BAB III
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikanya.

Dalam proses pembuatan suatu sistem mutlak dilakukan penelitian dan penganalisaan tentang sistem yang akan dibangun. Berikut adalah beberapa analisis yang dilakukan untuk membangun aplikasi AR Word Translator dengan implementasi Augmented Reality dengan OCR untuk penerjemah pada perangkat Android.

3.2 Analisis Masalah

Teks tertulis merupakan salah satu metode yang paling umum untuk menyampaikan informasi. Namun, ketika melakukan traveling atau berwisata ke luar negeri informasi tersebut ditemui dalam bahasa asing. Bagi beberapa individu perbedaan bahasa membuat informasi tersebut tidak dapat tersampaikan dengan baik.

Pada umumnya kamus saku digunakan untuk menerjemahkan teks berbahasa asing. Akan tetapi, penggunaan kamus dirasa kurang praktis untuk digunakan saat berwisata. Penggunaan kamus digital juga menjadi pilihan wisatawan untuk menerjemahkan teks dengan pencarian kata yang cepat dan mudah. Namun kamus digital masih memiliki kekurangan, yaitu wisatawan harus mengetikan teks yang ingin diterjemahkan. Maka dari itu dibutuhkan sebuah sistem yang mampu mendeteksi teks dan menerjemahkannya secara langsung dengan tingkat mobilitas yang tinggi. Salah satu teknologi yang dapat dimanfaatkan adalah teknologi *Augmented Reality* dengan OCR (*Optical Character Recognition*) pada perangkat mobile bersistem operasi Android.

Augmented Reality (AR) memungkinkan objek 3D *virtual* untuk diintegrasikan kedalam lingkungan nyata secara *real-time. Augmented Reality* menggunakan teknologi *computer vision, image processing* dan teknik *computer graphics* untuk menggabungkan konten digital ke dunia nyata. Hal ini memungkinkan interaksi *realtime* antara pengguna, benda nyata dan benda-benda virtual. *Optical Character Recognition* (OCR) adalah proses mengubah teks pada citra digital menjadi teks yang dapat diedit. Hal ini memungkinkan mesin atau komputer dapat mengenali karakter melalui mekanisme optik.

Oleh karena itu, penelitian kali ini mencoba untuk mengimplementasikan *Augmented Reality* dan OCR pada aplikasi penerjemah kata berbasis Android. Penelitian dilakukan untuk mengetahui bagaimana cara memanfaatkan *augmented reality* sebagai penerjemah *realtime* dengan *teks recognition* menggunakan OCR untuk mengoptimalkan proses penerjemahan teks.

3.3 Analisis Sistem Berjalan

Analisis sistem berjalan merupakan proses penguraian terhadap sistem yang telah ada atau sedang berjalan dengan tujuan untuk melakukan evaluasi yang ada, merumuskan tujuan yang ingin dicapai dengan sistem yang baru, serta menyususun rencana pembangunan sistem.

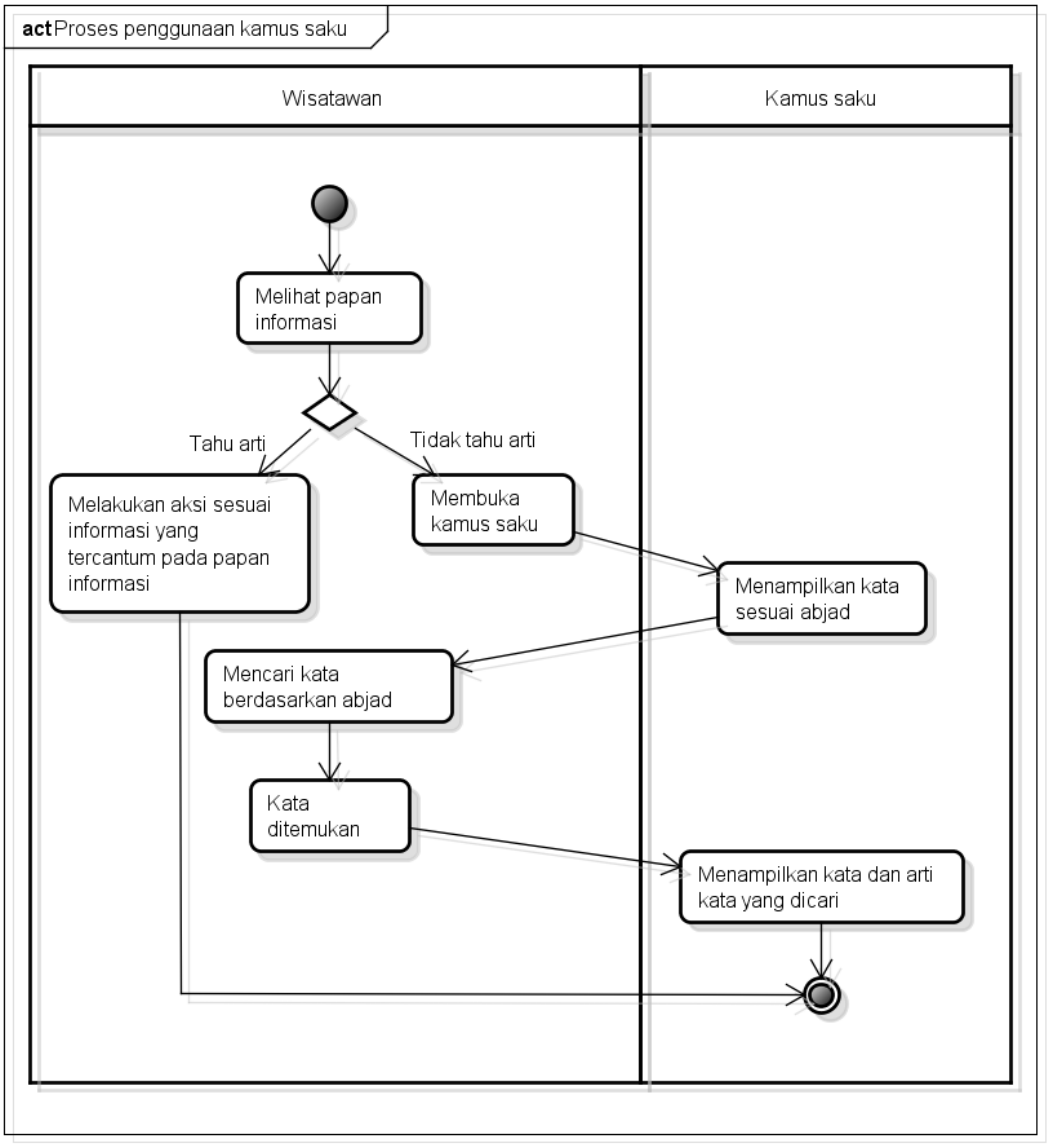
Pada umumnya proses penerjemahan dengan menggunakan kamus saku dilakukan dengan mencari setiap kata satu persatu yang telah tersusun sesuai alphabet. Setelah kata yang ditemukan akan tertera hasil terjemahan pada baris yang sama dengan kata yang dicari. Prosedur ini membutuhkan waktu yang relatif lama karena dalam pencariannya dilakukan secara manual.



Gambar 3.1 Penggunaan Kamus Saku

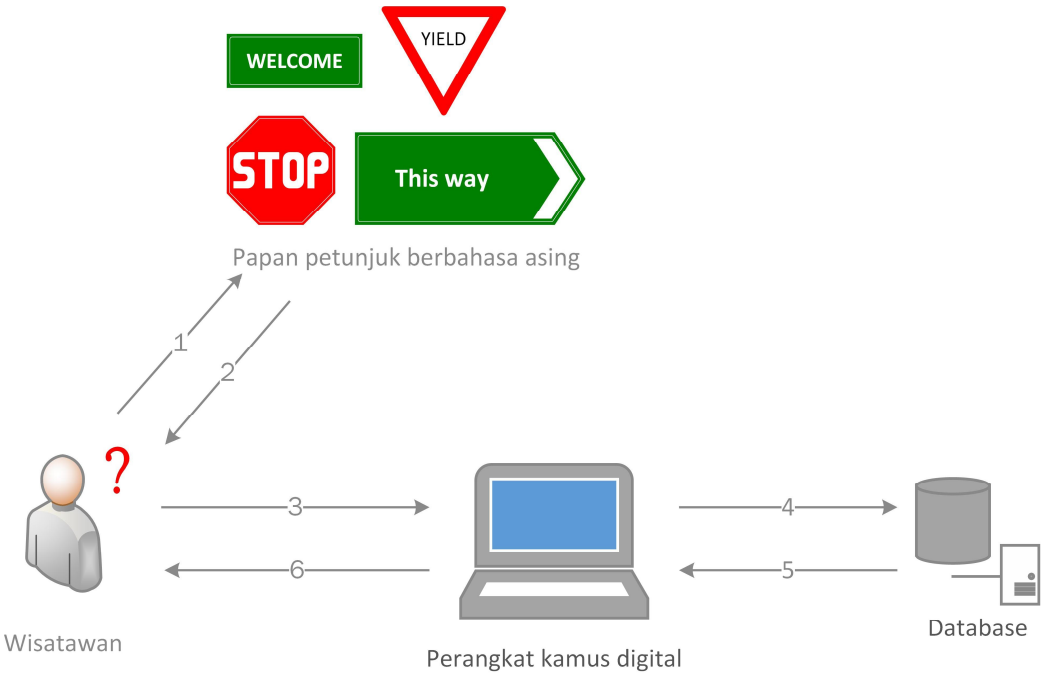
Keterangan :

1) Wisatawan melihat papan informasi yang memiliki teks berbahasa asing.
2) Dari papan informasi wisatawan mendapatkan kata yang tidak dipahami.
3) Wisatawan mencari kata yang tidak dipahami menggunakan kamus saku.
4) Dari kamus saku wisatawan mendapat arti atau penjelasan dari kata yang dicari.



Gambar 3.2 Diagram Aktivitas Proses Penggunaan Kamus Saku

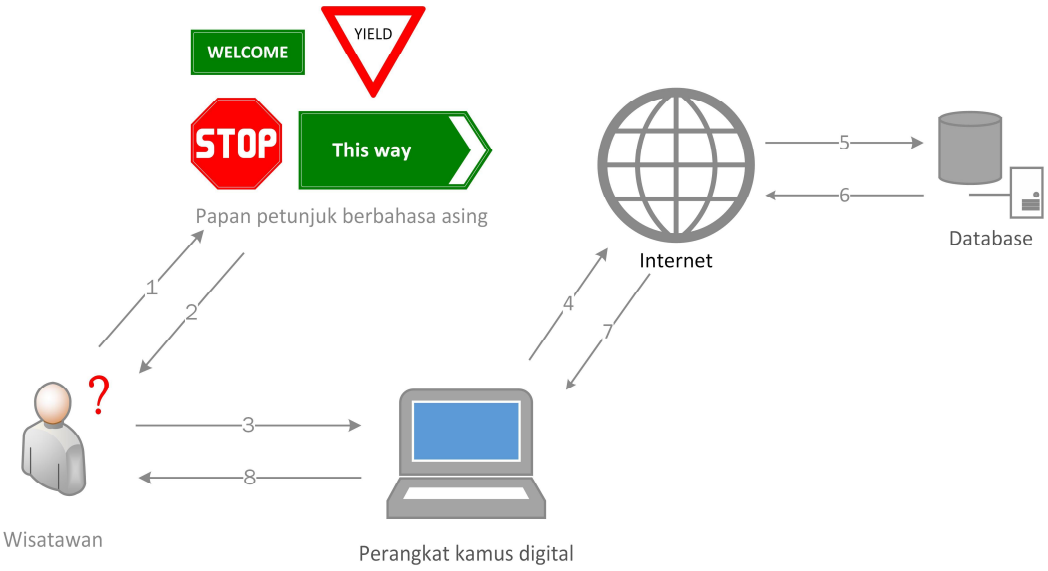
Kamus digital proses penerjemahan dilakukan dengan mengetikan kata yang ingin diterjemahkan, lalu melakukan pencarian dalam database pada perangkat untuk mendapatkan hasil terjemahan kemudian ditampilkan pada layar. Prosedur ini lebih cepat dibandingkan dengan penggunaan kamus saku. Terdapat dua jenis perangkat kamus digital, yaitu kamus digital offline dan kamus digital online.



Gambar 3.3 Penggunaan Perangkat Kamus Digital Offline

Keterangan :

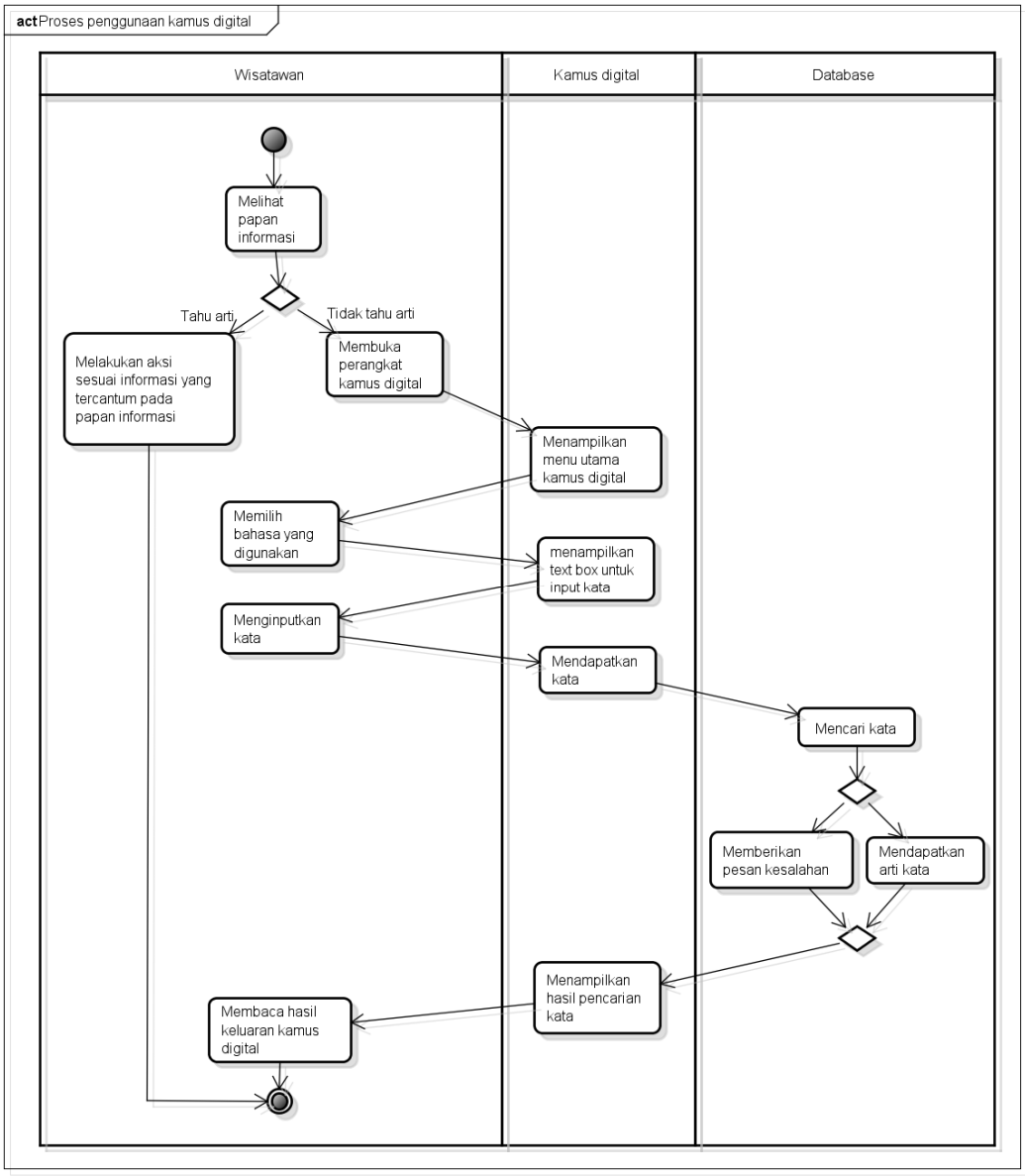
1) Wisatawan melihat papan informasi yang memiliki teks berbahasa asing.
2) Dari papan informasi wisatawan mendapatkan kata yang tidak dipahami.
3) Wisatawan memberi inputan ke perangkat kamus digital dengan mengetikan kata yang ingin diterjemahkan.
4) Perangkat kamus digital melakukan pencarian kata pada database perangkat kamus digital.
5) Database mendapatkan hasil pencarian dan mengembalikannya pada perangkat.
6) Perangkat kamus digital menampilkan hasil penerjemahan kepada wisatawan.



Gambar 3.4 Penggunaan Kamus Digital Online

Keterangan :

1) Wisatawan melihat papan informasi dengan bahasa asing.
2) Dari papan informasi wisatawan mendapatkan kata yang tidak dipahami.
3) Wisatawan memberi inputan ke perangkat kamus digital dengan mengetikan kata yang ingin diterjemahkan.
4) Perangkat kamus digital melakukan akses internet.
5) Dengan akses interternet, perangkat menuju ke database yang berada pada server.
6) Database melakukan pencarian kata dan mengirim kembali hasil terjemahan melalui internet.
7) Dari internet perangkat kamus digital mendapatkan hasil terjemahan.
8) Perangkat kamus digital menampilkan hasil penerjemahan kepada wisatawan.



Gambar 3.5 Diagram Aktivitas Proses Penggunaan Kamus Digital

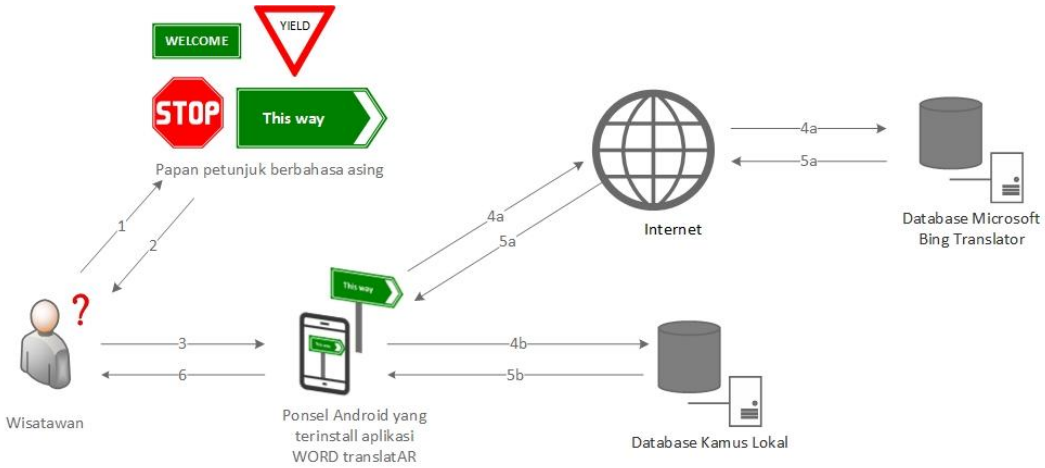
3.4 Analisis Sistem yang Dikembangkan

Analisis sistem yang dikembangkan merupakan penguraian dari suatu sistem yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

3.4.1 Deskripsi Sistem

Aplikasi yang akan dibangun merupakan aplikasi penerjemah bahasa yang dapat dioperasikan pada perangkat mobile menggunakan *platform* Android yang ditujukan untuk wisatawan yang tidak menguasai bahasa asing. Pada aplikasi ini akan diimpelemtasikan teknologi *Augmented Reality* dengan *Optical Character Recognition (OCR)* yang bertujuan untuk mempermudah wisatawan dalam menerjemahkan teks tanpa mengetik terlebih dahulu.

