terdapat pada aplikasi *Zona Helm Virtual Mirror* dengan teknologi *augmented reality*, yang digambarkan dengan *Use Case Diagram* yang dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Use Case Diagram* Aplikasi

3.7.2. Definisi Aktor

Aktor adalah abstraksi dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Berikut adalah aktor yang berperan dalam menjalankan sistem yang dibangun yang dapat dilihat pada Tabel 3.6.

Tabel 3.6 Aktor dalam *use case*

No.	Aktor	Deskripsi
1	Pengguna	Pemilik toko dan konsumen yang akan mengakses dan menggunakan fitur-fitur yang tersedia pada aplikasi. Pemilik toko sebagai pengguna dapat login menjadi admin.

3.7.3. Definisi *Use Case*

Diagram *use case* merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Tujuan pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasi sistem. Definisi dari setiap *use case* yang ada pada sistem dapat dilihat pada Tabel 3.7.

Tabel 3.7 Definisi *use case*

No.	Nama *Use Case*	Deskripsi
UC-01	Menambah Foto Helm	Fungsional untuk menambah foto helm
UC-02	Menghapus Foto Helm	Fungsional untuk menghapus foto helm
UC-03	Login Admin	Fungsional untuk login admin
UC-04	Melihat Foto Helm	Fungsional untuk menampilkan foto helm
UC-05	Menampilkan Model Helm 3D	Fungsional untuk menampilkan model 3D helm
UC-06	Menampilkan AR Helm	Fungsional untuk menampilkan model 3D helm ke posisi wajah
UC-07	Deteksi Wajah	Fungsional untuk mendeteksi posisi wajah di depan kamera

3.7.4. Skenario *Use Case*

Skenario *use case* menjelaskan urutan langkah-langkah dalam proses yang dilakukan sebuah *use case*. Adapun skenario *use case* dari sistem yang dibangun adalah sebagai berikut :

3.7.4.1. Skenario *Use Case* Menambah Foto Helm

Skenario *use case* memilih model helm dapat dilihat pada Tabel 3.8.

Tabel 3.8 Skenario *Use Case* Menambah Foto Helm

Nama *Use Case*	Menambah Foto Helm	
Tujuan dalam konteks	Pemilik Toko berhasil menambah foto helm	
Deskripsi	Fungsionalitas untuk menambah foto helm	
Kondisi Awal	Aktor berhasil login dan berada di halaman admin	
Kondisi Akhir Berhasil	Berhasil menambah foto helm	
Kondisi Akhir Gagal	Gagal menambahkan foto helm	
Aktor	Pengguna (pemilik toko)	
Include	Login	
Extend	-	
Trigger	Aktor menekan pilihan tambah foto helm	
Aliran Utama	**Langkah**	**Aksi**
	1.	Aktor menekan tombol tambah helm
	2.	Sistem menampilkan form tambah foto helm
	3.	Aktor mengisi deskripsi helm
	4.	Aktor memilih foto helm
	5.	Aktor menekan tombol simpan
	6.	Sistem menyimpan foto dan deskripsi ke database dan menampilkan pesan berhasil tambah foto
Kondisi lain	**Langkah**	**Aksi Percabangan**
	6.1.	Sistem menampilkan pesan gagal tambah foto

3.7.4.2. Skenario *Use Case* Menghapus Foto Helm

Skenario *use case* memilih model helm dapat dilihat pada Tabel 3.9.

Tabel 3.9 Skenario *Use Case* Menghapus Foto Helm

Nama *Use Case*	Menghapus Foto Helm	
Tujuan dalam konteks	Pemilik toko berhasil menghapus foto helm	
Deskripsi	Fungsionalitas untuk menghapus foto helm	
Kondisi Awal	Aktor berhasil login dan berada di halaman admin	
Kondisi Akhir Berhasil	Berhasil menghapus foto helm	
Kondisi Akhir Gagal	Gagal menghapus foto helm	
Aktor	Pengguna (pemilik toko)	
Include	Login	
Extend	-	
Trigger	Aktor menekan pilihan hapus foto helm	
Aliran Utama	**Langkah**	**Aksi**
	1.	Aktor menekan tombol hapus helm
	2.	Sistem menampilkan form hapus foto helm
	3.	Aktor memilih jenis helm yang akan dihapus
	4.	Aktor memilih nama helm yang akan dihapus
	5.	Sistem me*load* data helm yang dipilih dari database
	6.	Sistem menampilkan data helm yang dipilih
	7.	Aktor menekan tombol hapus
	8.	Sistem menghapus data helm dari database dan menampilkan pesan berhasil hapus data helm
Kondisi lain	**Langkah**	**Aksi Percabangan**
	8.1.	Sistem menampilkan pesan gagal hapus foto

3.7.4.3. Skenario *Use Case* Login Admin

Skenario *use case* melihat model helm dapat dilihat pada tabel 3.10.

Tabel 3.10 Skenario *Use Case* Login

Nama *Use Case*	Login Admin	
Tujuan dalam konteks	Pemilik toko berhasil login ke menu admin	
Deskripsi	Fungsional untuk login admin	
Kondisi Awal	Aktor berada di menu utama dan menekan tombol login admin	
Kondisi Akhir Berhasil	Berhasil login ke sistem dan menampilkan menu admin	
Kondisi Akhir Gagal	Gagal login dan tidak dapat masuk ke dalam menu admin	
Aktor	Pengguna (pemilik toko)	
Include	-	
Extend	-	
Trigger	Aktor menekan tombol login admin pada menu utama	
Aliran Utama	**Langkah**	**Aksi**
	1.	Aktor menekan tombol login admin
	2.	Sistem menampilkan form login
	3.	Aktor memasukkan username dan password
	4.	Sistem akan mencocokan username dan password
	5.	Jika sesuai, sistem menampilkan menu admin
Kondisi lain	**Langkah**	**Aksi Percabangan**
	5.1.	Jika username dan password tidak sesuai, sistem menampilkan pesan gagal login, dan kembali ke langkah 2

3.7.4.4. Skenario *Use Case* Melihat Foto Helm

Skenario *use case* melihat model helm dapat dilihat pada Tabel 3.11.

Tabel 3.11 Skenario *Use Case* Melihat Foto Helm

Nama *Use Case*	Melihat Foto Helm	
Tujuan dalam konteks	Pengguna dapat melihat foto-foto helm	
Deskripsi	Fungsional untuk menampilkan foto helm	
Kondisi Awal	Aktor berada di menu utama aplikasi	
Kondisi Akhir Berhasil	Aktor dapat melihat foto-foto helm	
Kondisi Akhir Gagal	-	
Aktor	Pengguna	
Include	-	
Extend	-	
Trigger	Aktor menekan tombol lihat helm pada menu utama	
Aliran Utama	**Langkah**	**Aksi**
	1.	Aktor memilih jenis helm dan menekan tombol lihat helm
	2.	Sistem me*load* data helm dari database
	3.	Sistem menampilkan data helm berdasarkan jenis helm
	4.	Aktor memilih salah satu foto helm
	5.	Sistem menampilkan *slideshow* foto helm

3.7.4.5. Skenario *Use Case* Menampilkan Model 3D Helm

Skenario *use case* menampilkan model 3D helm dapat dilihat pada Tabel 3.12.

Tabel 3.12 Skeneraio *Use Case* Menampilkan Model 3D Helm

Nama *Use Case*	Melihat Model 3D Helm	
Tujuan dalam konteks	Pengguna dapat melihat objek 3D Helm	
Deskripsi	Fungsional untuk menampilkan model 3D helm	
Kondisi Awal	Aktor berada di menu utama dan menekan tombol model helm 3D	
Kondisi Akhir Berhasil	Berhasil menampilkan objek 3D dengan rotasi	
Kondisi Akhir Gagal	File 3D tidak valid	
Aktor	Pengguna	
Include	-	
Extend	-	
Trigger	Aktor menekan tombol helm 3D pada menu utama	
Aliran Utama	**Langkah**	**Aksi**
	1.	Aktor menekan tombol helm 3D
	2.	Sistem meload objek tiga dimensi
	3.	Sistem menampilkan form utama Helm 3D
	4.	Sistem memunculkan objek 3D
	5.	Sistem merotasi objek 3D
	6.	Aktor memilih warna helm dengan tombol 1-4
	7.	Sistem menampilkan objek 3D Helm dengan warna lain
Kondisi lain	**Langkah**	**Aksi Percabangan**
	3.1	Objek 3D tidak valid

3.7.4.6. Skenario *Use Case* Menampilkan AR Model Helm

Skenario *use case* menampilkan AR Helm dapat dilihat pada Tabel 3.13.

Tabel 3.13 Skenario *Use Case* Menampilkan AR Model Helm

Nama *Use Case*	Menampilkan AR Helm	
Tujuan dalam konteks	Pengguna dapat mencoba objek 3d helm pada wajah	
Deskripsi	Fungsional untuk menampilkan model 3D helm ke posisi wajah	
Kondisi Awal	Pengguna memilih kamera dan menekan tombol AR Helm	
Kondisi Akhir Berhasil	Berhasil menampilkan objek 3D pada wajah aktor	
Kondisi Akhir Gagal	File 3D tidak valid, wajah tidak terdeteksi	
Aktor	Pengguna	
Include	Deteksi Wajah	
Extend	-	
Trigger	Aktor memilih kamera dan menekan tombol AR Helm pada menu utama	
Aliran Utama	**Langkah**	**Aksi**
	1.	Aktor memilih kamera dan menekan tombol AR Helm
	2.	Sistem mengaktifkan kamera
	3.	Sistem meload objek 3D Helm
	4.	Sistem menampilkan form utama AR
	5.	Aktor mengarahkan wajah ke kamera
	6. *include* : Deteksi Wajah	Sistem mendeteksi wajah aktor
	7.	Sistem memunculkan objek 3D pada lokasi wajah
	8.	Aktor memilih warna helm dengan tombol 1-4
	9.	Sistem menampilkan AR Helm dengan warna lain
Kondisi lain	**Langkah**	**Aksi Percabangan**
	4.1	Objek 3D tidak valid

3.7.4.7. Skenario *Use Case* Deteksi Wajah

Skenario *use case* deteksi wajah dapat dilihat pada Tabel 3.14.

Tabel 3.14 Skenario *Use Case* Deteksi Wajah

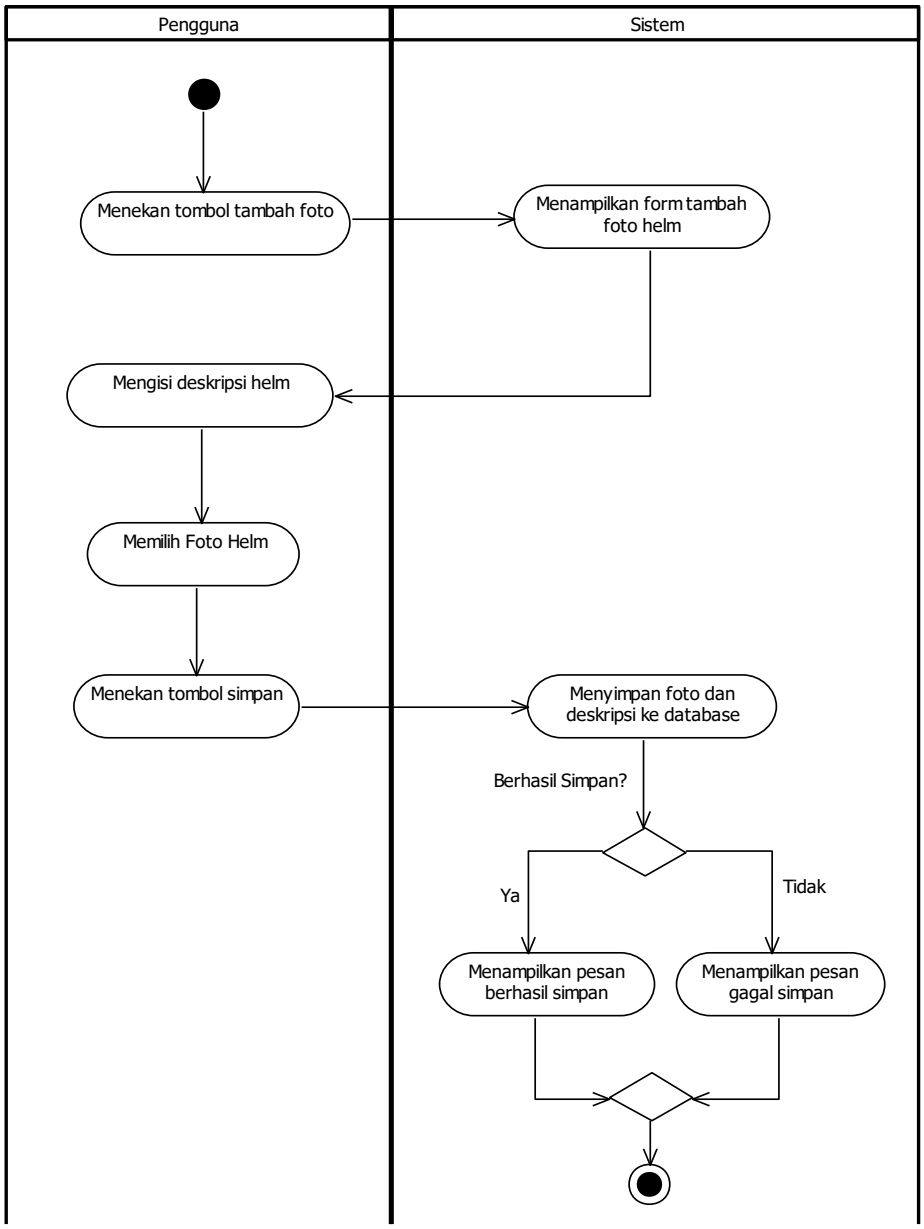
Nama *Use Case*	Deteksi Wajah	
Tujuan dalam konteks	Sistem dapat mendeteksi wajah	
Deskripsi	Fungsional untuk mendeteksi posisi wajah di depan kamera	
Kondisi Awal	Aktor berada di form utama AR dan mengarahkan wajah ke kamera	
Kondisi Akhir Berhasil	Berhasil mendeteksi wajah	
Kondisi Akhir Gagal	Gagal mendeteksi wajah	
Aktor	-	
Include	-	
Extend	-	
Trigger	Aktor pada menu AR dan mengarahkan wajah ke kamera	
Aliran Utama	**Langkah**	**Aksi**
	1.	Sistem mengambil citra dari *webcam*
	2.	Sistem mengubah format citra menjadi *grayscale*
	3.	Sistem mencari fitur wajah
	4.	Sistem berhasil mendeteksi wajah dan menampilakan lokasi deteksi wajah
Kondisi Lain	**Langkah**	**Aksi Percabangan**
	4.1	Gagal mendeteksi wajah dan kembali ke langkah 1

