
**TUGAS AKHIR – KI091391**

# Implementasi Pencocokan Perintah Suara Pada Aplikasi FaceTube

ERFAN AHMADIYONO
NRP. 5109 100 046

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Adhatus Solichah A., S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

*[Halaman ini sengaja dikosongkan]*

TUGAS AKHIR – KI091391

# Implementasi Pencocokan Perintah Suara Pada Aplikasi FaceTube

ERFAN AHMADIYONO
NRP. 5109 100 046

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Adhatus Solichah A., S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

*[Halaman ini sengaja dikosongkan]*

FINAL PROJECT – KI091391

# Implementation of Voice Command Matching in FaceTube Application

ERFAN AHMADIYONO
NRP. 5109 100 046

Supervisor I
Dwi Sunaryono, S.Kom., M.Kom.

Supervisor II
Adhatus Solichah A., S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

*[Halaman ini sengaja dikosongkan]*

## LEMBAR PENGESAHAN

**Implementasi Pencocokan Perintah Suara Pada Aplikasi FaceTube**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
**ERFAN AHMADIYONO**
NRP. 5109 100 046

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Dwi Sunaryono, S.Kom., M.Kom. .....
   NIP. 19720528 1997 02 1 001 (Pembimbing I)

2. Adhatus Solichah A., S.Kom., M.Sc. .....
   NIP. 19850826 2015 04 2 002 (Pembimbing II)

SURABAYA
AGUSTUS, 2016

*[Halaman ini sengaja dikosongkan]*

**Implementasi Pencocokan Perintah Suara Pada Aplikasi FaceTube**


| | |
|---|---|
| Nama Mahasiswa | : ERFAN AHMADIYONO |
| NRP | : 5109 100 046 |
| Jurusan | : Teknik Informatika, FTIF-ITS |
| Dosen Pembimbing I | : Dwi Sunaryono, S.Kom., M.Kom. |
| Dosen Pembimbing II | : Adhatus Solichah A., S.Kom., M.Sc. |


## ABSTRAK


*Publikasi berupa media audio-visual saat ini sedang marak digunakan oleh masyarakat di era teknologi yang makin cepat berkembang. Hal tersebut tentu berdampak pada perkembangan budaya serta kebiasaan penduduknya yang makin mengarah pada keinginan agar segala sesuatunya dapat direkam dan dipublikasikan dalam bentuk video. Peristiwa-peristiwa informatif seperti kecelakaan lalu lintas, bencana alam, kriminalitas, atau bahkan perkara lucu dan aneh seringkali tidak luput dari sorotan kamera. Meski teknik perekaman video saat ini makin mudah digunakan pada berbagai macam gadget canggih, namun tidak semua perangkat tersebut bekerja secara efisien. Hal tersebut dinilai karena masih belum ditemukan adanya aplikasi yang dapat melakukan penyebarluasan konten dan pemberian tautan referensi yang bekerja secara otomatis, sehingga pemublikasian media audio-visual ini masih tergolong lambat khususnya dalam kondisi darurat yang membutuhkan penyebarluasan secara cepat.*

*Tugas Akhir ini kemudian muncul dengan dibuatnya aplikasi bernama FaceTube dalam upaya untuk memberikan jalan keluar bagi permasalahan di atas. Aplikasi pada perangkat Android ini merupakan sebuah aplikasi perekam video yang memiliki fitur utama yaitu dapat melakukan perekaman dengan menggunakan perintah suara. FaceTube kemudian secara otomatis akan memberikan tautan referensi dan mengunggahnya ke situs web*

*berbagi video YouTube. Aplikasi ini juga akan melakukan posting status berupa link video yang akan diunggah pada akun Facebook pengguna. FaceTube sendiri menggunakan algoritma pencocokan string yaitu Levensthein Distance dan Soundex untuk mencocokkan perintah suara yang diterima aplikasi.*

*Dengan diciptakannya aplikasi FaceTube ini, publikasi video ke media sosial akan menjadi semakin hemat waktu dan tenaga karena pengguna tidak perlu melakukannya secara manual. Pengimplementasian kedua algoritma pencocokan string pada aplikasi juga dapat mengurangi kesalahan pendeteksian perintah suara untuk melakukan perekaman. Selain itu, adanya fitur pemberian tag reference berupa lokasi dan waktu perekaman menjadikan media audio-visual pada aplikasi FaceTube sebagai sumber informasi yang lebih akurat dari yang lainnya. Dengan demikian, pemilihan media publikasi melalui video dapat menjadi lebih efektif dan efisien.*

***Kata kunci: video, algoritma pencocokan string, perintah suara, FaceTube, Android.***

**Implementation of Voice Command Matching in FaceTube Application**


| | |
|---|---|
| Student's Name | : ERFAN AHMADIYONO |
| NRP | : 5109 100 046 |
| Major of Department | : Informatics, FTIF-ITS |
| Supervisor I | : Dwi Sunaryono, S.Kom., M.Kom. |
| Supervisor II | : Adhatus Solichah A., S.Kom., M.Sc. |


## ABSTRACT


*Audio visual base publication currently used by a worldwide community in the era of technology that became more rapidly developed. Thus, having an impact on the development of the culture and habits of its people itself that continues to lead to bigger needed for everything that can be recorded and published in the form of video. Informative events such as traffic accidents, natural disasters, crimes, or even amusing and strange matters often absent from camera lens. Although video recording technique now become easier to use on various sophisticated gadgets, not all of those devices can operated efficiently. This is presumably because there is no any capable application yet to publicize the video content and tag reference link that work automatically, so that audiovisual media publicized is still relatively slow especially during emergencies that need quick publicizing.*

*This Final Task then comes up with an application named FaceTube in order to give a way out of the problem above. This Android base application is a video recording application that has the main feature of voice command recording. FaceTube will automatically make reference tagging then upload it to the video sharing website like YouTube. This application will also do a status posting of the video link on users Facebook account. FaceTube uses string matching algorithm of Levensthein Distance and Soundex to match the voice command that application receive.*

*By creating this FaceTube application, publicizing video to social media will become even more time and energy saving because users do not have to do it manually. The implementation of both string matching algorithm for the application can also reduce errors in the voice command detection while the user does the recording. In addition, tag reference feature in the form of location and time record makes the audiovisual media such FaceTube as an accurate information source than the others. Therefore, choosing media publication through a video can be more effective and efficient.*

***Keywords: video, string matching algorithm, voice commands, FaceTube, Android.***

# KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT yang senantiasa melancarkan dan memudahkan segala urusan terutama dalam penyelesaian Tugas Akhir bidang keahlian Rekayasa Perangkat Lunak yang berjudul: “Implementasi Pencocokan Perintah Suara Pada Aplikasi FaceTube”.

Atas selesainya Tugas Akhir ini, banyak pihak yang secara langsung maupun tidak langsung telah berkontribusi dalam membantu penyelesaian. Kepada pihak-pihak tersebut, penulis ingin menyampaikan penghormatan dan terima kasih. Diantaranya disebutkan sebagai berikut.

1. Syukur Alhamdulillah kepada Allah SWT atas limpahan Rahmat dan pertolongannya dalam penyelesaian Tugas Akhir ini.
2. Kakak kedua penulis, Mas Laks yang selalu membantu dan mendukung penulis baik secara teknis maupun non teknis.
3. Orang tua saya yang selalu memberikan doa dan dukungannya.
4. Bapak Dwi Sunaryono selaku dosen pembimbing Tugas Akhir pertama dan yang telah memberikan kemudahan, semangat, arahan dan bimbingan dengan sabar hingga tugas akhir ini tuntas.
5. Ibu Licha selaku dosen pembimbing Tugas Akhir kedua yang juga telah memberikan waktu dan tenaga untuk membimbing saya dalam pengerjaan tugas akhir ini.
6. Bapak Radityo Anggoro selaku Pembina Tugas Akhir yang selama ini memberikan kemudahan, semangat serta bimbingannya kepada penulis.
7. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
8. Mas Yudi dan Staf TU Teknik Informatika ITS yang senantiasa memudahkan segala urusan penulis di jurusan.

9. Teman saya angkatan 2009 yang memberikan pengalaman berharga selama berada di Teknik Informatika ITS.
10. Dimas dan sahabat-sahabat saya diluar sana yang terus memberikan semangat dan kepercayaan diri untuk menyelesaikan Tugas Akhir ini.
11. Pihak-pihak lain yang tidak sengaja terlewat dan tidak dapat penulis sebutkan satu-persatu.

Penulis telah berusaha maksimal dalam penyusunan tugas akhir ini. Harapannya, Tugas Akhir ini dapat bermanfaat bagi kemajuan teknologi informasi dan dunia pendidikan di Indonesia. Penulis juga ingin menyampaikan permohonan maaf apabila terdapat kekurangan, kesalahan maupun kelalaian di dalamnya. Kritik dan saran yang membangun sangat diharapkan sebagai perbaikan selanjutnya.

Surabaya, Agustus 2016

Erfan Ahmadiyono

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR KODE SUMBER

*[Halaman ini sengaja dikosongkan]*

# BAB 1
# PENDAHULUAN

Pada bab ini dipaparkan garis besar Tugas Akhir. Garis besar Tugas Akhir ini meliputi latar belakang, tujuan dan manfaat pembuatan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

## 1.1. Latar Belakang

Publikasi berupa media *audio-visual* saat ini sedang marak digunakan oleh masyarakat di era teknologi yang makin cepat berkembang. Hal tersebut tentu berdampak pada perkembangan budaya serta kebiasaan penduduknya yang makin mengarah pada keinginan agar segala sesuatunya dapat direkam dan dipublikasikan dalam bentuk video. Peristiwa-peristiwa informatif seperti kecelakaan lalu lintas, bencana alam, kriminalitas, atau bahkan perkara lucu dan aneh seringkali tidak luput dari sorotan kamera.

Peningkatan kebutuhan masyarakat mengenai berita berupa video ini sempat memicu salah satu stasiun televisi swasta untuk berkreasi mengadakan acara dengan konsep masyarakat sebagai jurnalisnya. Sebagai contohnya adalah acara *Citizen Jurnalism* dalam *Wide Shot* di Metro TV. Dalam acara tersebut masyarakat dapat mengirimkan video berita hasil rekamannya kepada pihak terkait dan jika terpilih maka akan ditayangkan pada layar televisi. Hal ini banyak mendapat respon baik dari masyarakat, karena masyarakat Indonesia saat ini menginginkan berita yang jelas sumbernya terutama disertai dengan adanya video kejadian.

Meski teknik perekaman video saat ini makin mudah digunakan pada berbagai macam *gadget* canggih, namun tidak semua perangkat tersebut bekerja secara efisien. Hal tersebut dinilai karena masih belum ditemukan adanya aplikasi yang dapat melakukan penyebarluasan konten dan pemberian tautan referensi

yang bekerja secara otomatis, sehingga pemublikasian media *audio-visual* ini masih tergolong lambat khususnya dalam kondisi darurat yang membutuhkan penyebarluasan secara cepat. Dengan aplikasi ini maka pengguna tidak perlu repot dalam melakukan perekaman dan publikasi videonya, karena aplikasi ini selain dapat melakukan otomatisasi, juga dapat berjalan dengan perintah suara. Dengan mengimplementasikan algoritma pencocokan string maka aplikasi akan semakin fleksibel dalam melakukan pendeteksian perintah suara sehingga perekaman dan pembagian video pun akan menjadi lebih efisien.

## 1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan diatas, dapat dirumuskan beberapa permasalahan. Permasalahan tersebut adalah sebagai berikut:

1. Bagaimana cara membangun aplikasi unggah video pada YouTube dan *post* status berupa *link* pada Facebook yang keduanya dilakukan secara otomatis.
2. Bagaimana cara membangun aplikasi perekam video yang dapat berjalan dengan perintah suara.
3. Bagaimana menerapkan algoritma pencocokan *string* untuk mencocokkan perintah suara pengguna dengan data yang ada dalam sistem.
4. Bagaimana aplikasi yang dibangun dapat menyediakan fitur pemberian *tag reference* pada video yang diunggah.
5. Bagaimana aplikasi yang dibangun dapat mengganti penggunaan kamera depan atau belakang.

## 1.3. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan. Batasan tersebut sebagai berikut:

1. Aplikasi ini hanya dapat mengunggah video ke dalam situs YouTube saja, sedangkan Facebook hanya sebatas *link* referensi dari YouTube.
2. Perintah suara yang dapat dideteksi untuk melakukan perekaman oleh aplikasi hanya kata “rekam” saja.
3. Tag referensi pada video adalah berupa waktu dan lokasi video tersebut diambil.
4. Algoritma pencocokan *string* yang digunakan adalah algoritma *Levenshtein Distance* dan *Soundex*.
5. Durasi waktu perekaman adalah selama 1 menit setelah perintah suara untuk merekam dari pengguna dideteksi oleh aplikasi. Dengan durasi 1 menit tersebut diharapkan aplikasi dapat dengan cepat mengunggah video ke situs YouTube karena ukuran datanya yang relatif kecil dan aplikasi tidak akan banyak menghabiskan daya baterai *smartphone*.
6. *Front end* aplikasi perangkat komunikasi bergerak ini dibangun dengan menggunakan bahasa *native* untuk sistem operasi Android.
7. Aplikasi ini dapat dijalankan pada perangkat komunikasi bergerak dengan spesifikasi sistem operasi Android minimal 2.3 (Gingerbread).