Teknologi *augmented reality* akan digunakan pada saat pendeteksian teks yang akan diterjemahkan dan kemudian menampilkan hasil terjemahan dimana teks sebenarnya akan ditutupi oleh teks virtual hasil *augmented reality*. OCR berfungsi untuk mengkonversikan citra digital berbentuk teks yang telah diperoleh pada proses sebelumnya menjadi teks yang dapat diedit. Mekanisme optik pada OCR mampu mengenali setiap karakter yang ada pada teks yang dipindai untuk dilanjutkan pada proses penerjemahan. Proses penerjemahan akan dilakukan oleh Microsoft Bing Translator, dan hasil terjemahan akan kembali diubah menjadi virtual teks dengan *augmented reality*.



Gambar 3.6 Deskripsi Umum Sistem

Keterangan :

1) Wisatawan melihat papan informasi dengan bahasa asing.
2) Dari papan informasi wisatawan mendapatkan kata yang tidak dipahami.
3) Wisatawan memindai papan informasi menggunakan ponsel Android yang telah terinstall aplikasi Word TranslatAR dan memilih bahasa yang dituju.

4a) Jika bahasa yang dituju tidak tersedia, maka sistem akan melakukan akses ke internet. Kemudian, Microsoft Bing Translator akan melakukan penerjemahan.

4b) Jika bahasa yang dituju tersedia, maka sistem akan langsung melakukan penerjemahan oleh database kamus yang ada pada perangkat

5a) Microsoft Bing Translator mengirimkan kembali hasil terjemahan melalui internet ke ponsel Android.

5b) Database kamus lokal mengirimkan kembali hasil terjemahan ke ponsel Android

6) Ponsel Android menampilkan hasil terjemahan kepada wisatawan secara realtime.

3.4.2 Analisis Kebutuhan Data

Aplikasi yang dibangun akan dibagi menjadi 3 proses besar, yaitu deteksi kata, penerjemahan teks dan menampilkan hasil terjemahan. Pendeteksian dilakukan terhadap objek berupa teks dengan maksimal 24 karakter yang diambil secara *realtime* menggunakan kamera dari perangkat android tanpa melakukan proses penyimpanan terlebih dahulu. Objek berupa teks yang berbentuk citra digital akan berfungsi sebagai input dari sistem, kemudian citra tersebut dikonversi menjadi karakter yang dapat diedit. Setelah semua karakter diperoleh lalu dilakukan proses penerjemahan pada karakter tersebut dan menampilkannya kembali dalam bentuk teks dengan *augmented reality*.



Gambar 3.7 Diagram Blok Sistem Yang Akan Dibangun

3.4.3 Analisis Marker dan Target

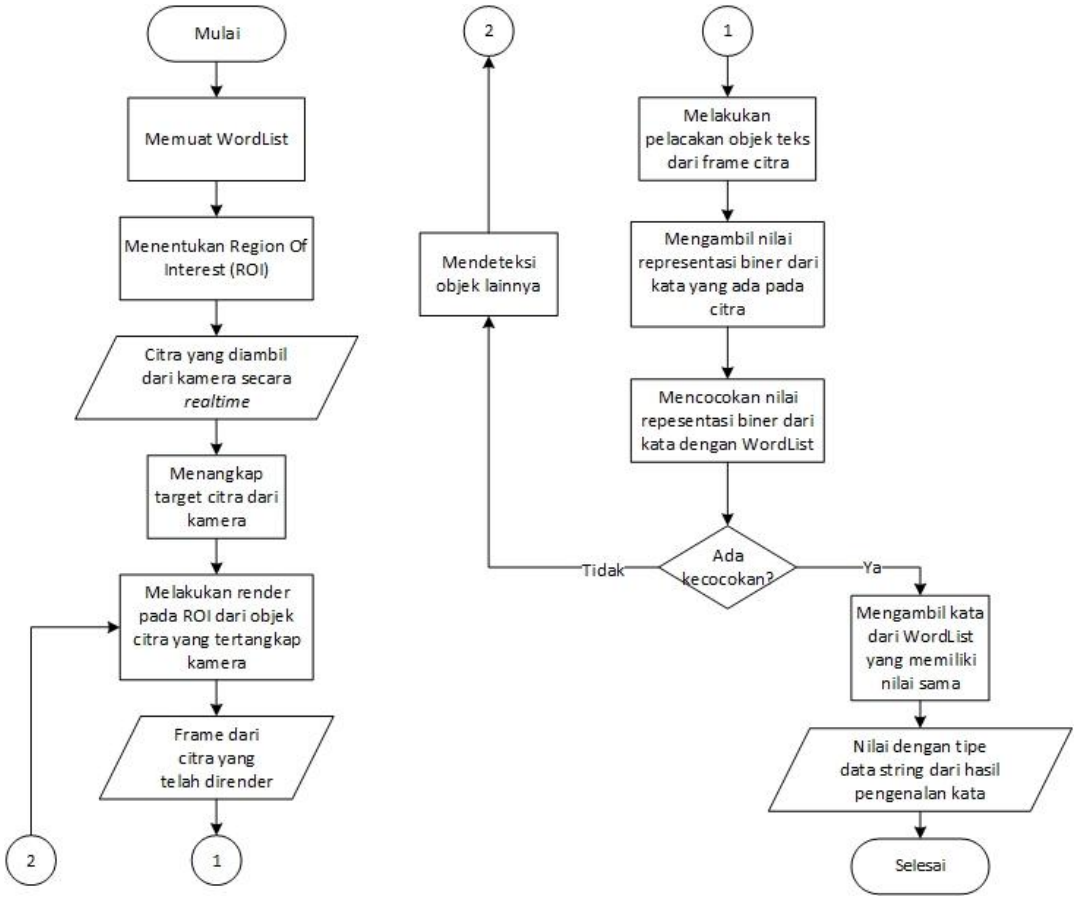
Pada sistem ini, target merupakan kata sehingga pendeteksian target menggunakan teks atau huruf sebagai *marker*-nya. Vuforia SDK dapat mengenali kata-kata dan melacaknya, dengan dua mode pelacakan yang tersedia yaitu "*Words*" dan "*Characters*." Pada penelitian ini metode yang dipakai menggunakan mode "*Words*", kata tersebut akan diakui (dan kemudian dilacak) jika termasuk dalam *wordlist* yang ada pada sistem.

Target kata merepresentasikan elemen tekstual yang dapat dikenali dan dilacak oleh *Vuforia*. Target kata yang valid ditandai dengan fitur berikut:

a. Sebuah kata yang memiliki panjang hingga 24 karakter.
b. Sebuah kata yang berisi hypens (kata majemuk tertutup, misalnya, *"mother-in-law"*).
c. Sebuah kata yang memiliki spasi (kata majemuk terbuka, misalnya, *"hot dog"*).
d. Sebuah kata yang tidak mengandung angka.
e. Sebuah kata dengan jenis encoding karakter UTF-8.

3.4.4 Deteksi dan Pengenalan Kata

Proses pendeteksian teks merupakan proses pertama dari 3 proses utama dalam sistem ini. Proses pendeteksian kata dilakukan untuk mendapatkan nilai dari teks yang dideteksi untuk kemudian digunakan pada proses penerjemahan. Proses pendeteksian teks dilakukan dengan teknologi *augmented reality*. Teknologi pendeteksian teks ini telah disediakan oleh Vuforia sebagai salah satu bentuk *tracking* untuk *augmented reality*. Gambar 3.8 berikut merupakan diagram alir dari proses deteksi kata dengan *augmented reality*.



Gambar 3.8 Diagram Alir Deteksi Text dengan Augmented Reality

Keterangan:

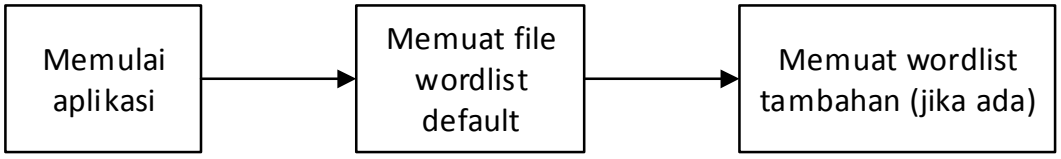
1) Inisialisasi pertama, sistem memuat *word list* untuk mempercepat pemilihan kata dalam pendeteksian.
2) Inisialisasi kedua, menentukan *Region Of Interest* (ROI) untuk memberikan daerah deteksi pada kamera dalam mendeketsi objek.
3) Memberikan inputan berupa citra yang diambil dari kamera secara *realtime.*
4) Kamera menangkap citra secara *realtime*. Dalam hal ini, kamera akan menangkap citra dengan mengambil *frame* tanpa melakukan *pause* seperti layakna menangkap *frame* dari video.

5) Melakukan *render* pada ROI dari objek citra yang tertangkap kamera untuk kemudian merepresentasikan setiap huruf yang ada pada citra tersebut kedalam bentuk biner dengan nilai 8-bit.
6) Menghasilkan frame dari citra yang telah direnderkemuadian dilakukan pelacakan objek teks dari frame citra.
7) Mencocokan nilai dari representasi biner dengan nilai pada *WordList* untuk menemukan kata yang cocok. Apabila ditemukan kecocokan maka sistem akan mengambil kata dari wordlist. Apabila tidak ditemukan kecocokan maka sistem akan melakukan pengenalan huruf dari objek lainnya.
8) Setelah ditemukan kecocokan dengan *wordlist* maka sistem akan menghasilkan keluaran berupa kata dengan tipe data *string* yang kemudian akan digunakan pada proses penerjemahan kata.

3.4.4.1 Memuat *Wordlist*

Seperti yang telah disebutkan, *wordlist* mewakili himpunan dasar kata-kata yang *Vuforia* dapat digunakan sebagai referensi untuk pencocokan elemen teks yang muncul dalam *field-of-view* kamera. *Wordlist* diambil dari file biner yang dikodekan dalam format *Vuforia Word List* (VWL). File *Wordlist default* VWL berisi lebih dari 100.000 kata bahasa Inggris yang sering digunakan sudah didistribusikan dan di bundel dengan *Vuforia* SDK.

Sebelum kamera di inisialisasi dan pendeteksian target dilakukan, sistem akan memuat *wordlist* ke dalam proses. Hal ini bertujuan untuk mengambil nilai dari semua daftar kata untuk pencocokan elemen teks yang muncul dari kamera. *wordlist* memiliki format biner khusus dari *Vuforia* yang dirancang untuk menyertakan satu set besar kata. Gambar 3.9 menunjukan diagram blok memuat wordlist.



Gambar 3.9 Diagram Blok Memuat WordList

Wordlist default dapat diperpanjang dengan daftar kata tambahan dan dapat juga disaring untuk mengecualikan kata-kata tertentu dari pengenalan atau untuk memastikan bahwa hanya beberapa kata-kata tertentu yang dapat dikenali.

a. Memuat *wordlist* dari file

Untuk memuat *wordlist* dari file, digunakan perintah *loadWordList()* yang telah disediakan oleh SDK dari *Vuforia*. Biasanya perintah *loadWordlist()* ini adalah untuk memuat *wordlist default* yang disediakan *Vuforia*. Contoh pemanggilan perintah *loadWordList()* adalah sebagai berikut.

```
// memuat wordlist dari file
wordlist.loadWordList("vuforia-default-english-
wordlist.vwl",
       WordList.STORAGE_TYPE.STORAGE_APPRESOURCE);
```

b. Menambah muatan kata-kata khusus dari file

Untuk menambah muatan kata-kata khusus dari file, digunakan perintah *addWordsFromFile()* yang telah disediakan oleh SDK dari *Vuforia*. File yang memuat kata-kata khusus tersebut perlu dibuat dahulu dengan ekstensi lst. Contoh pemanggilan perintah *addWordFromFile()* adalah sebagai berikut.

```
// menambah muatan kata-kata khusus dari file
wordlist.addWordsFromFile("my-custom-words.lst",
       WordList.STORAGE_TYPE.STORAGE_APPRESOURCE);
```

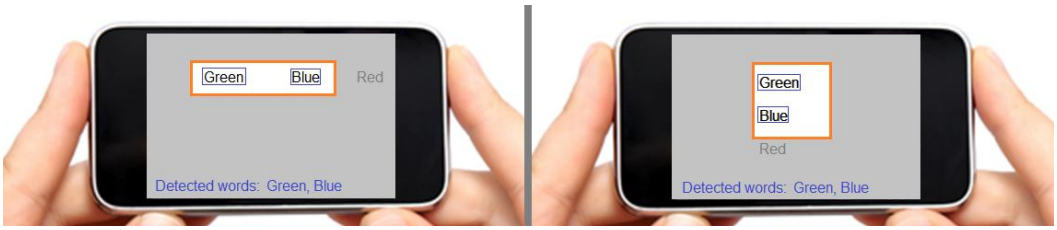
c. Menambah beberapa kata-kata khusus

Jika hanya beberapa kata yang akan ditambahkan pada *wordlist*, dapat juga digunakan perintah *addWordU()* yang telah disediakan oleh SDK dari *Vuforia.* Satu kali pemanggilan perintah *addWordU()* akan menambahkan satu kata kedalam *wordlist*. Contoh pemanggilan perintah *addWordU()* adalah sebagai berikut.

```
// menambah beberapa kata-kata khusus
wordList.addWordU( "region" );
wordList.addWordU( "interest" );
```

3.4.4.2 Menentukan *Region of Interest* (ROI)

Langkah pertama dalam mendeteksi sebuah teks adalah menentukan *Region of Interest* (ROI). ROI ditentukan untuk membatasi wilayah pelacakan dan deteksi dimana terdapat suatu teks. Gambar 3.10 adalah contoh dari *Region of Interest.*

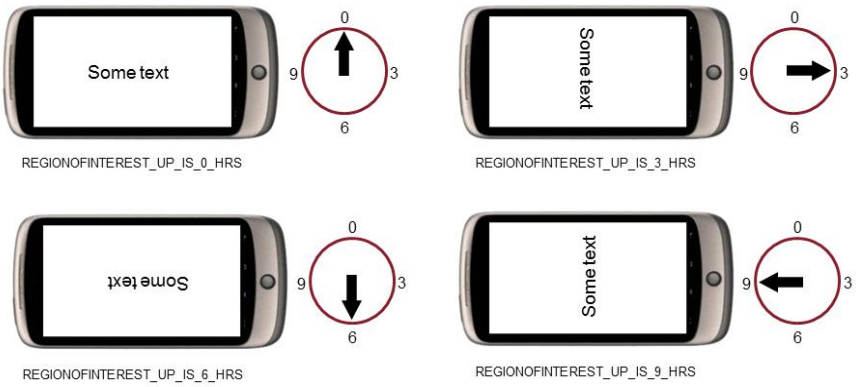


Gambar 3.10 Orientasi Region Of Interest

Ada hal yang perlu diperhatikan dalam mentukan ROI, yaitu ukuran dari ROI tersebut. Ukuran dari ROI untuk melakukan deteksi ini dapat mempengaruhi performansi dalam pendeteksian kata. Semakin luas ROI semakin lama juga waktu yang dibutuhkan untuk mendeteksi kata.

Setelah ROI ditetukan, akan ditambahkan informasi tentang arah. Arah yang dimaksud mendefinisikan orientasi teks dengan sumbu utama dari bingkai kamera seperti pada gambar 3.11. Arah diwakili oleh enum UP_DIRECTION, dan nilai dari enum tersebut dapat berupa salah satu dari nilai berikut (berdasarkan arah jarum jam) :

a. `REGIONOFINTEREST_UP_IS_0_HRS`
b. `REGIONOFINTEREST_UP_IS_3_HRS`
c. `REGIONOFINTEREST_UP_IS_6_HRS`
d. `REGIONOFINTEREST_UP_IS_9_HRS`



Gambar 3.11 Orientasi Region of Interest

Enumerasi UP_DIRECTION ini merupakan bagian dari kelas *TextTracker* sehingga untuk pemanggilannya, perlu dipanggil nama kelasnya dahulu. Pada penelitian ini, orientasi *Region Of Interest* yang digunakan adalah REGIONOFINTEREST_UP_IS_9 atau orientasi searah jarum jam 9. Rutin untuk melakukan inisialisasi orientasi ROI adalah sebagai berikut.

```
TextTracker.UP_DIRECTION upDirection =
        TextTracker.UP_DIRECTION.REGIONOFINTEREST_UP_IS_9_HRS;
```

Nilai atau ukuran dari ROI dapat ditentukan dengan memberikan nilai secara manual. Untuk menentukan nilai tersebut, perlu dibuat suatu persegi yang merepresentasikan ROI. *Vuforia* telah menyediakan kelas *Rectangle* (tipe data float) dan *RectangleInt* (tipe data integer) untuk menggambar persegi atau persegi panjang. Berikut rutin yang digunakan untuk merepresentasikan segi empat.

```
public RectangleInt(
           int leftTopX,
           int leftTopY,
           int rightBottomX,
           int rightBottomY
               );
```

Kemudian rutin tersebut diinstansiasi sebelum inisialisasi ROI untuk menggambarkan ROI kedalam segi empat dengan mengisi nilai-nilai parameternya sehingga nampak seperti berikut.

```
RectangleInt detectionRegion = new RectangleInt(20, 20, 460, 320);
```

Akan tetapi ukuran ROI yang digunakan pada penelitian ini tidak ditentukan dengan nilai yang statis, namun bergantung pada ukuran layar dari ponsel yang digunakan dengan margin kiri, kanan dan atas adalah 5% dari lebar layar ponsel dan tinggi ROI adalah 40% dari tinggi layar ponsel.

3.4.4.3 Menangkap Target dan *Rendering* Citra dari Kamera

Pada proses penangkapan target dari kamera, sistem mengambil citra dari *frame* yang ditangkap kamera. *Vuforia* akan mengaktifkan kamera dengan modus video yang dijalakankan secara *background* sehingga citra yang diambil berasal dari *frame buffer* video yang mengakibatkan pengambilan citra dapat dilakukan

secara *realtime*. Untuk mengaktifkan kamera dengan modus video yang dijalankan secara *backgroud* adalah dengan menginstansiasi kelas *VideoMode* dan *VideoBackgroundConfig* seperti pada rutin berikut.

```
VideoMode vm = CameraDevice.getInstance().getVideoMode(
        CameraDevice.MODE.MODE_DEFAULT);
VideoBackgroundConfig config = Renderer.getInstance()
        .getVideoBackgroundConfig();
```

Setalah melakukan instansiasi untuk modus kamera yang akan digunakan, kemudian tahapan selanjutnya adalah melakukan *render* pada objek yang tertangkap kamera. Adapun beberapa tahapan sebelum melakukan *render* menjadi citra yang dapat dilihat pada Gambar 3.12.