3.7.5. *Activity Diagram*

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang dirancang, bagaimana masing-masing fungsionalitas bekerja, dan bagaimana suatu fungsional berakhir. *Activity Diagram* memodelkan event-event yang terjadi pada use case. Adapun *Activity Diagram* dari aplikasi yang akan dibangun adalah sebagai berikut :

3.7.5.1. *Activity Diagram* Menambah Foto Helm

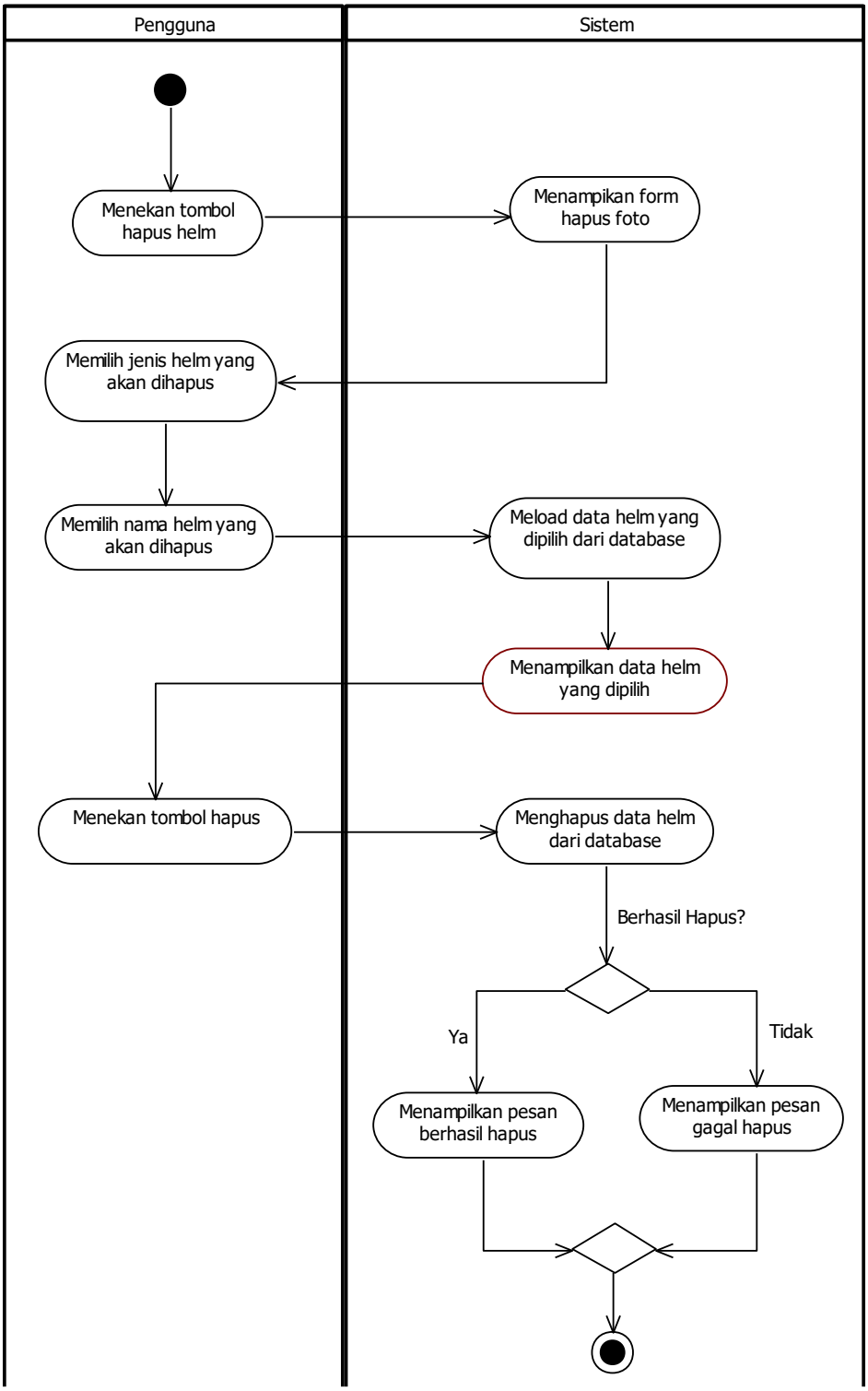
Activity diagram menambah foto helm dapat dilihat pada Gambar 3.11.



Gambar 3.11 *Activity Diagram* Menambah Foto Helm

3.7.5.2. *Activity Diagram* Menghapus Foto Helm

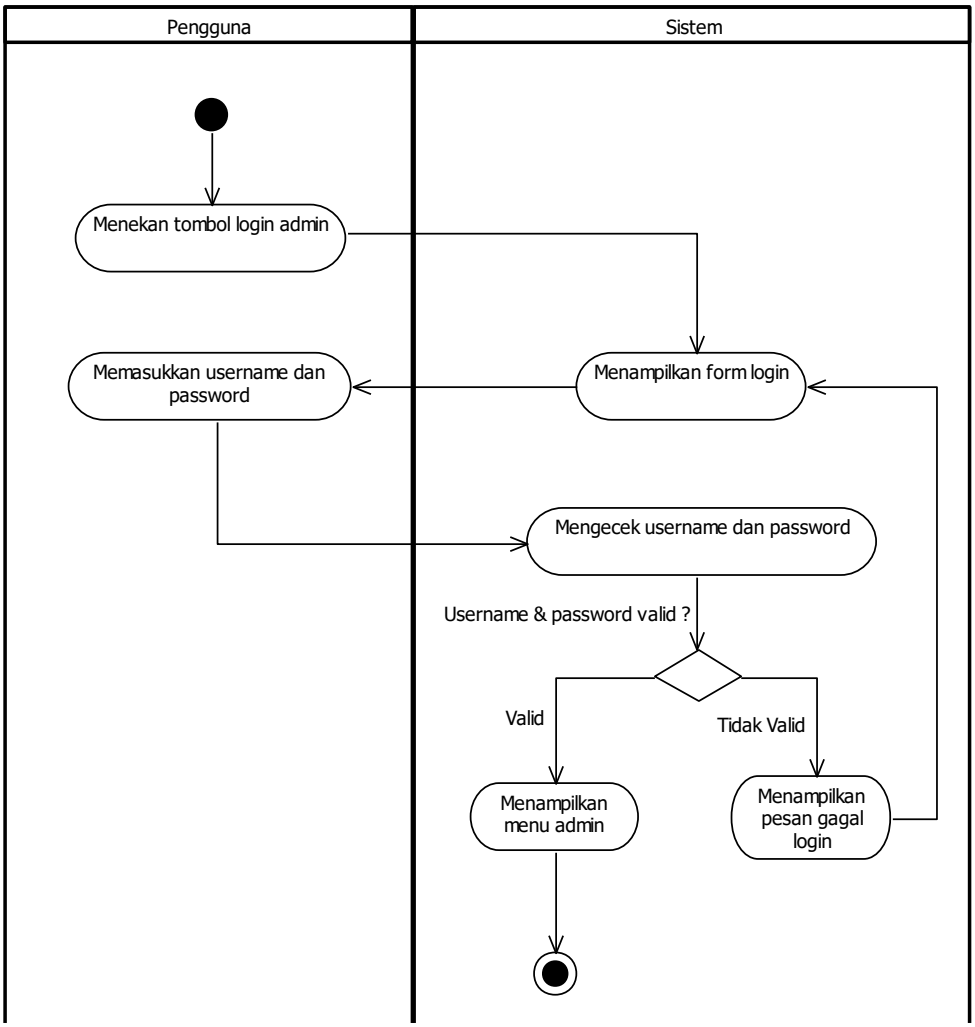
Activity diagram menghapus foto helm dapat dilihat pada Gambar 3.12.



Gambar 3.12 *Activity Diagram* Menghapus Foto Helm

3.7.5.3. *Activity Diagram* Login Admin

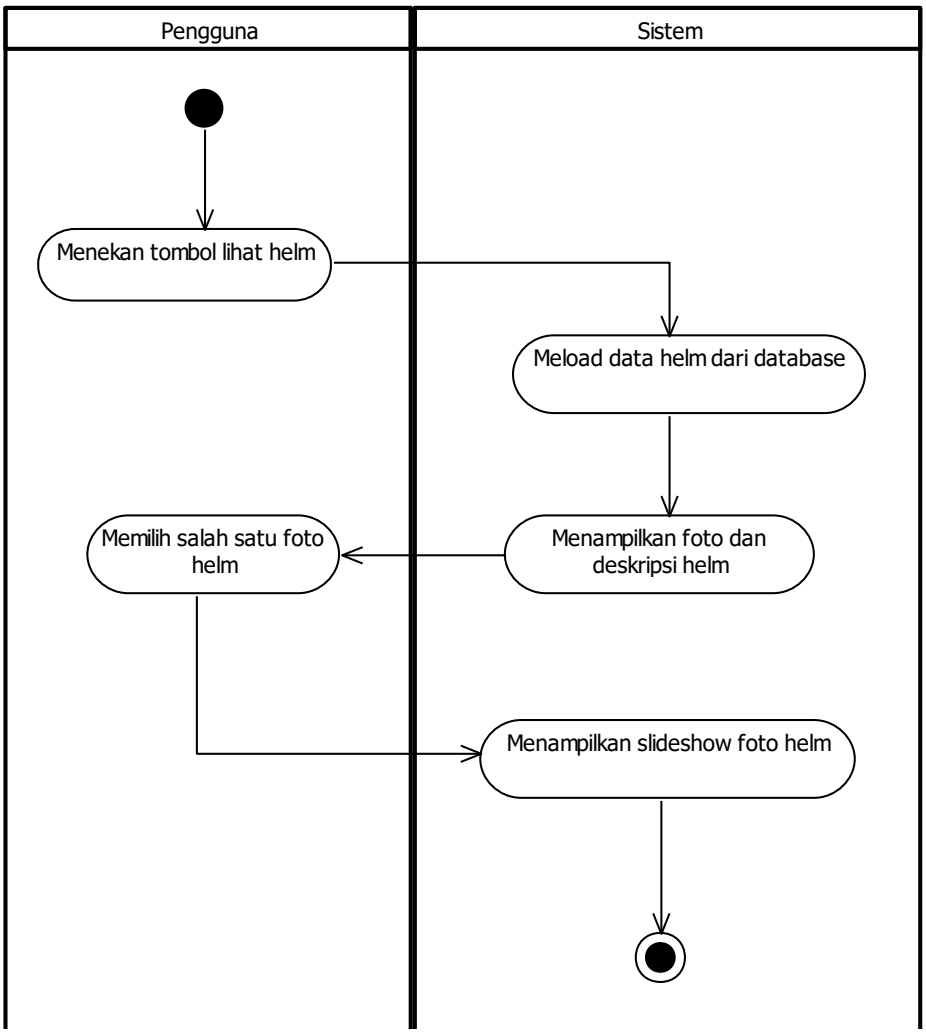
Activity diagram menghapus foto helm dapat dilihat pada Gambar 3.13.