## 1.4. Tujuan dan Manfaat

Tugas Akhir ini memiliki beberapa tujuan. Rincian dari tujuan tersebut dapat dituliskan sebagai berikut:

1. Membangun aplikasi yang dapat membantu mempermudah pengguna dalam proses perekaman video dan publikasinya ke dalam situs YouTube dan Facebook.
2. Mengimplementasikan algoritma pencocokan *string* untuk mencocokkan perintah suara pengguna dengan data yang ada di dalam sistem.
3. Membangun aplikasi yang dapat memberikan *tag reference* berupa lokasi dan waktu video tersebut diambil.

4. Membangun aplikasi perekam video yang dapat dijalankan dengan perintah suara.
5. Membuat aplikasi yang dapat berjalan dengan baik pada perangkat komunikasi bergerak berbasis Android.

Manfaat dari disusunnya Tugas Akhir ini adalah untuk membantu mempercepat proses publikasi video ke YouTube serta Facebook, serta membantu mempermudah masyarakat dalam perekaman video di segala situasi dan kondisi dengan hanya menggunakan perintah suara.

## 1.5. Metodologi

Pada bagian ini akan dijelaskan metodologi yang digunakan dalam proses pelaksanaan Tugas Akhir. Metodologi tersebut adalah sebagai berikut:

### 1.5.1. Penyusunan Proposal

Pada tahap ini penulis menyusun proposal tugas akhir sebagai langkah awal dalam pengerjaan tugas akhir. Pada proposal ini penulis menggagas penulisan tugas akhir untuk merancang suatu aplikasi yang mengimplementasikan algoritma pencocokan string untuk mendeteksi perintah suara pada aplikasi pengunggahan video otomatis yaitu FaceTube.

### 1.5.2. Studi Literatur

Tahap ini merupakan tahap pengumpulan informasi yang diperlukan untuk pengerjaan tugas akhir sekaligus mempelajarinya. Tahap–tahap studi literatur adalah sebagai berikut:

- Pengumpulan informasi terkait dengan aplikasi dan juga mencari informasi mengenai kebutuhan pengguna.
- Studi literatur tentang pemahaman konsep aplikasi android berbasis kamera untuk melakukan perekaman video serta pengunggahannya, konsep *login* YouTube

dan Facebook pada aplikasi, dan pendeteksian perintah suara.

- Mempelajari algoritma pencocokan *string Soundex* dan *Levenshtein Distance*.
- Mempelajari teknologi dan pustaka yang mendukung pembangunan aplikasi, seperti YouTube API, Facebook API dan Google API.

### 1.5.3. Analisis dan Desain Perangkat Lunak

Pada tahapan ini akan dilakukan analisa terhadap kebutuhan yang akan digunakan pada pengembangan perangkat lunak. Selain itu pada tahap ini juga dilakukan perancangan arsitektur aplikasi yang meliputi beberapa hal sebagai berikut:

- Menentukan *domain* permasalahan.
- Mendeskripsikan sistem secara umum.
- Menentukan spesifikasi kebutuhan perangkat lunak.
- Membuat pemodelan sistem menggunakan UML yang terdiri dari kasus penggunaan, spesifikasi kasus penggunaan, dan diagram aktivitas.
- Menentukan arsitektur sistem.
- Membuat rancangan implementasi algoritma pencocokan string.
- Membuat rancangan antarmuka pengguna.

### 1.5.4. Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun aplikasi android perekam video dengan perintah suara yang menggunakan algoritma pencocokan string. Sistem dibangun dengan berpedoman pada konsep-konsep yang sudah ditentukan pada tahap sebelumnya. Pembangunan perangkat lunak menggunakan alat bantu dan pustaka sebagai berikut:

- *Eclipse Kepler* sebagai kakas bantu *Integrated Development Environment* (IDE) untuk aplikasi

berbasis Java.

- *Java Development Kit* 1.8.0 dan *Java Runtime Environment* 8 sebagai pustaka yang mendukung pembangunan aplikasi Java.
- *Android SDK (Software Development Kit)* sebagai *library* dan *tools* yang dibutuhkan untuk membangun aplikasi berbasis Android.
- *YouTube API v3, Google Speech Recognition API, Google Sign-in API* dan *Google Maps Geocoding API* sebagai pustaka yang mendukung pembangunan aplikasi android yang terhubung dengan YouTube dan Google.
- *Facebook Login API* dan *Facebook Graph API* sebagai pustaka yang mendukung pembangunan aplikasi android yang terhubung dengan Facebook.

### 1.5.5. Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba terhadap sistem yang telah dibuat, untuk mengamati kinerja sistem, serta mengidentifikasi kesalahan. Proses pengujian yang akan dilakukan adalah menguji fungsionalitas dari aplikasi, mengevaluasi jalannya program, mendeteksi kesalahan-kesalahan yang mungkin terjadi, dan melakukan perbaikan bila terdapat kekurangan untuk menyempurnakan hasil. Pengujian juga dilakukan untuk mengevaluasi apakah program yang dibuat akan menghasilkan solusi sesuai dengan tujuan dan manfaat dari Tugas Akhir ini.

### 1.5.6. Penyusunan Buku Tugas Akhir

Tahap terakhir merupakan penyusunan laporan yang memuat dokumentasi mengenai pembuatan serta hasil implementasi perancangan yang telah dibuat. Buku Tugas Akhir yang dibuat, terdiri dari beberapa bagian yaitu sebagai berikut:

1. Pendahuluan
   1.1 Latar Belakang

## 1.6. Sistematika Penulisan

Sistematika penulisan buku Tugas Akhir dibagi menjadi beberapa bab sebagai berikut:

*Bab 1 Pendahuluan*

Bab ini berisi latar belakang masalah, tujuan dan manfaat dari pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika Tugas Akhir.

*Bab 2 Tinjauan Pustaka*

Bab ini membahas tentang teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

*Bab 3 Analisis dan Perancangan*

Bab ini membahas analisis dan perancangan perangkat lunak. Rancangan perangkat lunak meliputi rancangan data, arsitektur, dan proses.

*Bab 4 Implementasi*

Bab ini membahas cara mengimplementasikan hasil rancangan perangkat lunak ke dalam kode program dan pembuatan antarmuka.

*Bab 5 Uji Coba dan Evaluasi*

Bab ini membahas tentang uji coba subjektif dari aplikasi yang telah dikembangkan. Uji coba dan evaluasi langsung dilakukan oleh pengembang perangkat lunak menggunakan pengujian kotak hitam.

*Bab 6 Kesimpulan dan Saran*

Bab ini berisi kesimpulan dan saran dari hasil uji coba yang dilakukan.

# BAB 2
# TINJAUAN PUSTAKA

Pada bagian ini akan dibahas mengenai tinjauan pustaka yang menjadi dasar dari pembuatan Tugas Akhir ini. Beberapa teori dan pustaka yang mendasari pengerjaan Tugas Akhir ini adalah definisi pencocokan *string*, algoritma *Soundex*, algoritma *Levenshtein Distance*, *Application Programming Interface*, *YouTube*, *Facebook* dan *Google API*. Berikut ini penjelasan masing-masing tinjauan pustaka.

## 2.1 Pencocokan String

*String* adalah susunan dari karakter-karakter dan biasanya direpresentasikan sebagai struktur data *array*. *String* dapat berupa kata, frase, atau kalimat. Pencocokan *string* diartikan sebagai sebuah permasalahan untuk menemukan pola susunan karakter *string* di dalam *string* lain atau bagian dari isi teks [1].

## 2.2 Algoritma Soundex

Algoritma *Soundex* pertama kali dipatenkan oleh Margaret O'Dell dan Robert C. Russell pada tahun 1918 [3]. Algoritma *Soundex* mengambil masukan berupa sebuah kata atau nama. Algoritma ini menghasilkan sebuah *string* yang mengidentifikasi apakah sepasang kata tersebut mirip secara fonetik. *String* ini disebut dengan kode fonetis.

Fonetik adalah ilmu yang menyelidiki bunyi bahasa tanpa melihat fungsi bunyi itu sebagai pembeda makna dalam suatu bahasa. Pencocokan *string* fonetik adalah suatu teknik pencocokan *string* yang membandingkan suatu *string* dengan *string* yang lain berdasarkan kode fonetis masing-masing [2]. Sebuah *string* yang berbeda namun mempunyai cara pengucapan yang sama, akan memiliki kode fonetis yang sama. Contohnya, "akhmad" dengan "achmad".

Metode yang digunakan algoritma *Soundex* berdasarkan pada klasifikasi fonetis dari suara cara berbicara manusia (*human speech sound*) yaitu *labial*, *dental*, *alveolar*, *palato-alveolar*, *palatal*, *velar*, dan *glottal* [3]. Klasifikasi ini berdasarkan pada dimana manusia meletakkan bibir dan lidahnya ketika melafalkan bunyi tertentu. Penjelasan klasifikasi ini adalah sebagai berikut [1]:

1. *Labial* adalah bunyi bibir.
2. *Dental* adalah bunyi diartikulasi oleh ujung lidah dengan gigi atas.
3. *Alveolar* adalah bunyi diartikulasi oleh ujung lidah dengan punggung gigi.
4. *Palato-alveolar* adalah bunyi yang memiliki artikulasi *alveolar* diikuti dengan naiknya lidah sampai pada langit-langit mulut secara simultan.
5. *Palatal* adalah bunyi diartikulasi oleh bagian depan lidah dengan langit-langit keras.
6. *Velar* adalah bunyi diartikulasi oleh bagian belakang lidah dengan langit-langit lunak.
7. *Glottal* adalah bunyi diartikulasi oleh glottis.

Aturan pemberian kode fonetis per huruf pada algoritma *Soundex* [4] dapat dilihat pada Tabel 2.1.

Tabel 2.1 Aturan pemberian kode fonetis pada algoritma Soundex [4]

| Huruf | Kode | Klasifikasi Fonetis |
|---|---|---|
| A, E, H, I, O, U, W, Y | 0 | Diperlakukan sebagai bunyi vokal |
| B, F, P, V | 1 | *Labial* dan *labio-dental* |
| C, G, J , K, Q, S, X, Z | 2 | *Glottal* |
| D, T | 3 | *Dental-mute* |

| Huruf | Kode | Klasifikasi Fonetis |
|---|---|---|
| L | 4 | *Palatal fricative* |
| M, N | 5 | *Labio-nasal* dan *dental* |
| R | 6 | *Dental fricative* |

Langkah-langkah algoritma *Soundex* dalam menghasilkan kode fonetis dari sebuah *string* masukan adalah sebagai berikut [3]:

1. Ubah semua huruf menjadi huruf kapital, dan hilangkan tanda baca.
2. Pertahankan huruf pertama pada kata tersebut.
3. Ubah huruf lainnya menjadi kode fonetis berdasarkan tabel.
4. Hapus semua pasangan dari kode fonetis yang berurutan.
5. Hapus semua kode fonetis 0.
6. Tulis empat posisi pertama yang mengikuti pola:

   <*uppercase letter*><*digit*><*digit*><*digit*>. Ini adalah kode fonetis sebagai keluaran. Jika kode fonetis tidak sampai empat karakter, maka ditambahkan *digit* 0 sampai menjadi empat karakter.

Misalkan ada sebuah string masukan “Baharuddin”. Langkah-langkah pemberian kode fonetisnya adalah sebagai berikut. Ubah semua huruf menjadi huruf kapital (“BAHARUDDIN”). Pertahankan huruf pertama, lalu ubah huruf lainnya menjadi kode fonetis berdasarkan tabel (B000603305). Hapus semua pasangan dari kode fonetis yang berurutan (B060305). Hapus semua kode fonetis 0 (B635). Empat karakter pertama adalah kode fonetis sebagai keluaran (B635).

## 2.3. Algoritma Levenshtein Distance

Algoritma *Levenshtein Distance* dinamakan berdasarkan penemunya, yaitu Dr. Vladimir Levenshtein. Dr. Vladimir Levenshtein dikenal sebagai bapak dari teori koding (*father of coding theory*) di Rusia [5]. *Levenshtein Distance* saat ini adalah dasar dari banyak aplikasi komputer di bidang pemeriksaan ejaan.

*Levenshtein Distance* adalah jumlah minimal operasi yang dibutuhkan untuk mengubah suatu *string* ke *string* yang lain. Dengan algoritma ini, dapat diketahui tingkat perbedaan dua buah *string* dalam representasi angka. Operasi yang dilakukan ada 3 macam yaitu [6]:

1. *Insertion*

   *Insertion* adalah penyisipan sebuah karakter ke dalam sebuah *string* tertentu. Contohnya, penyisipan karakter ‘m’ ke dalam *string* “julah” pada posisi setelah karakter ‘u’ akan mengubah *string* “julah” menjadi “jumlah”.

2. *Deletion*

   *Deletion* adalah penghapusan sebuah karakter dari sebuah *string* tertentu. Contohnya, penghapusan karakter ‘a’ pada posisi terakhir *string* “algoritma” akan mengubah *string* “algoritma” menjadi “algoritm”.

3. *Substitution*

   *Substitution* adalah penggantian sebuah karakter dari sebuah *string* dengan karakter lain. Contohnya, penggantian karakter ‘l’ dengan karakter ‘b’ pada *string* “julah” akan mengubah *string* “julah” menjadi “jubah”.

Misalkan ada dua buah masukan *string* yaitu “kepala” dan “kapal”. Langkah untuk mengubah “kepala” menjadi “kapal” adalah sebagai berikut. Substitusi karakter ‘e’ dengan karakter ‘a’. Kemudian hapus karakter ‘a’ pada posisi terakhir. Jadi, jumlah operasi yang dilakukan adalah dua kali. Satu kali *substitution* dan satu kali *deletion*. *Levenshtein Distance* adalah

jumlah dari banyaknya operasi yang dilakukan. Sehingga, *Levenshtein Distance* dari “kepala” dan “kapal” adalah 2.

