BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis sistem 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. Pada tahap analisis sistem ini dibagi menjadi beberapa bagian yaitu :

1. Analisis Algoritma
2. Analisis Terhadap Tools AR Sejenis
3. Analisis Library *Beyond Reality Face*
4. Analisis Kebutuhan Non Fungsional
5. Analisis Kebutuhan Fungsional

3.1.1 Analisis Algoritma

Di sistem ini webcam berperan penting bagi perangkat lunak untuk mendapatkan video masukan. Kamera mengambil frame-frame video untuk dapat diterima oleh komputer. Komputer memproses citra digital yang diakuisisi oleh webcam frame demi frame. Komputer akan mendeteksi pola yang mirip dengan wajah dari setiap frame video tersebut, yang kemudian lokasi wajah dapat ditentukan. Dengan informasi tersebut, objek *virtual* digabungkan dengan video

dari webcam dan merendernya sesuai dengan informasi posisi yang diperoleh dari *face detector* tersebut. Proses terjadi secara *real-time* sehingga model *virtual* yang tampil di media display akan mengikuti pergerakan wajah. Gambaran umum sistem ini dapat dilihat pada *flowchart* gambar 3.1.



Gambar 3.1 *Flowchart* sistem *virtual mirror eyeglasses*

3.1.1.1 Inisialisasi Kamera, *Face Detector*, dan GUI

Di dalam sistem perangkat lunak *virtual mirror eyeglasses* ini tahap awal yang harus ditentukan adalah *face detector* yang akan digunakan, sumber input videonya, dan *graphical user interface* (GUI) yang akan di *load*. Semua hal

tersebut tergambar di proses inisialisasi. Inisialisasi adalah proses penentuan awal semua hal yang diperlukan untuk menjalankan proses selanjutnya.

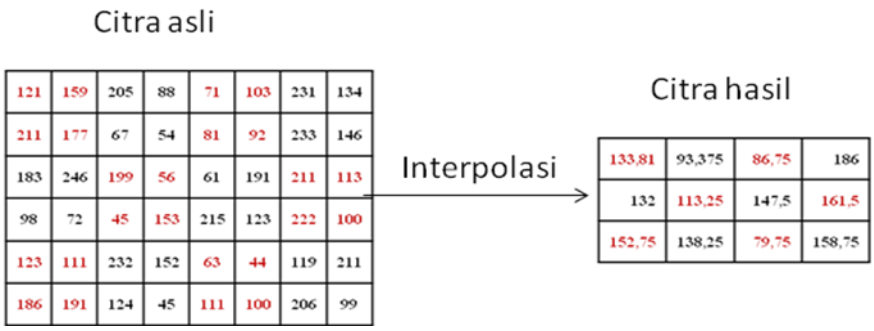
Pada inisialisasi ini, GUI diinisialisasikan terlebih dahulu untuk mendapatkan tampilan elemen-elemen *interface* sehingga memudahkan penggunaan bagi *user*. Inisialisasi dari kamera mulai dari menginisialisasi video masukan sampai pemrosesan *image*. *Face detector* akan mengatur hal-hal yang berkaitan dengan pendeteksian wajah.

3.1.1.2 *Pre-Processing Image*

Pada tahap pre-processing image dilakukan proses scaling, grayscaling, dan thresholding untuk menjadi inputan selanjutnya dalam metode *Haar Cascade*.

1. Tahap *Scaling*

   Scaling merupakan proses mengubah ukuran citra digital, hal ini perlu dilakukan agar semua citra digital memiliki ukuran yang sama. Untuk melakukan *scaling* di sistem ini dapat memanfaatkan package display object. Citra digital tersebut diperkecil dengan menggunakan metode interpolasi. Metode ini menggunakan rata-rata suatu region untuk mewakili region tersebut.



Gambar 3.2 Metode Interpolasi untuk memperkecil gambar

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

Tabel 3.1 Perhitungan nilai piksel hasil interpolasi

Nilai piksel citra asli	Nilai piksel citra hasil interpolasi
(121+159+211+177) / 4	133,81
(205+88+67+54) / 4	93,375
(71+103+81+92) / 4	86,75
(231+134+233+146) / 4	186
(183+246+98+72) / 4	132
(199+56+45+153) / 4	113,25
(61+191+215+123) / 4	147,5
(211+113+222+100) / 4	161,5
(123+111+186+191) / 4	152,75
(232+152+124+45) / 4	138,25
(63+44+111+100) / 4	79,75
(119+211+206+99) / 4	158,75

Berikut ini adalah citra hasil interpolasi :



Gambar 3.3 Penskalaan citra dengan metode Interpolasi

2. Tahap *Grayscaling*

Citra digital yang telah melalui proses penskalaan kemudian diubah menjadi citra dua warna dengan proses *grayscaling*. Proses pengubahan citra RGB menjadi citra *grayscale* adalah sebagai berikut :

Misalkan suatu citra wajah memiliki nilai :

R = 152

G = 132

B = 133

Maka nilai *grayscale* dari citra tersebut dapat dihitung seperti di bawah ini :

$$\mathbf{W = 0{,}2989}\boldsymbol{R} \mathbf{+ 0{,}5870}\boldsymbol{G} \mathbf{+ 0{,}1140}\boldsymbol{B} \quad \text{.....(3.1)}$$

w =(0,2989 x 152)+(0,5870 x 132)+(0,1140 x 133) = **138,0788**

Berikut ini adalah citra hasil *grayscaling* :

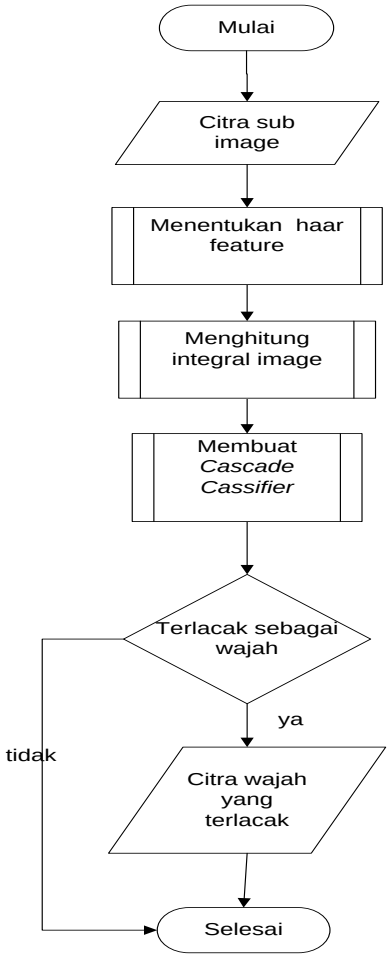


Gambar 3.4 Pengubahan citra RGB menjadi citra *grayscale*

3.1.1.3 Pendeteksian Metode *Haar Cascade Classifier*

Wajah merupakan salah satu bagian dari manusia yang memiliki ciri berbeda untuk setiap manusia. Pada penelitian ini digunakan metode haar cascade classifier sebagai metode untuk pengenalan pola wajah.

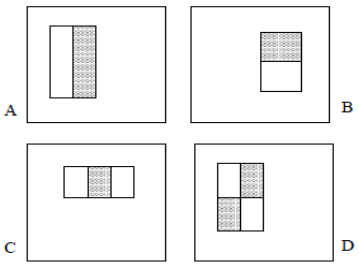
Perangkat lunak ini mengadopsi metode *Haar Cascade classifier* sebagai pendeteksi pola wajah, berikut ini alur proses metode *haar cascade classifier* :



Gambar 3.5 *Flowchart* Metode *Haar Cascades Classifier*

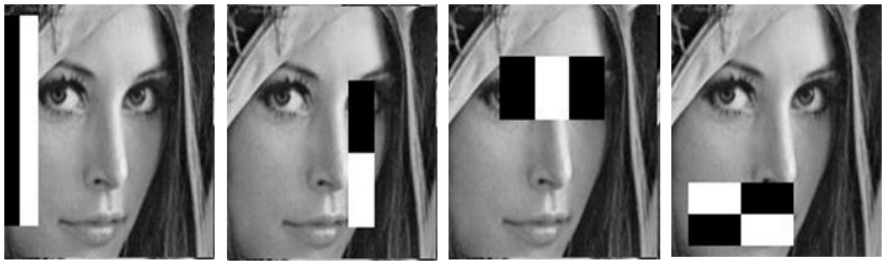
1. Proses Menentukan *Haar Feature*

Metode ini menggunakan *haar-like features* dimana perlu dilakukan *training* terlebih dahulu untuk mendapatkan suatu pohon keputusan dengan nama ***cascade classifier*** sebagai penentu apakah ada obyek atau tidak dalam tiap frame yang di proses. Terdapat tiga jenis fitur berdasarkan jumlah persegi panjang yang terdapat di dalamnya, seperti yang dapat dilihat pada gambar di bawah ini :



Gambar 3.6 *Haar-like feature* [10]

Setelah melakukan penskalaan sebelumnya kemudian dilakukan pencarian posisi wajah yaitu 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.



Gambar 3.7 Contoh pencarian dengan *haar-like feature*

2. Proses Menghitung *Integral Image*

Pada gambar 3.6 dapat dilihat bahwa fitur a dan b terdiri dari dua persegi panjang. Cara menghitung nilai dari fitur ini adalah mengurangkan nilai piksel pada area hitam dengan piksel pada area putih. Jika nilai perbedaannya itu di atas nilai ambang (*threshold*), maka dapat dikatakan bahwa fitur tersebut ada. Untuk menentukan ada atau tidaknya dari ratusan fitur *haar* pada sebuah citra dan pada skala yang berbeda secara efisien, digunakan satu teknik yang disebut *integral image*.