Gambar 3.13 *Activity Diagram* Login Admin

3.7.5.4. *Activity Diagram* Melihat Foto Helm

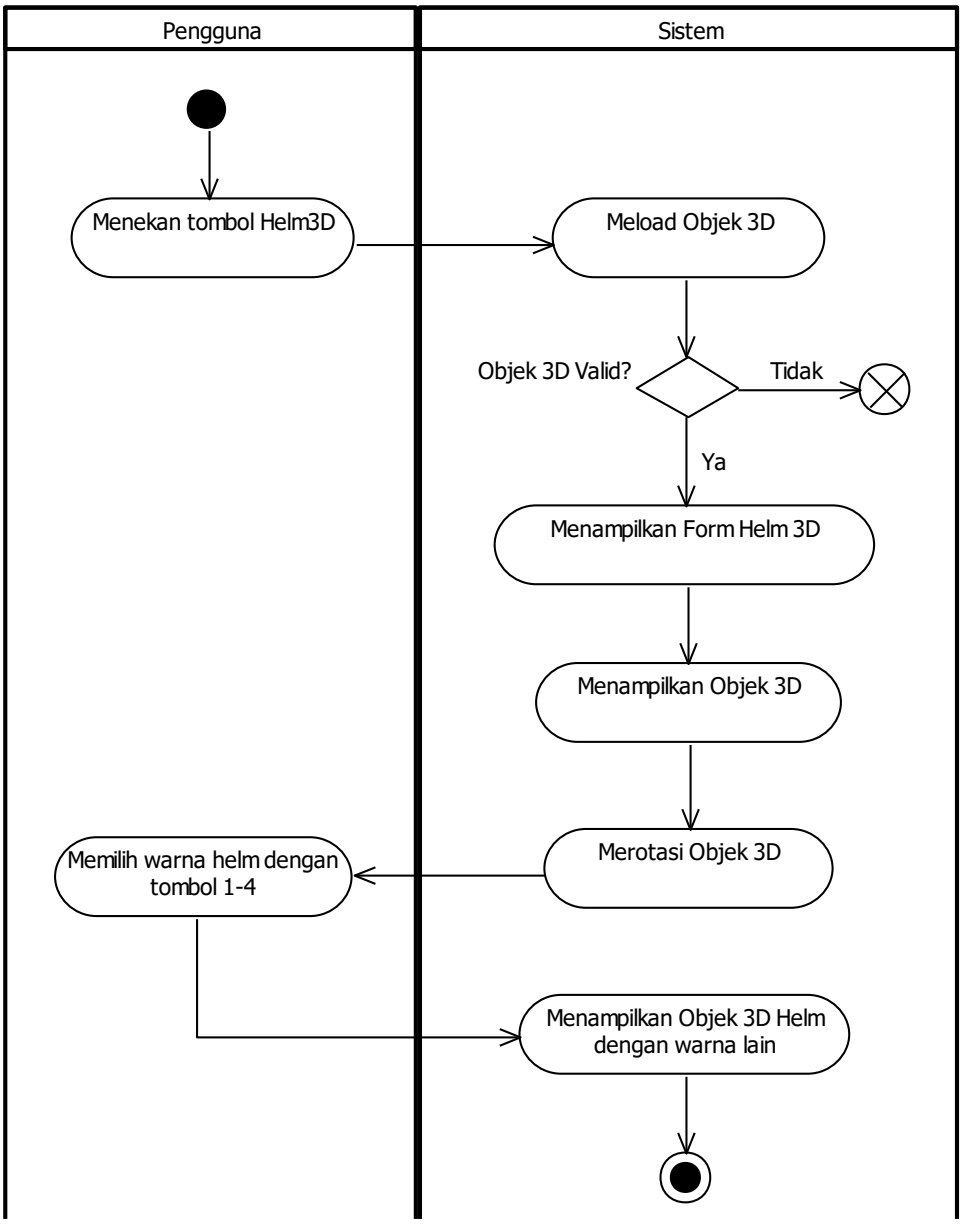
Activity diagram melihat foto helm dapat dilihat pada Gambar 3.14.



Gambar 3.14 *Activity Diagram* Melihat Foto Helm

3.7.5.5. *Activity Diagram* Menampilkan Model Helm 3D

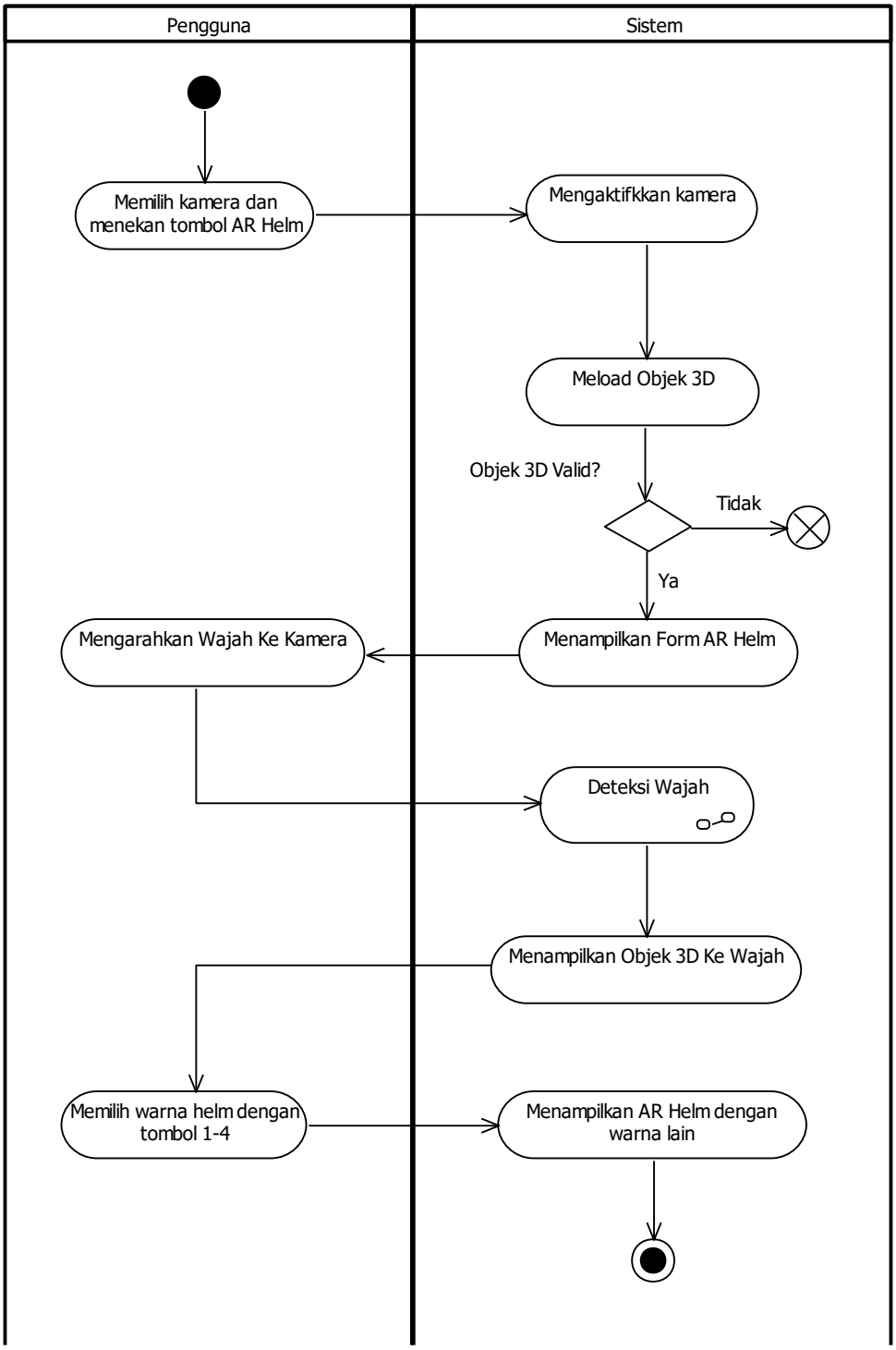
Activity diagram menampilkan model 3D helm dapat dilihat pada Gambar 3.15.



Gambar 3.15 *Activity Diagram* Menampilkan Model 3D Helm

3.7.5.6. *Activity Diagram* Menampilkan AR Helm

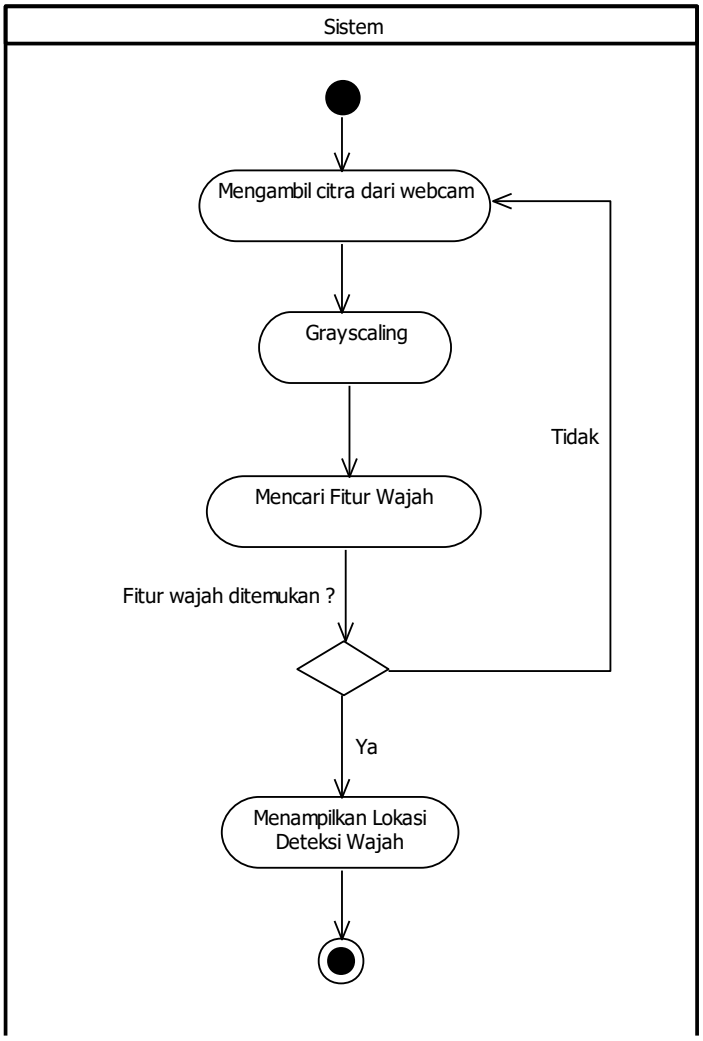
Activity diagram menampilkan AR helm dapat dilihat pada Gambar 3.16.



Gambar 3.16 *Activity Diagram* Menampilkan AR Model Helm

3.7.5.7. *Activity Diagram* Deteksi Wajah

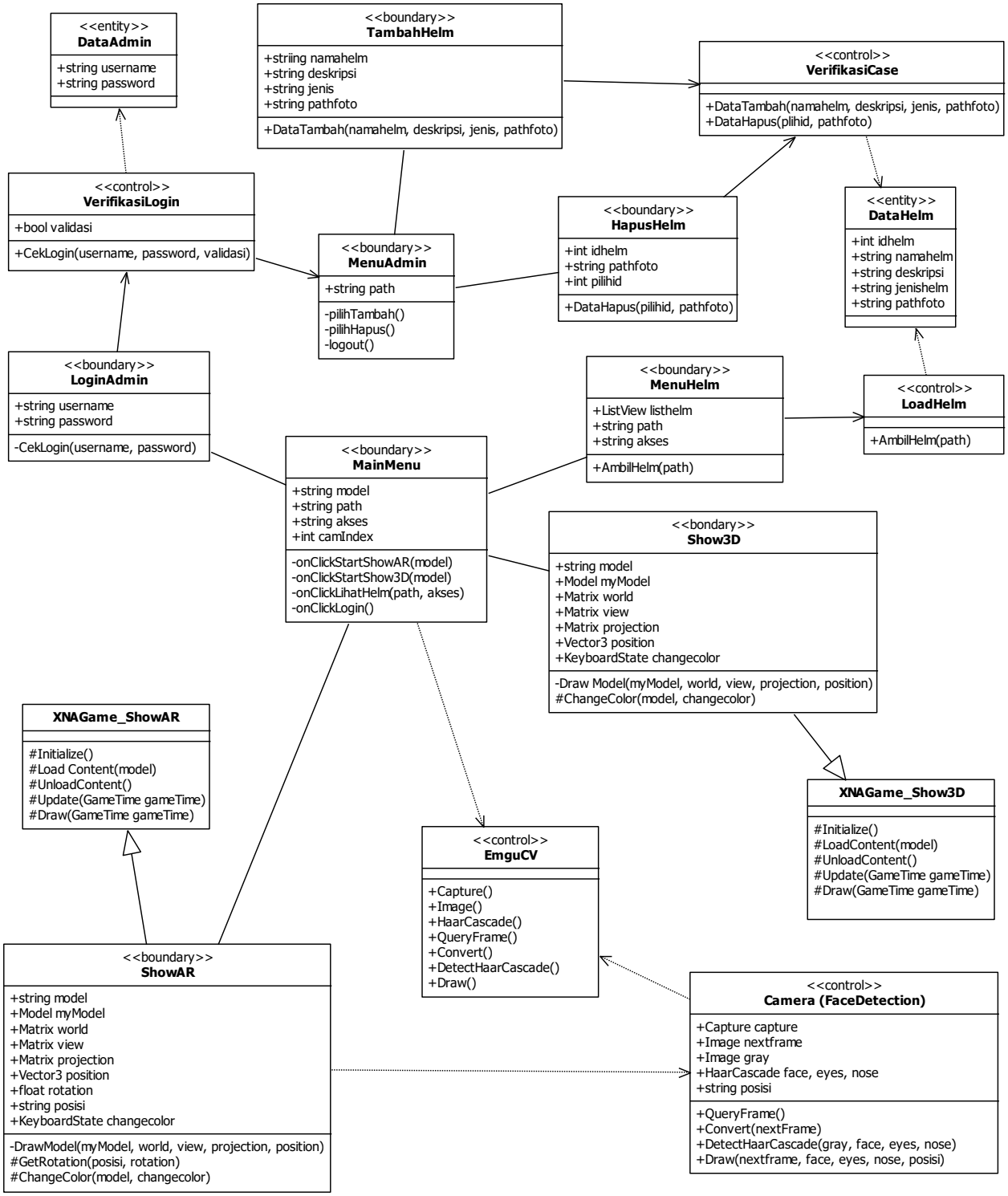
Activity diagram deteksi wajah dapat dilihat pada Gambar 3.17.