*Levenshtein Distance* menggunakan konsep *dynamic programming* dalam penerapannya. Algoritma ini menggunakan sebuah matriks berukuran panjang dari *string-string* yang menjadi masukan. *Levenshtein Distance* yang dihasilkan adalah angka yang terletak di kotak paling bawah kanan dari matriks, sebagai jumlah minimum operasi yang dibutuhkan untuk mengubah *string* awal menjadi *string* target. *Pseudocode* algoritma *Levenshtein Distance* ditunjukkan pada Gambar 2.1.

```
function LevenshteinDistance(s1[1..m],s2[1..n])
levDist ← array(0..m, 0..n)


for i ←0 to m
        for j ← 0 to n
                if i = 0
                        levDist[i,j] ← j
                else if j = 0
                        levDist[i,j] ← i
                else
                   if(s1[i]=s2[j])levDist[i,j] ← levDist[i-1,j-1];
                   else
                      levDist[i,j] ← 1 + minimum(levDist[i,j-1],
                                                 levDist[i-1,j],
                                                  levDist[i-1,j-1]
                                                );
return levDist[i,j];
```

Gambar 2.1 Algoritma Levenshtein Distance

Contoh matriks *Levenshtein Distance* untuk masukan “kepala” dan “kapal” ditunjukkan pada Gambar 2.2.

*String* pada baris pertama adalah *string* awal. Sedangkan *string* pada kolom pertama adalah *string* target. Angka pada setiap sel matriks adalah banyaknya operasi yang dibutuhkan untuk mengubah *substring* awal menjadi *substring* target. Misalnya pada baris pertama dan kolom pertama, nilainya adalah 0, karena tidak ada operasi yang diperlukan untuk mengubah *substring* awal “k”

menjadi *substring* target “k”. Pada baris kedua dan kolom kedua, nilainya adalah 1, karena ada satu operasi yang dilakukan, yaitu substitusi karakter ‘e’ dengan ‘a’. Demikian seterusnya sampai dengan baris terakhir kolom terakhir.

| | | K | E | P | A | L | A |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| K | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| A | 2 | 1 | 1 | 2 | 2 | 3 | 4 |
| P | 3 | 2 | 2 | 1 | 2 | 3 | 4 |
| A | 4 | 3 | 3 | 2 | 1 | 2 | 3 |
| L | 5 | 4 | 4 | 3 | 2 | 1 | 2 |

Gambar 2.2 Contoh matriks Algoritma Levensthein Distance

## 2.4. Antarmuka Pemrograman Aplikasi

Antarmuka Pemrograman Aplikasi atau yang dalam bahasa Inggris disebut *Application Programming Interface* (API) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh *programmer* saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan *programmer* untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

API dapat juga diartikan sebagai serangkaian instruksi dan standar pemrograman untuk mengakses aplikasi atau layanan berbasis web. Sebuah perusahaan *software* atau penyedia layanan berbasis web merilis API mereka kepada publik. Dengannya, pengembang lain dapat mendesain aplikasi yang memanfaatkan layanan mereka.

Sebagai contoh, Amazon merilis API sehingga para pengembang *web* dapat lebih mudah mengakses informasi produk-produk Amazon dari *website* mereka. Menggunakan API dari Amazon, *website* pihak ketiga dapat mem-*posting* link langsung ke produk-produk Amazon dengan harga aktual dan opsi "*buy now*" [7]. Pada aplikasi Facetube ini akan dipergunakan API dari Google, Facebook dan YouTube.

## 2.5. YouTube

YouTube adalah sebuah situs web berbagi video yang dibuat oleh tiga mantan karyawan PayPal pada Februari 2005. Situs ini memungkinkan pengguna mengunggah, menonton, dan berbagi video. Perusahaan ini berkantor pusat di San Bruno, California, dan memakai teknologi *Adobe Flash Video* dan *HTML5* untuk menampilkan berbagai macam konten video buatan pengguna, termasuk klip film, klip TV, dan video musik. Selain itu ada pula konten amatir seperti blog video, video orisinal pendek, dan video pendidikan [8].

Kebanyakan konten di YouTube diunggah oleh individu, meskipun perusahaan-perusahaan media seperti CBS, BBC, Vevo, Hulu, dan organisasi lain sudah mengunggah material mereka ke situs ini sebagai bagian dari program kemitraan YouTube. Pengguna tak terdaftar dapat menonton video, sementara pengguna terdaftar dapat mengunggah video dalam jumlah tak terbatas. Video-video yang dianggap berisi konten ofensif hanya bisa ditonton oleh pengguna terdaftar berusia 18 tahun atau lebih. Pada November 2006, YouTube, LLC dibeli oleh Google dengan nilai US$1,65 miliar dan resmi beroperasi sebagai anak perusahaan Google [9].

Pada aplikasi FaceTube digunakan *YouTube Data API v3 Client Library* untuk *Java* pada Android. YouTube Data API mendukung fitur-fitur utama dari YouTube seperti mengunggah video, membuat dan mengolah *playlist*, mencari konten, dan masih banyak lagi [15].

## 2.6. Facebook

Facebook adalah sebuah layanan jejaring sosial yang diluncurkan pada bulan Februari 2004, dan berkantor pusat di Menlo Park, California, Amerika Serikat. Pada September 2012, Facebook memiliki lebih dari satu miliar pengguna aktif [10], lebih dari separuhnya menggunakan telepon genggam [11]. Pengguna harus mendaftar sebelum dapat menggunakan situs ini. Setelah itu, pengguna dapat membuat profil pribadi, menambahkan pengguna lain sebagai teman, dan bertukar pesan, termasuk pemberitahuan otomatis ketika mereka memperbarui profilnya. Selain itu, pengguna dapat bergabung dengan grup pengguna dengan ketertarikan yang sama, diurutkan berdasarkan tempat kerja, sekolah atau perguruan tinggi, atau ciri khas lainnya, dan mengelompokkan teman-teman mereka ke dalam daftar seperti "Rekan Kerja" atau "Teman Dekat".

Facebook didirikan oleh Mark Zuckerberg bersama teman sekamarnya dan sesama mahasiswa Universitas Harvard, Eduardo Saverin, Andrew McCollum, Dustin Moskovitz, dan Chris Hughes [12].

Pada aplikasi FaceTube digunakan Facebook SDK untuk Android. Facebook SDK ini adalah cara termudah untuk mengintegrasikan aplikasi Android kita dengan Facebook. Dengan SDK tersebut kita dapat melakukan *Login, Share* dan *Send dialogs, Apps Events,* dan *Graph API* [16].

## 2.7. Google API

Google API adalah sebuah set antarmuka pemrograman aplikasi (API) yang dikembangkan oleh Google yang memungkinkan komunikasi dengan *Google Services* beserta integrasi untuk *service* lainnya. Contohnya meliputi *Search, Gmail, Translate* atau *Google Maps*. Aplikasi pihak ketiga dapat menggunakan API ini untuk mengambil manfaat dari atau memperluas fungsionalitas dari *service* yang telah ada. API

tersebut menyediakan fungsionalitas seperti *analytics*, perangkat belajar yang bertindak sebagai *service* (*the Prediction API)* atau akses ke data pengguna (ketika izin untuk membaca data diberikan). Contoh penting lainnya adalah *Google map* yang terpasang pada sebuah *website*, yang mana dapat diimplementasikan menggunakan *Static maps API, Places API* atau *Google Earth API.*

Penggunaan dari beberapa API membutuhkan otentikasi dan otorisasi yang menggunakan protokol *OAuth 2.0*. *OAuth 2.0* adalah protokol yang sederhana. Sebagai permulaan, dibutuhkan pengambilan *credentials* dari *Developer Console*. Setelah itu aplikasi klien dapat melakukan *request* token akses dari *Google Authorization Server* dan menggunakanya untuk otorisasi ketika mengakses servis dari Google API [13]. Terdapat library klien di berbagai macam bahasa yang memungkinkan pengembang untuk menggunakan Google API dari dalam kode mereka, seperti *Java*, *JavaScript*, *.NET*, *Objective-C*, *PHP* dan *Python* [14].

Jika kita hendak mengembangkan aplikasi Android dan Google API yang ingin kita butuhkan termasuk dalam *library Google Play Service* maka gunakan *library* tersebut untuk mendapatkan pengalaman dan performa terbaik. Akan tetapi jika Google API yang hendak digunakan tidak ada dalam *library Google Play Service* maka kita dapat menggunakan *Google API Client library* untuk *Java* yang dapat berjalan pada Android versi 1.5 keatas [17]. Untuk *menggunakan Google Play Service* pada aplikasi Android diperlukan beberapa penyetelan project kita dengan *Google Play service SDK* terlebih dahulu. Sedangkan untuk melakukan pengetesan aplikasi, yang diibutuhkan adalah perangkat Android yang kompatibel yang berjalan pada Android 2.3 keatas termasuk memiliki *Google Play Store* atau *emulator* Android dengan AVD yang menjalankan *platform* Google API pada Android 4.2.2 keatas [18].

*[Halaman ini sengaja dikosongkan]*

# BAB 3
# ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan mengenai analisis pembangunan aplikasi perekam video FaceTube. Analisis mencakup kebutuhan yang menjadi dasar dari rancangan sistem yang dikembangkan, yaitu: domain permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan perangkat lunak, dan analisis aktor. Perancangan meliputi perancangan skenario kasus penggunaan, arsitektur, algoritma, dan perancangan antarmuka.

## 3.1 Tahap Analisis

Tahap ini merupakan tahap mendefinisikan secara detail analisis – analisis dalam pembangunan sistem.

### 3.1.1. Domain Permasalahan

Domain permasalahan dalam aplikasi ini adalah bagaimana aplikasi dapat mendeteksi perintah suara secara optimal dengan mengimplementasikan algoritma pencocokan *string*. Kesalahan pendeteksian terjadi akibat jarak suara pengguna yang terlalu jauh dengan *microphone smartphone* ataupun suara pengguna yang kurang jelas. Kesalahan juga dapat disebabkan karena banyaknya suara lain yang terjadi di lingkungan pendeteksian perintah suara. Berdasarkan pengamatan pada pendeteksian perintah suara, kata "rekam" terkadang terdeteksi sebagai "bekam", "tekan", atau beberapa kata lain yang ketika dibaca berbunyi mirip seperti "rekam". Dari permasalahan tersebut maka diterapkan algoritma *Levenshtein Distance* dan *Soundex*. *Levenshtein Distance* akan mengecek seberapa miripkah string perintah suara yang terdeteksi aplikasi dengan *string* yang terdapat dalam sistem dalam bentuk persentase. Yang selanjutnya akan dilakukan pengecekan berdasarkan kemiripan bunyi pelafalannya menggunakan *Soundex*. Jika *string* suara yang dieksekusi oleh algoritma tersebut

memenuhi syarat maka suara tersebut tergolong perintah suara yang dapat menjalankan aplikasi untuk merekam video.

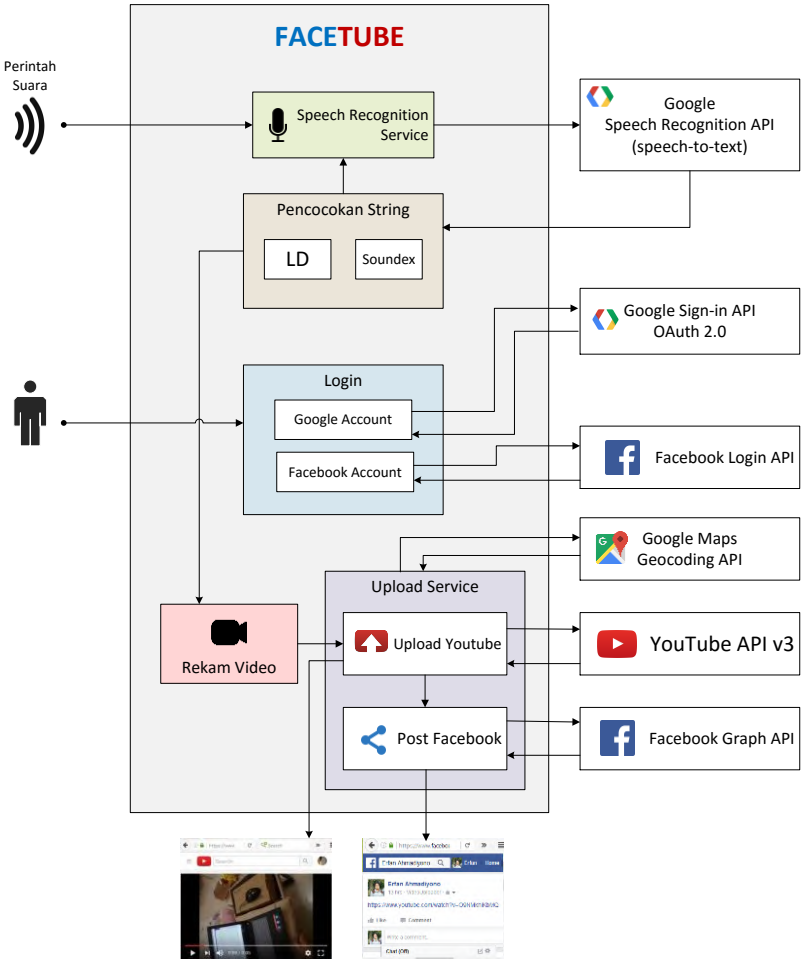
Permasalahan lain yang timbul adalah kebutuhan masyarakat akan aplikasi perekam video pada perangkat *smartphone* yang dapat membantu dalam keadaan terdesak maupun untuk meminimalisasi waktu publikasinya. Musibah berupa kecelakaan, bencana alam, serta peristiwa lain yang terjadi begitu cepat, tentu masyarakat tidak ingin melewatkan untuk memublikasikannya. Dari masalah tersebut, dibutuhkan sebuah aplikasi perekam video yang fleksibel dan juga dapat memberikan informasi lokasi serta waktu secara otomatis pada hasil rekaman yang dipublikasikan, Dengan begitu, diharapkan aplikasi ini dapat membantu masyarakat dalam perihal perekaman dan berbagi video.