Gambar 3.12 Flow Process Menangkap Target Citra dari Kamera

Berikut adalah penjelasan langkah-langkah pada Gambar 3.12 yang merupakan *flow process* dari proses menangkap target citra dari kamera.

a. Menentukan tinggi dan lebar layar

Tinggi dan lebar layar ditentukan dengan mengambil nilai dari layout sistem. Pertama instasiasi objek *metrics* dari kelas *DisplayMetrics* yang disediakan android untuk mengambil nilai dari ukuran layar.

```
DisplayMetrics metrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);
```



Gambar 3.13 Lebar dan Tinggi Layar

Ambil nilai lebar dan tinggi dari objek *metrics* guna mendapatkan ukuran lebar dan tinggi layar seperti pada Gambar 3.13. Kemudian simpan nilai tersebut ke dalam variabel bertipe data integer. Adapaun kode untuk mengambil nilai dari objek *metrics* adalah sebagai berikut.

```
int screenWidth = metrics.widthPixels;
int screenHeight = metrics.heightPixels;
```

b. Menentukan lebar dan tinggi ROI

Setelah menentukan orientasi ROI dengan operasi UP_DIRECTION, sebagaimana telah dijelaskan sebelumya lebar dan tinggi ROI ditentukan berdasarkan ukuran layar ponsel atau tablet. Terdapat tiga langkah yang perlu dilakukan untuk menentukan lebar dan tinggi ROI, diantaranya:

1) Menentukan lebar margin

Jika perangkat yang digunakan wisatawan adalah tablet, maka lebar margin adalah 20% dari lebar layar sedangkan jika perangkat yang digunakan adalah ponsel maka lebar margin adalah 5% dari lebar layar seperti pada Gambar 3.14. Adapaun kode untuk menghitung lebar adalah sebagai berikut.

```
int marginWidth = mIsTablet ? (screenWidth * 20) / 100
          : (screenWidth * 5) / 100;
```



Gambar 3.14 Lebar Margin 5% dari Lebar Layar Ponsel

2) Menentukan tinggi ROI

Jika perangkat yang digunakan wisatawan adalah tablet, maka tinggi ROI adalah 30% dari tinggi layar sedangkan jika perangkat yang digunakan adalah ponsel maka tinggi ROI adalah 40% dari tinggi layar seperti pada Gambar 3.15. Adapaun kode menghitung tinggi ROI adalah sebagai berikut.

```
int loupeHeight = mIsTablet ? (screenHeight * 30) / 100
        : (screenHeight * 40) / 100;
```



Gambar 3.15 Tinggi ROI 40% dari Tinggi Layar Ponsel

3) Menentukan lebar ROI

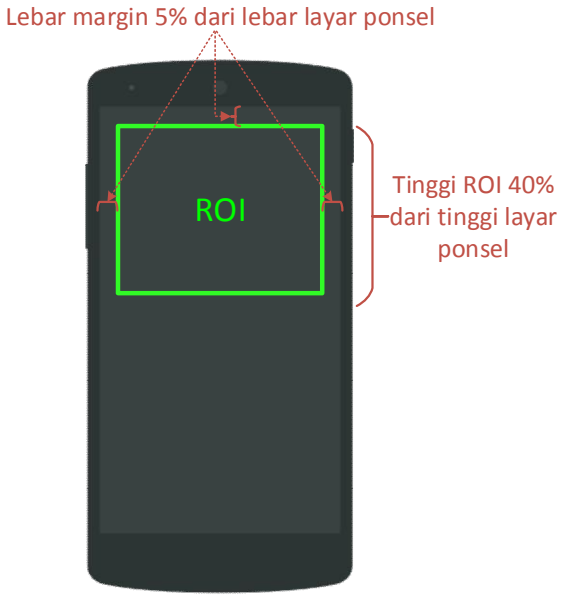
Lebar ROI adalah lebar layar dikurangi dengan dua kali lebar margin. Adapaun kode untuk menghitung tinggi ROI adalah sebagai berikut.

```
int loupeWidth = screenWidth - (2 * marginWidth);
```

c. Kalkulasi ROI

Untuk menentukan daerah mana yang merupakan daerah tangkapan target. Kemudian mengisi parameter pada operasi *setROI()* dengan parameter: titik tengah koordinat x = lebar layar / 2; titik tengah koordinat y = lebar margin + (tinggi ROI / 2); lebar ROI; tinggi ROI. Rutin untuk operasi *setROI()* adalah sebagai berikut.

```
setROI(
    screenWidth / 2,
    marginWidth + (loupeHeight / 2),
    loupeWidth,
    loupeHeight
    );
```



Gambar 3.16 Tampilan Akhir ROI yang Ditampilkan

Setelah melakukan konfigurasi ROI, maka *layout* atau susunan yang akan ditampilkan oleh sistem adalah seperti Gambar 3.16. Kotak yang berwarna hijau merupakan ROI untuk daerah deteksi.

d. Konversi koordinat layar ke koordinat kamera

Melakukan konversi koordinat layar dari ROI yang telah dikalkulasikan ke koordinat kamera, misalnya *cropping* gambar kamera untuk menyesuaikan aspek rasio layar yang berbeda. Untuk dimensi kamera, lebar ini selalu lebih

besar dari ketinggian (orientasi selalu lanskap). koordinat awal dimulai dari kiri atas layar kamera. Adapun rutin untuk mengkonversi koordinat layar ke koordinat kamera adalah sebagai berikut.

```
void screenCoordToCameraCoord(
    int screenX, int screenY,
    int screenDX, int screenDY,
    int screenWidth, int screenHeight,
    int cameraWidth, int cameraHeight,
    int[] cameraX, int[] cameraY,
    int[] cameraDX, int[] cameraDY
    );
```

e. Buat persegi dari koordinat kamera

Setelah mendapatkan koordinat kamera, kemudian tentukan sebuah persegi sebagai objek dari ROI yang telah dikalkulasi. Objek persegi ini, seperti yang sudah dijelaskan sebelumnya, memanfaatkan kelas *RectangleInt* untuk menggambarkan ROI ke layar untuk kemudian akan menjadi objek yang di-*render*.

f. Dapatkan data dan ukuran dari objek persegi

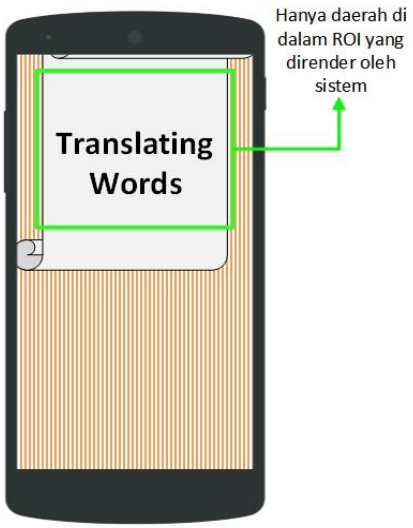
Mendapatkan data dan ukuran dari objek persegi ini bertujuan untuk memberikan nilai pada *viewport* dimana *viewport* sendiri merupakan area dimana objek yang menarik akan di-*render*. *Viewport* dinyatakan dalam koordinat *render-device-specific,* misalnya piksel untuk koordinat layar.

g. Render objek persegi menjadi citra

Sistem akan me-*render* objek yang ditangkap oleh kamera menjadi citra. Dalam melakukan render, sistem akan memanfaatkan kelas *Renderer* yang disediakan oeh *Vuforia* SDK dan kelas GLES20 yang disediakan oleh Android SDK.

```
State state = mRenderer.begin();
mRenderer.drawVideoBackground();
```

Operasi *begin()* akan memulai proses render dari *frame* dan operasi *drawVideoBackground()* akan memulai proses *video background*.



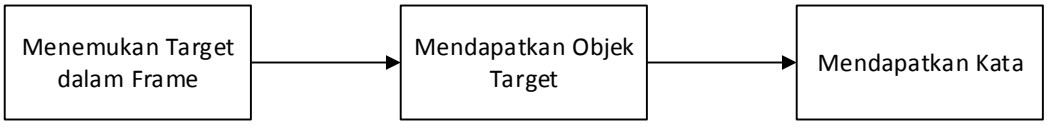
Gambar 3.17 Daerah yang Di-Render Oleh Sistem

Meskipun kamera dapat menangkap citra yang tampil di layar, citra yang di *render* oleh sistem hanya daerah di dalam ROI seperti pada Gambar 3.17. Daerah ini lah juga yang merupakan daerah pendeteksian kata.

Setelah sistem mendapatkan citra yang ditangkap oleh kamera yang mana citra tersebut merupakan *frame* dari video yang ditangkap kamera, kemudian sistem akan me-*render frame* tersebut untuk melakukan *binary masking*. *Masking* ini bertujuan untuk melakukan representasi citra ke biner.

3.4.4.4 Pelacakan Objek

Setelah sistem melakukan render pada target untuk mendapatkan citra dari setiap *frame* video, kemudian *Vuforia* akan melakukan pelacakan objek dimana objek tersebut merupakan teks (kata). Ada beberapa langkah yang perlu dilakukan untuk melakukan pelacakan teks dengan *Vuforia*. Gambar 3.18 berikut adalah langkah-langkah untuk melakukan pelacakan dan pengenalan kata dengan memanfaatkan API dari *Vuforia*.



Gambar 3.18 Alur Proses dari Pelacakan dan Pengenalan Kata

Berdasarkan Gambar 3.18 terdapat tiga proses utama dalam pelacakan dan pengenalan kata dengan *Vuforia*. Berikut adalah rincian dari ketiga langkah tersebut.

3.4.4.4.1 Menemukan Target dalam Frame

Sebelum melakukan pelacakan dan pengenalan kata, perlu dilakukan suatu pencarian terhadap target yang dapat dilacak pada *frame*. Dalam mencari target dalam *frame* yang telah di-*render*, dilakukan perulangan dengan rutin sebagai berikut.

```
for (int tIdx = 0; tIdx < state.getNumTrackableResults();
    tIdx++)
```

state.getNumTrackableResult() merupakan operasi dari kelas *State* yang berfungsi untuk mengambil nilai dari objek target yang sedang dilacak. Operasi ini telah disediakan oleh Vuforia dalam kelas *State.*

3.4.4.4.2 Menapatkan Objek Target

Untuk mendapatkan objek dari target yang sedang dilacak, pertama lakukan instansiasi objek dari kelas *TrackableResult* yang telah disediakan oleh vuforia untuk menampung hasil dari pelacakan.

```
TrackableResult result = state.getTrackableResult(tIdx);
```

state.getTrackableResult(intidx) merupakan operasi dari kelas *State* yang berfungsi untuk menyediakan akses pada objek tertentu dari *TrackableResult*. Operasi ini telah disediakan oleh Vuforia dalam kelas *State.*

Kemudian periksa apakah jenis objek *result* dari *TrackableResult* sama dengan jenis dari kelas *WordResult* dengan membandingkan operasi *result.getType()* dengan operasi *WordResult.getClassType()*. Untuk lebih jelasnya dapat dilihat pada rutin berikut.

```
if (result.isOfType(WordResult.getClassType()))
```

Jika kondisinya benar maka objek yang terlacak memiliki teks. Itu berarti dapat dilakukan pelacakan kata dari *frame* yang di deteksi tersebut. Kemudian langkah selanjutnya adalah mendapatkan kata.

3.4.4.4.3 Mendapatkan Kata

Setelah memeriksa jenis objek *result* dari *TrackableResult* dan asilnya adalah sama dengan jenis kelas *WordResult,* kemudian dilakukan pelacakan kata guna mendapatkan kata yang ada pada *frame.* Proses pelacakan kata ini dilakukan oleh *Vuforia.* Sistem yang dibangun hanya perlu mengintegrasikan *Vuforia* dengan memanggil kelas-kelas yang diperlukan untuk proses pelacakan kata ini. Langkah-langkah yang dilakukan untuk melakukan pelacakan kata adalah sebagai berikut.

a. Instansiasi Objek *wordResult* dari Kelas *WordResult*

```
WordResult wordResult = (WordResult) result;
```

Instansiasi objek *wordResult* ini berfungsi untuk merepresentasikan hasil pelacakan kata.

b. Instansiasi Objek *word* dari Kelas *Word*

```
Word word = (Word) wordResult.getTrackable();
```

Dimana *get.Trackable()* merupakan operasi dari kelas *WordResult* yang disediakan oleh Vuforia untuk mengembalikan nilai dari objek *Trackable* yang sesuai. Nilai tersebut merepresentasikan hasil dari proses pelacakan kata. *Vuforia* tidak menjelaskan proses bagaimana hasil pengenalan tersebut didapatkan. *Vuforia* hanya menjelaskan bahwa Setiap pixel dalam gambar diwakili oleh satu byte (dengan nilai 8-bit). Sebuah Nilai dari 255 menunjukkan area kosong, yaitu pixel yang tidak tertutup oleh huruf apapun dari kata. Jika pixel ditutupi oleh sebuah huruf, maka nilai pixel mewakili posisi dari huruf tersebut dalam kata, yaitu 0 untuk karakter pertama, 1 untuk kedua, 2 untuk yang ketiga, dan seterusnya.

c. Inisialisasi Objek *wordBoxSize* dari Kelas *Vec2F*

Vec2F merupakan kelas vektor 2 dimensi dari item dengan tipe data *float* yang disediakan oleh *Vuforia*. Rutin dari struct *Vec2F* adalah sebagai berikut.

```
class Vec2F
{
     Vec2F(){}

     Vec2F(constfloat* v)
     {
         for(inti=0; i<2; i++)
             data[i]= v[i];
     }

    Vec2F(float v0, float v1)
    {
         data[0] = v0;
         data[1] = v1;
    }

    float data[2];
};
```

Kemudian lakukan inisialisasi untuk memberikan nilai pada objek *wordBoxSize*. Nilai yang diberikan ini diambil dari operasi *word.getSize()* yang telah disediakan *Vuforia* pada kelas *Word*. Operasi *getSize()* tersebut berfungsi untuk mengembalikan nilai ukuran (lebar dan tinggi) dari kotak pembatas kata (*word bounding box*).

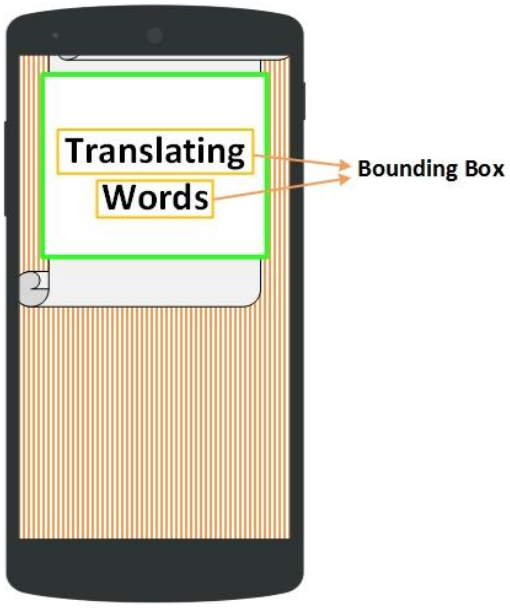
```
Vec2F wordBoxSize = word.getSize();
```

d. Instansiasi Objek *wordBox2D* dari Kelas Obb2D

Pemanggilan kelas Obb 2D ini bertujuan untuk menentukan kotak pembatas kata (*word bounding box*) dua dimensi. Nilai dari kotak pembatas kata ini diambil dari operasi *getObb2D()* dari objek *wordResult* pada kelas *WordResult*. Operasi *getObb2D()* ini berfungsi untuk mengembalikan kotak pembatas berorientasi pada jarak citra dari kata. Berikut adalah rutin untuk instansiasi objek *wordBox2D*.

```
Obb2D wordBox2D= wordResult.getObb2D();
```

Apabila terdapat kata yang terdeteksi oleh sistem, maka sistem akan melakukan *render* pada objek *wordBox2D* yang mana merupakan kotak pembatas kata. *Render* dilakukan untuk menampilkan kotak pembatas kata ke layar. Hal ini bertujuan untuk memberi tanda pada kata yang terdeteksi atau dikenali oleh sistem.



Gambar 3.19 Kotak Pembatas Kata

e. Mengabil Nilai dari Kata

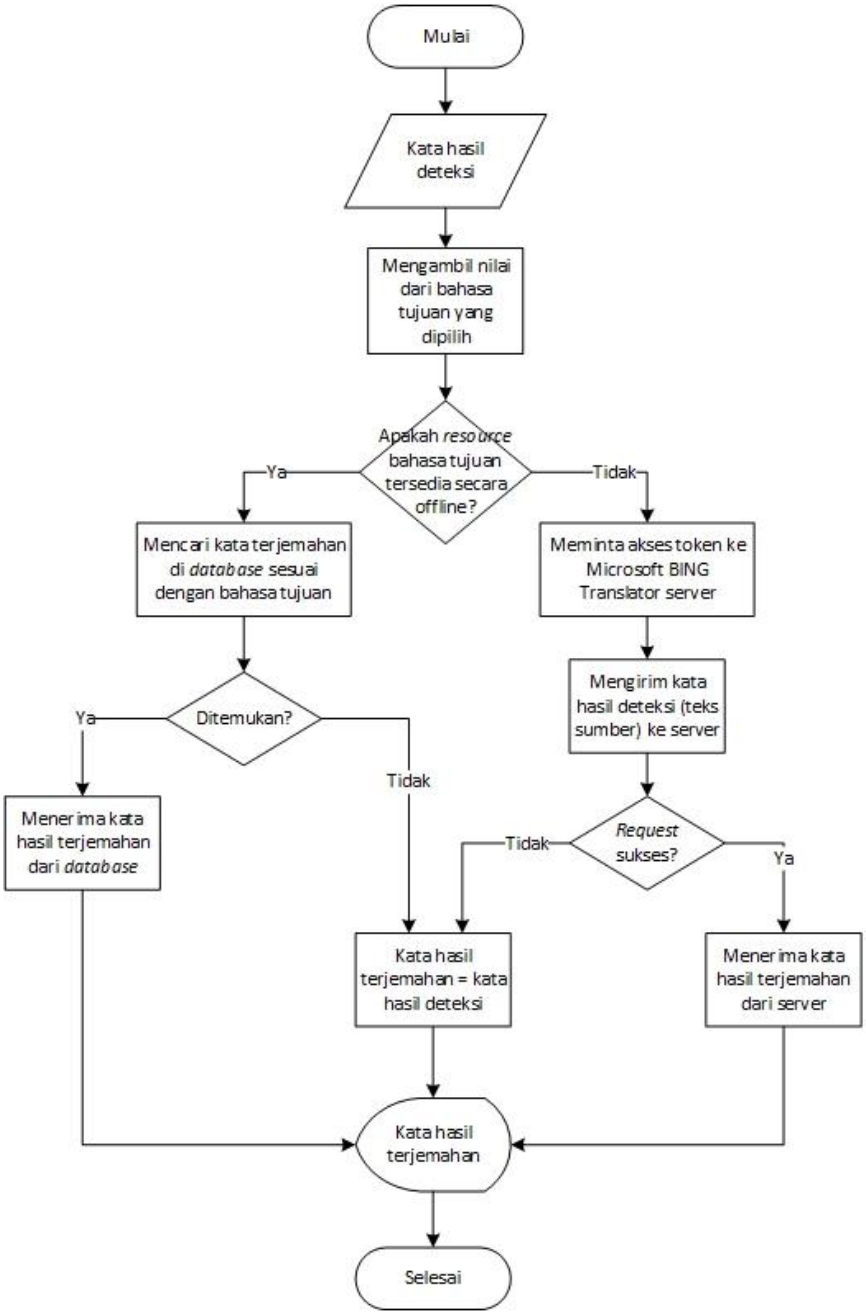
Langkah berikutnya dalam mendapatkan kata adalah mengambil nilai dari kata itu sendiri. *Vuforia* telah menyediakan fungsi untuk mengambil nilai dari kata yang terdeteksi. Fungsi tersebut merupakan operasi *getStringU()* yang ada pada kelas *Word.* Operasi ini mengambalikan nilai *string* dari karakter *Unicode* untuk kata yang terdeteksi. Berikut adalah rutin yang digunakan untuk mengambil nilai dari kata yang terdeteksi.

```
String wordU = word.getStringU();
```

Variabel *wordU* tersebut telah menampung nilai bertipe data *string* hasil dari pengenalan kata.

3.4.5 Penerjemahan Kata

Setalah setelah sistem mendapatkan nilai hasil pendeteksian kata, kemudian sistem akan menerjemahkannya sesuai dengan bahasa tujuan yang dipilih oleh wisatawan. Ada dua cara yang dilakukan untuk menerjemahan kata berdasarkan ketersediaan *resource* dari bahasa yang ada, yaitu secara *Offline* dan *Online*. Gambar 3.20 adalah diagram alir penerjemahan kata.