Gambar 3.17 *Activity Diagram* Deteksi Wajah

3.7.6. *Class Diagram*

Class Diagram menggambarkan struktur dan hubungan antra objek-objek yang ada pada sistem. Struktur itu meliputi atribut-atribut dan metode-metode yang ada pada masing-masing kelas. Untuk *class diagram* pada penelitian ini dapat dilihat pada Gambar 3.18.



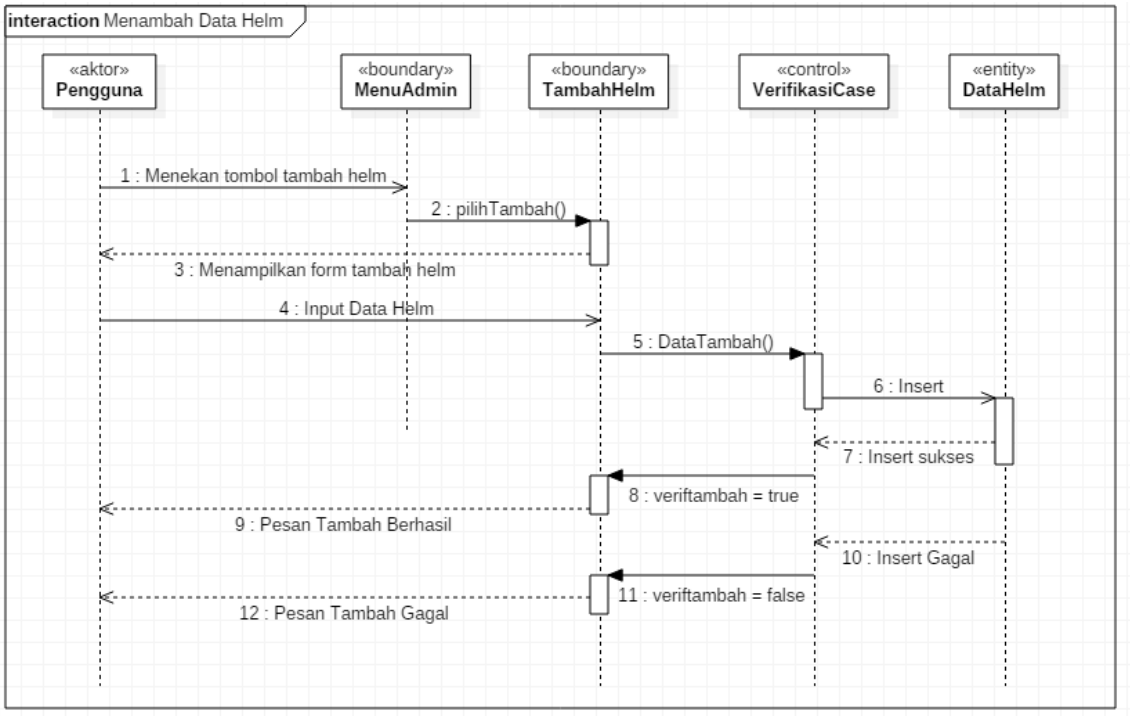
Gambar 3.18 ***Class Diagram***

3.7.7. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antara masing masing objek pada setiap *use case* dalam urutan waktu. Interaksi berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi. Adapun *sequence diagram* dari palikasi yang dibangun adalah sebagai berikut.

3.7.7.1. *Sequence Diagram* Menambah Foto Helm

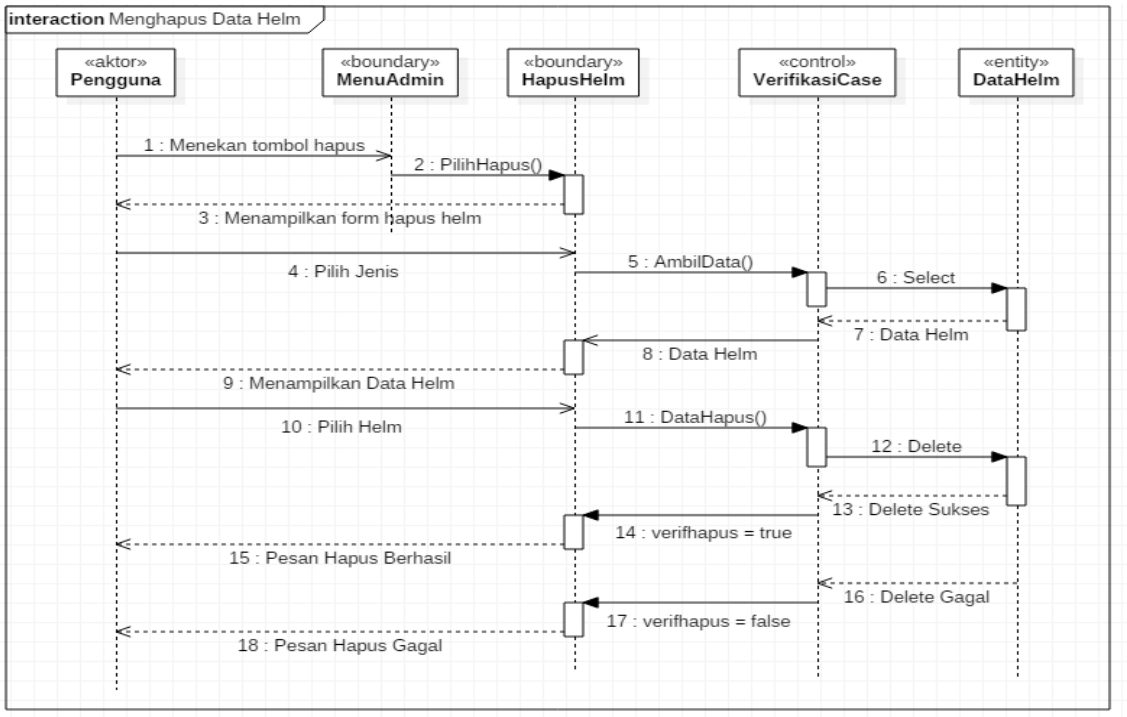
Sequence diagram menambah foto helm dapat dilhat pada Gambar 3.19.



Gambar 3.19 *Sequence Diagram* Menambah Foto Helm

3.7.7.2. *Sequence Diagram* Menghapus Foto Helm

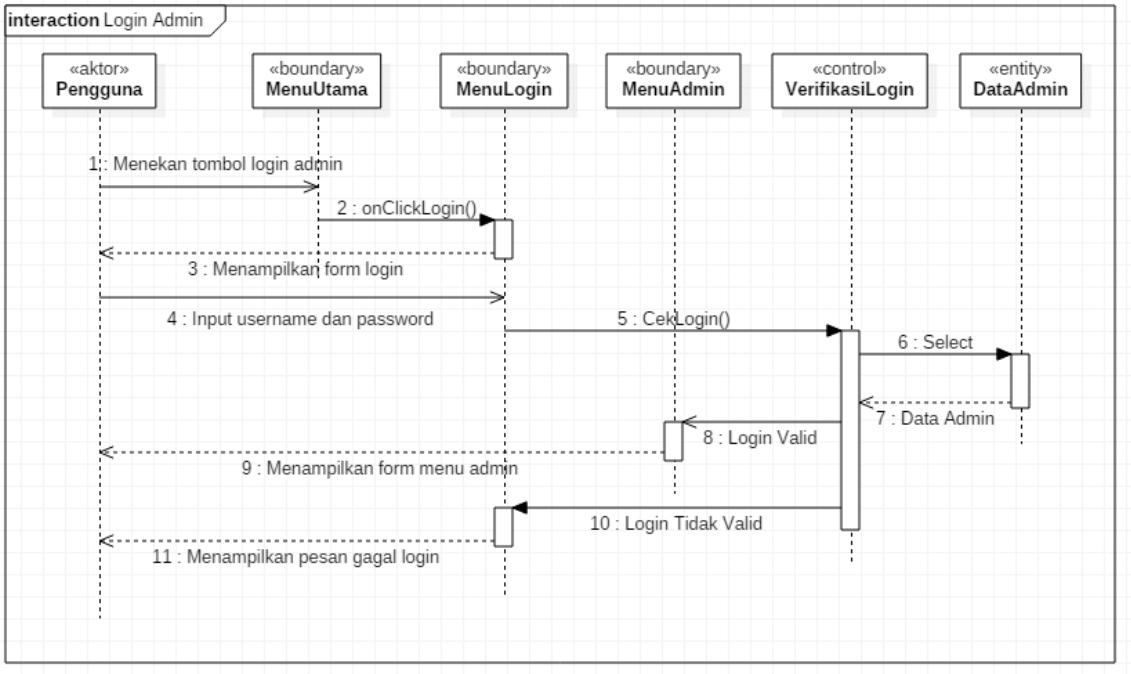
Sequence diagram menghapus foto helm dapat dilhat pada Gambar 3.20.



Gambar 3.20 *Sequence Diagram* Menghapus Foto Helm

3.7.7.3. *Sequence Diagram* Login Admin

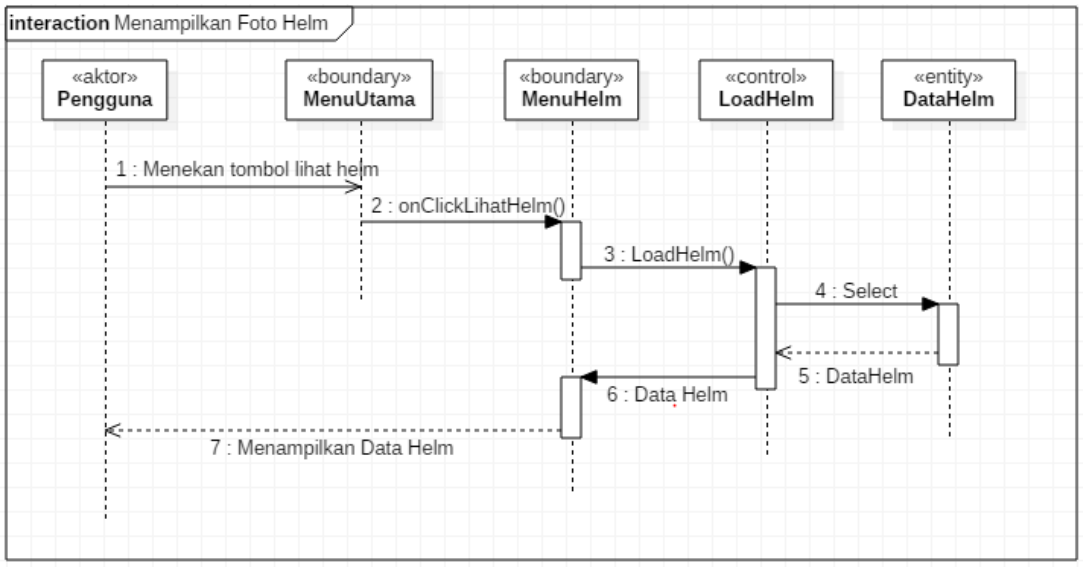
Sequence Diagram login admin dapat dilhat pada Gambar 3.21.