### 3.1.2. Deskripsi Umum Perangkat Lunak

Perangkat lunak yang akan dibangun dalam Tugas Akhir ini adalah aplikasi perekam video yang dapat berjalan menggunakan perintah suara. Video yang telah diambil akan secara otomatis diunggah oleh aplikasi ke akun YouTube pengguna dan akan dilakukan publikasi link video yang telah terunggah ke akun Facebook pengguna sebagai status. Dalam pendeteksian perintah suara akan diimplementasikan algoritma pencocokan string sehingga pendeteksian perintah suara akan semakin baik. Gambaran umum dari aplikasi FaceTube dapat digambarkan dengan diagram blok pada Gambar 3.1.

Penjelasan dari diagram blok pada Gambar 3.1 adalah sebagai berikut, jika pengguna menggunakan aplikasi ini untuk pertama kalinya maka diharuskan untuk melakukan login akun YouTube dan Facebook miliknya terlebih dahulu, kemudian aplikasi akan menunggu perintah suara dari pengguna untuk melakukan perekaman (*Speech Recognition Service*). Perintah suara yang masuk akan dikirimkan dan diproses oleh *Google Speech Recognition* dan aplikasi akan menerima penerjemahan

(*speech-to-text*) suara berupa *string*. Selanjutnya aplikasi akan melakukan proses pencocokan dengan algoritma yang telah diimplementasikan. Jika cocok dengan perintah suara yang ada pada sistem maka perekaman akan dimulai.




Gambar 3.1 Diagram Blok Aplikasi FaceTube

Kemudian dalam rentang waktu yang ditentukan perekaman video akan selesai dan aplikasi akan melakukan pengunggahan secara otomatis pada YouTube serta melakukan *post status* pada Facebook berupa *link*. Aplikasi akan memberikan tag referensi berupa lokasi dan waktu video diambil pada video yang diunggah di YouTube. Jika proses telah selesai maka aplikasi akan menampilkan notifikasi bahwa proses telah sukses dilakukan.

### 3.1.3. Spesifikasi Kebutuhan Perangkat Lunak

Berdasarkan hasil analisis terhadap domain permasalahan, serta deskripsi umum yang telah dilakukan, maka terdapat beberapa spesifikasi kebutuhan perangkat lunak yang harus dipenuhi agar sistem dapat berjalan dengan baik. Spesifikasi kebutuhan ini dibagi menjadi dua kategori, yaitu kebutuhan fungsional dan non fungsional.

#### 3.1.3.1. Kebutuhan Fungsional

Berdasarkan deskripsi umum sistem yang telah disebutkan pada sub bab 3.1.2, maka dapat disimpulkan menjadi beberapa proses bisnis. Dari proses bisnis tersebut dapat dipetakan menjadi sebuah fungsionalitas. Tabel 3.1 merupakan proses beserta pemetaannya terhadap fungsionalitas.

Tabel 3.1 Pemetaan proses bisnis dan fungsionalitas

| No. | Proses Bisnis | Fungsionalitas |
|---|---|---|
| 1. | Mengelola data *login* akun YouTube (Google) dan Facebook | Memasukkan data *login* pengguna berupa *username*/*email* dan *password* |
| | | Mengelola data *token login* agar pengguna tak perlu *login* berulang kali |

| No. | Proses Bisnis | Fungsionalitas |
|---|---|---|
| 2. | Mengelola perintah suara untuk perekaman video | Menerima perintah suara pengguna yang selanjutnya dikirimkan ke *google speech recognition* untuk mendapatkan *string* perintah suara |
| | | Mencocokkan *string* perintah suara yang terdeteksi dengan sistem menggunakan 2 algoritma |
| 3. | Melakukan pemilihan penggunaan kamera | Melakukan pengecekan apakah terdapat kamera depan untuk dapat melakukan perekaman video |
| 4. | Mengelola *service* pengenalan suara | Menghidupkan atau mematikan *service* pengenalan suara |
| 5. | Mengelola video pada aplikasi | Melakukan perekaman video dalam rentang waktu yang ditentukan |
| | | Memberikan tag referensi berupa lokasi dan waktu video diambil |
| | | Mengunggah video ke akun YouTube pengguna |
| | | Menerima *link* video dari YouTube dan melakukan *post status* pada akun Facebook pengguna |
| | | Menyimpan video ke dalam memori *smartphone* pengguna |

### 3.1.3.2. Kebutuhan non Fungsional

Berikut daftar kebutuhan non-fungsional yang harus dipenuhi agar aplikasi berjalan sesuai kebutuhan.

1. Koneksi internet
   Koneksi internet dibutuhkan untuk dapat mengambil dan mengirim informasi ke Google. Serta untuk dapat melakukan *login*, mengunggah video, mendeteksi lokasi dan *post status* ke akun sosial media.

2. Notifikasi
   Notifikasi *loading* dibutuhkan untuk memberitahukan bahwa aplikasi sedang melakukan proses unggah, karena proses tersebut membutuhkan waktu. Notifikasi *success* juga dibutuhkan untuk memberitahukan bahwa aplikasi telah sukses melakukan semua proses.

3. Akun sosial media
   Akun sosial media yaitu Facebook dan YouTube dibutuhkan untuk dapat melakukan proses unggah dan *post* status, karena pada dasarnya aplikasi ini mengolah video pada dua situs tersebut.

4. *GPS (Global Positioning System)*
   *GPS* pada perangkat *smartphone* diperlukan untuk mendapatkan lokasi yang akurat dari tempat pengambilan video.

### 3.1.4. Analisis Aktor

Berdasarkan deskripsi umum pada sub bab 3.1.2, maka dapat diketahui bahwa pengguna yang akan menggunakan sistem ini hanya ada satu saja, yakni pengguna. Karena hanya ada 1 sistem saja dan tidak memakai database tersendiri melainkan memakai API Google. Untuk pengunggahan akan ditangani secara otomatis

oleh sistem. Penjelasan mengenai pengguna yang disebut aktor dalam sistem dijelaskan pada Tabel 3.2.

Tabel 3.2 Pengguna Aplikasi

| Nama Aktor | Definisi |
|---|---|
| Pengguna | Orang yang berinteraksi dengan sistem atau aplikasi secara langsung (melakukan *login*, memilih opsi kamera dan memberikan perintah suara) |

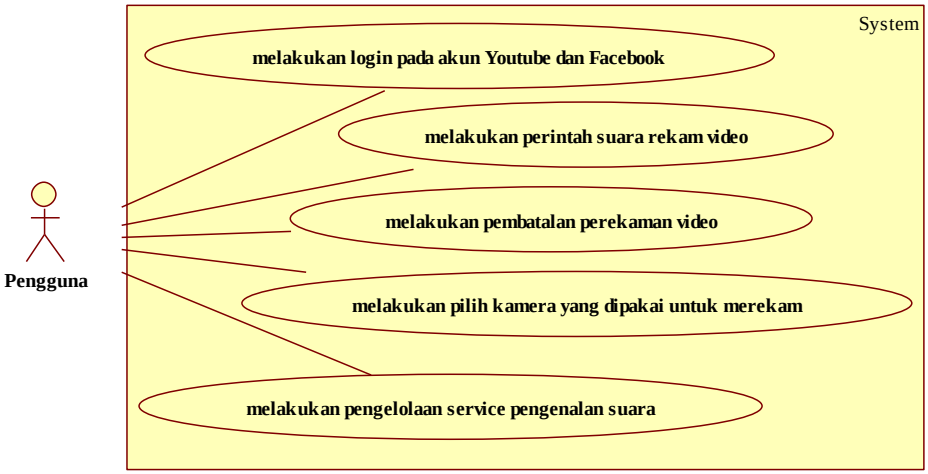
## 3.2. Tahap Perancangan

Tahap perancangan dalam sub bab ini dibagi menjadi beberapa bagian yaitu perancangan skenario kasus penggunaan, perancangan arsitektur sistem, perancangan algoritma, perancangan proses, serta perancangan antarmuka.

### 3.2.1. Perancangan Skenario Kasus Penggunaan

Berdasarkan spesifikasi kebutuhan fungsional yang telah disebutkan pada sub bab 3.1.3.1, kasus penggunaan yang akan digunakan pada sistem ini digambarkan pada Gambar 3.2. Kasus penggunaan merupakan kebutuhan pengguna yang paling utama dan harus ada dalam sebuah sistem. Dari gambar tersebut dapat diketahui bahwa terdapat 5 kasus penggunaan dalam sistem, yaitu melakukan *login* pada akun YouTube dan Facebook, melakukan perintah suara rekam video, melakukan pembatalan perekaman video, melakukan pilih kamera yang dipakai untuk merekam, dan melakukan pengelolaan servis pengenalan suara. Seluruh lima kasus penggunaan tersebut dilakukan oleh aktor pengguna.

Penjelasan dari masing-masing kasus penggunaan dalam sistem ini ditunjukkan pada Tabel 3.3. Tiap-tiap kasus penggunaan tersebut juga akan dijelaskan secara detil dengan menggunakan tabel spesifikasi kasus penggunaan pada sub bab berikutnya.




Gambar 3.2 Diagram Kasus Penggunaan Aplikasi dengan Aktor Pengguna

Tabel 3.3 Deskripsi Kasus Penggunaan

| No | Kode | Nama | Keterangan |
|---|---|---|---|
| 1 | KP-001 | Melakukan *login* pada akun YouTube dan Facebook | Pengguna wajib melakukan *login* pada akun YouTube dan Facebook miliknya |
| 2 | KP-002 | Melakukan perintah suara rekam video | Pengguna dapat memberikan perintah berupa suara kepada aplikasi untuk memulai perekaman video |
| 3 | KP-003 | Melakukan pembatalan perekaman video | Pengguna dapat membatalkan perekaman video |
| 4 | KP-004 | Melakukan pilih kamera yang dipakai untuk merekam | Pengguna dapat memilih kamera depan atau belakang yang digunakan untuk proses perekaman video |

| No | Kode | Nama | Keterangan |
|---|---|---|---|
| 5 | KP-005 | Melakukan pengelolaan *service* pengenalan suara | Pengguna dapat mematikan atau menyalakan *service* pengenalan perintah suara |

#### 3.2.1.1. Kasus Penggunaan Melakukan login pada akun Youtube dan Facebook (KP-001)

Pada kasus penggunaan melakukan *login* pada akun YouTube dan Facebook, nantinya pengguna diharuskan melakukan *login* akun miliknya di kedua situs media sosial tersebut terlebih dahulu sebelum hendak memakai aplikasi. Untuk rincian dari spesifikasi kasus penggunaan melakukan login ini dapat dilihat pada Tabel 3.4. Sedangkan untuk alur proses dari kasus penggunaan ini ditunjukkan pada diagram aktivitas pada Gambar 3.3.

#### 3.2.1.2. Kasus Penggunaan Melakukan perintah suara rekam video (KP-002)

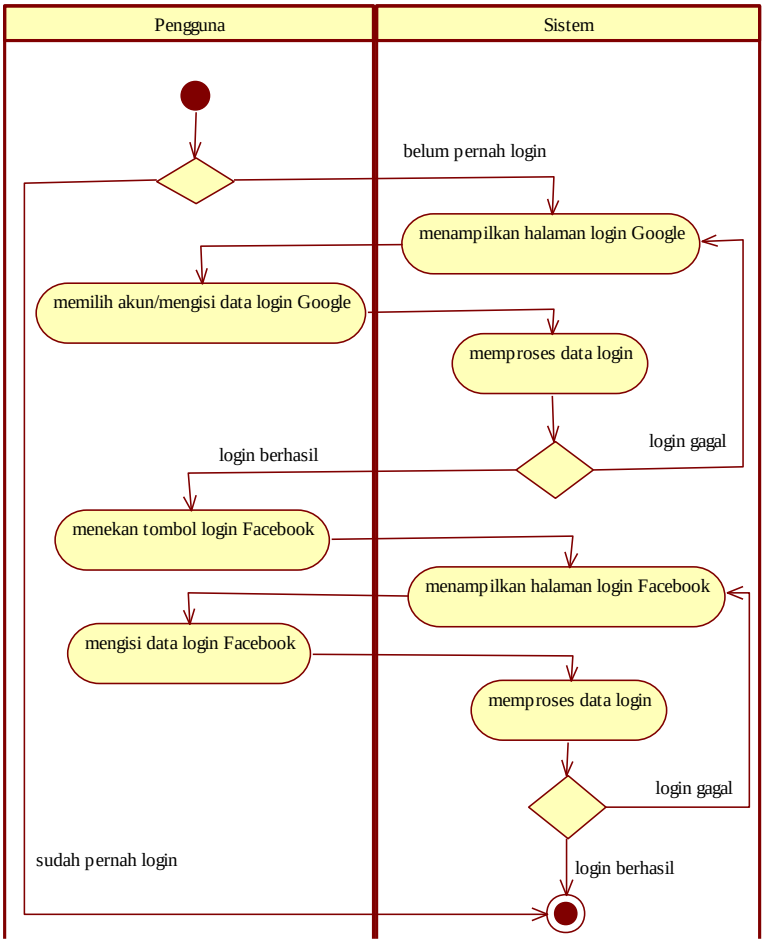
Kasus penggunaan melakukan perintah suara rekam video adalah kasus penggunaan yang digunakan oleh pengguna untuk memberi perintah suara “rekam” pada aplikasi agar memulai perekaman. Penjelasan lebih rinci terkait kasus penggunaan ini dapat dilihat pada Tabel 3.5. Perwujudan alur proses dari kasus penggunaan ini ditunjukkan pada diagram aktivitas pada Gambar 3.4.