Citra integral adalah sebuah citra yang nilai tiap pikselnya merupakan akumulasi dari nilai piksel atas dan kirinya. Sebagai contoh, piksel (a,b) memiliki nilai akumulatif untuk semua piksel (x,y) dimana $x \leq a$ dan $y \leq b$. Salah satu contoh seperti perhitungan berikut :



Gambar 3.8 Sampel perhitungan *integral image*

1	2	3
2	3	4
4	5	1

1	2	3
2	3	4
4	5	1

1	1+2	1+2+3
1+2	1+2+2+3	1+2+3+2+3+4
1+2+4	1+2+2+3+4+5	1+2+3+2+3+4+4+5+1

Maka hasil perhitungan citra integralnya adalah :

1	3	6
3	8	15
7	17	25

Nilai fitur = | (total piksel hitam) – (total piksel putih)|

= | {[6+0-(0+0)]+[25+3-(15+0)]} – [15+1-(6+0)]

= | 6 + 13 – 10 |

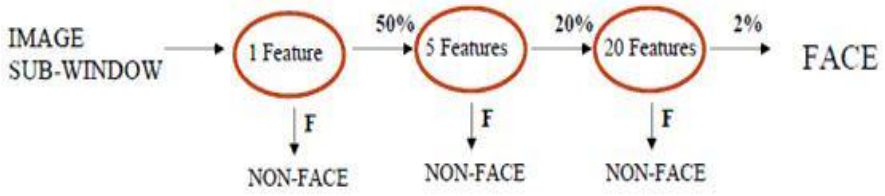
= 9

Nilai haar feature yang didapatkan di atas yaitu **9** adalah nilai perbedaan antar kotak (fitur putih dan fitur hitam) yang biasa disebut dengan *threshold*. *Threshold* ini digunakan sebagai parameter klasifikasi objek yang terdeteksi sebagai wajah atau tidak.

3. Proses Membuat *Cascade Classifier*

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*.

Di bawah ini adalah alur kerja dari klasifikasi bertingkat.



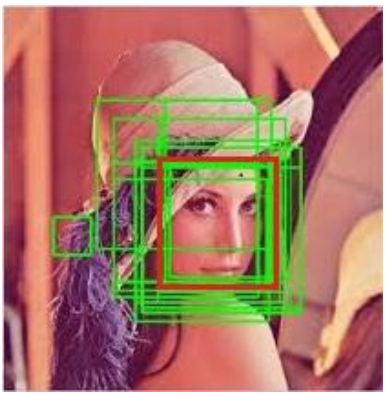
Gambar 3.9 Proses *Cascade Classifier*

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 3.9 di atas, dapat dijelaskan secara keseluruhan proses *cascade classifier*-nya sebagai berikut :

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

Sedangkan hasil dari filter pada proses cascade classifier dapat dilihat pada gambar 3.7.



Gambar 3.10 Hasil deteksi wajah

3.1.2 Analisis Terhadap Tools AR Sejenis

Pada subbab ini akan dibahas tentang analisis terhadap tools AR sejenis yaitu *face detection*. *Face detector* yang akan dibahas disini yaitu *library* Marilena yang merupakan *library* yang mendukung deteksi wajah pada flash dengan menggunakan ActionScript. Marilena telah berjalan baik untuk mendeteksi wajah atau benda lain yang berdasarkan algoritma *haar cascades*.

Marilena pada dasarnya berasal dari *library* OpenCV yang digunakan pada bahasa C yang dikonvesikan ke ActionScript oleh seorang pria berkebangsaan Jepang bernama Ohtsuka Masakazu. Lalu seorang pria bernama Mario Klingemann dari Quasimondo.com telah membuat beberapa perubahan untuk mengoptimasikan Marilena versi aslinya.

Adapun kelebihan dan kekurangan *library* Marilena ini sebagai berikut :

1. Kelebihan
   a. Merupakan *library open source* sehingga siapapun dapat dengan mudah menggunakannya serta memodifikasinya.
   b. Dapat mendeteksi objek lain selain wajah tergantung dari training set yang dipakai sebagai referensi pendeteksian.
2. Kekurangan
   a. *Library* yang masih sederhana, perlu optimasi dan modifikasi jika ingin meningkatkan hasil pendeteksiannya.
   b. Tidak dapat mendeteksi wajah pada posisi miring atau selain posisi tegak lurus.

3.1.3 Analisis Library *Beyond Reality Face*

Pada subbab ini akan dibahas tentang analisis *library Beyond Reality Face* (BRF) yang digunakan dalam pembangunan aplikasi *Virtual Mirror Eyeglasses* ini. Library ini merupakan library *Augmented Reality* pada Flash/Actionscript 3 dalam mendeteksi wajah. SDK ini memungkinkan untuk mendeteksi wajah, fitur-fitur, posisi, dan pose 3D-nya dalam Flash. Penggunaannya seperti berikut : masukkan sebuah objek BitmapData dan kelas *IBRFContentContainer* ke dalam kelas *BeyondRealityFaceManager*, perbaharui objek BitmapData dan jika wajah telah ditemukan, wajah dideteksi dan pose nya di-estimasi-kan. Analisis library berupa kelas-kelas dengan metode, properti, *event handler,* dan *event listener*, serta informasi penggunaannya.

3.1.3.1 *Class Interface IBeyondRealityFace*

IBeyondRealityFace merupakan interface yang akan diimplementasikan oleh kelas *BeyondRealityFaceManager*. Kelas ini mempunyai method setter dan getter yang nanti akan dipanggil sebagai properti dari instance pada kelas *BeyondRealityFaceManager*. Method-method yang terdapat pada kelas interface ini yaitu :

1. init() – inisialisasi *Beyond Reality Face*.

   Parameter yang dibutuhkan adalah :

   a. bitmapData:BitmapData – sebagai referensi untuk menganalisis, dengan membuat konten baru ke BitmapData ini, kemudian memanggil fungsi update(). BitmapData ini dapat di-update kemudian melalui bitmapData setter.

   b. contentContainer:IBRFContentContainer (default=null) – menggunakan satu dari content container yang diberikan untuk 3D overlay.

   Output dari method ini adalah :

   a. Boolean – true, jika init telah dilakukan. False, jika lib tidak diinisialisasi.

2. reset() – reset algoritma untuk mulai mendeteksi wajah kembali.
3. update() – hanya meng-update bitmapData instance sebelum memanggil fungsi update().

Sedangkan *public properties* yang ada pada kelas *IBeyondRealityFace* ini adalah sebagai berikut :

1. bitmapData : BitmapData – mengembalikan sumber BitmapData pada saat itu.
2. deleteLastDetectedFace : Boolean – mengembalikan, apakah *lastDetectedFace* menjadi null, ketika tidak ada wajah yang terdeteksi.
3. faceShape : Shape – mengembalikan referensi bentuk wajah. Objek ini mencakup semua poin bentuk wajah dengan beberapa referensi poin bantuan untuk mata, mulut, hidung, dan lain-lain.
4. isEstimatingFace : Boolean – mengembalikan, apakah estimasi wajah dibolehkan atau tidak.
5. isEstimatingPose : Boolean - mengembalikan apakah estimasi pose dibolehkan atau tidak.
6. isStabilizing : Boolean – mengembalikan apakah bentuk wajah distabilkan. True, untuk kemampuan bergerak cepat.
7. isTracking : Boolean – mengembalikan apakah *tracking* dibolehkan atau tidak.
8. lastDetectedFace : Rectangle – [read-only], mengembalikan wajah yang terakhir terdeteksi, akan terus update selama BRF dalam mode deteksi. Ketika BRF dalam mode estimasi tidak akan diupdate sampai estimasi wajah gagal.

9. lastTask : int – [read-only], mengembalikan *task*, yang telah dilakukan sebelum "*task"*.
10. nextTask : int – [read-only], mengembalikan *task*, yang akan dilakukan selanjutnya.
11. task : int – [read-only], mengembalikan *task*, yang telah dilakukan dalam update.
12. vars : BRFVars – [read-only], mengembalikan variable BRF. BRFVars mencakup kelas variable untuk 3 langkah algoritma : *face detection, face estimation,* dan *pose estimation.*

3.1.3.2 *Class BeyondRealityFaceManager*

Kelas *BeyondRealityFaceManager* ini men-dispatch sebuah "init" dan "ready" event. Mengimplementasikan kelas interface *IBeyondRealityFace*. Mehod-method yang terdapat pada kelas ini yaitu :

1. BeyondRealityFaceManager() – sebagai constructor. Mulai menginisialisasi SDK.

   Parameter yang dibutuhkan adalah :

   a. stage:Stage (default = null) – jika not null, otomatis inisialisasi proses validasi lisensi.
   b. params:Object (default = null)
2. init() – implementasi dari interface *IBeyondRealityFace*

   Parameter yang dibutuhkan adalah :

   a. bitmapData:BitmapData

b. contentContainer:IBRFContentContainer (default = null)

Output dari method ini yaitu :

a. Boolean - true, jika init telah dilakukan. False, jika lib tidak diinisialisasi.

3. reset() – implementasi dari interface *IBeyondRealityFace*
4. update() – implementasi dari interface *IbeyondRealityFace*

Dan memiliki properti implementasi dari kelas interface *IBeyondRealityFace* yaitu :

1. bitmapData : BitmapData – mengembalikan sumber BitmapData pada saat itu.
2. deleteLastDetectedFace : Boolean – mengembalikan, apakah *lastDetectedFace* menjadi null, ketika tidak ada wajah yang terdeteksi.
3. faceShape : Shape – mengembalikan referensi bentuk wajah. Objek ini mencakup semua poin bentuk wajah dengan beberapa referensi poin bantuan untuk mata, mulut, hidung, dan lain-lain.
4. isEstimatingFace : Boolean – mengembalikan, apakah estimasi wajah dibolehkan atau tidak.
5. isEstimatingPose : Boolean - mengembalikan apakah estimasi pose dibolehkan atau tidak.
6. isStabilizing : Boolean – mengembalikan apakah bentuk wajah distabilkan. True, untuk kemampuan bergerak cepat.