Gambar 3.21 *Sequence Diagram* Login Admin

3.7.7.4. *Sequence Diagram* Menampilkan Foto Helm

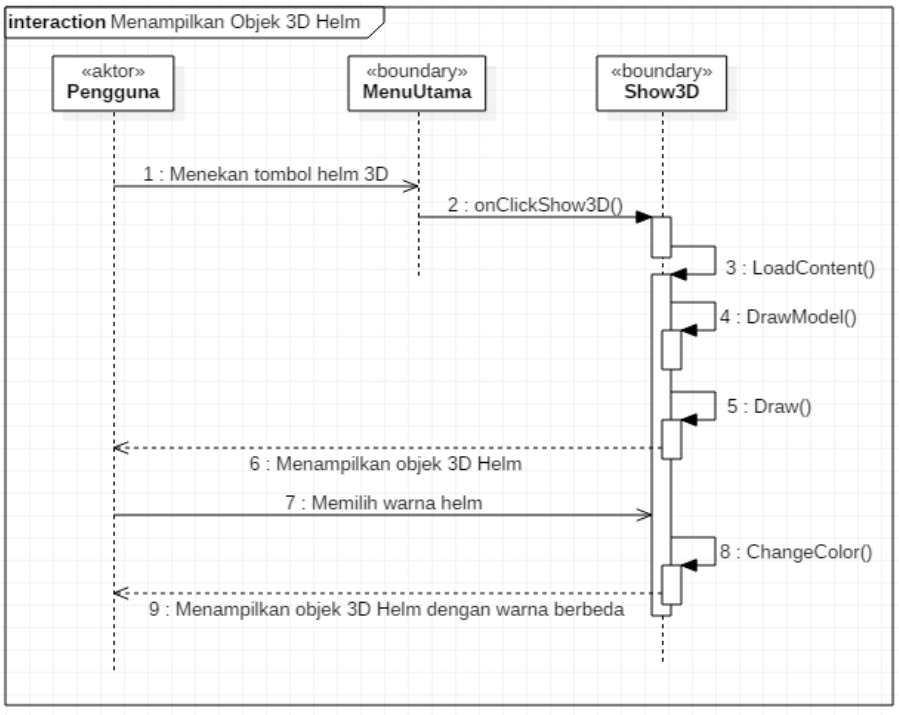
Sequence diagram menampilkan foto helm dapat dilhat pada Gambar 3.22.



Gambar 3.22 *Sequence Diagram* Menampilkan Foto Helm

3.7.7.5. *Sequence Diagram* Menampilkan Model 3D Helm

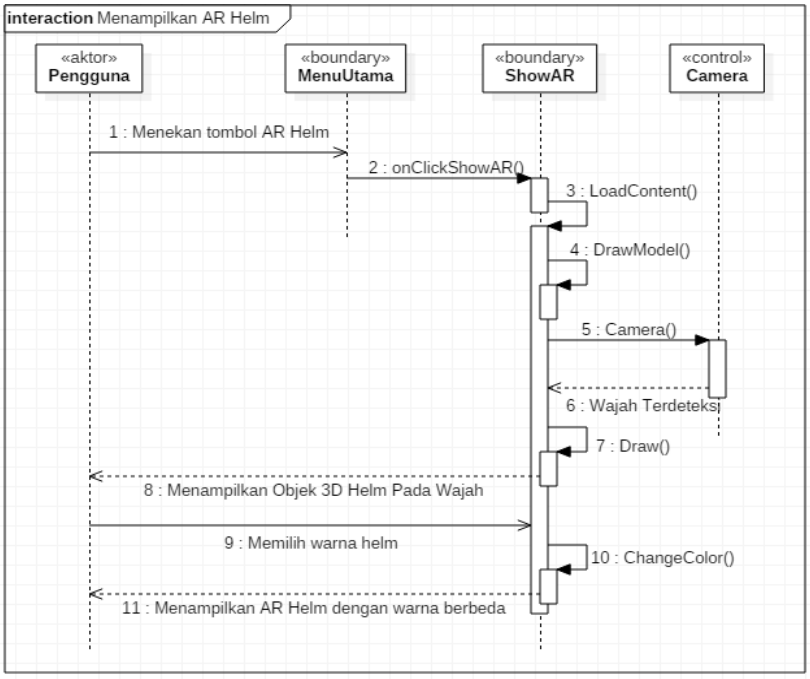
Sequence diagram menampilkan model 3D helm dapat dilihat pada Gambar 3.23.



Gambar 3.23 *Sequence Diagram* Menampilkan Model 3D Helm

3.7.7.6. *Sequence Diagram* Menampilkan AR Helm

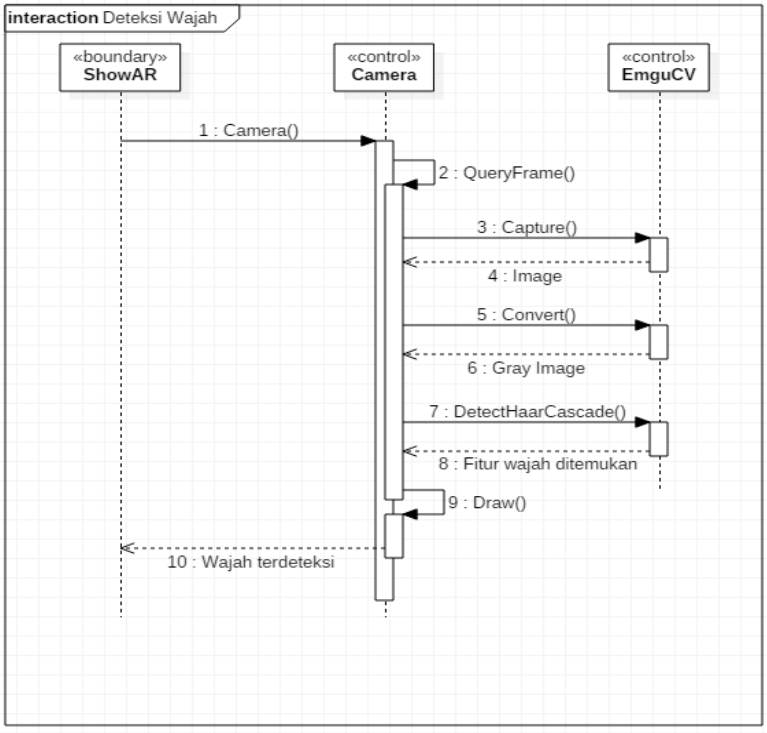
Sequence diagram menampilkan AR helm dapat dilihat pada Gambar 3.24.



Gambar 3.24 *Sequence Diagram* Menampilkan AR Helm

3.7.7.7. *Sequence Diagram* Deteksi Wajah

Sequence diagram deteksi wajah dapat dilihat pada gambar 3.25.



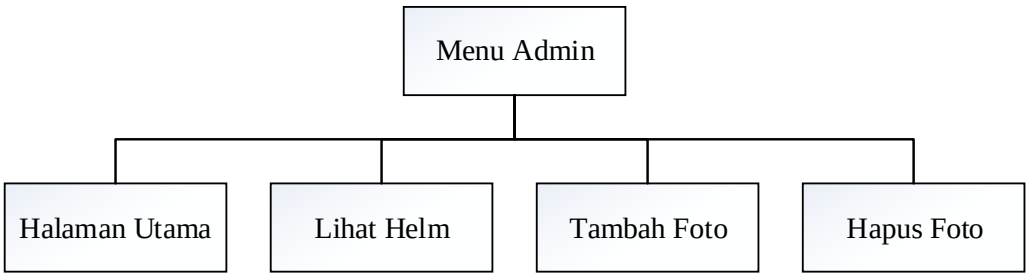
Gambar 3.25 *Sequence Diagram* Menampilkan Model Helm 3D

3.8. Perancangan Sistem

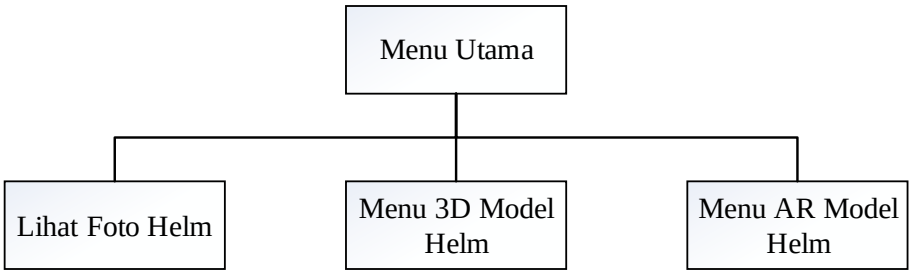
Perancangan sistem adalah penggambaran, perencanaan, dan pembuatan sketsa beberapa element yang terpisah ke dalam satu kesatuan yang utuh dan memiliki fungsi. Perancangan sistem pada aplikasi *augmented reality* ini meliputi perancangan struktur menu, perancangan antarmuka dan jaringan semantik.

3.8.1. Perancangan Struktur Menu

Perancangan struktur *menu* merupakan gambaran jalur pemakaian dari aplikasi sehingga aplikasi menjadi mudah dimengerti serta mudah digunakan dan juga saat menggunakan aplikasi, pengguna tidak mengalami banyak kesulitan dalam memilih *menu-menu* yang diinginkan. Perancangan struktur menu pada aplikasi ini terdapat dua struktur menu, yaitu struktur menu admin dan struktur menu konsumen. Struktur menu admin dapat dilihat pada Gambar 3.26. Struktur menu konsumen dapat dilihat pada Gambar 3.27.



Gambar 3.26 Perancangan Struktur Menu Admin



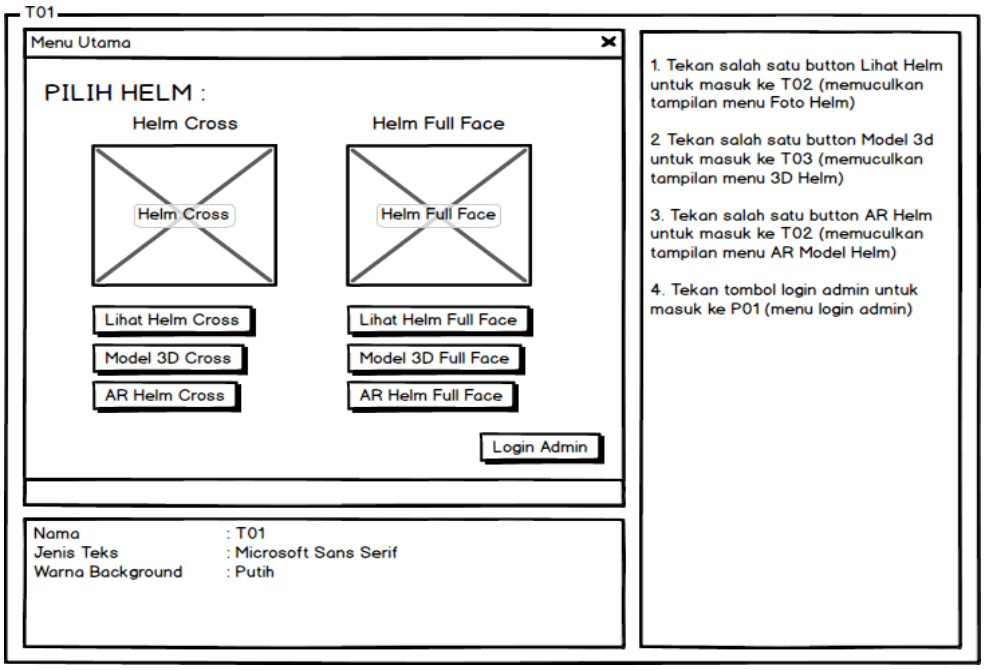
Gambar 3.27 Perancangan Struktur Menu Konsumen

3.8.2. Perancangan Antarmuka

Perancangan antarmuka merupakan suatu media komunukasi yang digunakan ketika *user* akan berinteraksi dengan sistem. Perancangan antarmuka pada aplikasi *augmented reality* ini adalah sebagai berikut :

3.8.2.1. Perancangan Antarmuka Menu Utama

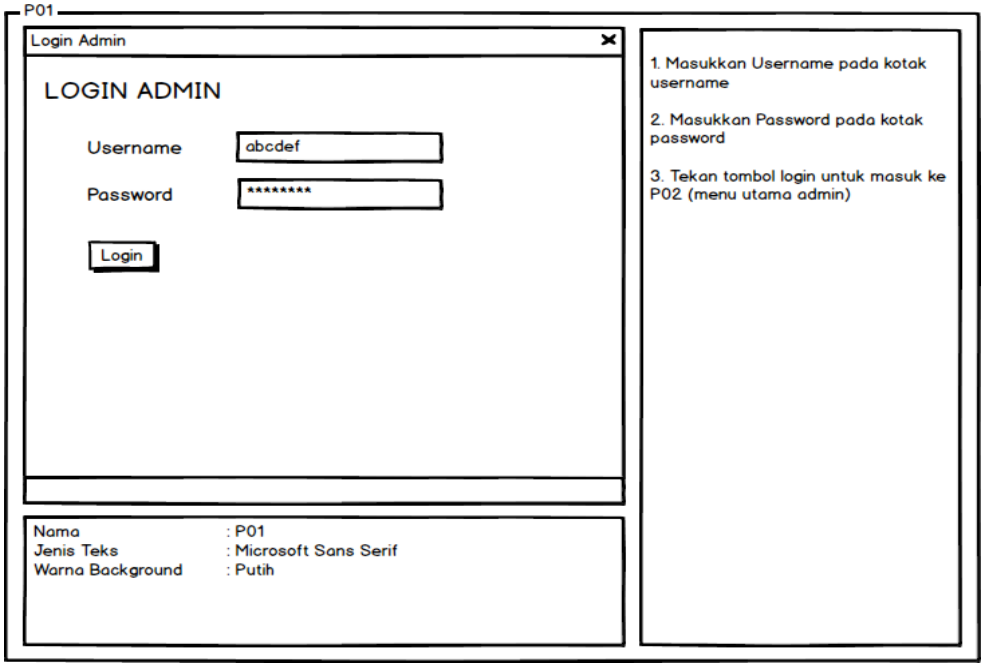
Perancangan antarmuka tampilan menu utama dapat dilihat pada Gambar 3.28.