Gambar 3.20 Diagram Alir Penerjemahan Kata

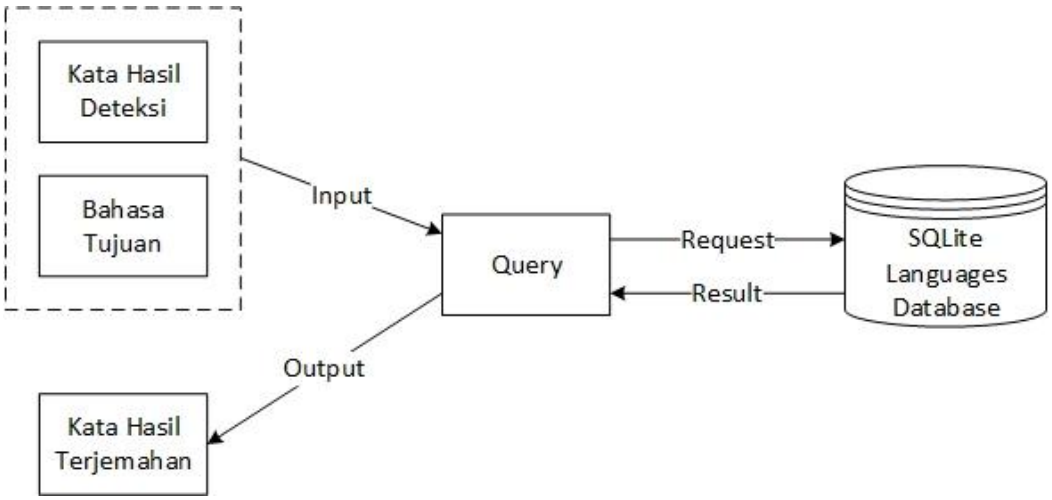
Keterangan:

1) Mengambil nilai dari hasil deteksi kata dan nilai dari bahasa tujuan yang dipilih wisatawan.
2) Mencari apakah bahasa tujuan yang dipilih wisatan memiliki *resource* secara *offline* atau tidak. Jika *resource* tersedia maka sistem akan menerjemakan kata secara *offline.* Jika *resource* tidak tersedia maka sistem akan menerjemahkan kata secara online dengan memanfaatkan Microsoft Bing Translator API.
3) Untuk terjemahan secara *offline* atau tanpa perlu terhubung dengan jaringan internet, sistem memanfaatkan *database* untuk menyimpan sistem daftar kata terjemahan.
4) Sistem akan mengeksekusi perintah SQLite untuk mencari kata terjemahan dari kata hasil deteksi sesuai dengan bahasa asal dan bahasa tujuan.
5) Sistem akan menerima hasil dari perintah SQLite tersebut. Jika kata terjemahan ditemukan maka hasil terjemahan merukan hasil dari *query* SQLite tersebut. Jika kata terjemahan tidak ditemukan maka nilai dari hasil deteksi kata merupakan nilai hasil terjemahan dengan kata lain, kata tidak diterjemahkan.
6) Untuk terjemahan secara *online* atau harus terhubung dengan jaringan internet, sistem akan meminta akses token dari server Microsoft Bing Translator.

Sistem akan mengirim kata hasil deteksi ke server. Jika *request* untuk terjemahan sukses maka server akan mengirimkan hasil terjemahan kata. Jika *request* gagal maka server akan mengirimkan pesan kesalahan dan kata pun tidak akan diterjemahkan.

3.4.5.1 Penerjemahan Kata Secara Offline

Penerjemahan kata secara *offline* dilakukan dengan memanfaatkan database untuk menyimpan *resource* dari kumpulan kata dalam berbagai bahasa. Walaupun bahasa yang disediakan tidak sebanyak yang disediakan oleh penyedia layanan terjemahan online, penerjemahan kata secara *offline* dapat menerjemahkan kata tanpa harus terhubung ke jaringan internet sehingga tidak tergantung dengan kecepatan dan ketersediaan jaringan internet di tempat wisatawan berada.



Gambar 3.21 Alur Proses Terjemahan Offline

Gambar 3.21 merupakan ilustrasi alur dari proses terjemahan secara *offline* dimana kata hasil deteksi dan bahasa tujuan diinputkan ke query, kemudian query akan melakukan *request* ke *database*. Kemudian, *database* akan mengirimkan hasil query dan hasil query inilah yang merupakan hasil terjemahan secara *offline*. Adapun tahapan-tahapan untuk melakukan terjemahan secara *offline* adalah sebagai berikut.

a. Membuat tabel pada DBMS SQLite.

Untuk menggunakan kamus pada terjemahan *offline* perlu disiapkan satu buah tabel untuk menampung data yang berisi kumpulan kata dalam berbagai bahasa. Struktur data untuk tabel kamus adalah seperti pada Tabel 3.1.

Tabel 3.1 Struktur File Tabel dict (kamus)

Nama Field	Tipe	Panjang	Kunci	Keterangan
_id	int	7	Primary key	Auto increment
en_us	varchar	24	-	-
id	varchar	24	-	-
fr	varchar	24	-	-
it	varchar	24	-	-
sp	varchar	24	-	-

Agar *database* dapat dikenali oleh Android maka perlu ditambahkan sebuah tabel lagi dengan nama *android_metadata*. Struktur file tabel *android_metadata* dapat dilihat pada Tabel 3.2

Tabel 3.2 Struktur File Tabel android_metadata

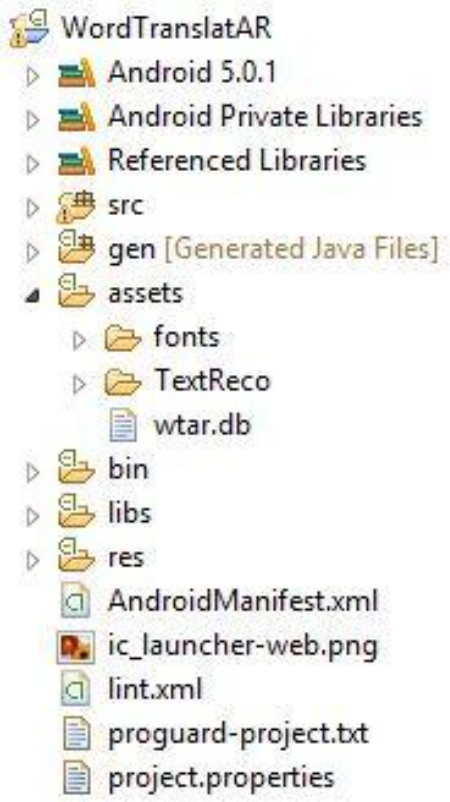
Nama Field	Tipe	Panjang	Kunci	Keterangan
locale	text	-	-	-

b. Mengisi tabel dengan kata-kata dalam berbagai bahasa.

Setelah tabel kamus tersedia, kemudian dilakukan operasi *insert* untuk mengisi tabel dengan kumpulan kata. Cara tercepat untuk melakukan *insert* dari ribuan kata adalah dengan melakukan *export* dari file ke *database*. File yang digunakan dapat berupa file teks ataupun file dari Microsoft Excel yang berformat .csv.

c. Menyimpan file SQLite ke folder assets.

Setelah tabel kamus terisi, simpan database SQLite dengan ekstensi file *.db, pada penelitian ini file database bernama *wtar.db*. Kemudian salin atau simpan file SQLite tersebut ke dalam foler assets di *project* Android seperti pada gambar 3.22



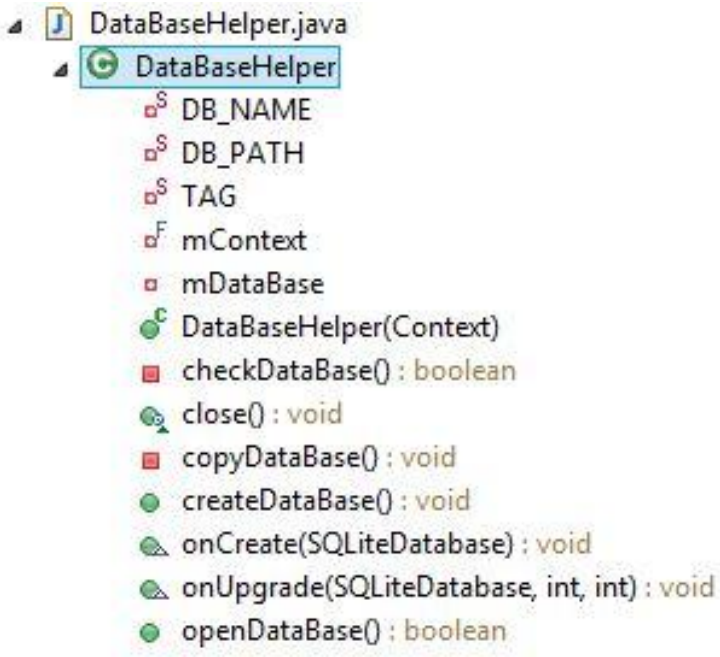
Gambar 3.22 Menyimpan File SQLite di Folder Assets

d. Membuat kelas penghubung dan penampung data

Untuk berinteraksi dengan *database*, perlu dibuat sebuah kelas yang berfungsi sebagai penghubung sistem dengan *database* dan dan sebuah kelas untuk menampung data.

1) Kelas DataBaseHelper

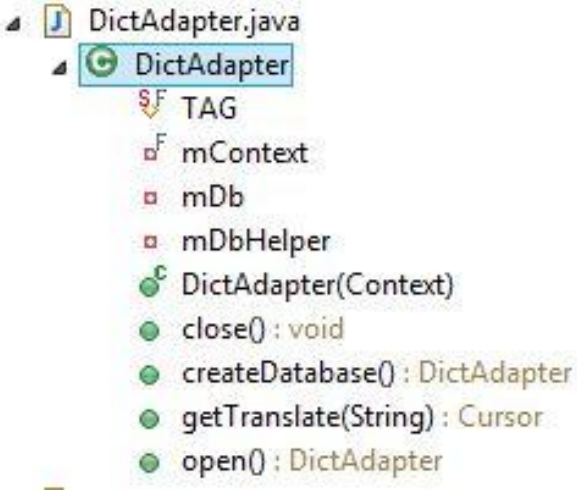
Kelas *DataBaseHelper* merupakan kelas yang mengimplementasikan kelas *SQLiteOpenHelper* yang disediakan oleh Android. Kegunaannya adalah untuk menghubungkan sistem dengan *database* SQLite. Struktur dari kelas *DataBaseHelper* dapat dilihat pada Gambar 3.23.



Gambar 3.23 Struktur Kelas DataBaseHelper

2) Kelas DictAdapter

Kelas *DictAdapter* merupakan kelas yang berfungsi untuk menampung data yang didapatkan dari *database*. Pada kelas ini terdapat operasi *getTranslate(String word)* yang berfungsi untuk mmelakukan terjemahan secara *offline* dan menampung data hasil terjemahan. Struktur dari kelas *DictAdapter* dapat dilihat pada gambar 3.24.



Gambar 3.24 Struktur Kelas DictAdapter

e. Mengeksekusi perintah SQLite untuk menerjemahkan kata.

Untuk menerjemahkan secara *offline* sistem akan mengambil nilai dari kata hasil deteksi dan nilai dari bahasa tujuan. Hal ini dilakukan pada operasi *getTranslate(String word).* Adapun rutin untuk operasi *getTranslate(String word)* adalah sebagai berikut.

```
public Cursor getTranslate(String word)
    {
        word = '"' + word + '"';
        try
        {
            String sql ="SELECT id FROM dict WHERE
           lower(en_us)=lower("+ word + ") LIMIT 0,1";

            Cursor mCur = mDb.rawQuery(sql, null);
            if (mCur!=null)
            {
               mCur.moveToNext();
            }
            return mCur;
        }
        catch (SQLException mSQLException)
        {
            Log.e(TAG, "getTranslate >>"+
                  mSQLException.toString());
            throw mSQLException;
        }
    }
```

Misalkan kata hasil deteksi adalah "*Translate"* dan bahasa tujuan adalah "Bahasa Indonesia". Kemudian nilai tersebut di inputkan kedalam query. Sebagai contoh, query untuk menerjemahkan kata "*Translate"* dengan bahasa tujuan "Bahasa Indonesia" adalah sebagai berikut.

```
SELECT id FROM dict WHERE lower(en_us)=lower('Translate') LIMIT
    0,1
```

Kemudian query yang dieksekusi akan mengakses data yang ada dari *database* dan melakukan pencarian kata terjemahan sesuai dengan bahasa tujuan yang dipilih.

f. Mengambil hasil dari *query* SQLite.

Setelah query dieksekusi, hasil dari query ditampung pada suatu *Cursor* dimana *cursor* ini berfungsi sebagai pointer dari data hasil query. Adapun rutin untuk melakukan terjemahan *offline* dan mengambil data dari hasil query adalah sebagai berikut.

```
Cursor c = mActivity.mDbHelper.getTranslate(kataHasilDeteksi);
if(Column.GetColumnValue(c, "id")!=""){
      KataHasilDeteksi = Column.GetColumnValue(c, "id");
}
else{
      KataHasilDeteksi = KataHasilDeteksi;
}
```

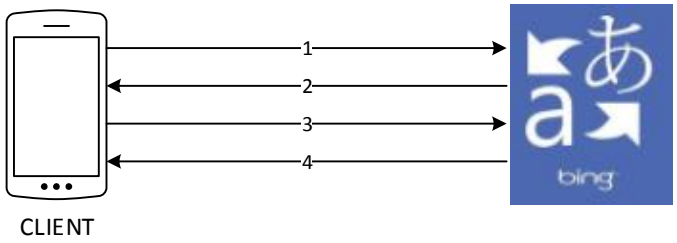
Proses terjemahan *offline* ini dilakukan setiap kali ada kata yang terdeteksi, sehingga sistem akan menerjemahankan setiap kata, bukan kalimat.

3.4.5.2 Penerjemahan Kata Secara Online

Pada penelitian ini, Microsoft Bing Translator dimanfaatkan untuk menerjemahan kata secara *online* apabila bahasa tujuan yang dipilih wisatawan tidak terdaftar sebagai bahasa yang memiliki *resource offline.* Microsoft Bing Translator API merupakan API yang digunakan untuk fitur penerjemah yang terdapat pada sistem. Secara default API ini akan menerjemahkan semua pesan yang diterima ke dalam bahasa yang digunakan oleh *user.*

Hasil dari deteksi kata akan ditampung dalam variabel untuk kemudian dikirimkan ke server Microsoft Translator oleh sistem untuk diterjemahkan. Apabila server Microsoft Translator memberikan balasan berupa pesan kesalahan atau kata yang dikirim tidak memliki terjemahan pada bahasa tujuan, maka sistem akan menampilkan kata awal (kata hasil deteksi) ke layar ponsel sebagai hasil terjemahan.

Gambar 3.25 adalah langkah-langkah menerjemahkan pesan menggunakan Microsoft Bing Translator API :



Gambar 3.25 Alur Penggunaan Microsoft Bing Translator

Keterangan :

1) *Client* meminta *access token* ke *server*.
2) *Server* merespon dengan memberikan *access token* yang dapat digunakan untuk mengakses Microsoft Bing Translator API.
3) *Client* mengirim teks sumber untuk diterjemahkan.
4) 'Jika *request* sukses, maka *server* akan membalas dengan sebuah niali string yang mewakili teks sumber yang telah diterjemahkan. Namun ketika *request* gagal, *server* akan mengirim pesan kegagalan.

3.4.5.2.1 Akses Token

Client memulai dengan meminta *access token* ke *server*, kemudian *server* akan merespon dengan memberikan *access token*ke *Client*. Parameter untuk permintaan token menggunakan *URL-encoded* dan dikirimkan melalui *HTTP request body.* Tabel 3.3 berikut menjelaskan parameter-parameter yang digunakan dalam meminta *access token.*

Tabel 3.3 Token *Request Input Parameter*

Parameter	Deskripsi
client_id	*Client id* yang didapatkan ketika mendaftarkan applikasi ke *Azure DataMarket*
client_secret	*Client secret* yang didapatkan ketika mendaftarkan applikasi ke *Azure DataMarket*
scope	Gunakan http://*API*.microsofttranslator.com URL sebagai nilai lingkup untuk Penterjemah *APIMicrosoft*.
grant_type	Gunakan "*client_credentials*" sebagai nilai *grant_type* untuk Penterjemah *APIMicrosoft*.

Respon untuk permintaan akses token berisi akses token yang dapat Anda gunakan untuk mengakses *TranslatorAPIMicrosoft*. Respon berupa *JSON-encoded* dengan beberapa attribut *output* dapat dilihat pada Tabel 3.4.

Tabel 3.4 Token *Request Output Parameter*

Parameter	Deskripsi
access_token	Akses token yang dapat Anda gunakan untuk otentikasi Anda akses ke *Microsoft Translator API.*
token_type	Format akses token.
expires_in	Jumlah detik dimana akses token valid.
scope	*Domain* dimana token ini berlaku. Untuk *Microsoft Translator API*, domain adalah http://*API*.microsofttranslator.com.

3.4.5.2.2 Menerjemahkan Pesan

Setelah *client* mendapatkan akses token, *client* dapat memulai menerjemahkan pesan teks dari suatu bahasa ke bahasa lain. Url yang digunakan untuk mengirimkan pesan memiliki format sebagai berikut :

`http://API.microsofttranslator.com/V2/Ajax.svc/Translate`

Berikut adalah parameter-parameter *query* yang dapat digunakan pada *Microsoft Translator API.*

Tabel 3.5 Parameter *Query Microsoft Translator API*

Parameter	Deskripsi
appid	Diperlukan. Sebuah *string* yang berisi "*Bearer*" + "" + akses token.
text	Diperlukan. Sebuah *string* yang mewakili teks untuk diterjemahkan. Ukuran teks harus tidak melebihi 10000 karakter.
from	Opsional. Sebuah *string* yang mewakili kode bahasa teks terjemahan. Jika dibiarkan kosong respon akan memberikan hasil dari *language auto-detection*
to	Diperlukan. Sebuah *string* yang mewakili kode bahasa untuk menerjemahkan teks ke bahasa target.
contentType	Diperlukan. Format teks yang diterjemahkan. Format yang didukung adalah "*text* / *plain*" dan "*text* / *html*".
category	Opsional. Sebuah string yang berisi kategori (*domain*) terjemahan. Defaultnya adalah "umum".

Jika *request* sukses, maka *server* akan memberikan sebuah nilai *string* yang mewakili teks yang diterjemahkan.

```
public String translate(String wordU, Language lang)throws
Exception
{
    translatedText = "";
    Translate.setClientId("1234567890987654321");
    Translate.setClientSecret("YuxgyLxvFd8q16llpzluLwPSnaZjrGusQyR
    4aPXvta4=");
    translatedText = Translate.execute(wordResult,
                                Language.ENGLISH, lang);
        return translatedText;
}
```

Operasi *translate()* ini akan dipanggil setiap kali sistem mendeteksi suatu kata. Sehingga hasil terjemahan sama seperti pada terjemahan *offline* yaitu dengan menterjemahkan setiap kata bukan dalam bentuk kalimat. Sistem akan mengirim *request* kepada *server* untuk menerjemahkan sebuah kata yang terdeteksi. Adapun rutin untuk memanggil operasi *translate()* adalah sebagai berikut.

```
if(mIC.isInetAvailable()==false){
    kataHasilDeteksi = "No Internet Connection";
}
else{
    try {
       kataHasilDeteksi = mActivity.mTranslator.translate(
           kataHasilDeteksi, mActivity.rLang);
    }
    catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
    }
}
```

3.4.6 Menampilkan Hasil Terjemahan dengan Augmented Reality

Setelah hasil pengenalan kata diterjemahkan ke bahasa tujuan, kemudian hasil terjemahan kata tersebut ditampilkan ke layar. Untuk menampilkannya kelayar, terdapat dua tahapan yaitu menentukan posisi objek kemudian melakukan *render* terhadap *frame*.