Tabel 3.4 Spesifikasi kasus penggunaan login

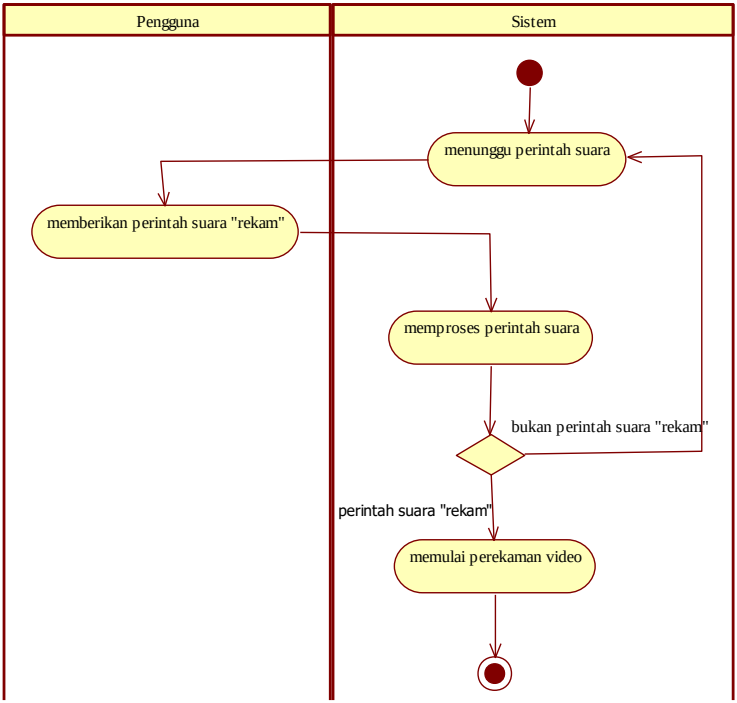
| | |
|---|---|
| **Nama Kasus Penggunaan** | Melakukan login pada akun YouTube dan Facebook |
| **Nomor** | KP-001 |
| **Deskripsi** | Proses ini digunakan untuk melakukan *login* pada akun |

| | |
|---|---|
| | YouTube dan Facebook |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Pengguna membuka aplikasi |
| **Alur Normal** | 1. Sistem menampilkan jendela *login* Google (untuk YouTube memakai akun Google)<br>2. Pengguna memilih akun Google (jika sudah terdapat *login* akun Google pada *smartphone*) atau mengisi data *login* akun Google<br>3. Sistem memproses data *login*<br>4. *Login* akun Google berhasil<br>5. Pengguna menekan tombol *login* Facebook<br>6. Sistem menampilkan jendela *login* Facebook<br>7. Pengguna mengisi data *login* akun Facebook<br>8. Sistem memproses data *login*<br>9. *Login* akun Facebook berhasil |
| **Alur Alternatif** | **Pengguna sudah pernah melakukan login**<br>1. Sistem menampilkan halaman utama<br>**Login gagal**<br>1. Sistem kembali menampilkan halaman *login* |

| | |
|---|---|
| **Kondisi Akhir** | Pengguna berhasil *login* dan sistem menampilkan halaman utama |




Gambar 3.3 Diagram Aktivitas Login




Gambar 3.4 Diagram Aktivitas melakukan perintah suara rekam

Tabel 3.5 Spesifikasi kasus penggunaan melakukan perintah suara rekam

| | |
|---|---|
| **Nama Kasus Penggunaan** | Melakukan perintah suara rekam video |
| **Nomor** | KP-002 |
| **Deskripsi** | Proses ini digunakan untuk melakukan perekaman video dengan perintah suara |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Sistem menghidupkan *service* pengenalan suara |
| **Alur Normal** | 1. Sistem menunggu perintah suara |

| | |
|---|---|
| | pengguna<br>2. Pengguna mengatakan kata perintah “rekam”<br>3. Sistem memproses perintah suara yang masuk<br>4. Sistem memulai perekaman video |
| **Alur Alternatif** | **Perintah suara yang terdeteksi bukan perintah suara untuk melakukan perekaman**<br>1. Sistem kembali menunggu perintah suara |
| **Kondisi Akhir** | Sistem melakukan perekaman video dengan durasi yang telah ditentukan |

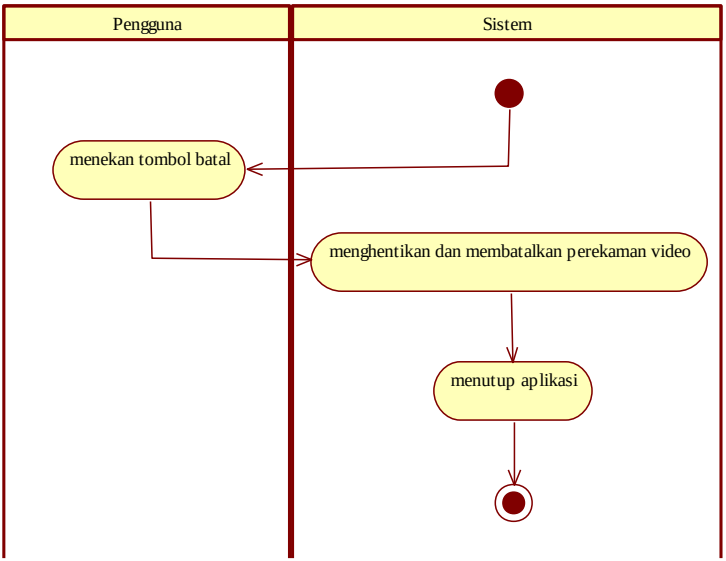
#### 3.2.1.3. Kasus Penggunaan Melakukan pembatalan perekaman video (KP-003)

Kasus penggunaan melakukan pembatalan perekaman video adalah kasus penggunaan dimana pengguna menekan tombol Batal untuk membatalkan perekaman video. Penjelasan lebih rinci terkait kasus penggunaan ini dapat dilihat pada Tabel 3.6. Perwujudan alur proses dari kasus penggunaan ini ditunjukkan pada diagram aktivitas pada Gambar 3.5.

#### 3.2.1.4. Kasus Penggunaan Melakukan pilih kamera yang dipakai untuk merekam (KP-004)

Kasus penggunaan melakukan pilih kamera yang dipakai adalah kasus penggunaan yang digunakan oleh pengguna untuk memilih kamera mana yang hendak dipakai untuk melakukan perekaman video (jika terdapat lebih dari 1 kamera di perangkat pengguna). Penjelasan lebih rinci terkait kasus penggunaan ini

dapat dilihat pada Tabel 3.7. Perwujudan alur proses dari kasus penggunaan ini ditunjukkan pada diagram aktivitas pada Gambar 3.6.

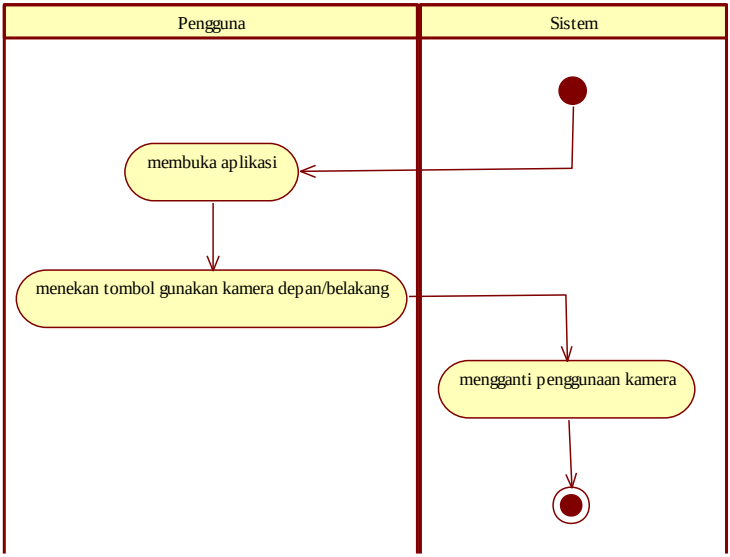



Gambar 3.5 Diagram Aktivitas pembatalan perekaman video

Tabel 3.6 Spesifikasi kasus penggunaan pembatalan perekaman

| | |
|---|---|
| **Nama Kasus Penggunaan** | Melakukan pembatalan perekaman video |
| **Nomor** | KP-003 |
| **Deskripsi** | Proses ini digunakan untuk melakukan pembatalan perekaman video |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Sistem sedang melakukan proses perekaman video |
| **Alur Normal** | 1. Pengguna menekan tombol batal |

| | |
|---|---|
| | 2. Sistem menghentikan dan membatalkan perekaman video<br>3. Sistem menutup aplikasi |
| **Alur Alternatif** | - |
| **Kondisi Akhir** | Sistem kembali menunggu perintah suara pengguna |




Gambar 3.6 Diagram Aktivitas pemilihan kamera

Tabel 3.7 Spesifikasi kasus penggunaan pemilihan kamera

| | |
|---|---|
| **Nama Kasus Penggunaan** | Melakukan pilih kamera yang dipakai untuk merekam |
| **Nomor** | KP-004 |
| **Deskripsi** | Proses ini digunakan untuk melakukan penggantian kamera yang hendak dipakai pengguna |

| | |
|---|---|
| | untuk melakukan perekaman video |
| **Aktor** | Pengguna |
| **Kondisi Awal** | Sistem menggunakan kamera belakang/depan untuk perekaman |
| **Alur Normal** | 1. Pengguna membuka aplikasi<br>2. Pengguna menekan tombol gunakan kamera depan/belakang<br>3. Sistem mengganti penggunaan kamera |
| **Alur Alternatif** | - |
| **Kondisi Akhir** | Sistem menggunakan kamera depan/belakang untuk melakukan perekaman |

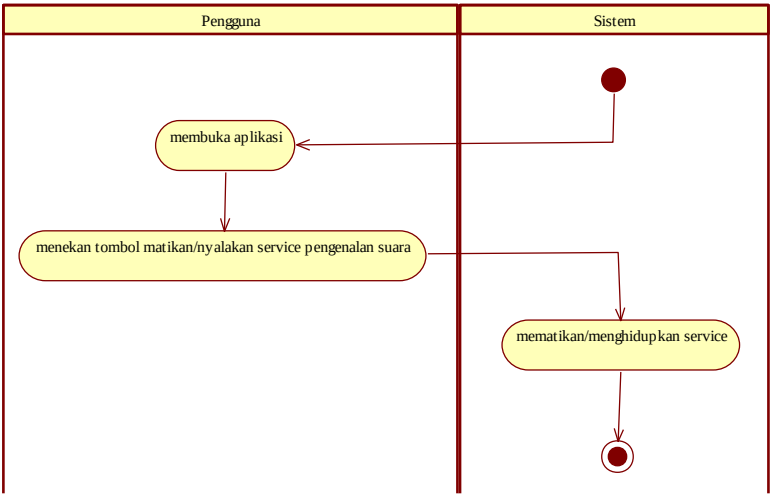
#### 3.2.1.5. Kasus Penggunaan Melakukan pengelolaan service pengenalan suara (KP-005)

Kasus penggunaan melakukan pengelolaan *service* pengenalan suara adalah kasus penggunaan yang digunakan oleh pengguna untuk menghidupkan/mematikan *service* dari pengenalan perintah suara. Sehingga pengguna dapat lebih menghemat daya *smartphone* jika sedang tidak menginginkan aplikasi terus berjalan pada *background* sistem. Penjelasan lebih rinci terkait kasus penggunaan ini dapat dilihat pada Tabel 3.8. Perwujudan alur proses dari kasus penggunaan ini ditunjukkan pada diagram aktivitas pada Gambar 3.7.