7. isTracking : Boolean – mengembalikan apakah *tracking* dibolehkan atau tidak.
8. lastDetectedFace : Rectangle – [read-only], mengembalikan wajah yang terakhir terdeteksi, akan terus update selama BRF dalam mode deteksi. Ketika BRF dalam mode estimasi tidak akan diupdate sampai estimasi wajah gagal.
9. lastTask : int – [read-only], mengembalikan *task,* yang telah dilakukan sebelum "*task".*
10. nextTask : int – [read-only], mengembalikan *task,* yang akan dilakukan selanjutnya.
11. task : int – [read-only], mengembalikan *task,* yang telah dilakukan dalam update.
12. vars : BRFVars – [read-only], mengembalikan variable BRF. BRFVars mencakup kelas variable untuk 3 langkah algoritma : *face detection, face estimation,* dan *pose estimation.*

3.1.3.3 *Class Interface IBRFContentContainer*

IBRFContentContainer merupakan sebuah kelas interface untuk mendapatkan 3D pose dari SDK. Mehod-method yang terdapat dalam kelas interface ini adalah :

1. isValidPose() – method ini mengembalikan nilai apakah 3D pose saat ini valid atau tidak. Method ini juga digunakan BRF ketika menghitung pose 3D.

Output dari method ini yaitu :

a. Boolean – true jika valid dan sebaliknya.

2. resetPose() – method ini mengatur ulang 3D pose ke nilai default, ketika 3D pose tidak valid.
3. updatePose() – method ini dipanggil ketika ditemukan 3D pose yang baru.

   Parameter yang dibutuhkan adalah :

   a. resultMatrix:ResultMatrix – 3D matriks yang berisi rotasi dan translasi kamera.

3.1.3.4 *Class BRFVars*

BRFVars merupakan sebuah kelas yang mengatur variabel-variabel untuk *face detection*, *face estimation*, dan *pose estimation.* Tidak ada method dalam kelas ini. Sedangkan *public properties* yang ada pada kelas *BRFVars* ini adalah sebagai berikut :

1. faceDetectionVars : FaceDetectionVars – merupakan properti untuk *face detection.*
2. faceEstimationVars : FaceEstimationVars – merupakan properti untuk *face estimation.*
3. poseEstimationVars : PoseEstimationVars – merupakan properti untuk *pose estimation.*

3.1.3.5 *Class FaceDetectionVars*

FaceDetectionVars merupakan sebuah kelas yang mengatur variabel-variabel *face detection.* Ketika jauh dari kamera wajah lebih kecil, ketika dekat ke kamera maka wajah menjadi lebih besar. *Face detection* mendeteksi ukuran wajah menggunakan properti *baseScale,* dimulai dari *baseScale,* dan kenaikan oleh *scaleIncrement* dan berakhir dengan skala lebih kecil sama dengan *maxScale*.

Untuk nilai default ini berarti *baseScale* : 4.0, *scaleIncrement* : 0.5, *maxScale* : 5.0, sehingga ada 3 iterasi yaitu dengan skala 4.0, 4.5, dan 5.0. Dengan *baseScale* 4.0 dan *maxScale* 5.0 berarti sebuah jarak normal ”wajah ke webcam”.

Method-method yang terdapat dalam kelas ini adalah :

1. updateFaceDetectionROI() – method yang melakukan update *region of interest* (ROI) wajah yang terdeteksi. BRF bekerja dengan dimensi BitmapData 640x480, *region of interest* (ROI) tidak boleh melebihi ukuran tersebut.

   Parameter yang dibutuhkan yaitu :

   a. x : Number
   b. y : Number
   c. width : Number
   d. height : Number

2. updateLeftEyeDetectionROI() - method yang melakukan update *region of interest* (ROI) mata kiri yang terdeteksi. BRF bekerja dengan dimensi BitmapData 640x480, *region of interest* (ROI) tidak boleh melebihi ukuran tersebut.

Parameter yang dibutuhkan yaitu :

a. x : Number
b. y : Number
c. width : Number
d. height : Number

3. updateRightEyeDetectionROI() - method yang melakukan update *region of interest* (ROI) mata kanan yang terdeteksi. BRF bekerja dengan dimensi BitmapData 640x480, *region of interest* (ROI) tidak boleh melebihi ukuran tersebut.

   Parameter yang dibutuhkan yaitu :

   a. x : Number
   b. y : Number
   c. width : Number
   d. height : Number

Sedangkan *public properties* yang ada pada kelas *FaceDetectionVars* ini adalah sebagai berikut :

1. baseScale : Number = 4.0 – yaitu ukuran minimal wajah untuk dideteksi. Misal dengan nilai baseScale : 2.0 mendeteksi wajah yang lebih kecil, tapi dapat menghasilkan kesalahan dikarenakan *noise.*
2. faceDetectionROI : Rectangle – [read-only], properti ini untuk *region of interest* pendeteksian wajah. Nilai default-nya adalah (50, 50, 540, 380).

3. leftEyeDetectionROI : Rectangle – [read-only], properti untuk *region of interest* mata kiri. Nilai default-nya adalah (200, 100, 240, 280).
4. rightEyeDetectionROI : Rectangle – [read-only], properti untuk *region of interest* mata kanan. Nilai default-nya adalah (200, 100, 240, 280).
5. maxScale : Number = 5.0 – yaitu properti untuk ukuran maksimal wajah untuk dideteksi.
6. minRectsToFind : int = 12 – yaitu properti untuk jumlah persegi yang harus dideteksi untuk memulai estimasi wajah. Nilai default-nya adalah 12. Semakin banyak persegi yang terdeteksi semakin menjadi lebih baik hasil inisial persegi wajah.
7. scaleIncrement : Number = 0.5 – properti ukuran langkah yang ditambahkan ke dalam baseScale tiap iterasi. Nilai default-nya adalah 0.5.

3.1.3.6 *Class FaceEstimationVars*

Pada kelas *FaceEstimationVars* ini terdapat variabel-variabel untuk estimasi wajah (*face estimation*). Method yang terdapat pada kelas ini cuma satu yaitu :

1. updateShapeCenterBounds() – method yang meng-update *shape center bounds.* BRF bekerja dengan 640x480 BitmapData, ROI-nya tidak boleh melebihi dimensi tersebut.

   Parameter yang dibutuhkan yaitu :

a. x : Number
b. y : Number
c. width : Number
d. height : Number

Sedangkan *public properties* yang ada pada kelas *FaceEstimationVars* ini adalah sebagai berikut :

1. maxShapeDimension : int = 400 – ketika sebuah bentuk (*shape*) wajah di-estimasi-kan ukurannya diperiksa menjadi lebih kecil dari *maxShapeDimension,* jika tidak *tracking* diatur ulang. Nilai default-nya adalah 400px.
2. minShapeDimension : int = 100 – ketika sebuah bentuk (*shape*) wajah di-estimasi-kan ukurannya diperiksa menjadi lebih besar dari *minShapeDimension,* jika tidak *tracking* diatur ulang. Nilai default-nya adalah 100px.
3. shapeCenterBounds : Rectangle – [read-only], properti untuk mengembalikan nilai *shape center bounds*. Ketika bentuk (*shape*) wajah di-estimasi-kan, wajah harus terletak di dalam batas-batas (*bounds*) agar menjadi valid. Nilai default-nya adalah (50, 50, 540, 380).
4. trackingAccuracy : int – properti untuk mengembalikan nilai akurasi *tracking* saat ini. Nilai default-nya adalah 0, bisa salah satu yaitu : 0 (paling akurat), 1 (akurat, tapi lebih cepat daripada 0), 2 (kurang

akurat, tapi lebih cepat daripada 1), atau 3 (paling kurang akurat, tapi paling cepat).

3.1.3.7 *Class PoseEstimationVars*

Pada kelas *PoseEstimationVars* ini terdapat variabel-variabel untuk pose estimasi (*pose estimation*). Tidak ada method dalam kelas ini. Properti-properti yang ada pada kelas *PoseEstimationVars* ini antara lain :

1. contentContainer : IBRFContentContainer - properti ini mengembalikan *content container*.
2. focalLength : Number – properti ini mengembalikan *focal length* yang ada saat itu.
3. pupilDistance : Number – properti ini mengembalikan *pupil distance* yang ada saat itu.
4. resultMatrix : ResultMatrix – [read-only], properti ini mengembalikan *ResultMatrix* yang sama dengan *IBRFContent* dapatkan dalam *updateMatrix*.

3.1.3.8 *Class ResultMatrix*

Pada kelas *ResultMatrix* ini dihitung matriks posisi 3D. Perhitungan menggunakan nilai *r* sebagai nilai rotasi dan nilai *t* sebagai nilai translasi. Method-method yang terdapat dalam kelas ini adalah :

1. ResultMatrix() – merupakan constructor.

2. reset() – method untuk mengatur ulang semua nilai rotasi *r* dan nilai translasi *t*.
3. setResetValue() – method yang mengatur nilai reset dari rotasi *r* dan translasi *t*.

   Parameter yang dibutuhkan yaitu :

   a. rawData : Vector.<Number>
4. toString() – method untuk menampilkan ResultMatrix dalam bentuk *string*.

   Output yang dihasilkan adalah : String.
5. update() – method yang meng-update nilai-nilai rotasi *r* dan translasi *t* yang telah diatur oleh setResetValue().