3.4.6.1 Menentukan Posisi Objek

Untuk menentukan posisi objek, pertama periksa jika kedua kata berada dalam garis yang sama. Kedua kata dikatakan berada dalam garis yang sama jika titik tengah (pada sumbu Y) dari titik pertama adalah diantara nilai dari titik kedua.

```
public int compareTo(WordDesc w2)
{
    WordDesc w1 = this;
    int ret = 0;

    int mid1Y = (w1.Ay + w1.By) / 2;

    if ((mid1Y < w2.By) && (mid1Y > w2.Ay))
    {
        // kedua kata berada dalam garis yang sama
        ret = w1.Ax - w2.Ax;
    } else
    {
        // kedua kata berada dalam garis yang berbeda
        ret = w1.Ay - w2.Ay;
    }
        Log.e(LOGTAG, "Compare result> " + ret);
    return ret;
}
```

Tujuan dari pemerikasaan kata berada pada garis yang sama atau tidak adalah untuk menentukan posisi kata hasil terjemahan yang akan ditampilkan ke layar. Apabila kedua kata berada pada garis yang berbeda maka hasil terjemahan yang ditampilkan pun demikian. Jika yang terdeteksi lebih dari satu kata maka kata kedua letaknya akan berada di bawah kata pertama. Posisi letak kata hasil terjemahan akan ditampilkan dibawah persegi panjang ROI dan menggunakan *center alignment* atau konten berada di tengah layar.

3.4.6.2 *Rendering Frame* untuk Menampilkan Objek

Setelah menentukan posisi untuk menampilkan hasil dari kata yang telah diterjemahkan, kemudian dilakukan *render frame* untuk menampilkan objek tersebut ke layar. Untuk melakukan *render* tersebut *Vuforia* telah menyediakan kelas untuk melakukannya dan ditambah juga dengan kelas GLES20 dari Android SDK.

Berikut adalah proses *rendering* yang dilakukan dengan *Vuforia*.

```
if (wordU != null)
{
    float wordx = -obb.getCenter().getData()[1];
    float wordy = obb.getCenter().getData()[0];

    if (mWords.size() < MAX_NB_WORDS)
    {
        mWords.add(new WordDesc(wordU,
        (int) (wordx - wordBoxSize.getData()[0] / 2),
        (int) (wordy - wordBoxSize.getData()[1] / 2),
        (int) (wordx + wordBoxSize.getData()[0] / 2),
        (int) (wordy + wordBoxSize.getData()[1] / 2)));
    }


    Matrix44F mvMat44f =
            Tool.convertPose2GLMatrix(result.getPose());
    float[] mvMat = mvMat44f.getData();
    float[] mvpMat = new float[16];
    Matrix.translateM(mvMat, 0, 0, 0, 0);
    Matrix.scaleM(mvMat, 0,
            wordBoxSize.getData()[0] - TEXTBOX_PADDING,
    wordBoxSize.getData()[1] - TEXTBOX_PADDING, 1.0f);
    Matrix.multiplyMM(mvpMat, 0, vuforiaAppSession
            .getProjectionMatrix().getData(), 0, mvMat, 0);

    GLES20.glUseProgram(shaderProgramID);
    GLES20.glLineWidth(3.0f);
    GLES20.glVertexAttribPointer(vertexHandle, 3, GLES20.GL_FLOAT,
                 false, 0, mQuadVerts);
    GLES20.glEnableVertexAttribArray(vertexHandle);
    GLES20.glUniform1f(lineOpacityHandle, 1.0f);
    GLES20.glUniform3f(lineColorHandle, 1.0f, 0.447f, 0.0f);
    GLES20.glUniformMatrix4fv(mvpMatrixHandle,
                1, false, mvpMat, 0);
    GLES20.glDrawElements(GLES20.GL_LINES,
                NUM_QUAD_OBJECT_INDICES,
                GLES20.GL_UNSIGNED_SHORT, mQuadIndices);
    GLES20.glDisableVertexAttribArray(vertexHandle);
    GLES20.glLineWidth(1.0f);
    GLES20.glUseProgram(0);
}

// Draw the region of interest
GLES20.glDisable(GLES20.GL_DEPTH_TEST);
drawRegionOfInterest(ROICenterX, ROICenterY, ROIWidth, ROIHeight);
GLES20.glDisable(GLES20.GL_BLEND);
```

Setelah melakukan *render* maka sistem akan menampilkan terjemahan ke layar ponsel. Hasil terjemahan ini akan ditampilkan secara *realtime* saat sistem dapat mengenali kata yang ada pada objek. Untuk lebih lengkapnya, ilustrasi hasil tampilan di layar dapat dilihat pada Gambar 3.26.



Gambar 3.26 Menampilkan Objek Hasil Terjemahan

Gambar 3.26 merupakan ilustrasi hasil akhir untuk menampilkan kata hasil terjemahan. Kata hasil terjemahan berada tepat dibawah ROI untuk deteksi dengan menggunakan *center alignment*.

3.5 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional merupakan penguraian kebutuhan-kebutuhan non fungsional (batasan sistem) menjadi kebutuhan yang lebih detail berdasarkan kebutuhan yang ada. Spesifikasi non-fungsional melibatkan beberapa kebutuhan, perangkat lunak, perangkat keras dan pengguna.

3.5.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak merupakan penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi perangkat lunak dan berhubungan dengan proses pembangunan perangkat lunak.

Perangkat lunak yang digunakan dalam pengembangan aplikasi, diantaranya :

a. Perangkat lunak PC :
    1) Sistem operasi (Windows Vista atau lebih baru)
    2) Vuvoria SDK
    3) Android NDK
    4) Eclipse Bundle ADT
b. Perangkat lunak pada *smartphone* yang digunakan adalah sistem operasi Android dengan spesifikasi sistem operasi minimum adalah versi 2.3 *Gingerbread.*

3.5.2 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras merupakan penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi perangkat keras dan berhubungan dengan proses pembangunan perangkat lunak. Perangkat keras yang digunakan dalam pengembangan aplikasi, diantaranya dapat dilihat pada tabel 3.6.

Tabel 3.6 Perangkat Keras yang Digunakan

Perangkat Keras PC	Perangkat Keras *Smartphone*
a. *Processor* minimal *DualCore* 2.0 GHz (disesuaikan dengan sistem operasi yang digunakan). b. RAM 2 GB atau lebih (disesuaikan dengan sistem operasi yang digunakan). c. *Hardisk* minimal 20 GB (untuk Windows 7) disesuaikan dengan sistem operasi yang digunakan.	a. *Smartphone* atau *tablet* bersistem operasi Android. b. CPU minimal *SingleCore* 1.0 GHz ARM Cortex A8. c. RAM minimal 512 MB. d. Memiliki kamera. e. Dapat terhubung ke jaringan internet melalui konektifitas 2G/3G/LTE maupun melalui Wi-Fi.

3.5.3 Analisis Kebutuhan Perangkat Pikir

Analisis kebutuhan perangkat pikir merupakan penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi pengguna dan berhubungan dengan perangkat lunak.

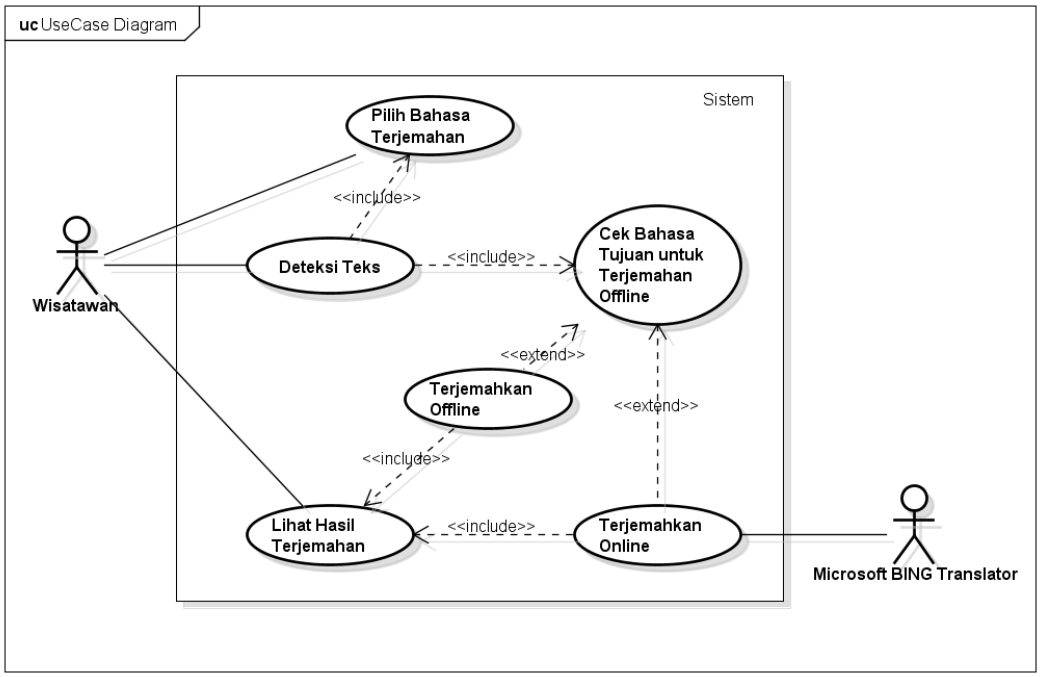
Aplikasi yang akan dibangun ditujukan bagi pengguna dengan perangkat berbasis Android, terutama bagi para wisatawan yang sedang berkunjung ke negara lain dan kurang memahami bahasa pada yang digunakan di negara tersebut.

3.6 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan penguraian dari kebutuhan yang berkaitan dengan fungsi atau proses transformasi yang harus mampu dikerjakan oleh perangkat lunak seperti fitur, karakteristik, dan batasan lainnya yang berhubungan dengan sistem. Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Tahap-tahap pemodelan dalam analisis tersebut antara lain *Use Case Diagram*, *Activity Diagram, Sequence Diagram*, dan *Class Diagram*.

3.6.1 *Use Case Diagram*

Didalam sistem ini terdapat 2 aktor yaitu wisatawan dan Microsoft Bing Translator. Wisatawan merupakan pengguna yang memiliki hak akses untuk konfigurasi, deteksi teks dan melihat hasil terjemahan. Sedangkan Microsoft Bing Translator merupakan API yang melakukan proses penerjemahan teks. Hak akses kedua aktor tersebut dapat dilihat pada diagram *Use Case* pada gambar 3.27 berikut.



Gambar 3.27 *Use Case Diagram* Word TranslatAR

3.6.1.1 Definisi Aktor

Definisi aktor merupakan penjelasan dari aktor yang ada pada diagram *use case.* Berikut merupakan definisi aktor dari *use case* pada gambar 3.27.

Tabel 3.7 Definisi Aktor Pada Diagram *Use Case* Word TranslatAR

No	Aktor	Definisi
1.	Wisatawan	Pengguna yang memiliki hak askes dalam konfigurasi sistem, deteksi teks dan melihat hasil terjemahan.
2.	Microsoft Bing Translator	API yang melakukan proses terjemahan teks secara online.

3.6.1.2 Definisi *Use Case* Word TranslatAR

Definisi *use case* merupakan penjelasan dari *use case* yang ada pada diagram *use case.* Berikut merupakan definisi aktor dari *use case* pada gambar 3.27.

Tabel 3.8 Definisi *Use Case* Pada Diagram *Use Case* Word TranslatAR

No	*Use case*	Definisi
1.	Pilih Bahasa Terjemahan	Merupakan fungsional untuk memilih bahasa yang akan digunakan untuk proses penerjemahan.
2.	Deteksi Teks	Merupakan proses pendeteksian teks yang akan diterjemahkan.
3.	Cek Bahasa Tujuan untuk Terjemahan Offline	Merupakan proses pengecekan untuk ketersediaan resource terjemahan offline yang ada dalam sistem.
4.	Terjemahkan Offline	Merupakan proses penerjemahan yang akan dilakukan oleh sistem.
5.	Terjemahkan Online	Merupakan proses penerjemahan yang akan dilakukan oleh Microsoft Bing Translator.
6.	Lihat Hasil Terjemahan	Merupakan proses akhir dari sistem dengan menampilkan hasil terjemahan teks.

3.6.1.2.1 Pilih Bahasa Terjemahan

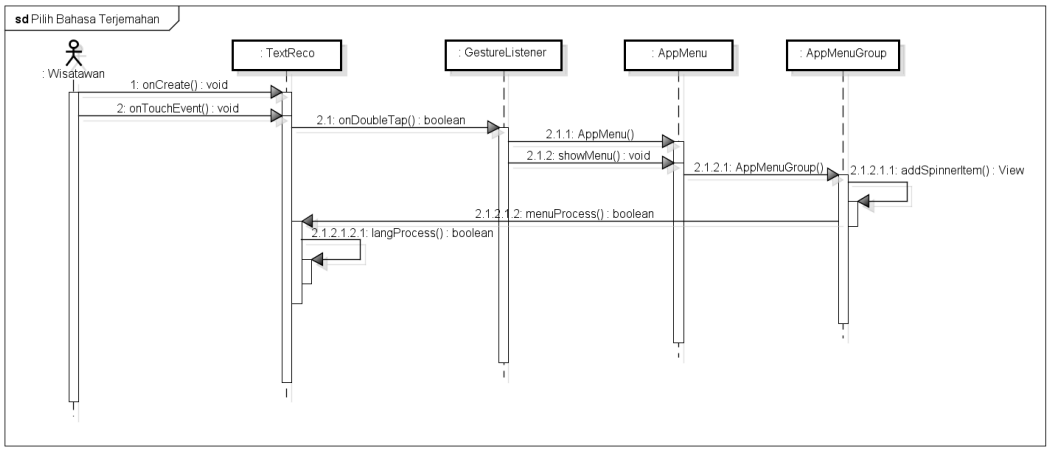
Pilih Bahasa Terjemahan merupakan fungsional yang berfungsi untuk menentukan bahasa. Pemilihan bahasa ditentukan oleh wisatawan sesuai dengan bahasa yang dideteksi dan bahasa yang akan diterjemahkan.

Tabel 3.9 Skenario *Use Case* Pilih Bahasa Terjemahan

Use Case	**Pilih Bahasa Terjemahan**
Related Requirement	Sistem menyediakan halaman konfigurasi sistem
Goal in Context	Wisatawan dapat memilih bahasa terjemahan
Preconditions	Sistem menampilkan halaman konfigurasi sistem
Successful End Conditions	Pilih bahasa terjemahan berhasil
Failed End Conditions	Tidak memilih bahasa terjemahan
Primary Actors	Wisatawan
Secondary Actors	
Trigger	Sistem menampilkan pilih bahasa terjemahan bawaan/default
Included Class	AppMenuGroup, AppMenu, TextReco, Language

Main Flow	Step	Action
	1	Sistem menampilkan menu konfigurasi sistem
	2	Wisatawan memilih bahasa awal terjemahan
	3	Wisatawan memilih bahasa tujuan terjemahan
	4	Sistem melakukan perubahan bahasa asal dan bahasa tujuan
Extensions	Step	Branching Action

Sistem menampilkan menu konfigurasi sistem kemudian wisatawan memilih bahasa asal dan bahasa tujuan pada menu pilih bahasa lalu sistem menyimpan konfigurasi bahasa yang telah dipilih. Gambar 3.28 menunjukan *sequence diagram* pilih bahasa terjemahan



Gambar 3.28 Sequence Diagram Pilih Bahasa Terjemahan

3.6.1.2.2Deteksi Teks

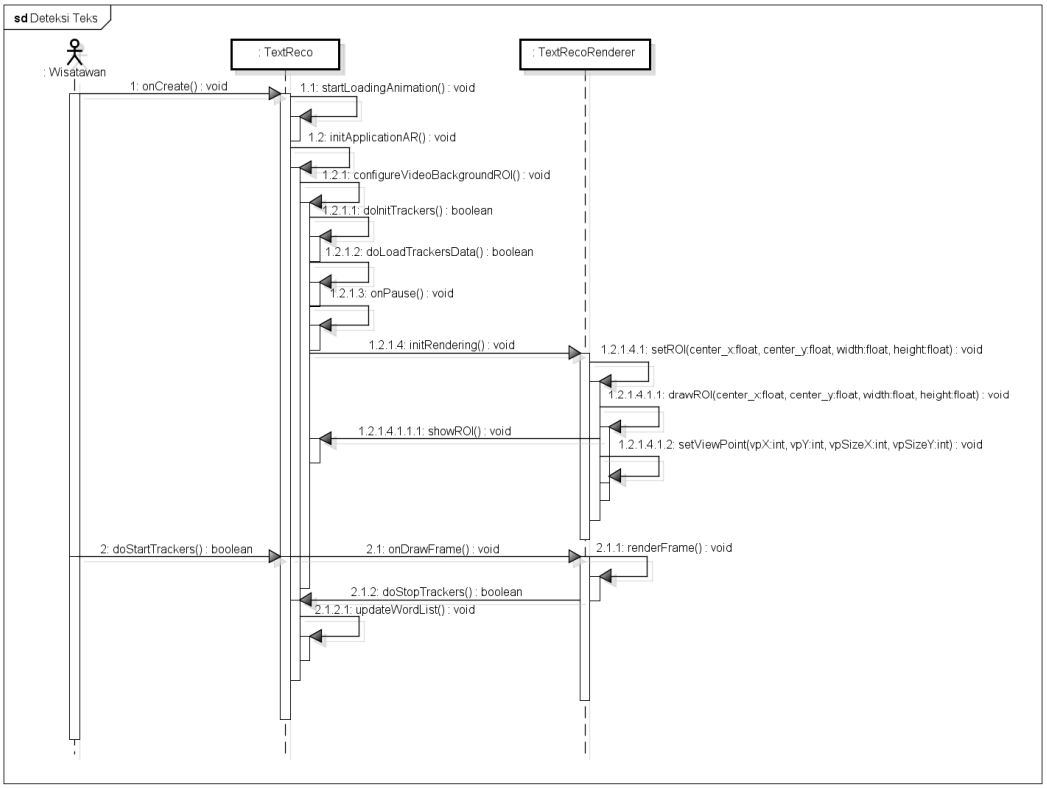
Deteksi Teks merupakan kegiatan yang dilakukan oleh wisatawan untuk menentukan teks yang akan diterjemahkan. Teks akan dipindai menggunakan kamera dengan teknologi *augmented reality.*

Tabel 3.10 Skenario *Use Case* Deteksi Teks

Use Case	**Deteksi Teks**
Related Requirement	Sistem menyediakan antarmuka kamera
Goal in Context	Sistem dapat melanjutkan ke proses terjemahan
Preconditions	Sistem menampilkan antarmuka kamera
Successful End Conditions	Deteksi teks berhasil
Failed End Conditions	Deteksi teks gagal
Primary Actors	Wisatawan
Secondary Actors	
Trigger	Sistem menampilkan halaman utama berupa tampilan kamera dengan area deteksi

Included Class	TextReco, TextRecoRenderer, TextTracker, TrackableResult, WordList, WordResult	
Main Flow	Step	Action
	1	Wisatawan menjalankan aplikasi
	2	Sistem menampilkan antarmuka kamera
	3	Wisatawan menyorotkan kamera pada teks yang ada pada papan petunjuk
	4	Sistem mendeteksi teks yang ada pada papan petunjuk
	5	Sistem mencocokan teks dengan wordlist yang ada
	6	Sistem menerjemahkan hasil deteksi teks ke bahasa tujuan
Extensions	Step	Branching Action
	6.1	Tidak melakukan penerjemahan

Deteksi teks diawali dengan menampilkan antarmuka kamera, kemudian wisatawan menyorotkan teks yang akan diterjemahkan. Sistem akan mendeteksi teks yang berada pada wilayah pelacakan lalu mencocokan dengan wordlist yang ada. Jika ada kecocokan maka diteruskan ke proses terjemahan. Gambar 3.29 menunjukan *sequence diagram* deteksi teks.