Gambar 3.28 Perancangan Antarmuka Tampilan Menu Utama

3.8.2.2. Perancangan Antarmuka Login Admin

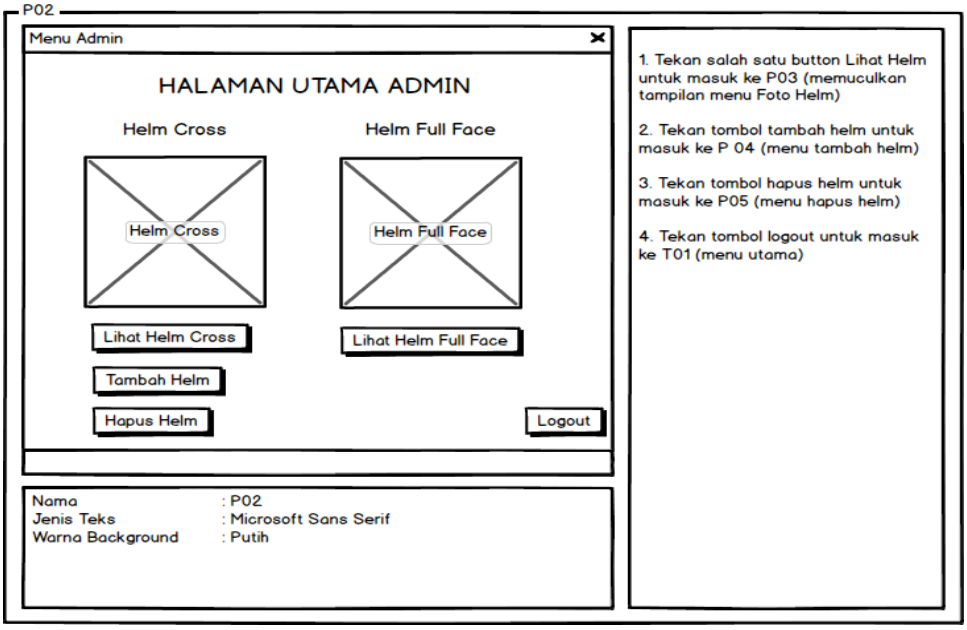
Perancangan antarmuka tampilan menu login admin dapat dilihat pada Gambar 3.29.



Gambar 3.29 Perancangan Antarmuka Tampilan Login Admin

3.8.2.3. Perancangan Antarmuka Halaman Utama Admin

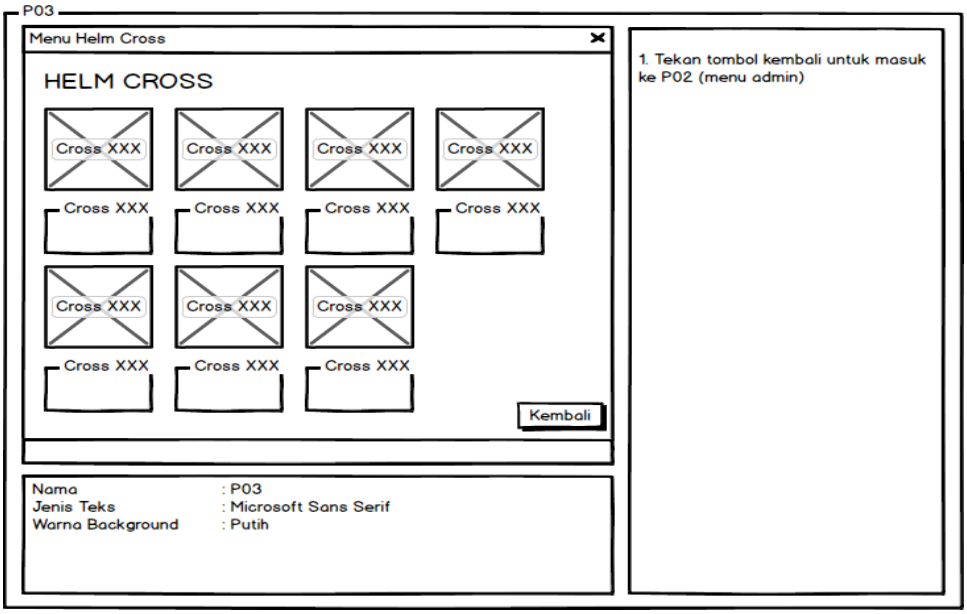
Perancangan antarmuka tampilan halaman utama admin dapat dilihat pada Gambar 3.30.



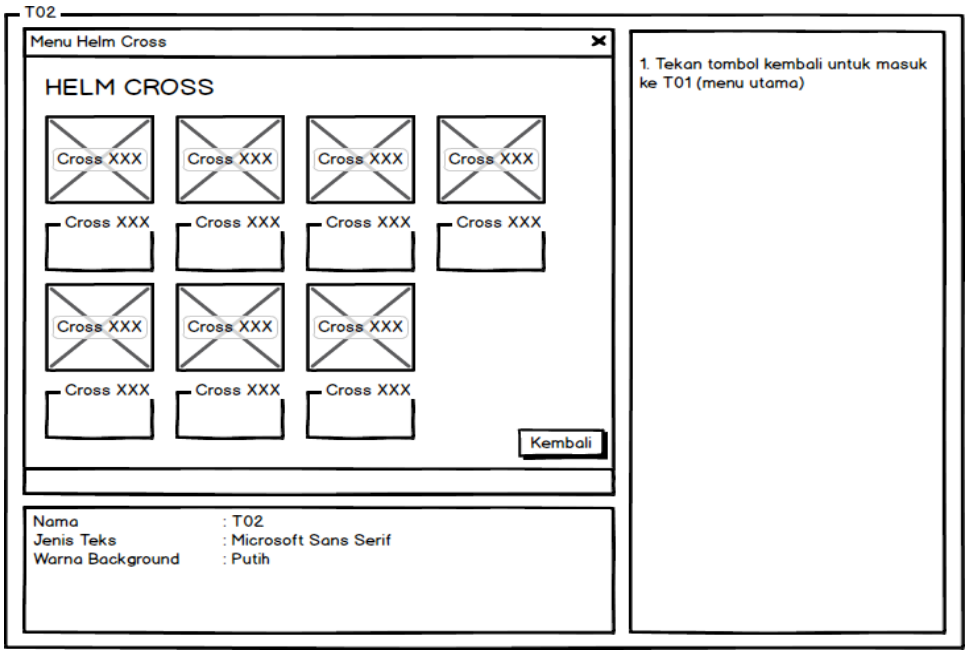
Gambar 3.30 Perancangan Antarmuka Halaman Utama Admin

3.8.2.4. Perancangan Antarmuka Lihat Helm

Perancangan antarmuka tampilan lihat helm memiliki 2 tampilan menu, yaitu tampilan antarmuka lihat helm admin yang dapat dilihat pada Gambar 3.31 dan tampilan antarmuka lihat helm konsumen yang dapat dilihat pada Gambar 3.32.



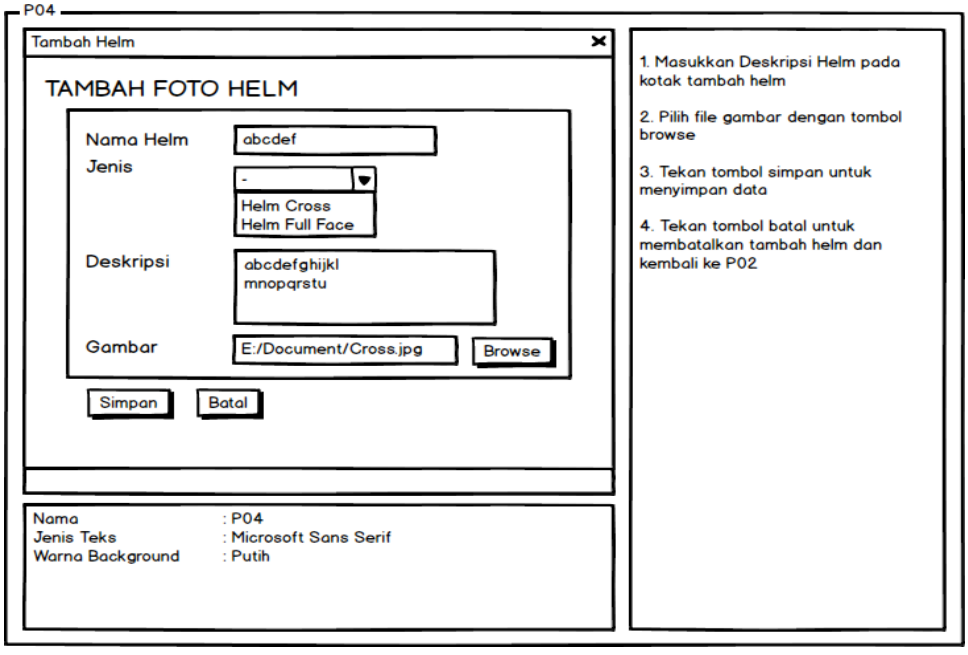
Gambar 3.31 Perancangan Antarmuka Lihat Helm Admin



Gambar 3.32 Perancangan Antarmuka Lihat Helm Konsumen

3.8.2.5. Perancangan Antarmuka Tambah Helm

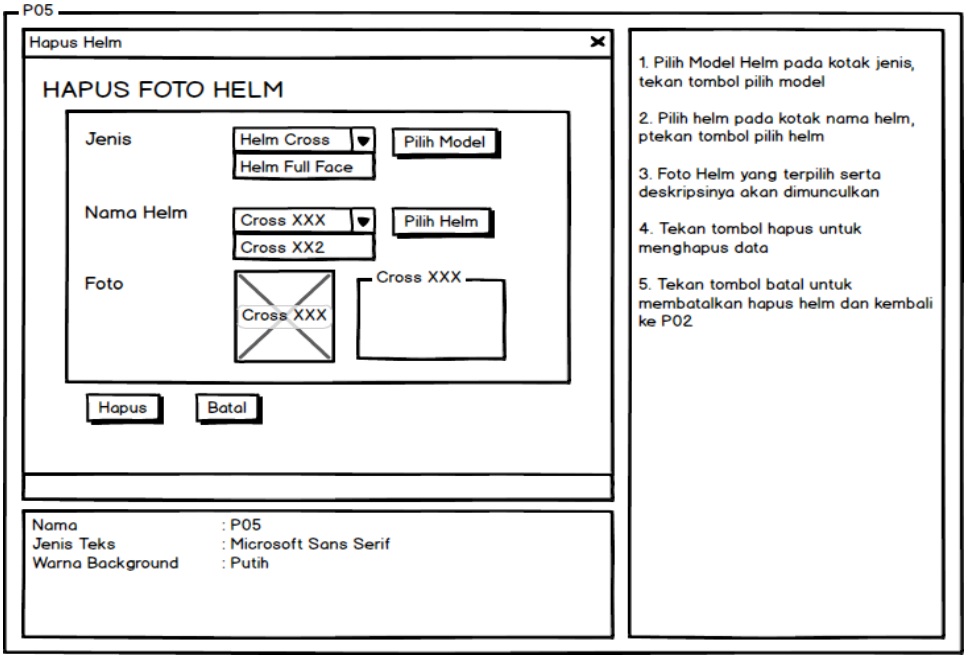
Perancangan antarmuka tampilan tambah helm dapat dilihat pada Gambar 3.33.



Gambar 3.33 Perancangan Antarmuka Tambah Helm

3.8.2.6. Perancangan Antarmuka Hapus Helm

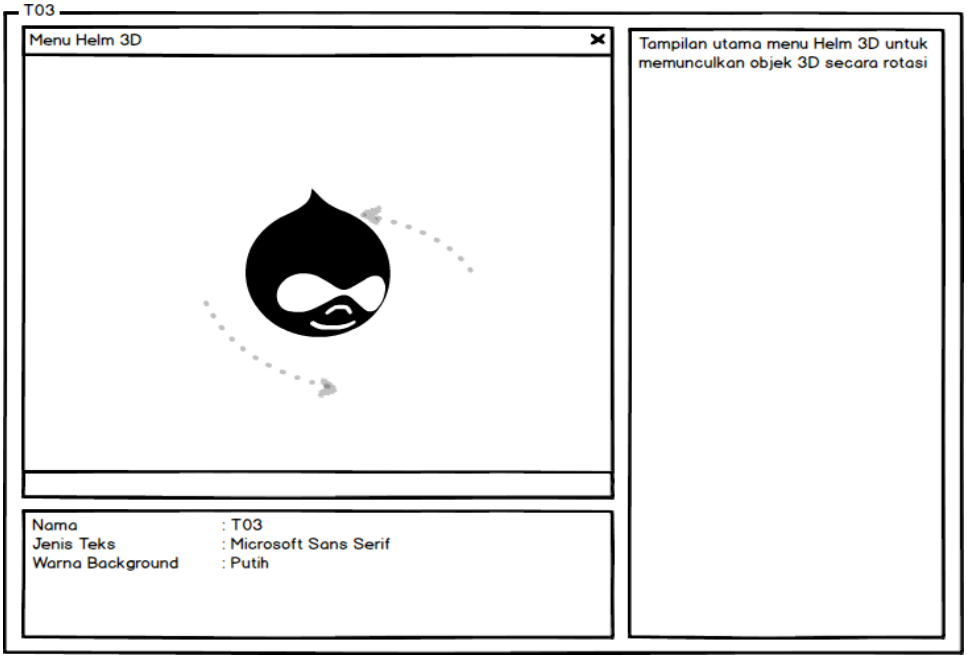
Perancangan antarmuka tampilan tambah helm dapat dilihat pada Gambar 3.34.