Tabel 3.8 Spesifikasi kasus penggunaan pengelolaan service pengenalan suara

| | |
|---|---|
| **Nama Kasus Penggunaan** | Melakukan pengelolaan service pengenalan suara |
| **Nomor** | KP-005 |
| **Deskripsi** | Proses ini digunakan untuk menghidupkan/mematikan *service* |

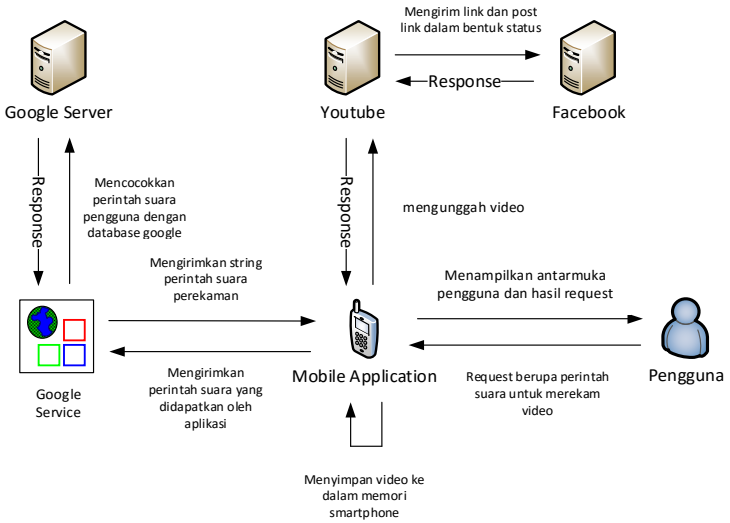
| | |
|---|---|
| | dari pengenalan perintah suara |
| **Aktor** | Pengguna |
| **Kondisi Awal** | *Service* pengenalan suara sedang berjalan/mati |
| **Alur Normal** | 1. Pengguna membuka aplikasi<br>2. Pengguna menekan tombol Matikan/Nyalakan *Service* Pengenalan Suara<br>3. Sistem mematikan/menghidupkan *service* |
| **Alur Alternatif** | - |
| **Kondisi Akhir** | Sistem memulai/menghentikan *service* pengenalan suara |




Gambar 3.7 Diagram Aktivitas pengelolaan service pengenalan suara

### 3.2.2. Perancangan Arsitektur

Perangkat lunak yang akan dibangun pada Tugas Akhir ini adalah perangkat lunak berbasis Android. Gambar 3.8 merupakan ilustrasi arsitektur dari aplikasi FaceTube yang akan dibangun.




Gambar 3.8 Arsitektur Sistem Aplikasi FaceTube

Pada Gambar 3.8 dapat dilihat bahwa arsitektur sistem aplikasi FaceTube terdiri dari beberapa proses, yakni dimulai dari pengguna yang memberikan *input* perintah suara pada aplikasi yang selanjutnya akan diproses oleh *Google service*. Setelah didapatkan *string* perintah suara dari Google, aplikasi akan memprosesnya dengan menggunakan algoritma. Jika *string* tersebut lolos uji algoritma (dinyatakan cocok dengan data yang ada pada sistem) maka aplikasi akan melakukan perekaman. Setelah perekaman selesai maka aplikasi akan mengunggah video ke YouTube dan kemudian menyimpan *id link* video tersebut yang selanjutnya akan dipublikasikan menjadi status pada akun

Facebook pengguna. Aplikasi juga akan menyimpan video tersebut ke dalam memori *smartphone* pengguna.

### 3.2.3. Perancangan Algoritma Pencocokan String

Pada sub bab ini akan dijelaskan mengenai perancangan proses dari algoritma pencocokan *string* yang akan diimplementasikan dalam aplikasi.

Algoritma pencocokan *string* yang akan diimplementasikan adalah algoritma *Levenshtein Distance* dan algoritma *Soundex*. Kedua algoritma ini digunakan untuk mengantisipasi buruknya pendeteksian suara oleh mikrofon *smartphone*, agar aplikasi lebih optimal dalam melakukan pendeteksian.

Perintah suara yang diterima aplikasi akan dikirimkan ke *Google Speech Recognition* untuk dilakukan pendeteksian perintah suara. Kemudian Google akan mengirimkan hasil konversi suara berupa *string*. Sistem akan mengambil hasil teratas dari pendeteksian dan memprosesnya dengan kedua algoritma yang dipakai aplikasi untuk menentukan apakah perintah suara yang masuk termasuk perintah untuk melakukan perekaman atau bukan. Jika *string* yang telah diproses memenuhi syarat maka aplikasi akan melakukan perekaman, jika tidak maka aplikasi akan kembali pada kondisi menunggu perintah suara.

#### 3.2.3.1. Perancangan Algoritma Levenshtein Distance

Algoritma Levenshtein Distance diimplementasikan dengan melakukan sistem persentase. Persentase tersebut diperoleh dari seberapa mirip *string* yang didapat dari *Google Speech Recognition* dengan yang dimaksud oleh sistem (yaitu “REKAM”) dengan menggunakan jarak atau *distance*. Jarak yang dimaksud adalah seberapa banyak proses yang dilakukan untuk mengubah *string* masukan menjadi *string* tujuan.

Persentase yang ditentukan adalah lebih dari 90%, jika kurang dari persentase tersebut maka *string* perintah suara akan

dianggap tidak sama dengan data yang ada pada sistem sehingga aplikasi tidak akan melakukan perekaman. Rancangan algoritma Levenshtein Distance digambarkan dengan diagram alur seperti yang ditunjukkan pada Gambar 3.9.

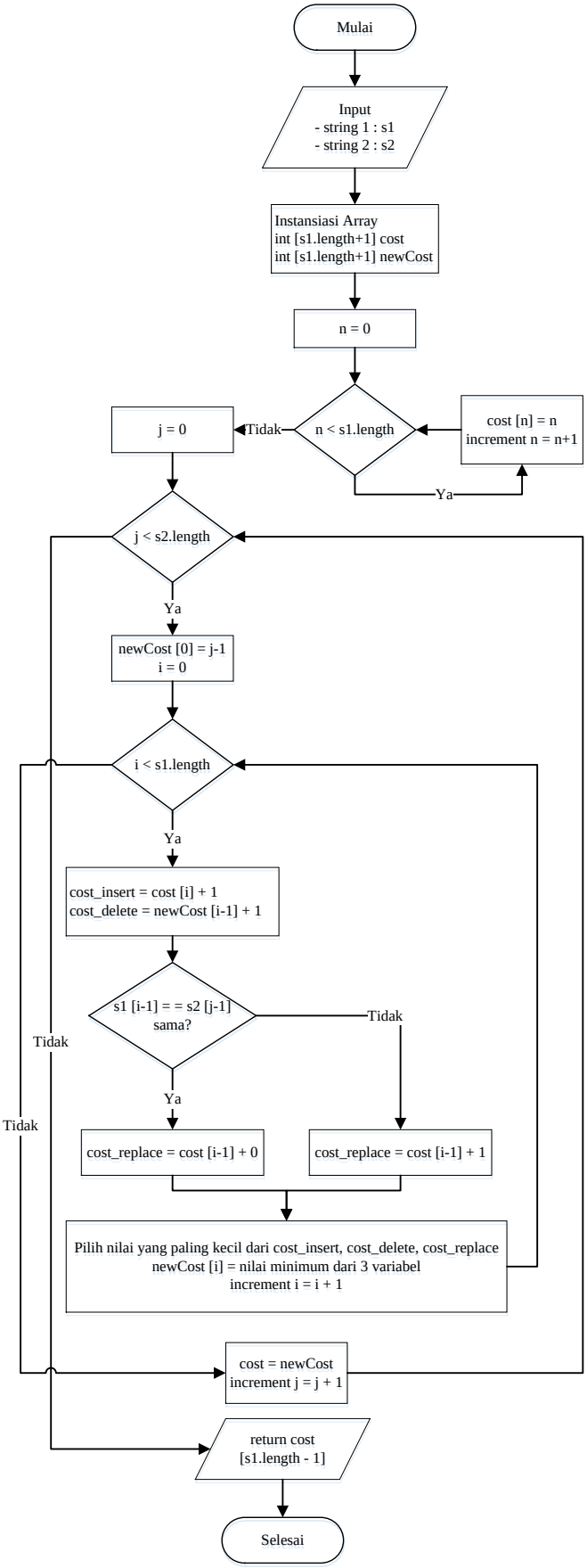
### 3.2.3.2. Perancangan Algoritma Soundex

Algoritma *Soundex* diimplementasikan dengan penyesuaian. Setelah *string* yang sebelumnya dilakukan pengecekan oleh algoritma *Levenshtein Distance* telah dinyatakan cocok maka selanjutnya akan diproses dengan algoritma *Soundex*.

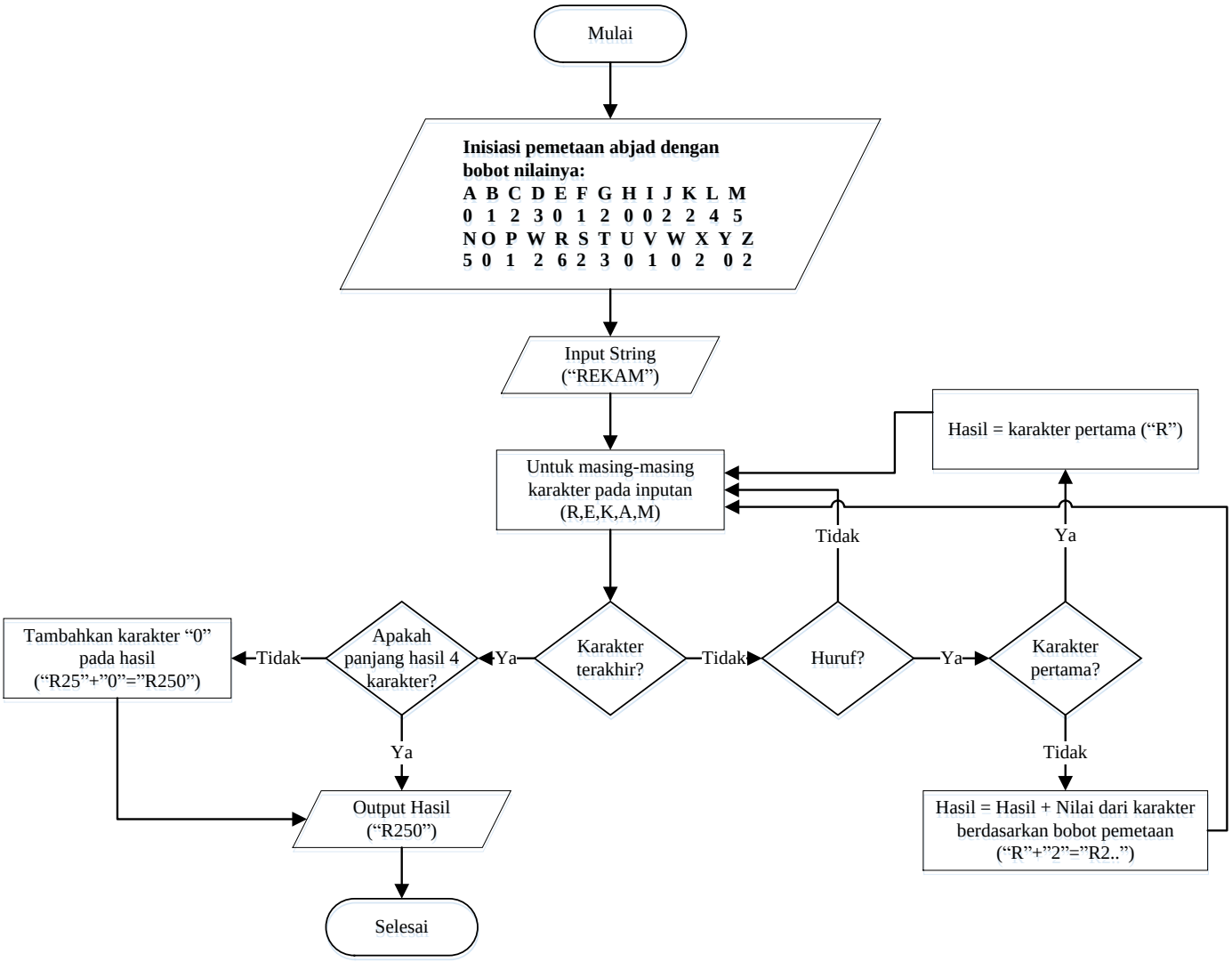
Algoritma ini akan membandingkan *string* perintah suara dengan kode fonetis yang telah ditentukan sistem sebagai kode fonetis dari perintah suara "REKAM" yakni "R250". *String* perintah suara yang dinyatakan cocok oleh algoritma pertama akan diubah menjadi kode fonetis. Yang selanjutnya akan dilakukan pencocokan dengan kode fonetis pada sistem. Jika kode fonetis sama atau memiliki persentase kemiripan lebih dari 90% maka *string* dianggap benar atau cocok. Dalam algoritma *Soundex* huruf pertama pada *string* akan diabaikan, dikarenakan huruf "R" dalam kata "REKAM" sering terjadi kesalahan pendeteksian. Sehingga hasil konversi *string* yang dilakukan pencocokan adalah kode fonetis "250" ("EKAM"). Rancangan algoritma *Soundex* digambarkan dengan diagram alur seperti yang ditunjukkan pada Gambar 3.10.

### 3.2.4. Perancangan Proses

Pada sub bab ini akan dijelaskan mengenai rancangan proses aplikasi yang digunakan untuk pencapaian suatu fungsi pada aplikasi. Proses yang ada pada sistem ini adalah proses *login*, proses pencocokan perintah suara, proses perekaman video, proses menyimpan video, unggah video, pemberian *tag reference* dan *post* status.