   Parameter yang dibutuhkan adalah :

   a. rawData : Vector.<Number>

Sedangkan *public properties* yang ada pada kelas *ResultMatrix* ini adalah sebagai berikut :

1. r00 : Number = 0
2. r00Reset : Number = 0
3. r01 : Number = 0
4. r01Reset : Number = 0
5. r02 : Number = 0
6. r02Reset : Number = 0
7. r10 : Number = 0
8. r10Reset : Number = 0

9. r11 : Number = 0
10. r11Reset : Number = 0
11. r12 : Number = 0
12. r12Reset : Number = 0
13. r20 : Number = 0
14. r20Reset : Number = 0
15. r21 : Number = 0
16. r21Reset : Number = 0
17. r22 : Number = 0
18. r22Reset : Number = 0
19. tx : Number = 0
20. txReset : Number = 0
21. ty : Number = 0
22. tyReset : Number = 0
23. tz : Number = 0
24. tzReset : Number = 0

3.1.3.9 *Class Shape*

Pada kelas *Shape* ini menyimpan informasi *tracking* dari algoritma yang terdiri dari beberapa vector.<point> yang dapat digunakan untuk membaca kembali hasil *tracking.* Agar algoritma dapat men-*track* bentuk (*shape*) wajah dengan tepat, maka jangan mengubah nilai poin dari vector.<point> dan juga jangan menggunakan fungsi-fungsi yang ada dalam kelas ini.

Public properties yang terdapat dalam kelas ini yaitu :

1. Nelems : int – [read-only], properti untuk mengembalikan jumlah dari semua poin.
2. pointLeftEyeCenter : Point – properti untuk titik pusat mata kiri.
3. pointNoseBase : Point – properti untuk titik pusat hidung bagian bawah.
4. pointNoseTip : Point – properti untuk ujung hidung.
5. pointRightEyeCenter : Point – properti untuk titik pusat mata kanan.
6. pointsLeftBrow : Vector.<Point> - properti untuk titik-titik alis mata kiri.
7. pointsLeftEye : Vector.<Point> - properti untuk titik-titik mata kiri.
8. pointsLowerLip : Vector.<Point> - properti untuk titik-titik bibir bawah.
9. pointsRightBrow : Vector.<Point> - properti untuk titik-titik alis mata kanan.
10. pointsRightEye : Vector.<Point> - properti untuk titik-titik mata kanan.
11. pointsUpperLip : Vector.<Point> - properti untuk titik-titik bibir atas.
12. shapePoints : Vector.<Point> - properti untuk semua titik *shape* dalam satu vektor.

3.1.4 Analisis Kebutuhan Non Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem, dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

Pada analisis kebutuhan sistem non-fungsional ini dijelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis pengguna.

3.1.4.1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun aplikasi AR *virtual mirror eyeglasses* ini adalah sebagai berikut:

1. *Microsoft Windows 7*
2. Library *Beyond Reality Face*
3. Flash IDE menggunakan Adobe Flash Professional CS5.5
4. Compiler menggunakan Adobe Flex SDK
5. 3DS Max sebagai 3D model editor

3.1.4.2 Analisis Kebutuhan Perangkat Keras

Kebanyakan untuk aplikasi yang berbasis *Augmented Reality* pasti membutuhkan grafis video yang relatif tinggi. Hal ini dikarenakan model-model grafis yang digunakan menggunakan teknik *rendering.*

Berdasarkan studi literatur terhadap proyek-proyek yang mengembangkan teknologi *augmented reality,* maka diperoleh spesifikasi minimum perangkat keras untuk teknologi *augmented reality* seperti berikut ini:

1. Processor minimal 1,8 GHz atau lebih.
2. *Random Access Memory* (RAM) 1 GB.
3. VGA card minimal 128 MB
4. Webcam 1,3 Megapixel
5. Koneksi internet minimal 153 Kbps

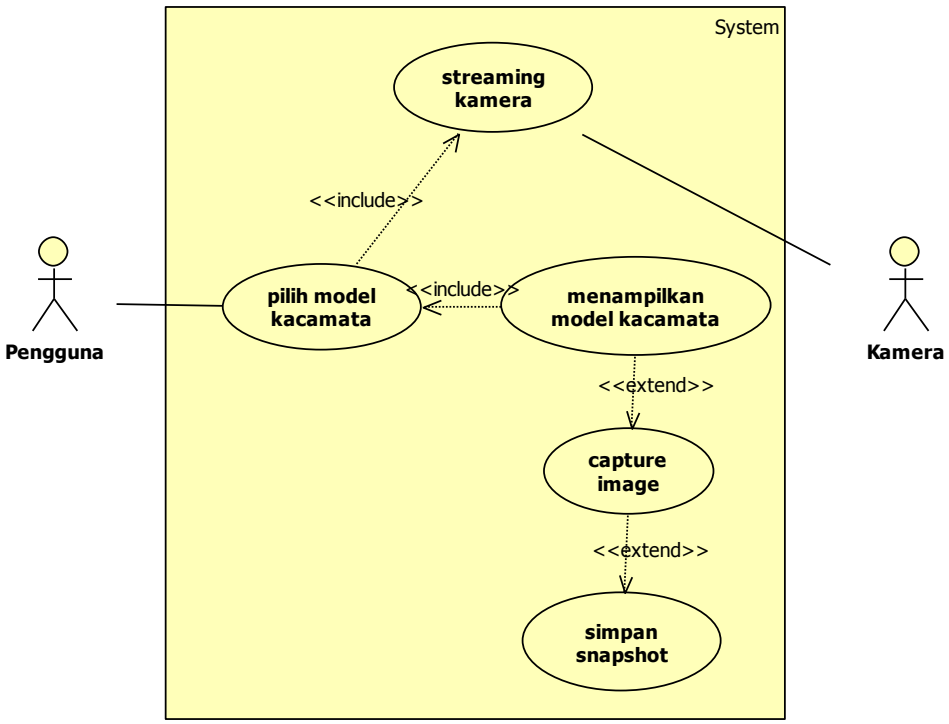
3.1.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional ini dimodelkan dengan menggunakan *UML (Unified Modeling Language).* Dimana tahap-tahap perancangan yang dilakukan dalam membangun aplikasi *virtual mirror eyeglasses* antara lain *Use Case Diagram, Class Diagram, Sequence Diagram,* dan *Activity Diagram.*

3.1.5.1 *Use Case Diagram*

Diagram *Use Case* merupakan pemodelan untuk kelakuan (*behaviour*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat [8].

Berikut ini adalah perancangan proses-proses yang terdapat pada aplikasi AR *virtual mirror eyeglasses*, yang digambarkan dengan *Use Case Diagram* yang dapat dilihat berikut ini:



Gambar 3.11 *Use Case Diagram* aplikasi *Virtual Mirror Eyeglasses*

1. **Definisi *Actor***

   Definisi *Actor* berfungsi untuk menjelaskan *Actor* yang terdapat pada *Use Case Diagram*. Definisi *Actor* diterangkan pada Tabel 3.1.

Tabel 3.2 Definisi *Actor*.

No	*Actor*	Deskripsi
1.	Pengguna	Orang yang menggunakan aplikasi
2.	Kamera	Alat input yang digunakan aplikasi

2. Definisi *Use Case*

Definisi *Use Case* berfungsi untuk menjelaskan fungsi *Use Case* yang terdapat pada *Use Case* Diagram. Definisi Use Case diterangkan pada Tabel 3.2.

Tabel 3.3 Definisi *Use Case.*

No	*Use Case*	Deskripsi
1.	streaming kamera	Proses untuk memberi izin akses kamera
2.	Menampilkan model kacamata	Proses menampilkan model kacamata
3.	Pilih model kacamata	Proses untuk memilih model kacamata yang diinginkan
4.	Capture image	Proses untuk meng-*capture* objek yang ditangkap kamera
5.	Simpan snapshot	Proses melakukan penyimpanan terhadap hasil tangkapan/*capture* kamera

3. Skenario *Use Case*

Skenario *Use Case* menggambarkan alur penggunaan sistem dimana setiap skenario digambarkan dari sudut pandang aktor, seseorang, atau piranti yang berinteraksi dengan perangkat lunak dalam berbagai cara.

3.1 Skenario *Use Case* Streaming Kamera

Use case streaming kamera ini menggambarkan keterlibatan kamera yang berperan untuk menangkap *image* yang berada di depan kamera untuk selanjutnya diproses oleh sistem.

Tabel 3.4 Skenario *Use Case* Streaming Kamera

Identifikasi		
Nomor	1	
Nama	Streaming kamera	
Tujuan	Mengaktifkan kamera	
Deskripsi	Proses untuk mengaktfkan kamera	
Aktor	Kamera	
Skenario Utama		
Kondisi awal	Aplikasi telah berjalan	
Aksi Aktor		Reaksi Sistem
1. Kamera mengirim permintaan koneksi		
		2. Sistem mengirim respon jawaban *allow* dari user
3.Kamera mengirim gambar yang ditangkap ke sistem		
		4.Sistem mengolah gambar dari kamera
		5.Sistem menampilkan gambar yang telah diolah ke layar.
		6.Sistem melakukan pendeteksian wajah
Kondisi akhir	Gambar user dan posisi wajah ditampilkan ke layar	
Skenario Alternatif		
Aksi Aktor		Reaksi Sistem
1.Kamera mengirim permintaan koneksi		
		2.Sistem mengirim respon jawaban *deny* dari user
		3.Kamera mati
Kondisi Akhir	Sistem tidak dapat menampilkan apapun ke layar	

3.2 Skenario *Use Case* Pilih Model Kacamata

Use case pilih model kacamata ini menggambarkan proses dimana pengguna memilih model kacamata yang diinginkan.

Tabel 3.5 Skenario *Use Case* Pilih Model Kacamata

Identifikasi		
Nomor	2	
Nama	Pilih model kacamata	
Tujuan	Memilih model kacamata yang disediakan	
Deskripsi	Proses untuk memilih model kacamata	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna berada di awal antarmuka aplikasi	
Aksi Aktor		Reaksi Sistem
1.Memilih salah satu model kacamata		
		2.Sistem mencari alamat file model yang dipilih
		3.Sistem me-*load* model
Kondisi akhir	Sistem berhasil me-*load* model ke layar	

3.3 Skenario *Use Case* Menampilkan Model Kacamata

Use case menampilkan model kacamata ini menggambarkan proses sistem menampilkan model kacamata setelah dipilih oleh pengguna.