Gambar 3.34 Perancangan Antarmuka Hapus Helm

3.8.2.7. Perancangan Antarmuka Menu Helm 3D

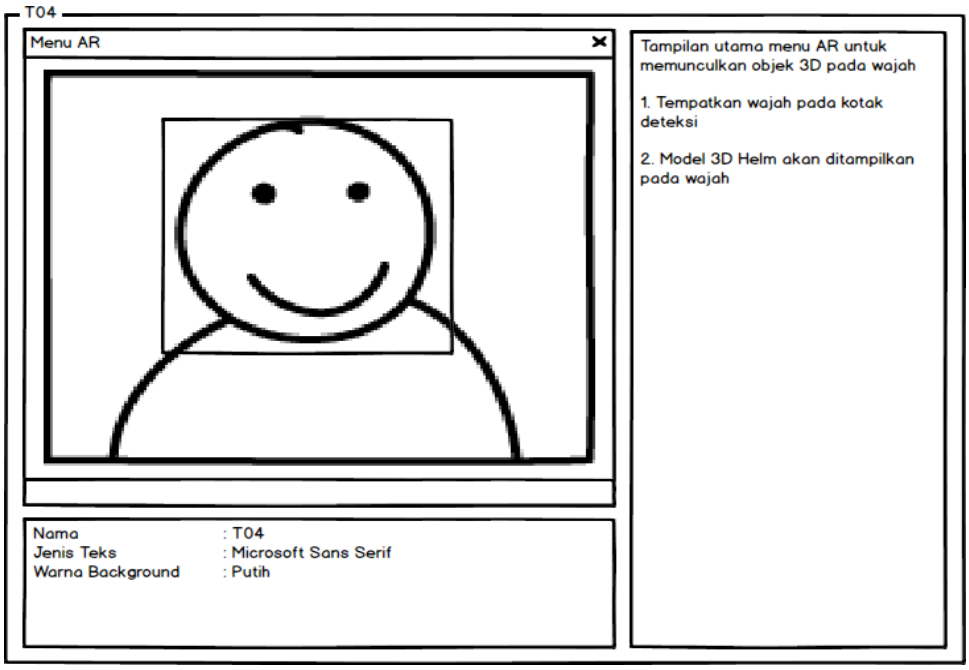
Perancangan antarmuka menu Helm 3D dapat dilihat pada Gambar 3.35.



Gambar 3.35 Perancangan Antarmuka Tampilan Menu Helm 3D

3.8.2.8. Perancangan Antarmuka Tampilan Menu AR

Perancangan antarmuka tampilan menu AR dapat dilihat pada gambar 3.36.



Gambar 3.36 Perancangan Antarmuka Tampilan Menu AR

3.8.3. Perancangan Pesan

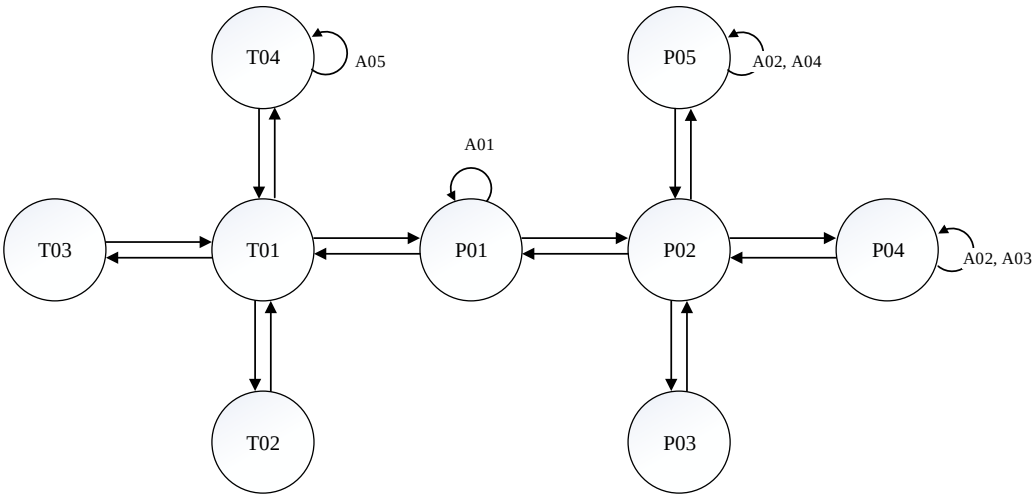
Perancangan antarmuka pesan pada aplikasi *augmented reality* ini dapat dilihat pada gambar 3.37.



Gambar 3.37 Perancangan Antarmuka Pesan

3.8.4. Jaringan Semantik

Jaringan semantik aplikasi *augmented reality* ini dapat dilihat pada Gambar 3.38.



Gambar 3.38 Jaringan Semantik Aplikasi

Keterangan:

T01 : Tampilan Menu Utama	P04 : Tampilan Menu Tambah Helm
T02 : Tampilan Lihat Helm (Konsumen)	P05 : Tampilan Menu Hapus Helm
T03 : Tampilan Menu Helm 3D	A01 : Tampilan Pesan Gagal Login
T04 : Tampilan Menu AR Helm	A02 : Tampilan Pesan Data Kosong
P01 : Tampilan Login Admin	A03 : Tampilan Pesan Simpan Berhasil
P02 : Tampilan Menu Admin	A04 : Tampilan Pesan Hapus Berhasil
P03 : Tampilan Lihat Helm (Admin)	A05 : Tampilan Pesan Kamera Tidak Aktif

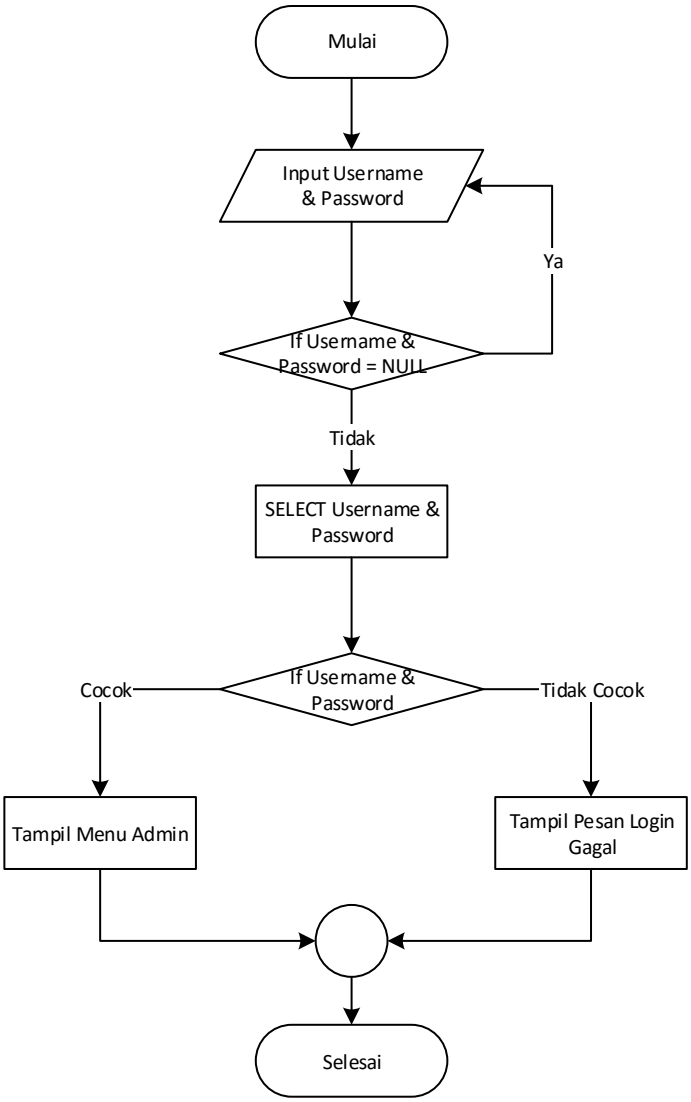
3.8.5. Perancangan *Method*

Perancangan *method* merupakan kumpulan metode yang ada pada sistem yang dirancang, dan berfungsi untuk menjalankan sistem. Berikut adalah perancangan *method* yang ada dalam sistem :

1. Perancangan *method* Cek Login

Method Cek Login digunakan pada saat pengguna ingin masuk ke dalam menu admin, dimana *method* ini berfungsi untuk mengvalidasi *username* dan *password* pengguna. Jika *username* dan *password* sudah benar dan sesuai dengan *database*, maka pengguna dapat masuk ke dalam menu admin untuk menambah dan

menghapus foto helm. Tetapi jika salah, maka sistem akan menampilkan pesan gagal login dan menampilkan kembali *form* login. *Method* Cek Login dapat dilihah pada Gambar 3.39.

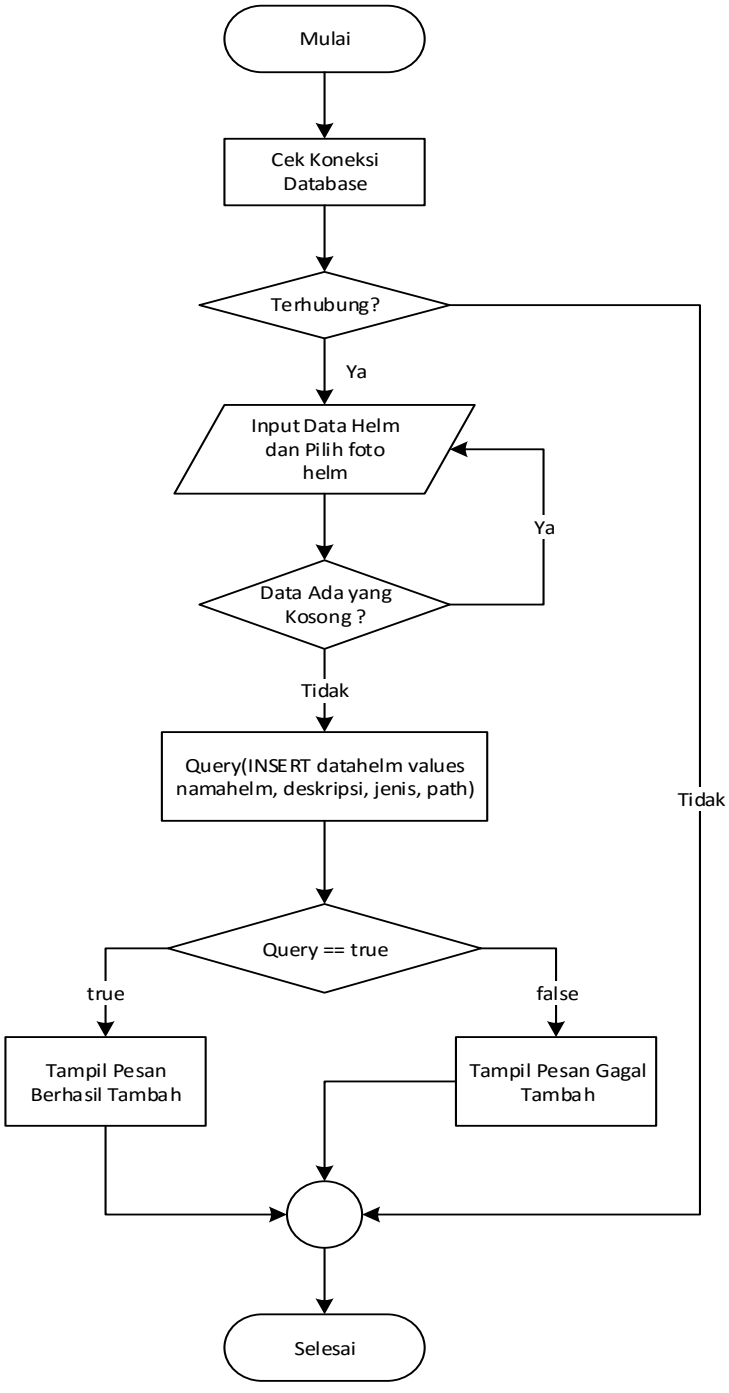


Gambar 3.39 Perancangan *method* Cek Login

2. Perancangan *method* Data Tambah

Method Data Tambah digunakan pada saat pengguna selaku admin ingin menambahkan foto helm pada menu tambah foto, dimana *method* ini berfungsi untuk mengvalidasi data yang masuk dalam *database*. Jika data sudah benar, maka sistem akan menyimpan data ke dalam *database*. Tetapi jika salah, maka sistem

akan menampilkan pesan gagal tambah. *Method* Data Tambah dapat dilihah pada Gambar 3.40.