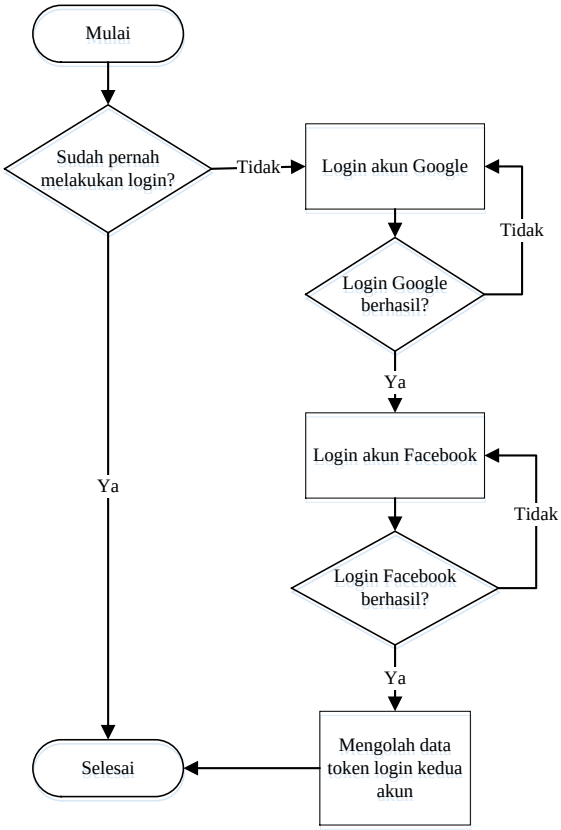

Gambar 3.9 Diagram alur perancangan algoritma Levensthein Distance




Gambar 3.10 Diagram alur perancangan algoritma Soundex

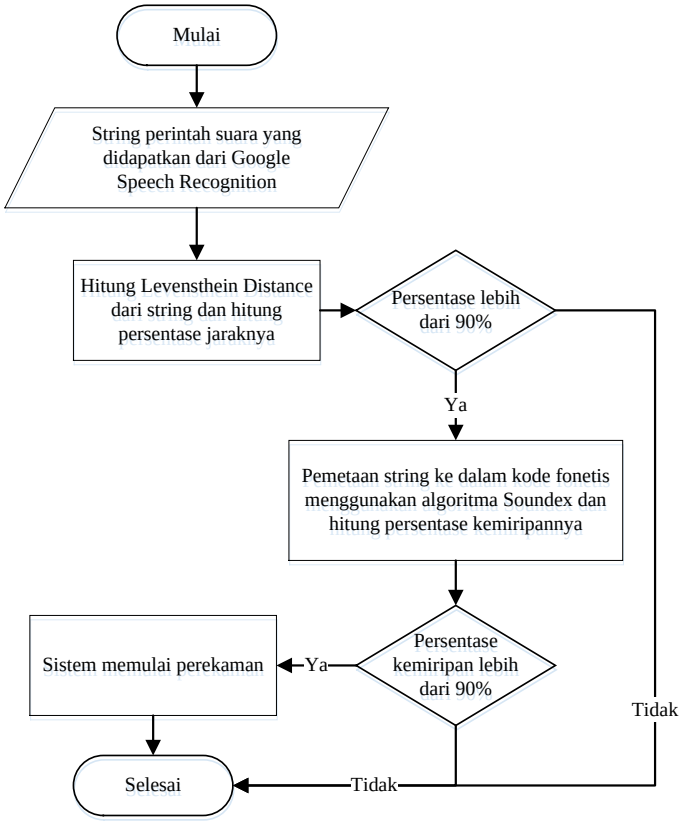
### 3.2.4.1. Rancangan Proses Login

Rancangan alur proses *login* ditunjukkan pada Gambar 3.11. Pada proses ini pengguna melakukan *login* akun YouTube dan akun Facebook miliknya. Akun YouTube akan memakai *login* akun Google. Aplikasi akan membutuhkan proses *login* karena fungsi utama aplikasi ini berkaitan dengan kedua sosial media tersebut.




Gambar 3.11 Diagram alur proses login

Untuk akun YouTube adalah bersifat wajib karena sebagai sarana pengunggahan video, sedangkan Facebook tidak bersifat wajib, dalam artian jika pengguna tidak memiliki akun Facebook aplikasi tetap dapat berjalan. Hanya saja aplikasi akan berhenti sampai proses pengunggahan saja. Jika pengguna telah melakukan *login* sebelumnya maka tidak perlu melakukan *login* terus menerus saat hendak menggunakan aplikasi karena sistem dirancang untuk dapat mengolah data token *login* kedua akun.



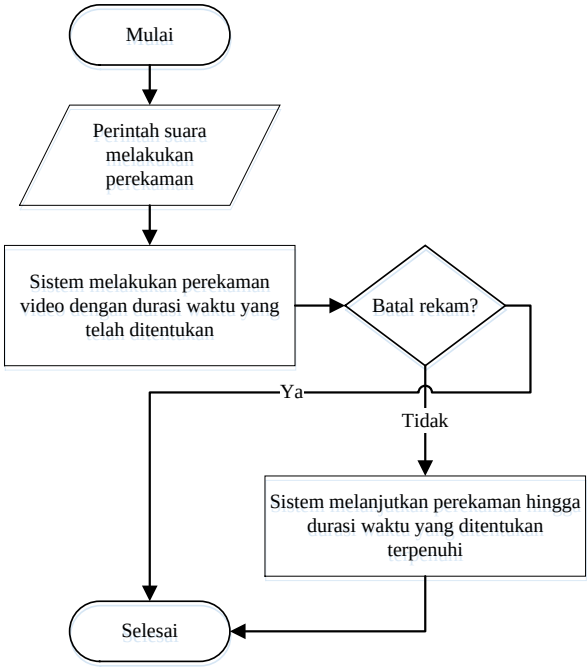

Gambar 3.12 Diagram alur proses pencocokan perintah suara

### 3.2.4.2. Rancangan Proses Pencocokan Perintah Suara

Pada proses ini sistem akan melakukan pencocokan *string* perintah suara dengan menggunakan algoritma *Levensthein Distance* dan algoritma *Soundex*. Jika *string* perintah suara dari pengguna memenuhi kriteria yang ditentukan (cocok dengan data yang terdapat dalam sistem) maka aplikasi akan melakukan perekaman, dan jika tidak cocok maka aplikasi akan kembali menjalankan *service* pengenalan suaranya. Rancangan alur proses pencocokan perintah suara ditunjukkan pada Gambar 3.12.

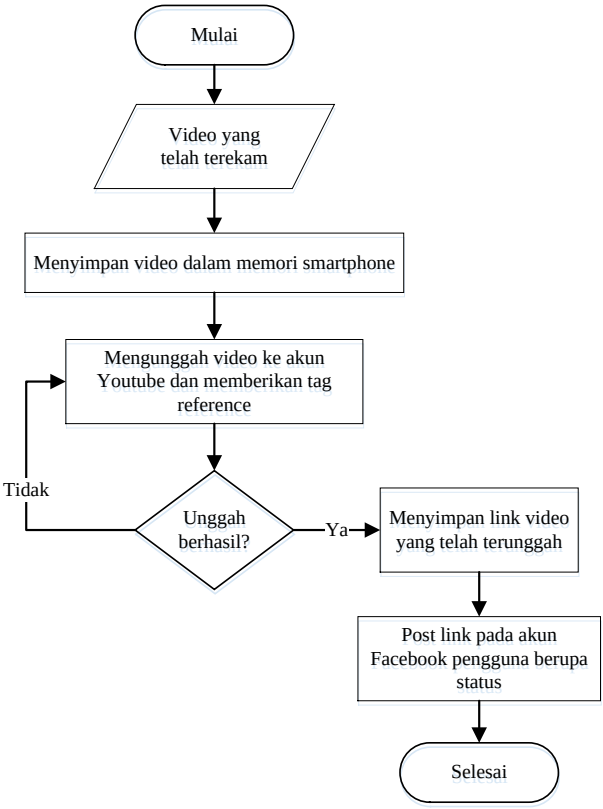
### 3.2.4.3. Rancangan Proses Perekaman Video

Rancangan alur proses perekaman video ditunjukkan pada Gambar 3.13.




Gambar 3.13 Diagram alur proses perekaman video

Pada proses ini sistem melakukan perekaman video setelah *string* perintah suara yang sebelumnya telah diproses memenuhi beberapa kriteria yang telah ditentukan algoritma. Pengguna dapat melakukan pembatalan perekaman pada saat aplikasi sedang melakukan perekaman. Pengguna hanya perlu menekan tombol batal pada saat perekaman dan sistem akan menghentikan proses perekamannya.




Gambar 3.14 Diagram alur proses menyimpan video, pemberian tag reference, unggah video dan post status

#### 3.2.4.4. Rancangan Proses Menyimpan Video, Unggah Video, Pemberian Tag Reference dan Post Status

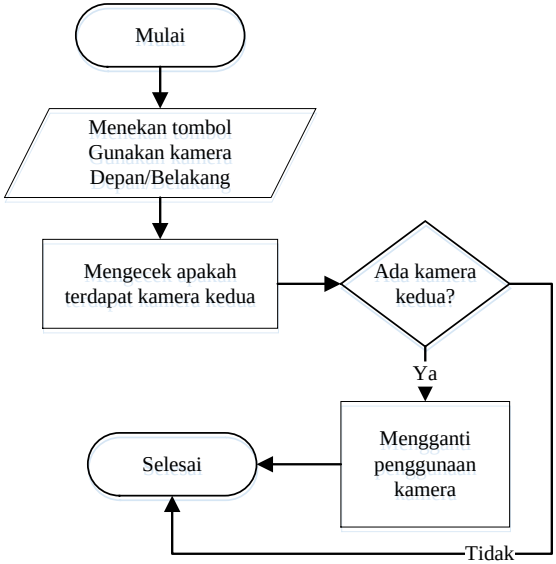
Pada proses ini pertama-tama sistem akan menyimpan video ke dalam memori *smartphone* pengguna. Dan selanjutnya sistem akan melakukan pengunggahan video ke akun YouTube dengan memberikan *tag reference* berupa waktu dan lokasi pada video tersebut. Proses terakhir adalah sistem melakukan *post status* berupa *link* tautan video YouTube pada akun Facebook pengguna. Jika pengunggahan tidak berhasil maka sistem akan mencoba mengunggah kembali video tersebut. Seluruh proses tersebut dilakukan secara otomatis oleh sistem. Rancangan proses unggah video dan *post status* ditunjukkan pada Gambar 3.14.

#### 3.2.4.5. Rancangan Proses Pilih Kamera dan Kelola Service

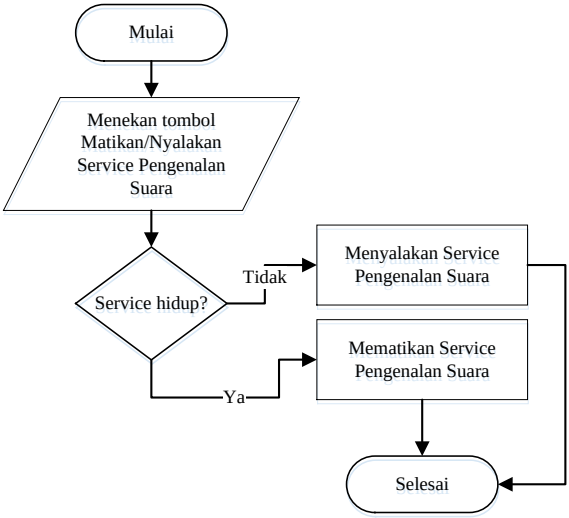
Pada proses ini pengguna memilih penggunaan kamera depan atau belakang dari perangkat *smartphone* miliknya. Jika pengguna menekan tombol Gunakan kamera depan/belakang pada halaman utama maka kamera akan berganti tergantung penggunaan pada saat itu. Sedangkan kelola *service* adalah proses untuk menghidupkan/mematikan servis pengenalan perintah suara. Sehingga pengguna dapat menghemat daya baterai *smartphone* ketika tidak menggunakan aplikasi. Rancangan proses pilih kamera ditunjukkan pada Gambar 3.15. Rancangan proses kelola *service* ditunjukkan pada Gambar 3.16.

### 3.2.5. Perancangan Antarmuka

Pada sub bab ini akan dijelaskan mengenai rancangan antarmuka pengguna. Antarmuka pengguna digunakan sebagai penghubung antara pengguna dan aplikasi. Terdapat 2 antarmuka pada aplikasi ini yang akan dijabarkan yaitu antarmuka halaman utama dan halaman perekaman video.



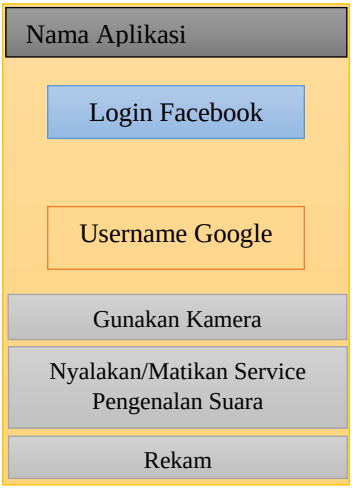

Gambar 3.15 Diagram alur proses memilih penggunaan kamera




Gambar 3.16 Diagram alur proses kelola servis pengenalan suara

### 3.2.5.1 Rancangan Antarmuka Halaman Utama

Halaman utama merupakan tampilan utama pada saat pengguna membuka aplikasi. Pada halaman ini terdapat beberapa tombol untuk menjalankan fungsi utama aplikasi. Diantaranya terdapat tombol untuk mengganti penggunaan kamera, tombol untuk menghidupkan dan mematikan *service* pengenalan suara, serta tombol untuk merekam secara manual (tanpa perintah suara). Tombol untuk perekaman secara manual ini digunakan untuk mempermudah pengguna jika lingkungan sekitar pengguna tidak memungkinkan untuk memberikan perintah suara kepada aplikasi (terlalu ramai atau terlalu banyak suara di sekitar lingkungan penggunaan aplikasi). Untuk jendela *login* Google yaitu berupa jendela *pop up*, sedangkan untuk *login* Facebook terdapat tombol *login* yang nantinya juga akan mengeluarkan jendela *pop up login* saat pengguna menekannya. Setelah pengguna berhasil *login* (Google) nantinya terdapat indikator yang menampilkan *username* akun Google pengguna yang menandakan bahwa pengguna telah *login*. Rancangan antarmuka halaman login dapat dilihat pada Gambar 3.17.