Tabel 3.6 Skenario *Use Case* Menampilkan Model Kacamata

Identifikasi	
Nomor	3
Nama	Menampilkan model kacamata
Tujuan	Menampilkan model kacamata pada layar
Deskripsi	Proses untuk menampilkan model kacamata pada layar
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Model kacamata yang dipilih telah berhasil di-*load*

Aksi Aktor	Reaksi Sistem
1.Posisi wajah pengguna didepan kamera	
	2.Sistem menentukan koordinat kacamata sesuai posisi wajah
3.Pengguna melakukan pergerakan	
	4.Sistem memindahkan kacamata sesuai posisi wajah yang terus di-*update*
Kondisi akhir	Sistem menampilkan model kacamata virtual yang telah dipilih pada wajah pengguna

3.4 Skenario *Use Case* Capture Image

Use case capture image ini menggambarkan proses sistem meng-*capture* atau menangkap *image* hasil streaming kamera.

Tabel 3.7 Skenario *Use Case* Capture Image

Identifikasi	
Nomor	4
Nama	Capture image
Tujuan	Menangkap/mengambil gambar hasil streaming kamera
Deskripsi	Proses untuk menangkap/mengambil gambar dari kamera
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Sistem dan kamera telah berjalan
Aksi Aktor	Reaksi Sistem
1. Menekan tombol capture kamera	
	2.Sistem men-*stop* video *drawing*
	2.Sistem mengambil gambar hasil *drawing* kamera dan menampilkannya di layar
	3.Sistem menampung hasil *capture* sementara
Kondisi akhir	Sistem mengambil dan menampilkan gambar hasil streaming kamera pada layar

3.5 Skenario *Use Case* Simpan Snapshot

Use case simpan snapshot ini menggambarkan proses penyimpanan gambar hasil *capture* dari kamera streaming ke direktori pengguna.

Tabel 3.8 Skenario *Use Case* Simpan Snapshot

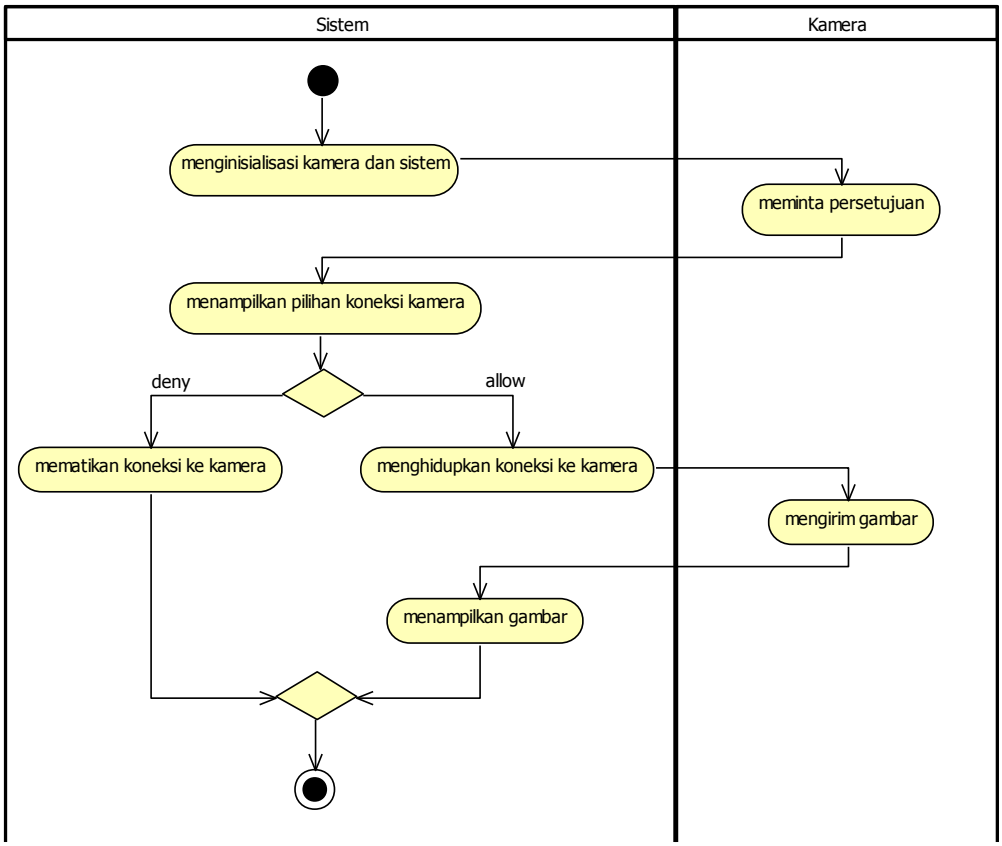
Identifikasi		
Nomor	5	
Nama	Simpan snapshot	
Tujuan	Menyimpan hasil capture kamera berupa snapshot	
Deskripsi	Proses untuk menyimpan gambar hasil capture kamera berupa snapshot	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Sistem telah mengambil dan menampilkan gambar hasil streaming kamera pada layar	
Aksi Aktor		Reaksi Sistem
1.Menekan tombol simpan snapshot		
		2.Sistem mengubah/*encode* gambar yang telah ditangkap dalam bentuk ByteArray
		3.Sistem menyimpan file gambar dengan menggunakan method FileReference.save()
4.Pengguna memilih direktori tempat penyimpanan gambar		
Kondisi akhir	Sistem menyimpan hasil capture kamera berupa snapshot di direktori pengguna	

3.1.5.2 *Activity Diagram*

Diagram aktivitas atau *Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis [8].

1. *Activity Diagram* Streaming Kamera

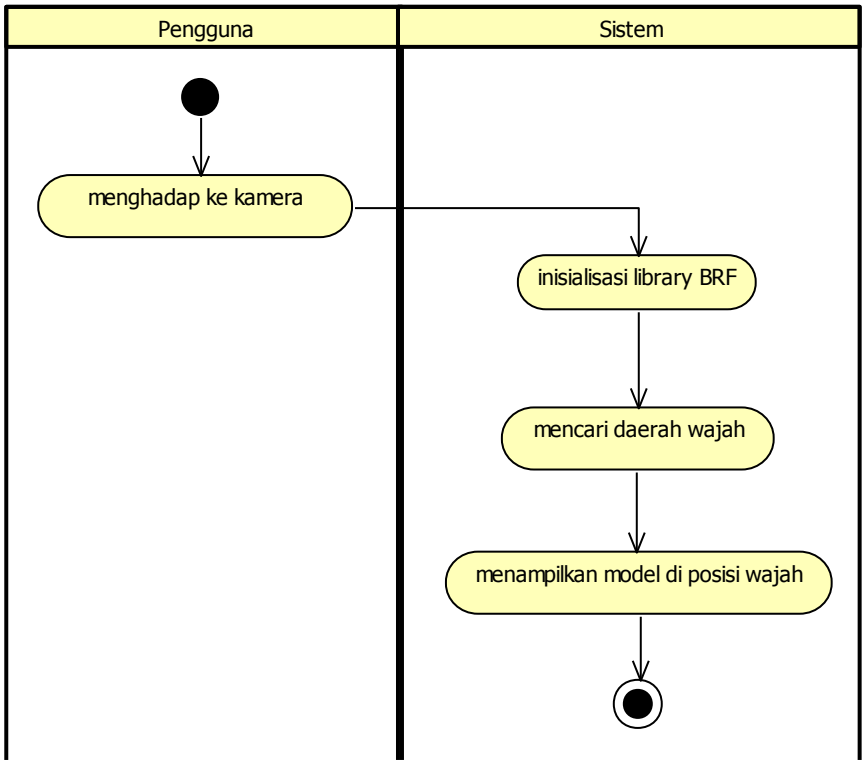
Proses streaming kamera memiliki alur proses seperti digambarkan sebagai berikut :



Gambar 3.12 *Activity Diagram* Streaming Kamera

2. *Activity Diagram* Menampilkan Model Kacamata

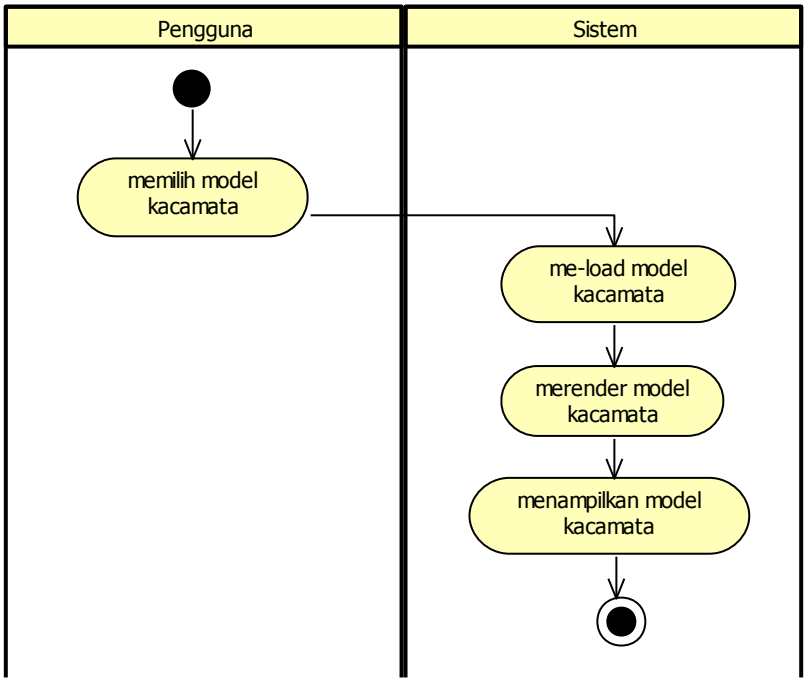
Proses menampilkan model kacamata memiliki alur proses seperti digambarkan sebagai berikut :