Gambar 3.29 *Sequence Diagram* Deteksi Teks

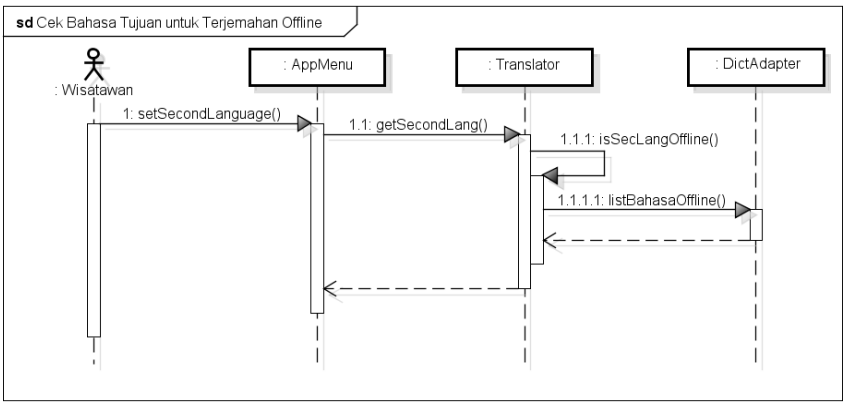
3.6.1.2.3Cek Bahasa Tujuan untuk Terjemahan Offline

Cek bahasa tujuan untuk terjemahan offline merupakan proses pengecekan untuk ketersediaan resource terjemahan offline yang ada dalam sistem.

Tabel 3.11 Skenario *Use Case* Cek Bahasa Tujuan untuk Terjemahan Offline

Use Case	**Cek bahasa tujuan untuk terjemahan offline**	
Related Requirement	Sistem menyediakan daftar bahasa terjemahan untuk terjemahan secara offline	
Goal in Context	Sistem dapat melanjutkan terjemahan secara offline sesuai dengan bahasa tujuan yang tersedia secara offline	
Preconditions	Sistem mengambil nilai dari bahasa tujuan yang dipilih wisatawan	
Successful End Conditions	Melanjutkan terjemahan secara offline	
Failed End Conditions	Melanjutkan terjemahan secara online	
Primary Actors	Sistem	
Secondary Actors		
Trigger	Sistem mencari bahasa terjemahan yang tersedia secara offline	
Included Class	TextRecoRenderer, Language	
Main Flow	Step	Action
	1	Wisatawan memilih bahasa awal
	2	Wisatawan memilih bahasa tujuan
	3	Sistem melakukan pengecekan resource
	4	Sistem melakukan penerjemahan offline
Extensions	Step	Branching Action

Pengecekan resource dilakukan sistem untuk mengetahui apakah bahasa tujuan yang dipilih wisatawan memiliki hasil terjemahan dalam database sistem. Jika bahasa tujuan cocok dengan salah satu daftar bahasa yang dimiliki database sistem, maka sistem akan melakukan penerjemahan secara offline. Gambar 3.30 merupakan *sequence diagram* cek bahasa tujuan untuk terjemahkan offline.



Gambar 3.30 Sequence Diagram Cek Bahasa Tujuan untuk Terjemahan Offline

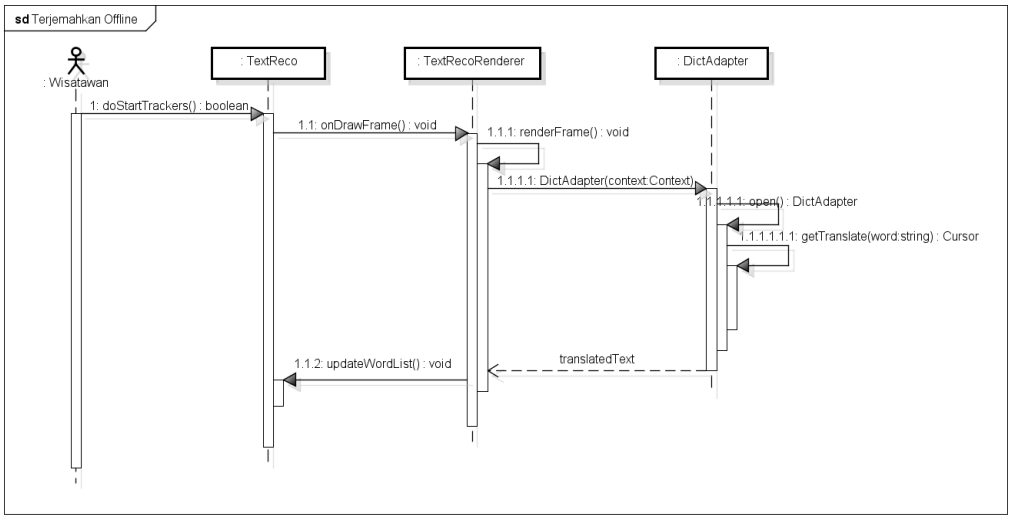
3.6.1.2.4Terjemahan Offline

Terjemhakan Teks merupakan proses penerjemahan yang dilakukan oleh sistem. Teks didapat dari hasil pendeteksian oleh kamera pada proses sebelumnya.

Tabel 3.12 Skenario *Use Case* Terjemahkan Teks

Use Case	**Terjemahkan Offline**	
Related Requirement	Sistem menyediakan resource terjemahan offline	
Goal in Context	Sistem mendapatkan hasil terjemahan	
Preconditions	Sistem mendapatkan teks yang telah dideteksi	
Successful End Conditions	Penerjemahan teks berhasil	
Failed End Conditions	penerjemahan teks gagal	
Primary Actors	Kamus offline	
Secondary Actors		
Trigger	Sistem memiliki teks yang telah dideteksi dari kamera	
Included Class	TextRecoRenderer, DataBaseHelper, DictAdapter, Language	
Main Flow	Step	Action
	1	Sistem memiliki nilai dari teks hasil deteksi
	2	Sistem membaca bahasa tujuan terjemahan dari konfigurasi
	3	Mengirim nilai dari teks yang akan diterjemahkan ke kamus offline
	4	Kamus offline menerima nilai yang dikirim oleh sistem
	5	Kamus offline menerjemahkan nilai dari hasil deteksi teks ke bahasa tujuan
	6	Kamus offline mengirim nilai dari hasil terjemahan ke sistem
	7	Sistem menerima hasil terjemahan
	8	Sistem menampilkan hasil terjemahan
Extensions	Step	Branching Action
	3.1	Menerjemahkan nilai dari hasil deteksi teks ke bahasa tujuan

Penerjemahan secara offline dilakukan dengan terlebih dahulu mencocokan ketersediaan resource. Jika bahasa tujuan tersedia maka sistem akan mengirim nilai dari deteksi teks ke database sistem. Database akan melakukan pencarian untuk bahasa tujuan. Jika kata ditemukan maka database akan mengirim hasil terjemahan ke sistem. Jika tidak maka database akan mengirim pesan bahwa kata yg dicari tidak ditemukan. Gambar 3.31 menunjukan *sequence diagram* terjmahkan offline.



Gambar 3.31 Sequence Diagram Terjemahkan Offline

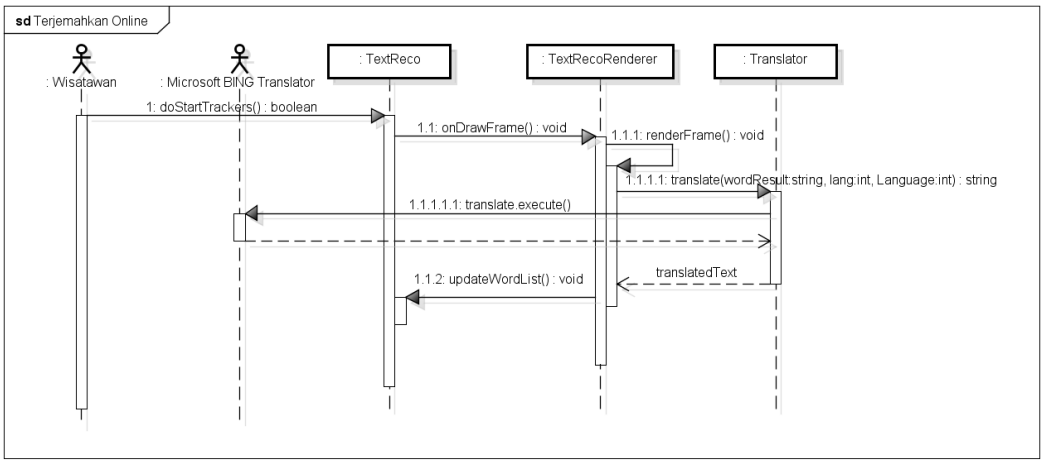
3.6.1.2.5Terjemahan Online

Terjemhakan Teks merupakan proses penerjemahan yang dilakukan oleh Microsoft Bing Translator.Teks didapat dari hasil pendeteksian oleh kamera pada proses sebelumnya.

Tabel 3.13 Skenario *Use Case* Terjemahkan Teks

Use Case	**Terjemahkan Online**	
Related Requirement	Sistem menyediakan teks yang telah dideteksi	
Goal in Context	Sistem mendapatkan hasil terjemahan	
Preconditions	Sistem mendapatkan teks yang telah dideteksi	
Successful End Conditions	Penerjemahan teks berhasil	
Failed End Conditions	penerjemahan teks gagal	
Primary Actors	Microsoft Bing Translator	
Secondary Actors		
Trigger	Sistem memiliki teks yang telah dideteksi dari kamera	
Included Class	TextRecoRenderer, Translator, Language, InternetConnection	
Main Flow	Step	Action
	1	Sistem memiliki nilai dari teks hasil deteksi
	2	Sistem membaca bahasa tujuan terjemahan dari konfigurasi
	3	Mengirim nilai dari teks yang akan diterjemahkan ke translator
	4	Translator menerima nilai yang dikirim oleh sistem
	5	Translator menerjemahkan nilai dari hasil deteksi teks ke bahasa tujuan
	6	Translator mengirim nilai dari hasil terjemahan ke sistem
	7	Sistem menerima hasil terjemahan
	8	Sistem menampilkan hasil terjemahan
Extensions	Step	Branching Action
	3.1	Menerjemahkan nilai dari hasil deteksi teks ke bahasa tujuan

Penerjemahan secara online dilakukan dengan bantuan dari Microsoft Bing Translator. Sistem mengirim nilai dari deteksi teks dan konfigurasi bahasa tujuan, lalu Microsoft Bing Translator akan menerjemahkannya. Setelah diterjemahkan, Microsoft Bing Translator mengirim kembali hasil terjemahan ke sistem. Gambar 3.32 menunjukan *sequence diagram* terjemahkan online



Gambar 3.32 Sequence Diagram Terjemahkan Online

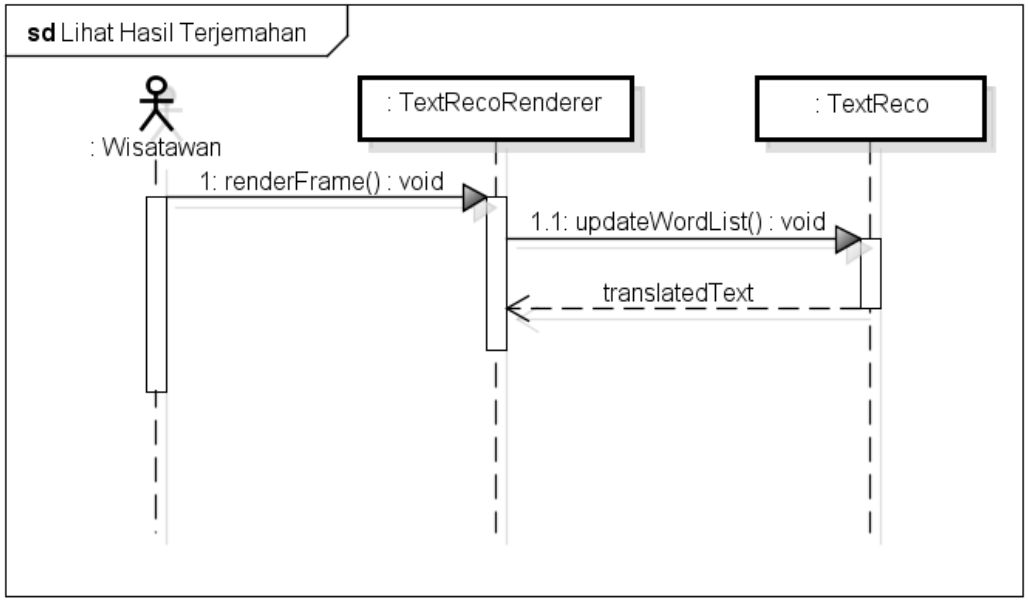
3.6.1.2.6Lihat Hasil Terjemahan

Lihat Hasil Terjemahan merupakan tampilan akhir dari proses penerjemahan. Microsoft Bing Translator mengirimkan hasil terjemahan, kemudian sistem menampilkannya secara *realtime*.

Tabel 3.14 Skenario Use Case Lihat Hasil Terjemahan

Use Case	**Lihat Hasil Terjemahan**	
Related Requirement	Sistem menyediakan teks yang telah diterjemahkan	
Goal in Context	Wisatawan melihat hasil terjemahan	
Preconditions	Sistem mendapatkan teks yang telah diterjemahkan	
Successful End Conditions	Menampilkan hasil terjemahan berhasil	
Failed End Conditions	Menampilkan hasil terjemahan gagal	
Primary Actors	Wisatawan	
Secondary Actors		
Trigger	Sistem memiliki teks yang telah diterjemahkan dari translator	
Included Class	TextReco, TextRecoRenderer	
Main Flow	Step	Action
	1	Sistem menampilkan hasil terjemahan
	2	Wistawan membaca hasil terjemahan
Extensions	Step	Branching Action

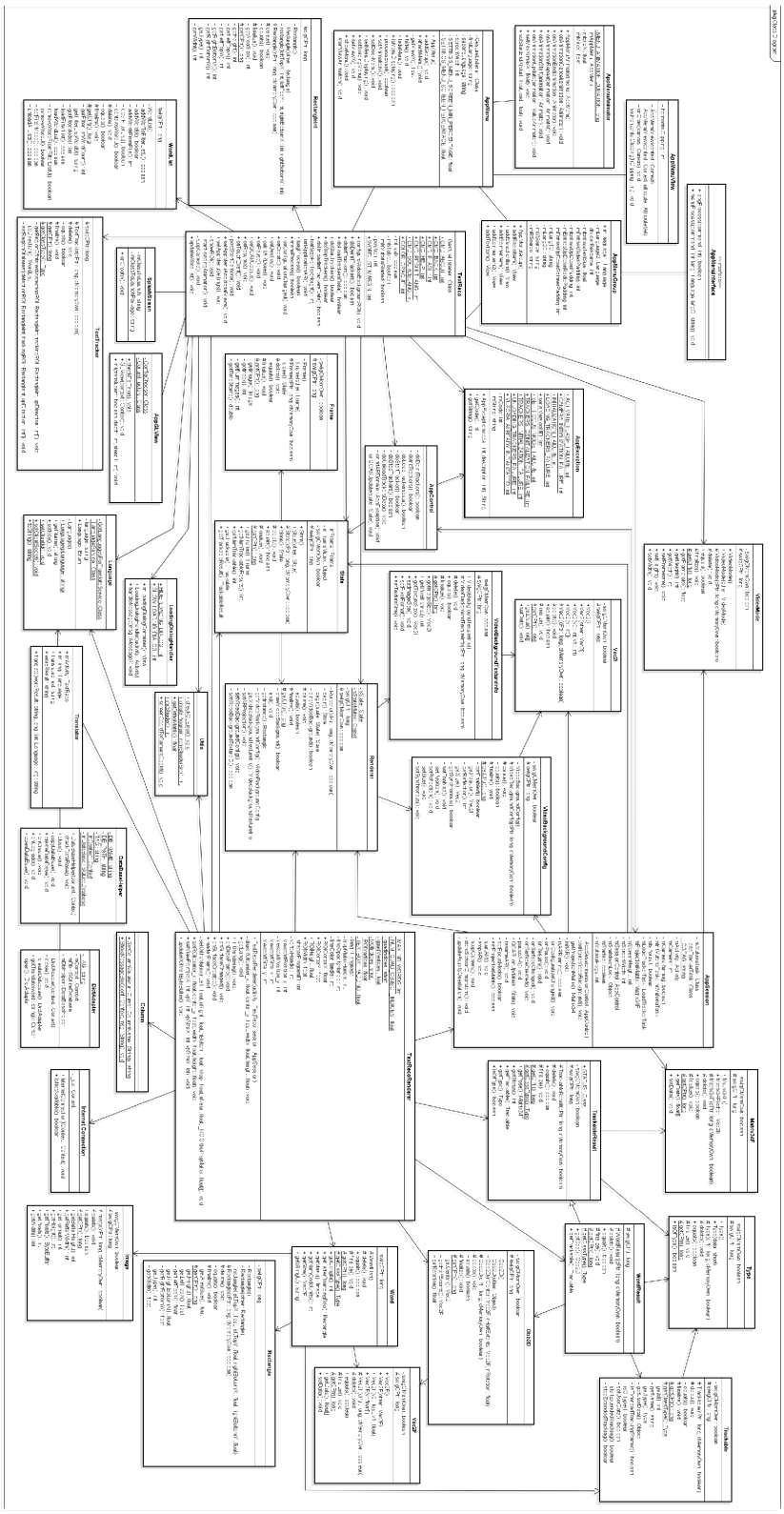
Sistem menerima hasil penerjemahan kata dari Microsoft Bing Translator atau dari database sistem kemudian menampilkannya pada layar. Wisatawan akan membaca hasil dari terjemahan kata langsung pada layar. Gambar 3.33 menunjukan *sequence diagram* lihat hasil terjemahan.



Gambar 3. 33 Sequence Diagram Lihat Hasil Terjemahan

3.6.2 *Class Diagram*

Class diagram menggambarkan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur ini meliputi atribut-atribut dan metode-metode yang ada pada masing-masing class. Hubungannya dari masing-masing class yang ada dapat dilihat pada Gambar 3.34.



Gambar 3. 34 Class Diagram

Dalam penelitian ini terdapat tiga jenis kelas yang digunakan. Ketiga kelas tersebut adalah kelas *Boundary*, kelas *Entity* dan kelas *Control* dapat dilihat pada tabel 3.15.