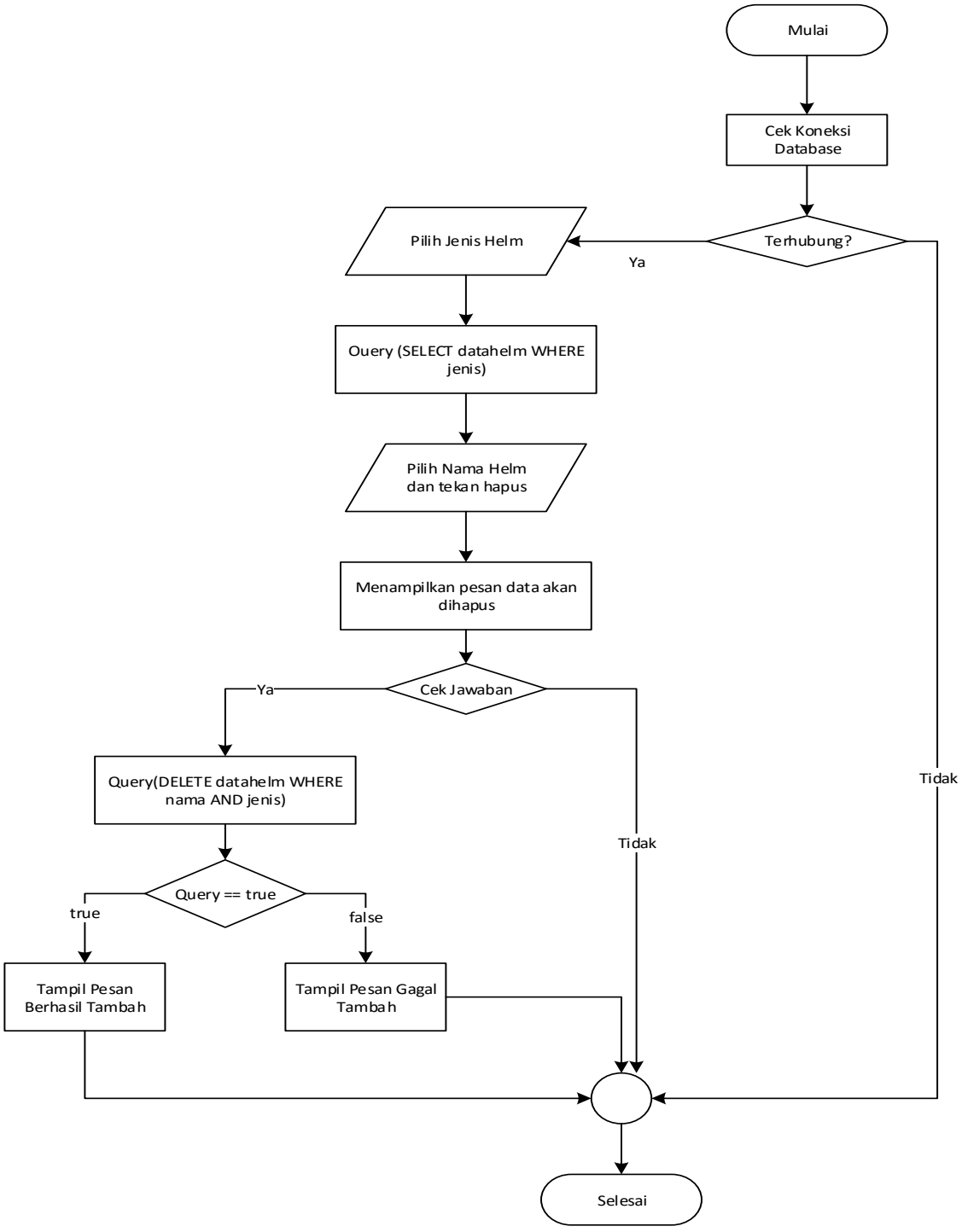


Gambar 3.40 Perancangan *method* Data Tambah

3. Perancangan *method* Data Hapus

Method Data Hapus digunakan pada saat pengguna selaku admin ingin menghapus foto helm pada menu hapus foto, dimana *method* ini berfungsi untuk

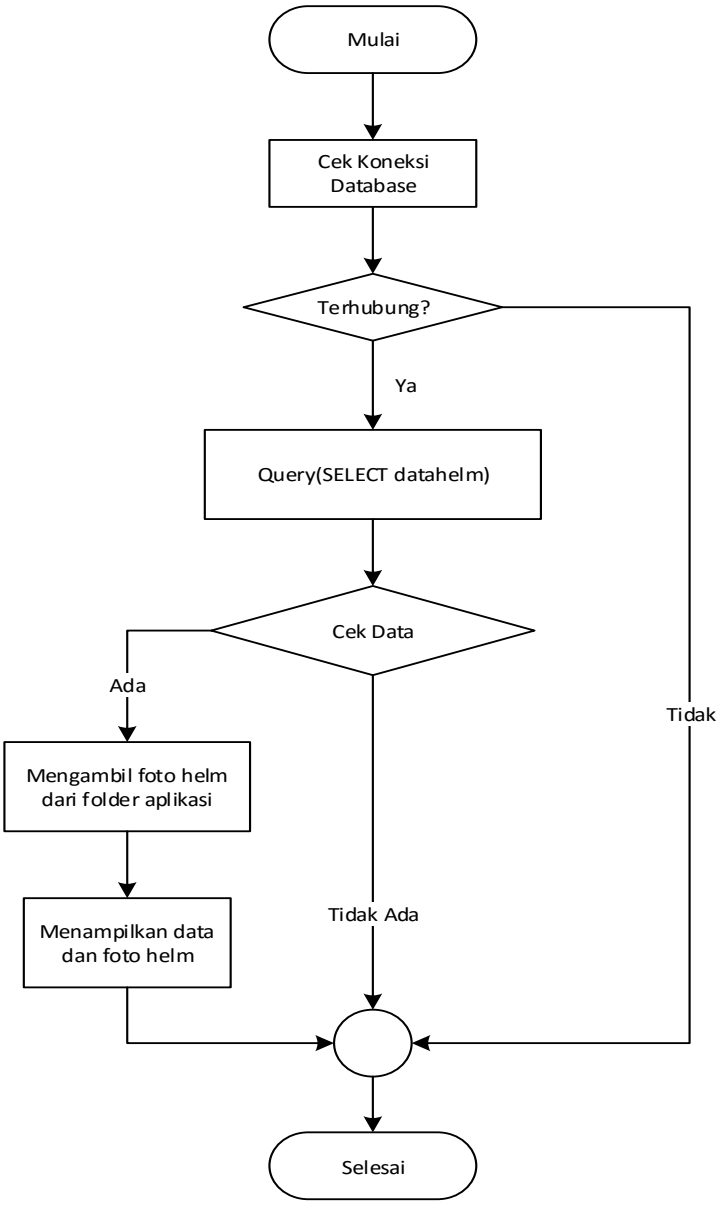
mengvalidasi data yang akan dihapus dari *database*. Jika data sudah benar, maka sistem akan menghapus data dari *database*. Tetapi jika salah, maka sistem akan menampilkan pesan gagal hapus. *Method* Data Hapus dapat dilihah pada Gambar 3.41.



Gambar 3.41 Perancangan *method* Data Hapus

4. Perancangan *method* Ambil Helm

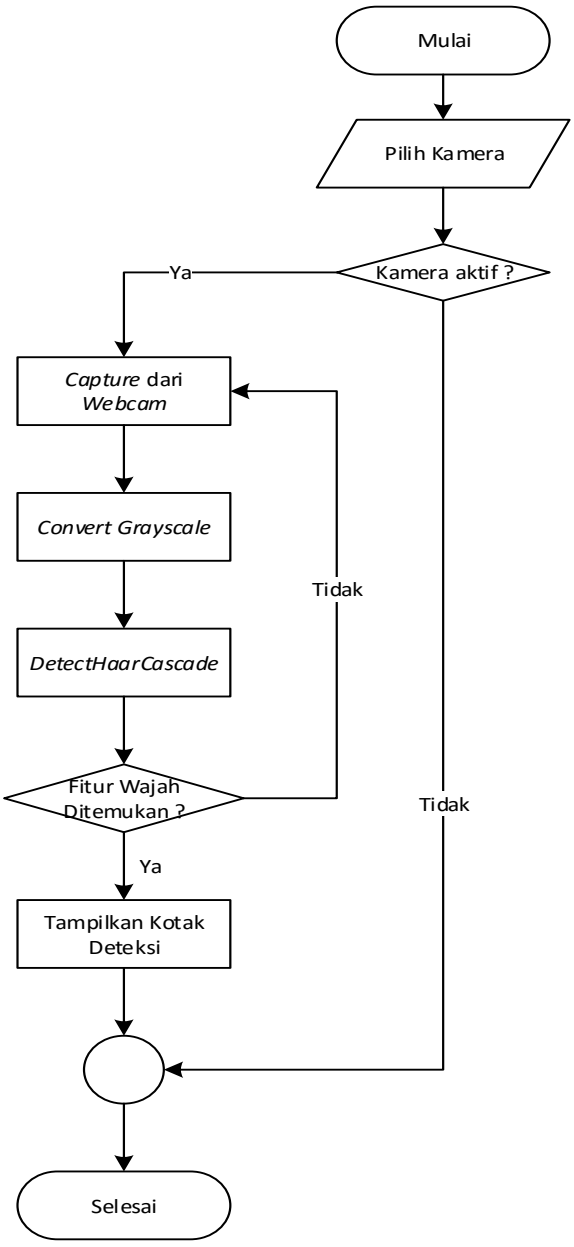
Method Ambil digunakan untuk mengvalidasi data yang diambil dari database. Jika data sudah benar, maka sistem akan menampilkan data dari *database*. Tetapi jika salah, maka sistem akan menampilkan pesan data kosong. *Method* Ambil Data dapat dilihah pada Gambar 3.42.



Gambar 3.42 Perancangan *method* ambil helm

5. Perancangan *method DetectHaarCascade*

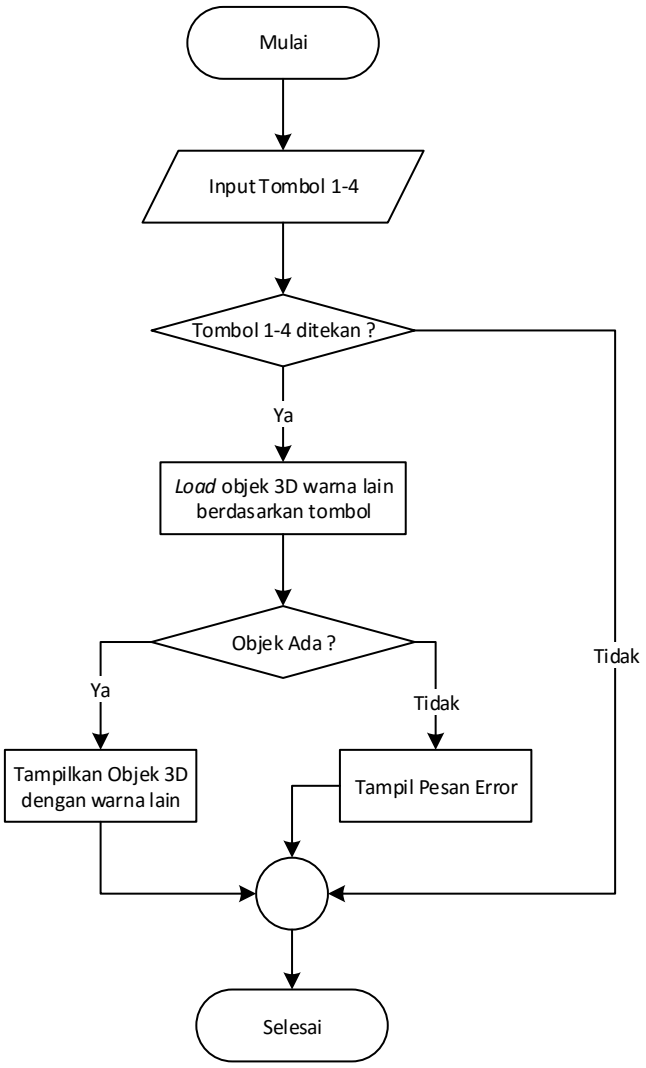
Method DetectHaarCascade digunakan pada saat sistem membaca wajah pengguna untuk dideteksi fitur-fitur wajahnya sehingga dapat ditampilkan objek 3D Helm. Jika terdeteksi oleh *HaarCascade*, maka sistem akan menampilkan kotak wajah terdeteksi sehingga objek 3D dapat muncul. Tetapi jika tidak terdeteksi, maka sistem akan mencoba kembali mencari fitur wajah pada *image capture* dari *webcam*. *Method DetectHaarCascade* dapat dilihah pada Gambar 3.43.



Gambar 3.43 Perancangan *method DetectHaarCascade*

6. Perancangan *method ChangeColor*

Method ChangeColor digunakan pada saat pengguna ingin mengubah warna objek 3D helm pada menu Helm 3D maupun menu AR Helm, dimana *method* ini berfungsi untuk mengvalidasi data masukan dari pengguna. Jika data masukan berupa angka 1 hingga 4, maka sistem akan mengubah warna objek 3D helm. Tetapi jika data masukan selain tombol 1 hingga 4, maka sistem tidak akan melakukan perubahan warna. *Method ChangeColor* dapat dilihah pada Gambar 3.44.



Gambar 3.44 Perancangan *method ChangeColor*