Gambar 3.13 *Activity Diagram* Menampilkan Model Kacamata

3. *Activity Diagram* Pilih Model Kacamata

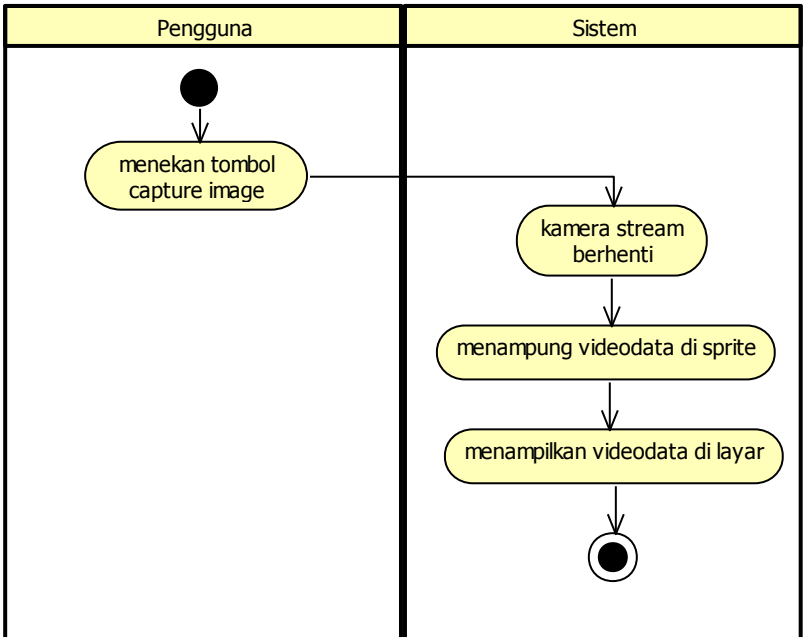
Proses pilih model kacamata memiliki alur proses seperti digambarkan sebagai berikut :



Gambar 3.14 *Activity Diagram* Pilih Model Kacamata

4. *Activity Diagram* Capture Image

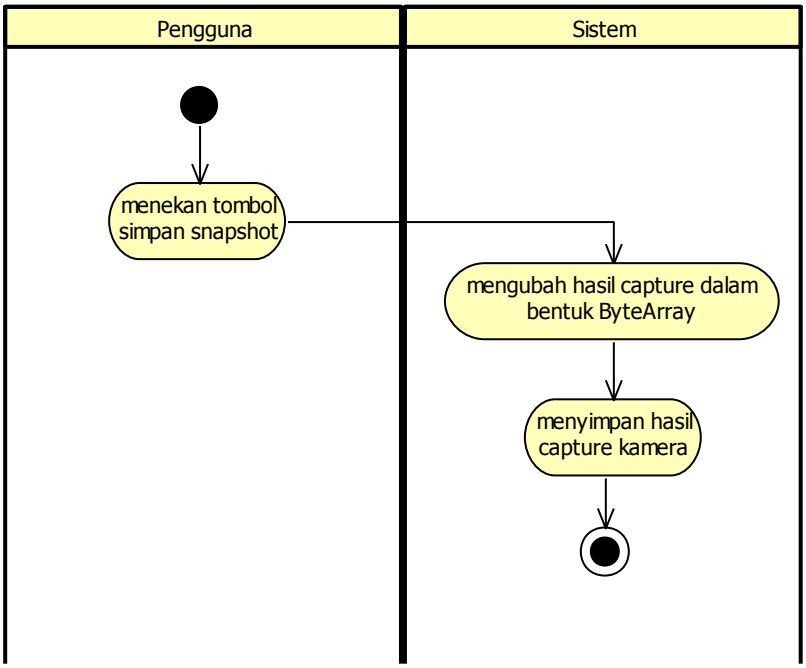
Proses capture image memiliki alur proses seperti digambarkan sebagai berikut :



Gambar 3.15 *Activity Diagram* Capture Image

5. *Activity Diagram* Simpan Snapshot

Proses simpan snapshot memiliki alur proses seperti digambarkan sebagai berikut :

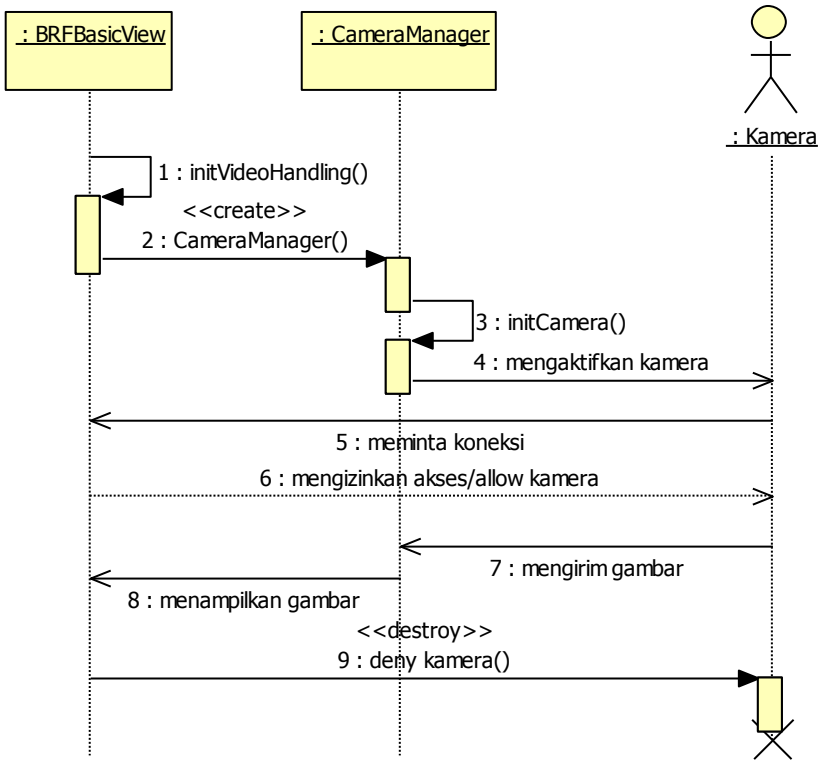


Gambar 3.16 *Activity Diagram* Simpan Snapshot

3.1.5.3 *Sequence Diagram*

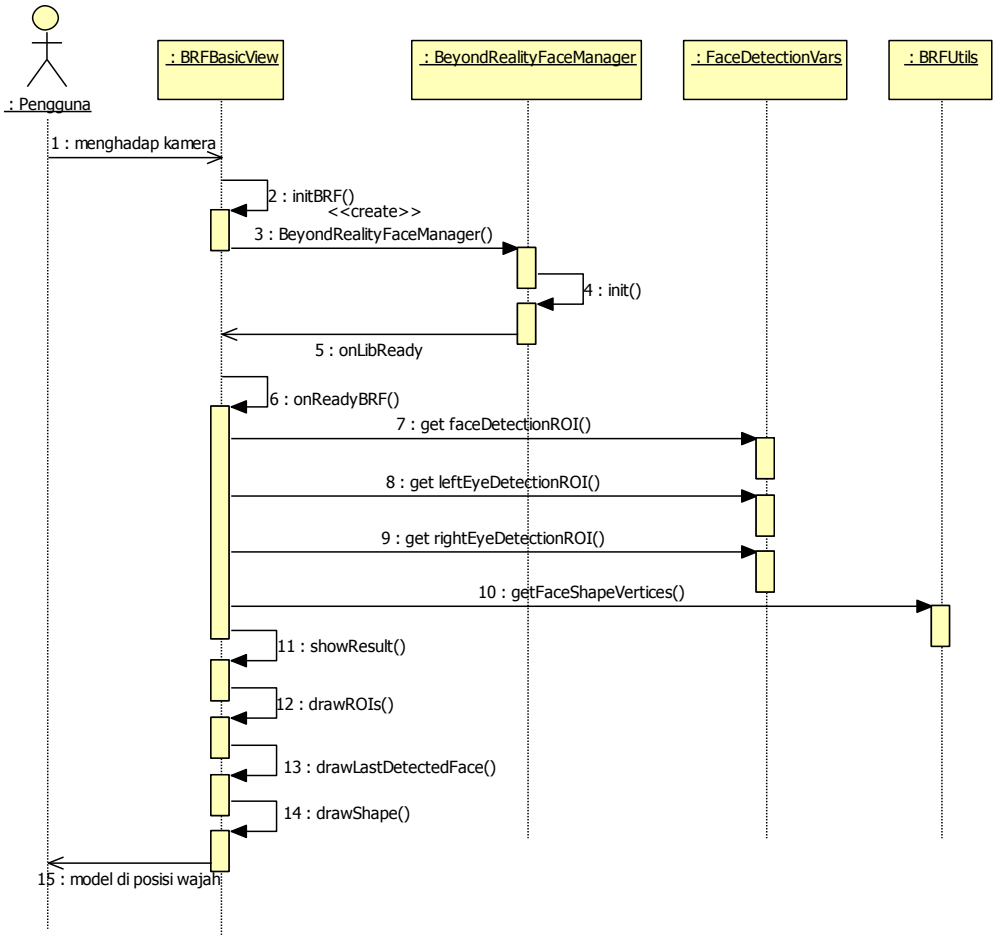
Diagram sequence atau *Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek [8].