Tabel 3.15 Deskripsi Kelas

No	NamaKelas	Jenis Kelas	Keterangan
1	AppControl	Boundary	Merupakan *interface* yang diimplementasikan pada *activity* yang menggunakan AppSession
2	AppException	Entity	Digunakan untuk mengembalikan pesan kesalahan pada *activity* pada saat proses Vuforia
3	AppGLView	Control	Merupakan kelas pendukung Vuforia untuk mengkonfigurasi OpenGL
4	AppMenu	Boundary	Kelas yang mengatur menu konfigurasi
5	AppMenuAnimator	Control	Kelas yang mengatur animasi dari menu konfigurasi
6	AppMenuGroup	Boundary	Kelas yang mengatur konten dari menu konfigurasi
7	AppMenuInterface	Boundary	Merupakan *interface* yang diimplementasikan pada *activity* untuk mengambil fungsi yang sebelumnya digunakan pada instasiasi AppMenu
8	AppMenuView	Control	Merupakan kelas yang mengatur view pada menu konfigurasi
9	AppSession	Control	Merupakan kelas yang mengatur sesi dari proses Vuforia
10	Column	Control	Merupakan kelas pendukung untuk mengambil nilai dari kolom pada DataBaseHelper
11	DataBaseHelper	Control	Merupakan kelas yang menghubungkan aplikasi dengan file SQLite
12	DictAdapter	Control	Merupakan kelas adapter untuk menampung data dari SQLite
13	Frame	Entity	Frame adalah kumpulan dari satu *snapshot* kamera yang berbeda-beda
14	Image	Entity	Merupakan kelas dari lib Vuforia untuk mengembalikan objek CameraDevice
15	InternetConnection	Control	Kelas untuk menguji ada tidaknya koneksi internet
16	Language	Control	Merupakan kelas dari *Bing Translator* yang mengatur bahasa yang digunakan untuk proses terjemahan *online*
17	LoadingDialog Handler	Control	Kelas pendukung untuk dialog *loading*
18	Matrix34F	Entity	Matrix dengan ukuran 3x4 dengan tipe data *float*
19	Obb2D	Entity	Merupakan kelas yang digunakan untuk mewakili kotak pembatas berorientasi 2 dimensi

20	Rectangle	Entity	Merupakan kelas yang digunakan untuk membuat bangun segi enpat
21	RectangleInt	Entity	Merupakan kelas yang digunakan untuk menggambar ROI ke layar
22	Renderer	Control	Merupakan kelas yang digunakan untuk me-*render* objek yang ditangkap oleh kamera menjadi citra
23	SplashScreen	Boundary	Kelas yang menampilkan halaman pembuka dari aplikasi
24	State	Control	Penampakan konsisten pada keadaan *augmented reality* termasuk *frame* kamera dan semua yang dapat dideteksi
25	TextReco	Boundary	Merupakan kelas utama dari sistem ini yang berfungsi untuk mengatur *wordlist*, *tracker* dan juga tampilan hasil *render* ke layar
26	TextRecoRenderer	Control	Merupakan kelas yang berfungsi untuk melakukan *render* dan melakukan deteksi kata
27	TextTrackable Result	Entity	Merupakan kelas yang digunakan untuk menampung hasil dari pelacakan
28	TextTracker	Control	Merupakan kelas yang digunakan untuk mengenali dan mendeteksi teks (kata) dan memaparkan API untuk inisialisasi
29	Trackable	Entity	Kelas utama untuk seluruh objek yang dapat dikenali.
30	Translator	Control	Merupakan kelas yang memfasilitasi proses terjemahan secara *online* antara sistem dengan *Bing Translator*
31	Type	Control	Kelas pendukung dari sistem *type* internal Vuforia
32	Utils	Control	Kelas pendukung yang untuk mengakses dan konfigurasi OpenGL
33	Vec2F	Entity	*Vector* 2 dimensi dengan nilai folat
34	Vec2I	Entity	*Vector* 2 dimensi dengan nilai integer
35	VideoBackgroundConfig	Control	Kelas yang disediakan Vuforia untuk konfigurasi video *background*
36	VideoBackground TextureInfo	Control	Kelas yang disediakan Vuforia untuk informasi tekstur video *background*
37	VideoMode	Control	Mengimplementasikan akses ke kamera bawaan ponsel
38	Word	Entity	Merepresentasikan elemen tunggal dari kata
39	WordList	Entity	Daftar kata yang dapat dideteksi dan dikenali oleh TextTracker
40	WordResult	Entity	Hasil yang dapat dikenali untuk sebuah kata yang dideteksi

3.6.3 *Activity Diagram*

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Aktivity* diagram merupakan state diagram khusus, dimana sebagian besar state adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behavior* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

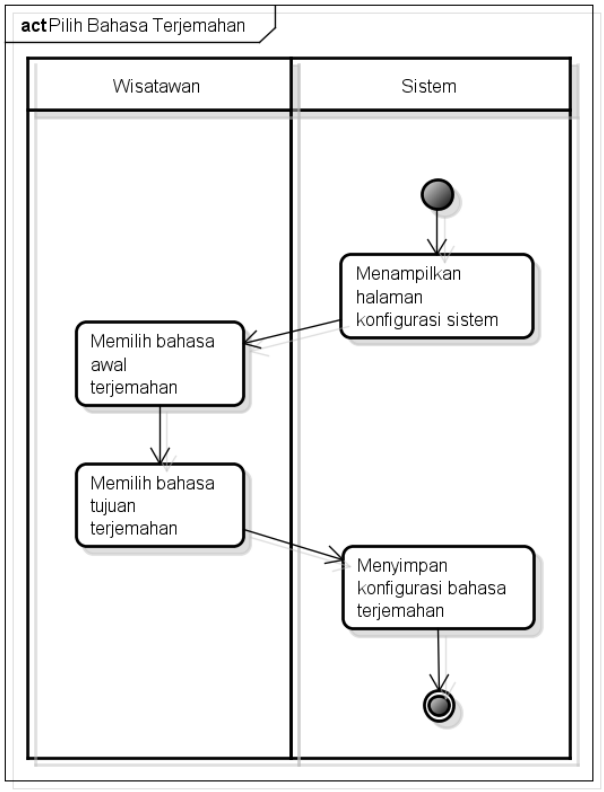
3.6.3.1 Aktivitas Pilih Bahasa Terjemahan

Aktivitas yang terdapat dalam pilih bahasa terjemahan adalah memilih bahasa tujuan dan bahasa asal yang akan diterjemahkan. Dengan melakukan pemilihan bahasa, sistem dapat melakukan proses penerjemahan karena bahasa asal dan tujuan telah diketahui. Tahapan-tahapan yang terjadi di dalam sistem adalah sebagai berikut.

a. Menampilkan halaman konfigurasi sistem, pada halaman ini terdapat menu untuk memilih bahasa tujuan dan bahasa asal.
b. Memilih bahasa awal terjemahan, aksi ini dilakukan oleh wisatawan untuk memilih bahasa awal pada halaman yang telah disediakan sistem.
c. Memilih bahasa tujuan terjemahan, aksi ini dilakukan oleh wisatawan untuk memilih bahasa tujuan pada halaman yang telah disediakan sistem.
d. Menyimpan konfigurasi bahasa terjemahan, setelah bahasa awal dan tujuan dipilih oleh wisatawan maka sistem akan menyimpan konfigurasi tersebut untuk digunakan pada proses penerjemahan.

Seluruh aktivitas pilih bahasa terjemahan dapat dilihat tahapannya pada diagram aktivitas pada gambar 3.35

.



Gambar 3.35 Diagram Aktivitas Pilih Bahasa Terjemahan

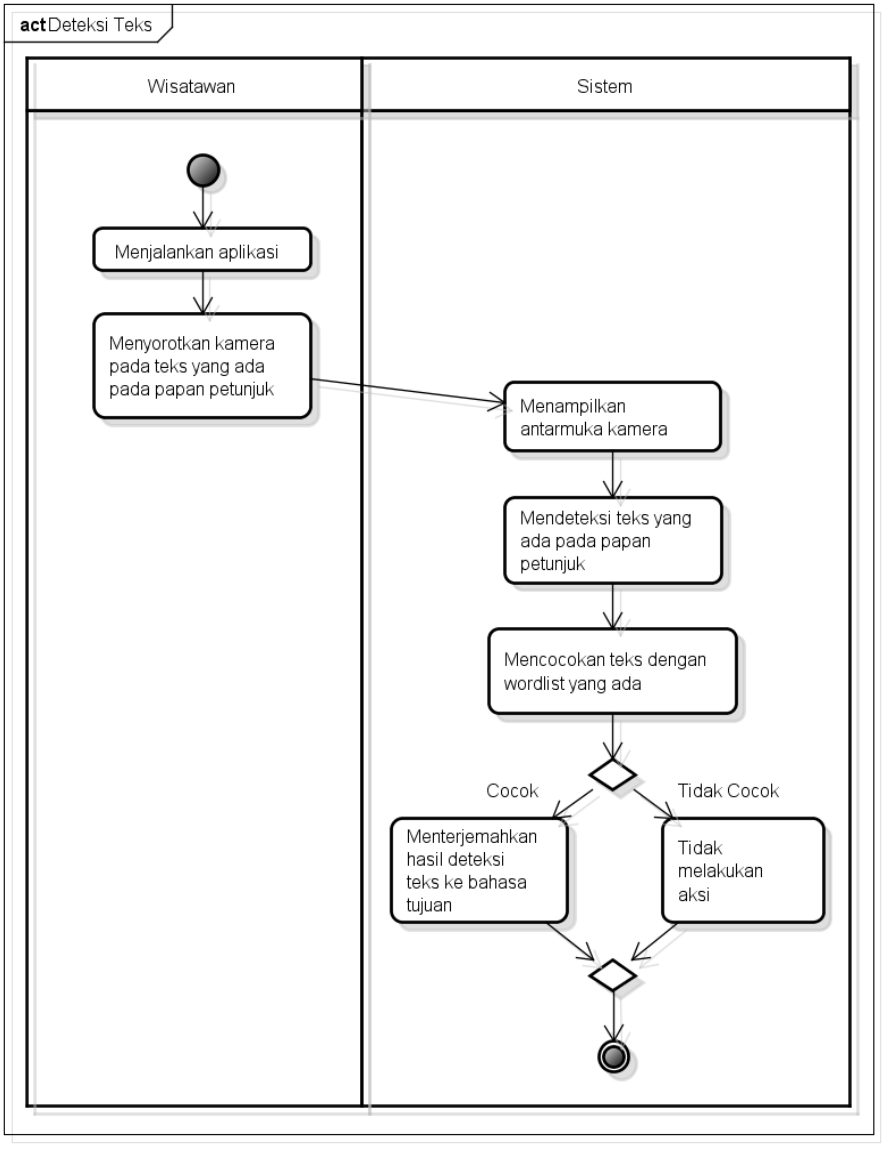
3.6.3.2 Aktivitas Deteksi Teks

Aktivitas yang terdapat dalam deteksi teks adalah mendeteksi teks dengan menyorotkan kamera pada teks yang akan diterjemahkan, kemudian sistem akan melakukan pencocokan dengan wordlist yang tersedia. Tahapan-tahapan yang terjadi di dalam sistem adalah sebagai berikut.

a. Menjalanjan aplikasi, aksi ini dilakukan oleh wisatawan untuk mengaktifkan kamera.
b. Menampilkan antarmuka kamera, sistem menampilkan atarmuka kamera setelah wisatawan menjalankan aplikasi.
c. Menyorotkan kamera pada teks yang ada pada papan petunjuk, wisatawan diharuskan menyorotkan kamera pada teks hingga teks berada dalam wilayah pendeteksian kamera.
d. Mendeteksi teks yang ada pada papan petunjuk, sistem melakukan pendeteksian terhadap teks yang telah terlacak.

e. Mencocokan teks dengan wordlist yang ada, pencocokan dilakukan dengan wordlist yang tersedia pada sistem.
f. Menerjmahkan hasil deteksi teks ke bahasa tujuan, jika teks yang di deteksi cocok dengan wordlist maka sistem akan melanjutkan proses ke terjemahan kata.
g. Tidak melakukan aksi, jika teks yang di deteksi tidak cocok dengan wordlist maka sistem tidak akan melakukan proses penerjemahan kata.

Seluruh aktivitas pilih bahasa terjemahan dapat dilihat tahapannya pada diagram aktivitas pada gambar 3.36.



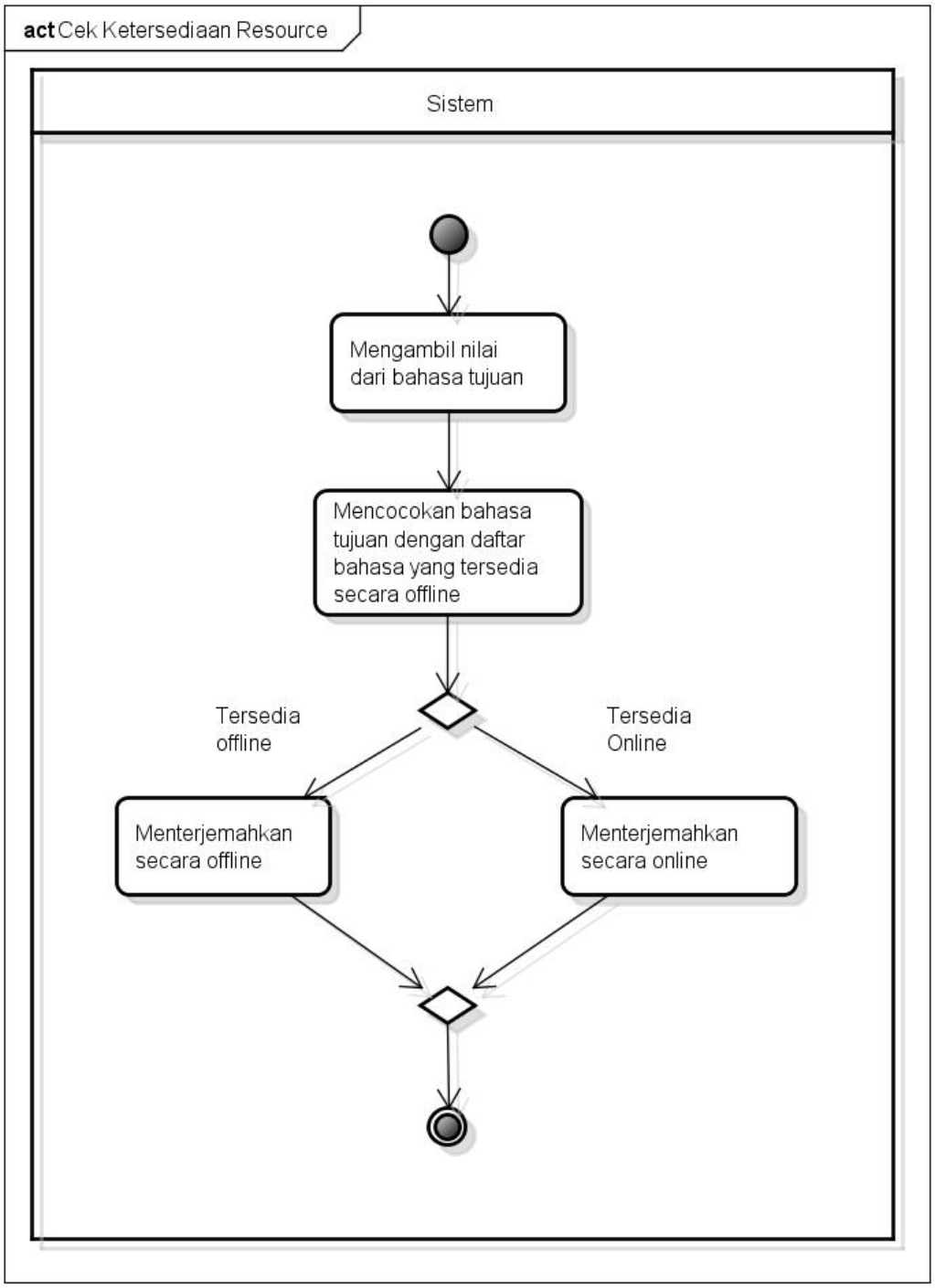
Gambar 3.36 Diagram Aktvitas Deteksi Teks

3.6.3.3 Aktivitas Cek Bahasa Tujuan untuk Terjemahan Offline

Aktivitas cek bahasa tujuan untuk terjemahan offline sepenuhnya dilakukan oleh sistem, dimana sistem akan mencocokan bahasa tujuan dengan bahasa yang disediakan sistem untuk penerjemahan. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

a. Mengambil nilai dari bahasa tujuan, nilai tersebut akan digunakan untuk mencocokan bahasa.
b. Mencocokan bahasa tujuan dengan daftar bahasa yang tersedia secara offline, sistem mencocokan bahasa tujuan dengan bahasa yang disediakan sistem.
c. Menerjemahkan secara offline, jika bahasa tujuan cocok dengan salah satu daftar bahasa yang ada pada database sistem maka sistem akan melakukan penerjemahan secara offline dimana hasil terjemahan telah tersedia dalam sistem.
d. Menerjemahkan secara online, jika bahasa tujuan tidak cocok dengan salah satu daftar bahasa yang ada pada sistem maka sistem akan melakukan penerjemahan secara online dimana hasil terjemahan akan diambil dari Microsoft Bing Translator.

Seluruh aktivitas pilih bahasa terjemahan dapat dilihat tahapannya pada diagram aktivitas pada gambar 3.37.



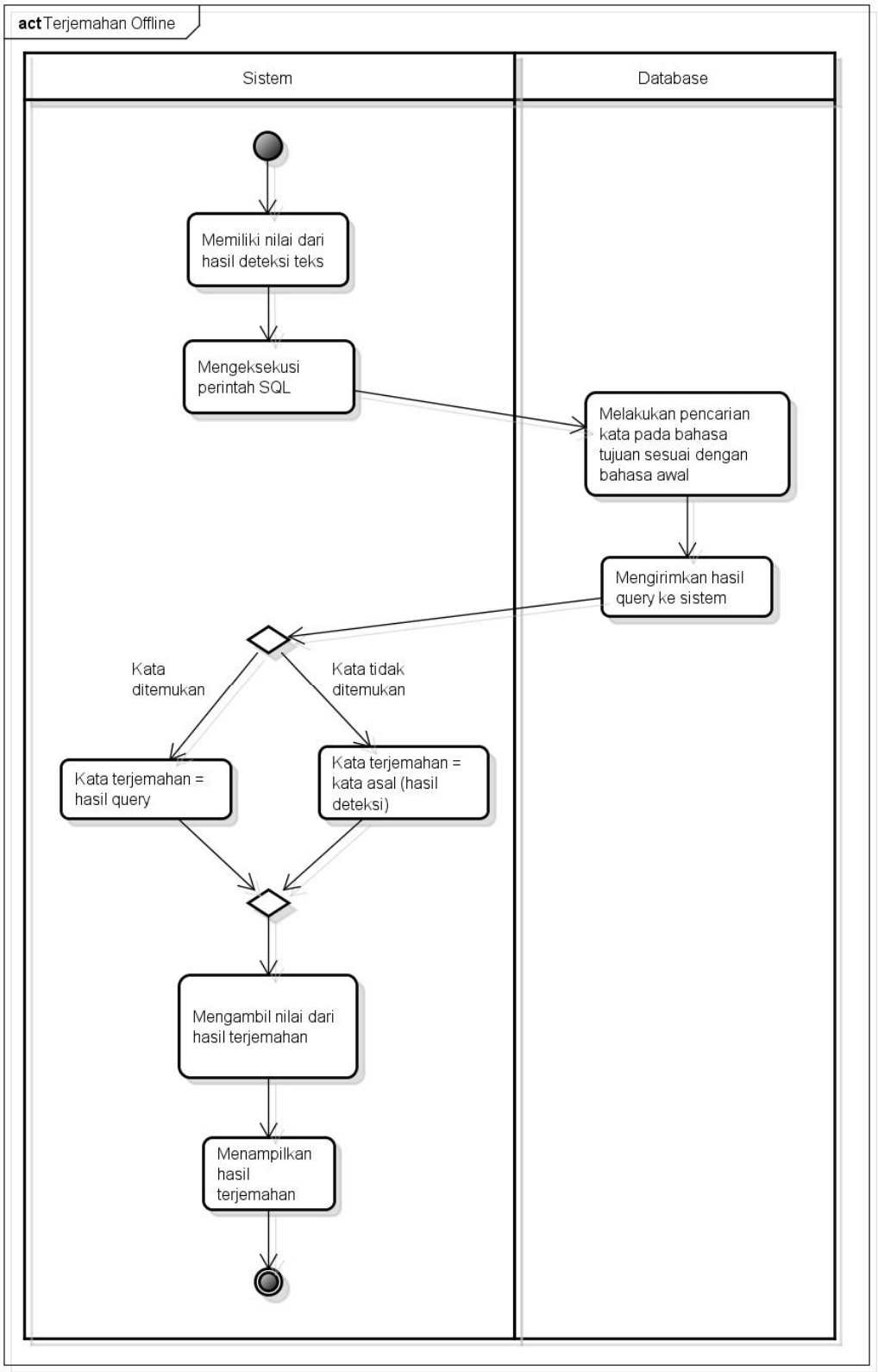
Gambar 3.37 Diagram Aktivitas Cek Bahasa Tujuan untuk Terjemahan Offline

3.6.3.4 Aktivitas Terjemahkan Offline

Aktivitas terjemahkan offline adalah menerjemahkan kata yang telah dideteksi tanpa harus terkoneksi ke internet. Sistem akan melakukan penerjemahan dengan mengambil hasil dari daftar bahasa yang terdapat database sistem dimana daftar bahasa harus cocok dengan bahasa tujuan saat dideteksi. Tahapan-tahapan yang terjadi di dalam sistem adalah sebagai berikut.

a. Memiliki nilai dari hasil deteksi teks, sistem telah memiliki nilai dari hasi deteksi teks.
b. Mengeksekusi perintah SQL, sistem melakukan eksekusi peritah SQL untuk pencarian kata.
c. Melakukan pencarian kata pada bahasa tujuan sesuai dengan bahasa awal, pencarian dilakukan dalam database sistem.
d. Mengirimkan hasil query ke sistem, jika kata ditemukan maka database mengirimkan hasil query ke sistem untuk ditampilkan.
e. Kata terjemahan = hasil query, jika kata ditemukan maka kata terjemahan adalah hasil query dari pencarian di database.
f. Mengirimkan pesan kata tidak ditemukan, jika kata tidak ditemukan maka database akan mengirimkan pesan bahwa kata tidak ditemukan.
g. Kata terjemahan = kata asal (hasil deteksi), jika kata tidak ditemukan maka kata terjemahan adalah kata asal, database mengembalikan kata asal untuk menandakan bahwa kata asal tersebut tidak ditemukan terjemahannya.
h. Mengirim nilai dari hasil terjemahan ke sistem, database mengirim semua hasil penerjemahan baik yang ditemukan atau tidak ditemukan.
i. Menerima hasil terjemahan, sistem menerima hasil terjemahan dari database.
j. Menampilkan hasil terjemahan, sistem menampilkan hasil terjemahan dari database.