1. *Sequence Diagram* Streaming Kamera



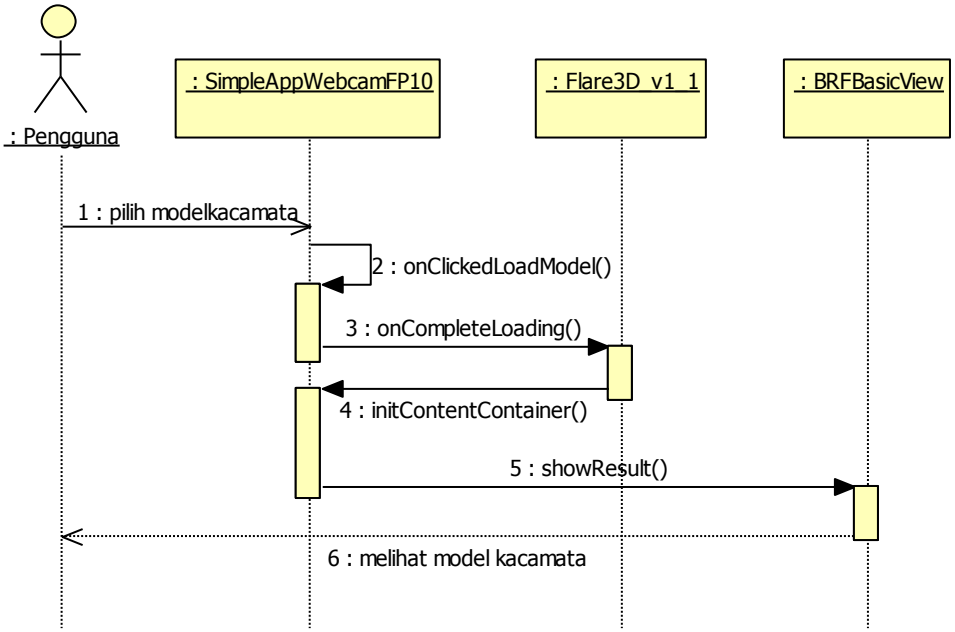
Gambar 3.17 *Sequence Diagram* Streaming Kamera

2. *Sequence Diagram* Menampilkan Model Kacamata



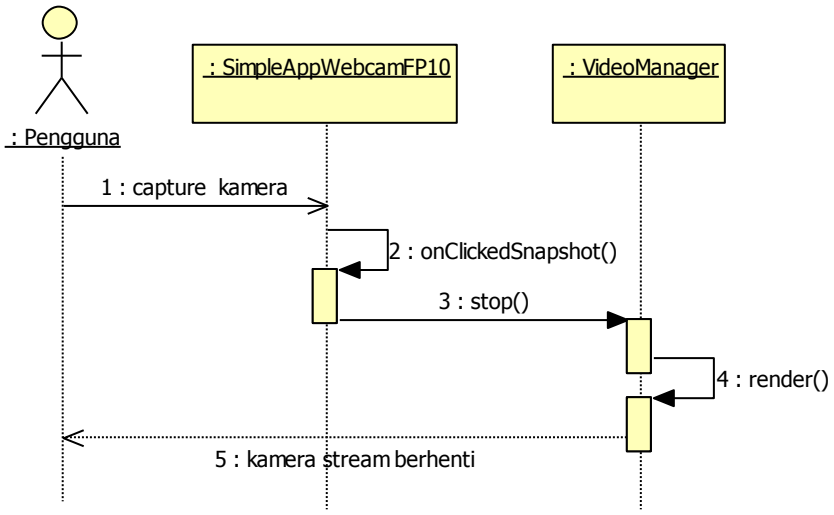
Gambar 3.18 *Sequence Diagram* Menampilkan Model Kacamata

3. *Sequence Diagram* Pilih Model Kacamata



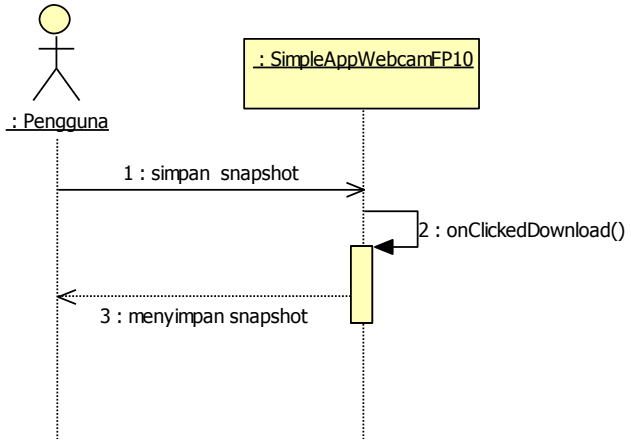
Gambar 3.19 *Sequence Diagram* Pilih Model Kacamata

4. *Sequence Diagram* Capture Image



Gambar 3.20 *Sequence Diagram* Capture Image

5. *Sequence Diagram* Simpan Snapshot

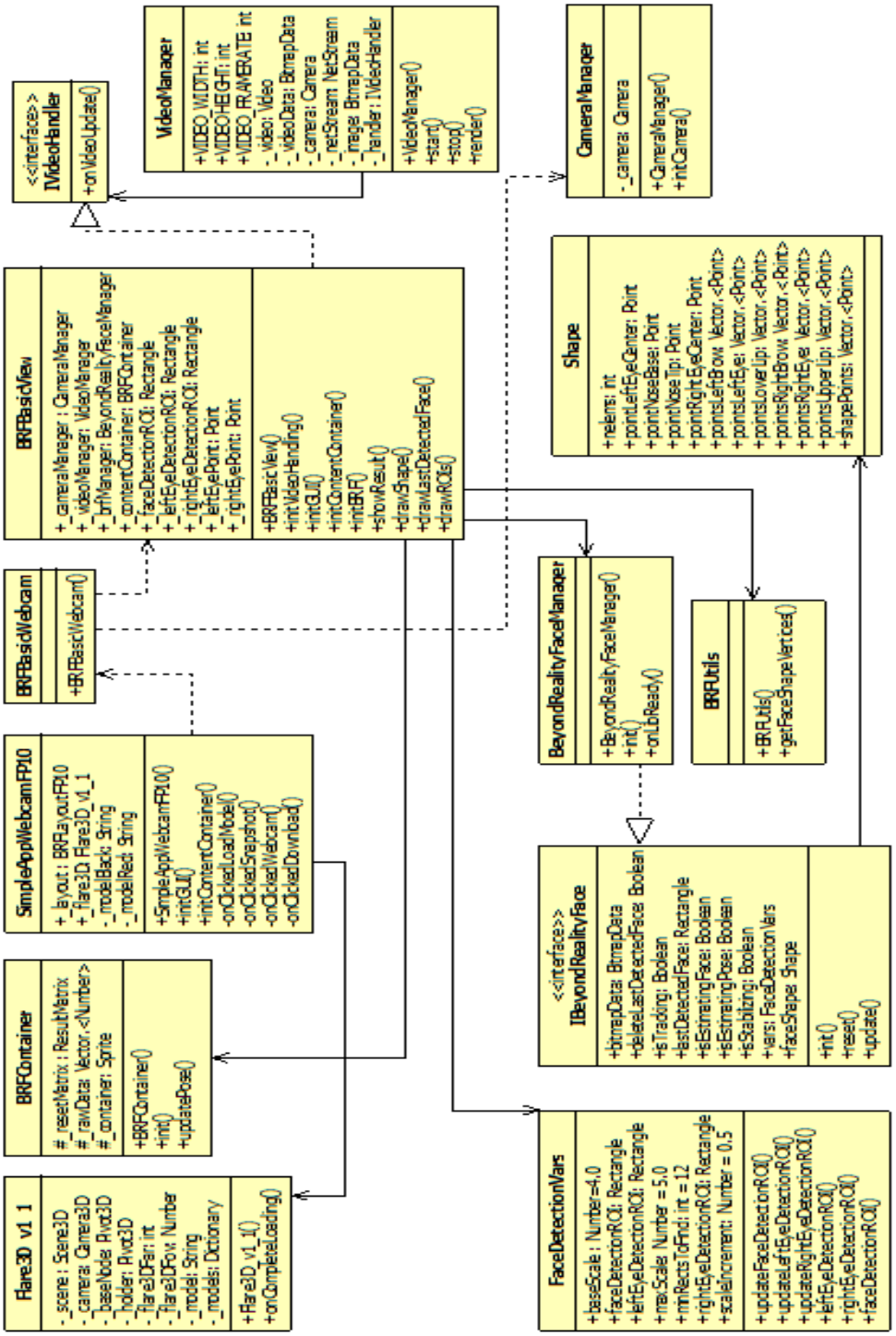


Gambar 3.21 *Sequence Diagram* Simpan Snapshot

3.1.5.4 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem [8].

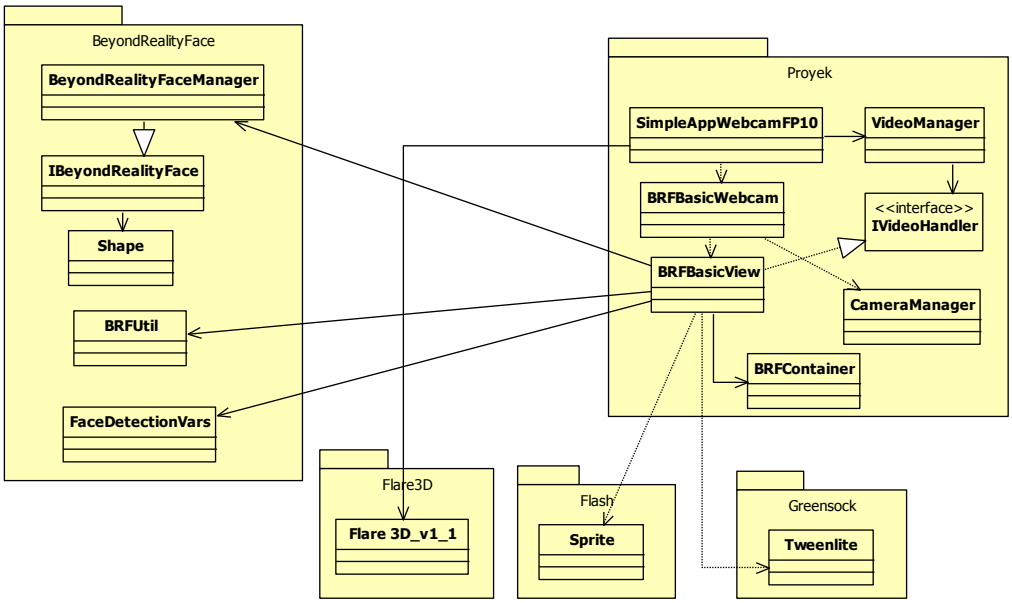
Berikut ini adalah perancangan struktur sistem yang terdapat pada aplikasi AR *virtual mirror eyeglasses,* yang digambarkan dengan *Class Diagram* yang dapat dilihat pada halaman berikutnya.