Seluruh aktivitas pilih bahasa terjemahan dapat dilihat tahapannya pada diagram aktivitas pada gambar 3.38.



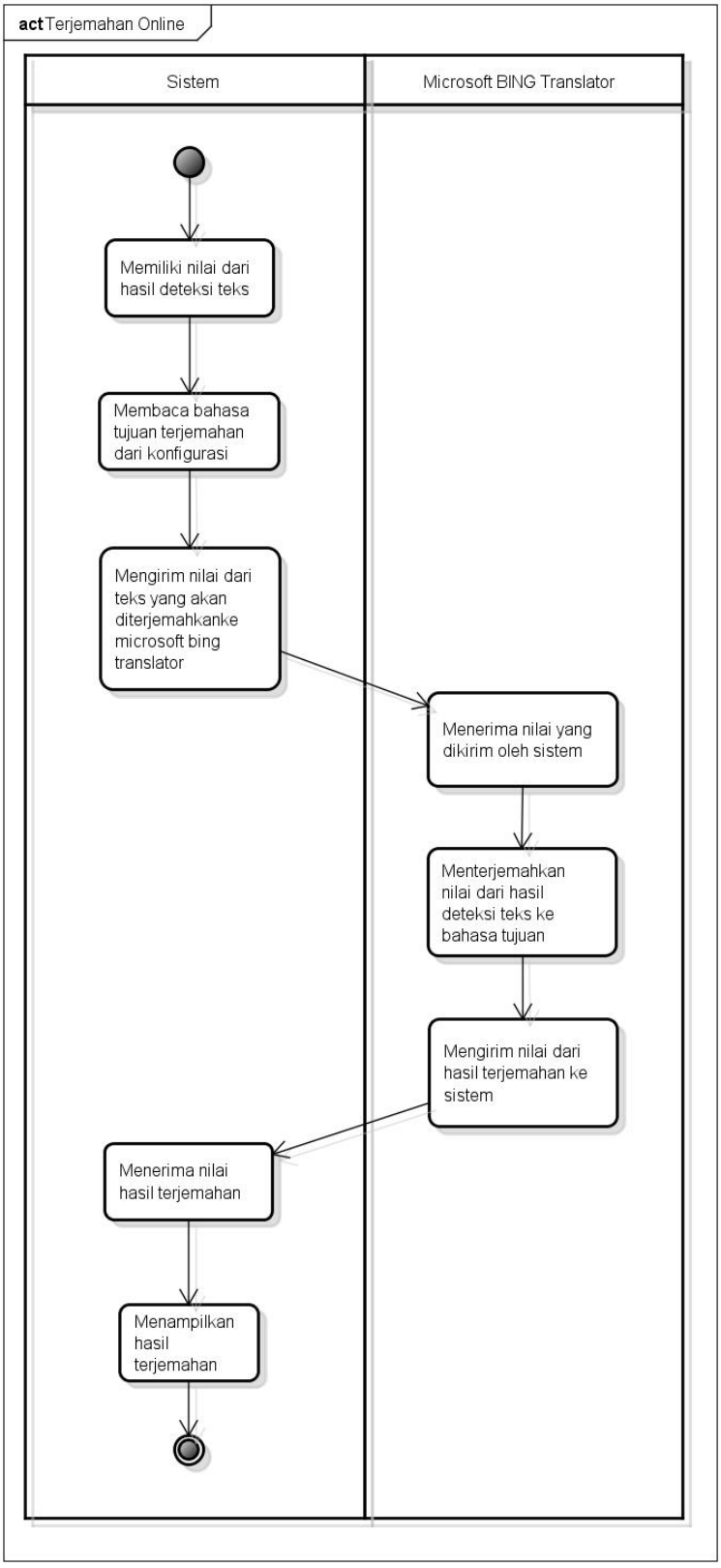
Gambar 3.38 Aktivitas Terjemahkan Offline

3.6.3.5 Aktivitas Terjemahkan Online

Aktivitas terjemahkan online adalah menerjemahkan kata yang telah dideteksi dengan kondisi harus terkoneksi ke internet untuk melakukan penerjemahan dengan Microsoft Bing Translator. Sistem akan melakukan penerjemahan dengan mengambil hasil dari Microsoft Bing Translator secara online. Tahapan-tahapan yang terjadi di dalam sistem adalah sebagai berikut.

a. Memiliki nilai dari hasil deteksi teks, sistem telah memiliki nilai dari hasi deteksi teks.
b. Membaca bahasa tujuan terjemahan dari konfigurasi, sistem membaca bahasa tujuan dan mendapatkan nilai bahasa tujuan tersebut.
c. Mengirim nilai dari teks yang akan diterjemahkan ke Microsoft Bing tanslator, sistem mengirim nilai ke Microsoft Bing Translator.
d. Menerima nilai yang dikirm oleh sistem, Microsoft Bing Translator menerima nilai yang dikirim oleh sistem.
e. Menerjemahkan nilai dari hasil deteksi teks ke bahasa tujuan, Microsoft Bing Translator menerjemahkan nilai yang dari hasil deteksi ke bahasa tujuan.
f. Mengirim nilai dari hasil terjemahan ke sistem, Microsoft Bing Translator mengirim nilai hasil terjemahan ke sistem untuk ditampilkan.
g. Menerima nilai hasil terjemahan, sistem menerima hasil terjemahan dari Microsoft Bing Translator.
h. Menampilkan hasil terjemahan, sistem menampilkan hasil terjemahan dari Microsoft Bing Translator.

Seluruh aktivitas pilih bahasa terjemahan dapat dilihat tahapannya pada diagram aktivitas pada gambar 3.39.



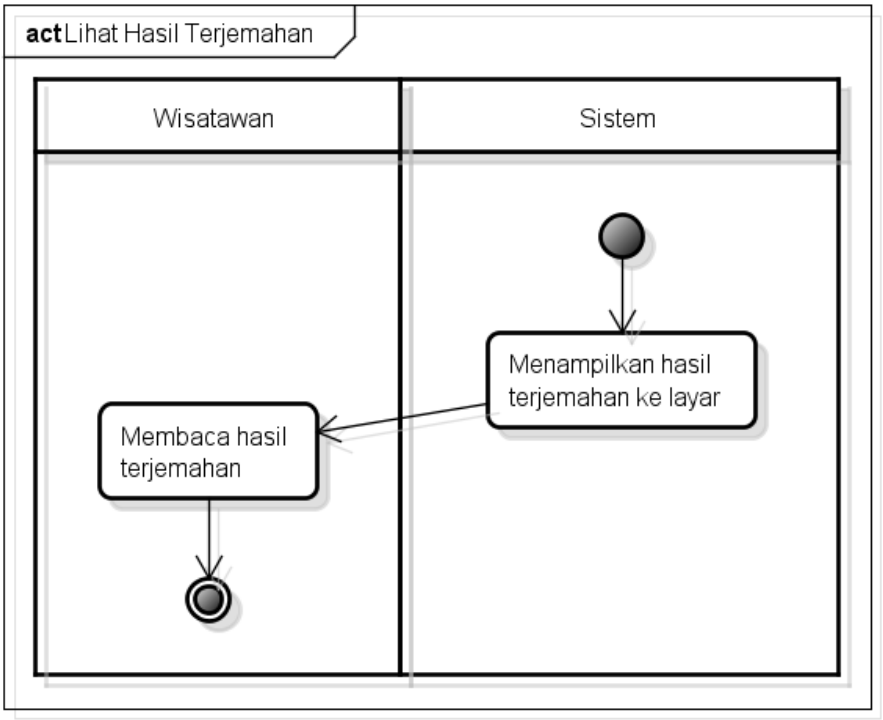
Gambar 3. 39 Diagram Aktivitas Terjemahkan Online

3.6.3.6 Aktivitas Lihat Hasil Terjemahan

Aktivitas lihat hasil terjemahan adalah menampilkan hasil terjemahan yang telah diterjemahkan secara offline maupun online dan akan dilihan oleh wisatawan. Tahapan-tahapan yang terjadi di dalam sistem adalah sebagai berikut.

a. Menampilkan hasil terjemahan ke layar, sistem akan menampilkan hasil terjemahan ke layar.
b. Membaca hasil terjemahan, wisatawan akan membaca hasil terjemahan yang dilampilkan di layar.

Seluruh aktivitas pilih bahasa terjemahan dapat dilihat tahapannya pada diagram aktivitas pada gambar 3.40.



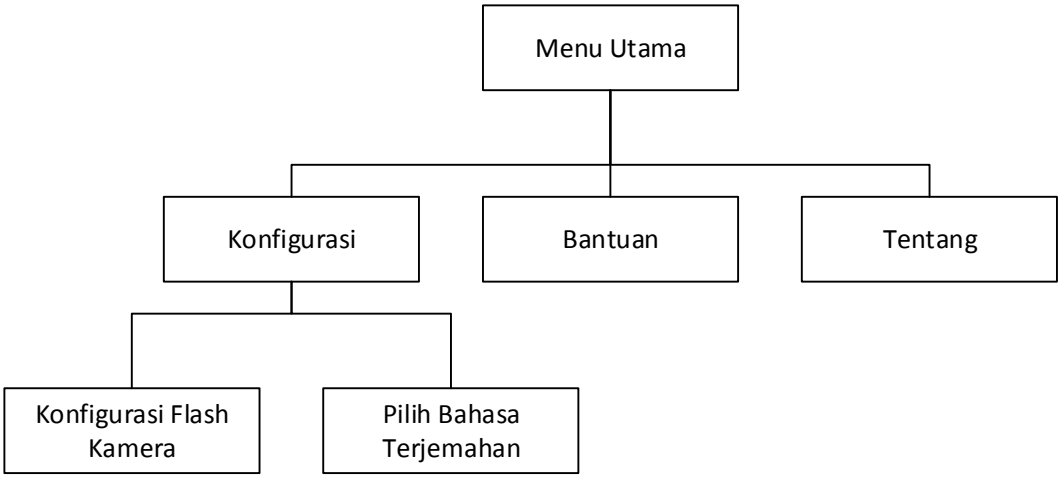
Gambar 3.40 Diagram Aktivitas Lihat Hasil Terjemahan

3.7 Perancangan Sistem

Perancangan sistem bertujuan untuk menspesifikasikan aspek-aspek teknik yang menjadi solusi dalam perencanaan. Pada tahap ini perancangan akan didefinisikan secara detail untuk mengatasi masalah-masalah yang lebih teknis, berkaitan dengan kegiatan implementasi seperti perancangan menu, perancangan antarmuka, dan jaringan semantik.

3.7.1 Perancangan Struktur Menu

Perancangan struktur menu merupakan gambaran alur pemakaian aplikasi. Terdapat 5 menu yang terdiri dari menu utama yaitu kamera, konfigurasi flash kamera, pilih bahasa terjemahan, bantuan dan tentang. Perancangan struktur menu dari aplikasi ini dapat dilihat pada gambar 3.41.



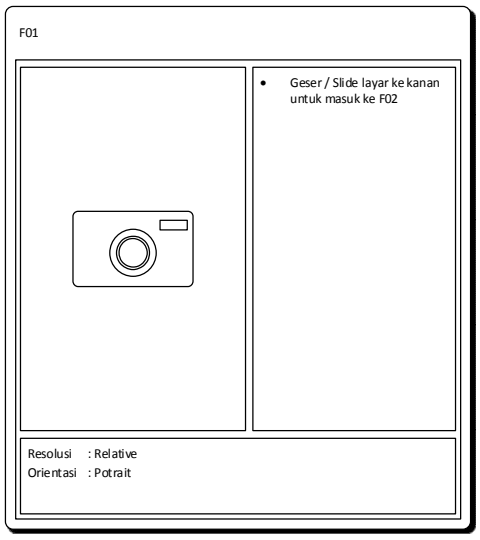
Gambar 3.41 Struktur Menu

3.7.2 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga mempermudah dalam mengimplementasikan aplikasi serta memudahkan dalam pembuatan aplikasi.

3.7.2.1 Perancangan Antarmuka Menu Utama

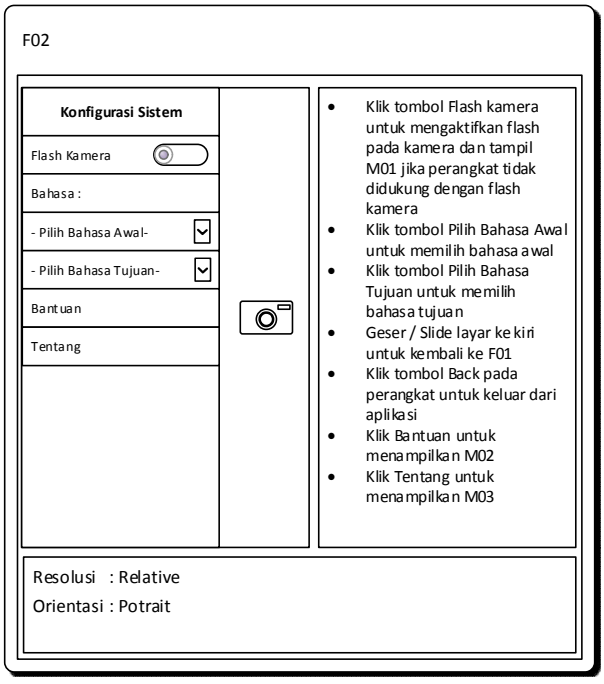
Menu utama merupakan halaman utama yang akan diakses oleh wisatawan, dengan antarmuka langsung pada kamera. Adapun perancangan antarmuka untuk menu utama yang dibangun adalah seperti gambar 3.42.



Gambar 3.42 Perancangan Antarmuka Menu Utama

3.7.2.2 Perancangan Antarmuka Menu Konfigurasi Sistem

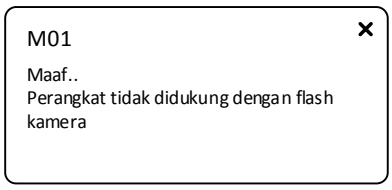
Halaman menu konfigurasi merupakan halaman dimana wisatawan dapat memilih menu konfigurasi seperti menyalakan flash kamera dan memilih bahasa terjemahan. Adapun perancangan antarmuka untuk menu konfigurasi sistem yang dibangun adalah seperti gambar 3.43



Gambar 3.43 Perancangan Antarmuka Menu Konfigurasi Sistem

3.7.2.3 Perancangan Antarmuka Pesan Flash Kamera

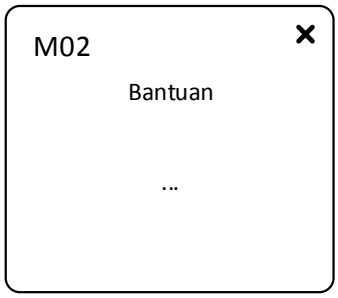
Pesan flash kamera akan muncul apabia pada perangkat tidak didukung dengan flash kamera sehingga fungsi ini tidak dapat dijalankan. Adapun perancangan antarmuka untuk menu utama yang dibangun adalah seperti gambar 3.44.



Gambar 3.44 Perancangan Antarmuka Pesan Flash Kamera

3.7.2.4 Perancangan Antarmuka Pesan Bantuan

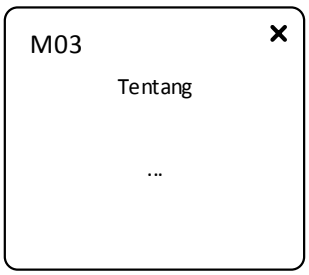
Pesan bantuan akan muncul apabia wisatawan memilih menu bantuan pada halaman konfigurasi. Adapun perancangan antarmuka untuk pesan bantuan yang dibangun adalah seperti gambar 3.45.



Gambar 3.45 Perancangan Antarmuka Bantuan

3.7.2.5 Perancangan Antarmuka Pesan Tentang

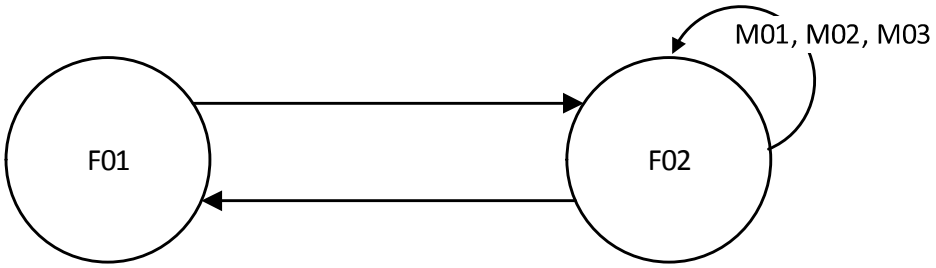
Pesan tentang akan muncul apabia wisatawan memilih menu tentang pada halaman konfigurasi. Adapun perancangan antarmuka untuk pesan tentang yang dibangun adalah seperti gambar 3.46.



Gambar 3.46 Perancangan Antarmuka Pesan Tentang

3.7.3 Jaringan Semantik

Jaringan semantik merupakan gambaran tentang relasi dari masing-masing halaman yang bisa diakses oleh pengguna. Jaringan semantik ini, digunakan sebagai panduan penggunaan menu-menu dalam sebuah aplikasi. Perancangan semantik dapat dilihat pada Gambar 3.47.



Gambar 3.47 Jaringan Semantik