Gambar 3.22 *Class Diagram* Aplikasi Virtual Mirror Eyeglasses

3.1.5.5 *Package Diagram*

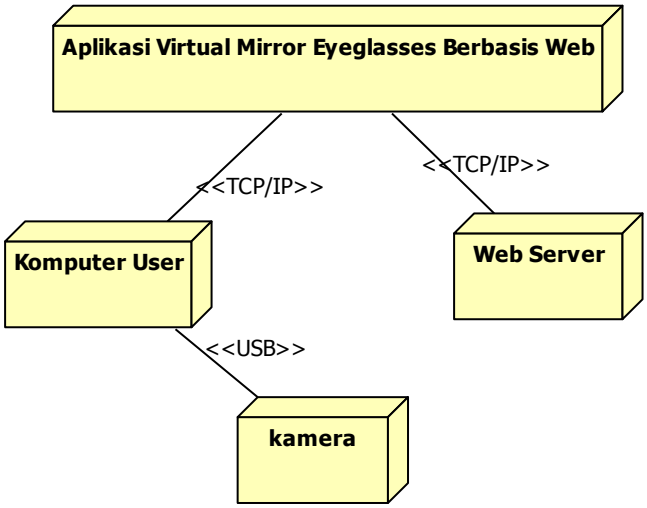
Package Diagram adalah diagram yang menggambarkan cara mengelola class-class yang dibuat dan keterkaitannya. Package juga mengelompokkan class-class sesuai dengan kategori tertentu, berikut ini gambar diagram package :



Gambar 3.23 *Package Diagram*

3.1.5.6 *Deployment Diagram*

Deployment diagram menggambarkan bagaimana software dan hardware bekerja sama. Berikut ini gambar deployment diagram :



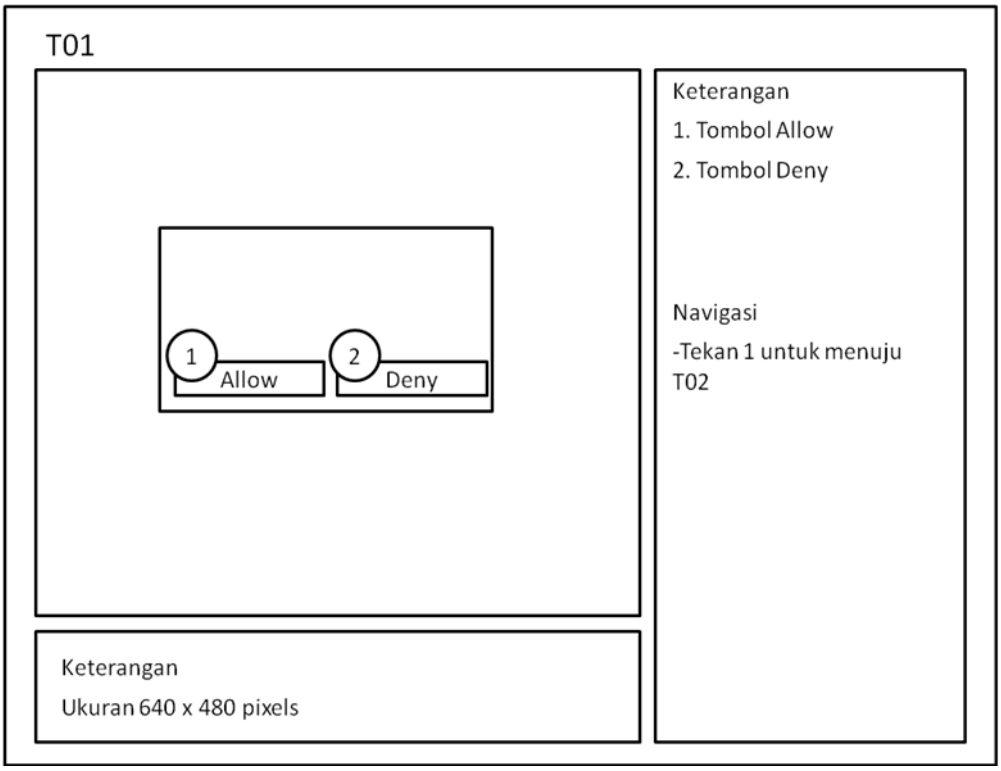
Gambar 3.24 *Deployment Diagram*

3.2. Perancangan Antarmuka

Desain yang dibuat berupa aplikasi berbasis web yang mudah untuk digunakan oleh pengguna, serta informasi yang dihasilkan aplikasi tersebut dapat dimengerti oleh pengguna. Perancangan antarmuka bertujuan untuk memberikan gambaran sistem yang dibuat. Objek tiga dimensi merupakan objek yang akan ditampilkan pada wajah pengguna yang dapat dilihat oleh pengguna dengan bantuan *webcam*.

1. Perancangan Antarmuka Streaming Kamera

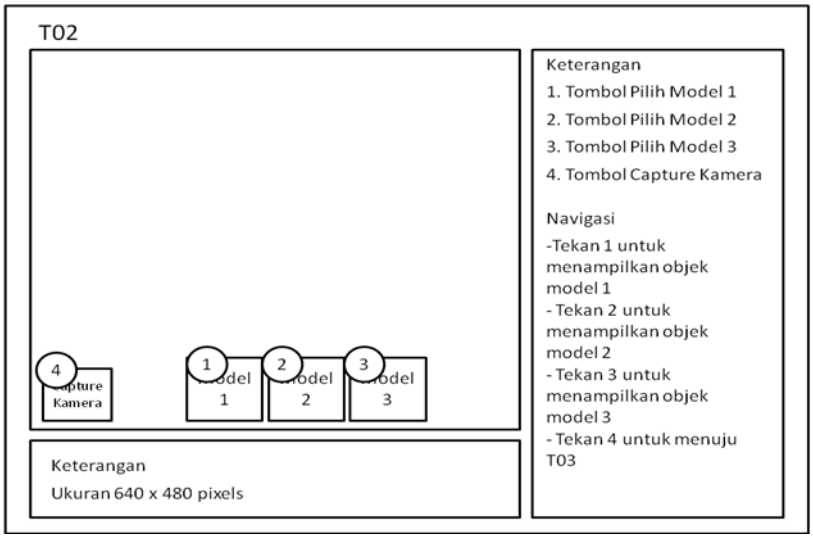
Perancangan antarmuka Streaming Kamera pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.25 Perancangan Antarmuka Streaming Kamera

2. Perancangan Antarmuka Pilih Model Kacamata

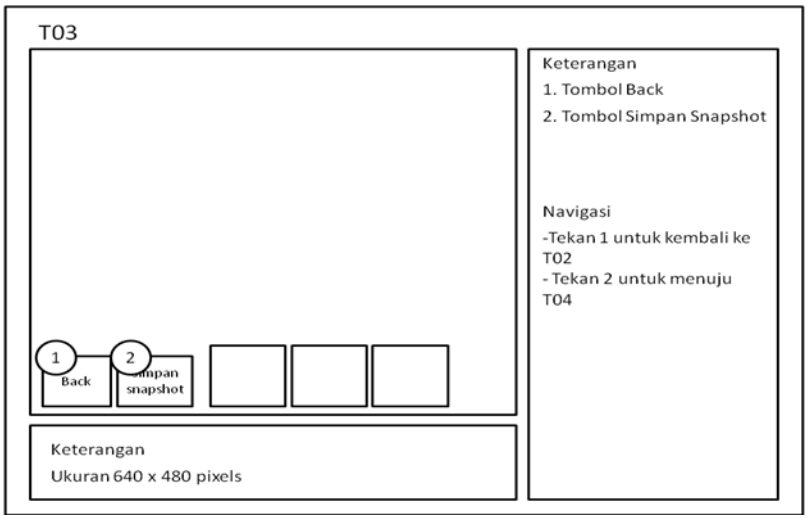
Perancangan antarmuka Pilih Model Kacamata pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.26 Perancangan Antarmuka Pilih Model Kacamata

3. Perancangan Antarmuka Capture Image

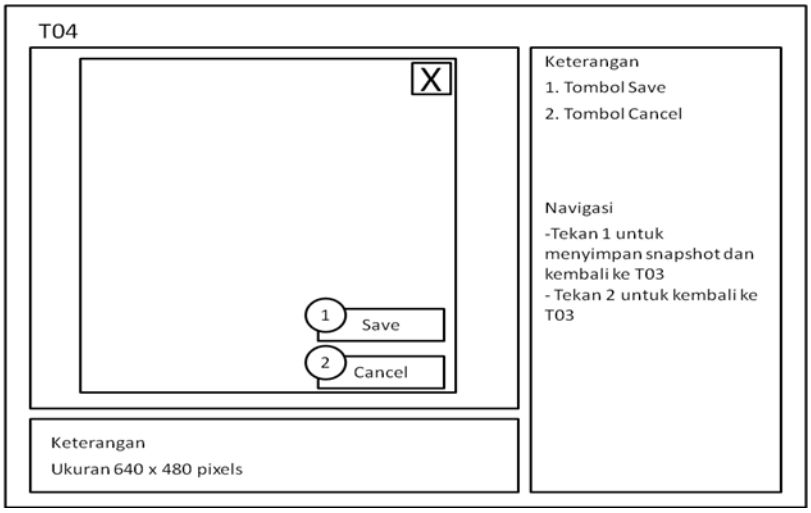
Perancangan antarmuka Capture Image pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.27 Perancangan Antarmuka Capture Image

4. Perancangan Antarmuka Simpan Snapshot

Perancangan antarmuka Simpan Snapshot pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.28 Perancangan Antarmuka Simpan Snapshot