Gambar 3.17 Rancangan Antarmuka Halaman Utama

### 3.2.5.2. Rancangan Antarmuka Halaman Perekaman Video

Halaman perekaman video adalah antarmuka yang menampilkan *preview* tampilan gambar dari proses perekaman video yang sedang berlangsung. Pada halaman ini pengguna dapat membatalkan perekaman jika menekan tombol Batal, dan sistem akan menutup aplikasi. Kemudian aplikasi akan kembali menjalankan *service* pengenalan suara. Rancangan antarmuka halaman perekaman video ditunjukkan pada Gambar 3.18.


Gambar 3.18 Rancangan Antarmuka Halaman Perekaman Video

# BAB 4
# IMPLEMENTASI PERANGKAT LUNAK

Bab ini membahas implementasi dari perancangan aplikasi yang meliputi lingkungan pembangunan, implementasi algoritma, implementasi proses, implementasi antarmuka pengguna, serta implementasi terkait dalam pengembangan aplikasi.

## 4.1. Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

### 4.1.1. Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang dipakai dalam pembuatan aplikasi ini memiliki spesifikasi sebagai berikut.

1. Prosesor Intel(R) Core(TM) i3-5005U CPU @ 2.00 GHz 2.00 GHz.
2. Memori (RAM) 8.00 GB.
3. Tipe sistem 64-bit.

### 4.1.2. Lingkungan Pembangunan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk membuat aplikasi ini yakni sebagai berikut.

1. Sistem operasi Windows 10 Pro (64-bit).
2. Eclipse Kepler SDK sebagai IDE.
3. StarUML v5.0.2.1570.

## 4.2. Implementasi Algoritma

Pada sub bab ini akan dijelaskan mengenai implementasi algoritma pencocokan string yang terdiri dari algoritma *Levensthein Distance* dan *Soundex*.

### 4.2.1. Implementasi Algoritma Levenshtein Distance

Algoritma pencocokan string *Levensthein Distance* diimplementasikan sebagai fungsi dalam sebuah kelas algoritma. Proses algoritma ini diimplementasikan seperti yang sudah digambarkan dan dijelaskan alurnya pada sub bab 3.2.3.1 mengenai perancangan algoritma *Levensthein Distance*. Implementasi proses algoritma *Levensthein Distance* dapat dilihat pada Kode Sumber 4.1.

```
private int LevenshteinDistance(String s0, String s1) {
    int len0 = s0.length() + 1;
        int len1 = s1.length() + 1;

        // array jarak
    int[] cost = new int[len0];
    int[] newcost = new int[len0];

    // nilai awal pada s0
    for (int i = 0; i < len0; i++) {
        cost[i] = i;
    }

    // komputasi array jarak
    // nilai transformasi untuk setiap huruf dalam s1
    for (int j = 1; j < len1; j++) {
        // nilai awal pada s1
        newcost[0] = j - 1;
        // nilai transformasi untuk setiap huruf dalam s0
        for (int i = 1; i < len0; i++) {
            // pencocokan huruf pada kedua string
            int match = (s0.charAt(i - 1) == s1.charAt(j - 1)) ? 0 : 1;

            // nilai komputasi untuk setiap transformasi
            int cost_replace = cost[i - 1] + match;
            int cost_insert = cost[i] + 1;
            int cost_delete = newcost[i - 1] + 1;

            // menjaga nilai minimum
            newcost[i] = Math.min(Math.min(cost_insert, cost_delete),
                    cost_replace);
        }
        // menukar nilai/array nilai baru
        int[] swap = cost;
        cost = newcost;
        newcost = swap;
    }
    // jarak adalah nilai untuk trasnformasi semua huruf pada
    // kedua string
    return cost[len0 - 1];
}
```

Kode Sumber 4.1 Implementasi algoritma Levenshtein Distance

Pada implementasinya digunakan 2 *array* 1 dimensi (`cost` dan `newcost`). Kemudian akan dihitung jaraknya, yaitu nilai komputasi untuk setiap proses *transformasi* (*replace, insert* dan *delete*) antar kedua string yang dicocokkan. Parameter fungsi ini adalah dua *String* yang akan dihitung *Levensthein Distance*-nya. Tipe nilai kembalian dari fungsi ini adalah *Integer*.

### 4.2.2. Implementasi Algoritma Soundex

Algoritma pencocokan string *Soundex* diimplementasikan sebagai fungsi dalam sebuah *kelas* algoritma. Implementasi algoritma *Soundex* dapat dilihat pada Kode Sumber 4.2.

Proses algoritma ini diimplementasikan seperti yang sudah digambarkan dan dijelaskan alurnya pada sub bab 3.2.3.2 mengenai perancangan algoritma *Soundex*. Pada algoritma ini dilakukan pemetaan kode fonetis pada string perintah suara dan dicocokkan dengan pemetaan yang ada pada sistem. Parameter fungsi ini adalah *String* yang akan dicari kode fonetisnya. Tipe nilai kembalian dari fungsi ini adalah *String*.

## 4.3. Implementasi Proses

Pada sub bab ini akan diuraikan implementasi dari proses aplikasi FaceTube yang terdiri dari proses *login*, proses pencocokan perintah suara, perekaman video, pilih kamera, kelola servis, serta proses menyimpan video, unggah video, pemberian *tag reference* dan *post status*. Kode Sumber lengkap dari implementasi semua proses dapat dilihat pada LAMPIRAN A KODE SUMBER.

### 4.3.1. Implementasi Proses Login

Proses *login* aplikasi diimplementasikan dengan *login* dari akun YouTube dan Facebook pengguna. Berikut ini adalah penjabaran masing-masing proses *login*.

```
/* Implements mapping from: AEHIOUWYBFPVCGJKQSXZDTLMNR to:
   00000000111122222222334556 */
public static final char[] MAP = {
  // A B C D E F G H I J K L M
  '0', '1', '2', '3', '0', '1', '2', '0', '0', '2', '2', '4', '5',
  // N O P W R S T U V W X Y Z
  '5', '0', '1', '2', '6', '2', '3', '0', '1', '0', '2', '0', '2' };

public String Soundex(String s) {
  // Algorithm works on upper case (mainframe era).
  String t = s.toUpperCase(Locale.getDefault());

  StringBuffer res = new StringBuffer();
  char c, prev = '?';

  // Main loop: find up to 4 chars that map.
  for (int i = 0; i < t.length() && res.length() < 4
       && (c = t.charAt(i)) != ','; i++) {
      // Check character = alphabetic.
      // Also, skip double letters.
      if (c >= 'A' && c <= 'Z' && c != prev) {
          prev = c;
          // First char is installed unchanged, for sorting.
          if (i == 0)
              res.append(c);
          else {
              char m = MAP[c - 'A'];
              if (m != '0')
                  res.append(m);
          }
      }
  }

  if (res.length() == 0)
      return null;

  for (int i = res.length(); i < 4; i++)
      res.append('0');

  return res.toString();
}
```

Kode Sumber 4.2 Implementasi algoritma Soundex

#### 4.3.1.1. Login YouTube (Google API)

Proses *login* ke YouTube menggunakan Google API (*OAuth2*). Implementasi proses pada Kode Sumber 4.3 akan memunculkan menu *pop-up* yang berisi daftar akun Google yang tersedia pada perangkat *smartphone* pengguna. Pengguna bisa memilih satu akun yang tersedia ataupun menambah akun lain yang belum terdapat pada daftar tersebut. Proses selanjutnya diimplementasikan pada Kode Sumber 4.4 yaitu nama akun yang telah terautentifikasi, akan disimpan pada *SharedPreference* dari

perangkat pengguna. Untuk kemudian sewaktu-waktu bisa dimuat oleh aplikasi untuk proses-proses yang membutuhkannya.

```
public class MainActivity extends Activity implements
 UserFragment.Callbacks,{

     public void onCreate(Bundle savedInstanceState) {
     ...
     // Check to see if the proper keys and play-list IDs
     // have been set up
     if (isCorrectlyConfigured()) {
          credential = GoogleAccountCredential.usingOAuth2(
                  getApplicationContext(),
                                          Arrays.asList(Auth.SCOPES));
          // set exponential back-off policy
          credential.setBackOff(new ExponentialBackOff());
          loadAccount();
          if (mChosenAccountName == null) chooseAccount();
          credential.setSelectedAccountName(mChosenAccountName);
     }
              ...

 }

 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data)
 {
     switch (requestCode) {
          ...
     case REQUEST_ACCOUNT_PICKER:
          if (resultCode == Activity.RESULT_OK && data != null
                  && data.getExtras() != null) {
              String accountName = data.getExtras().getString(
                      AccountManager.KEY_ACCOUNT_NAME);
              if (accountName != null) {
                  mChosenAccountName = accountName;
                  credential.
                                          setSelectedAccountName(accountName);
                  saveAccount();
                  mUserFragment.setProfileInfo(accountName);
              }
          }
          break;
                      ...
     }
 }
}
```

Kode Sumber 4.3 Implementasi proses login Youtube (Google API)

### 4.3.1.2. Login Facebook (Facebook API)

Proses *login* ke Facebook menggunakan Facebook API. Untuk pengguna yang baru pertama kali menggunakan aplikasi ini, implementasi proses pada Kode Sumber 4.5 akan menampilkan

layar persetujuan dari Facebook, yang menampilkan fitur-fitur apa saja yang akan diakses oleh aplikasi ini, yang membutuhkan izin dari pengguna.

```
public class MainActivity extends Activity implements
  UserFragment.Callbacks,{
  …
      private void chooseAccount() {
      startActivityForResult(credential.newChooseAccountIntent(),
              REQUEST_ACCOUNT_PICKER);
  }

  private void saveAccount() {
      SharedPreferences sp = PreferenceManager
              .getDefaultSharedPreferences(this);
      sp.edit().putString(VideoUploadActivity.ACCOUNT_KEY,
      mChosenAccountName).commit();
  }

  private void loadAccount() {
      SharedPreferences sp = PreferenceManager
              .getDefaultSharedPreferences(this);
      mChosenAccountName =
      sp.getString(VideoUploadActivity.ACCOUNT_KEY, null);
      invalidateOptionsMenu();
  }

  private boolean isCorrectlyConfigured() {
      return Auth.KEY.length() != 0;
  }
}
```

Kode Sumber 4.4 Implementasi menyimpan dan menampilkan nama akun Google pengguna

Kode Sumber 4.6 menjelaskan tentang pengelolaan token pada proses *login* Facebook. Untuk keperluan aplikasi ini diperlukan izin "*publish_action*" untuk keperluan *posting* ke Facebook pengguna. Izin tersebut memerlukan kita sebagai pengembang aplikasi untuk mengirimkan aplikasi kita untuk di-*review* oleh Facebook.

Cara yang lain adalah dengan melakukan *login* untuk kedua kalinya dengan menggunakan *loginWithPublishPermission*. Pada aplikasi ini digunakan cara kedua, yang dilakukan langsung setelah *login* yang pertama berhasil.

```
public class MainActivity extends Activity implements
  FacebookCallback<LoginResult>{
  ...
  @Override
  public void onCreate(Bundle savedInstanceState) {
                ...
       FacebookSdk.sdkInitialize(context);
       callbackManager = CallbackManager.Factory.create();

       loginButton = (LoginButton) this.findViewById(R.id.login_button);
       loginButton.setReadPermissions("user_friends");
       loginButton.setReadPermissions("public_profile");
       loginButton.setReadPermissions("email");
       // loginButton.setReadPermissions("publish_actions");
       // Callback registration
       loginButton.registerCallback(callbackManager, this);

       mIsPost = false;

       accessTokenTracker = new AccessTokenTracker() {
            @Override
            protected void onCurrentAccessTokenChanged(AccessToken arg0,
                     AccessToken arg1) {
                if (arg1 == null) {
                     // what to do when user logout
                     mIsPost = false;
                }
            }
       };
  }
}
```

Kode Sumber 4.5 Implementasi proses login Facebook

Kedua proses *login* tersebut, *login* yang pertama dengan *login* yang kedua dengan *loginWithPublishPermission*, menghasilkan *Activity Result Code* yang sama, yaitu 64206. Sehingga proses *login* yang kedua akan dipanggil berkali-kali, karena proses *login* yang kedua ini dilakukan ketika proses *login* berhasil. Oleh karena itu dibutuhkan *AccessTokenTracker*, yang digunakan untuk pelacakan perubahan *token* yang didapat dari Facebook API setelah proses *login* yang pertama. Perubahan *token* ini tidak akan terjadi pada *login* yang kedua.

### 4.3.2. Implementasi Proses Pencocokan Perintah Suara

Kode Sumber 4.7 adalah kode sumber dari implementasi proses pencocokan perintah suara secara keseluruhan.

```
public class MainActivity extends Activity implements
  FacebookCallback<LoginResult>{
  …
  @Override
  public void onCancel() {
       // TODO Auto-generated method stub
       Log.d("FBLOGIN", "Cancel");
  }

  @Override
  public void onError(FacebookException arg0) {
       // TODO Auto-generated method stub
       Log.d("FBLOGIN", "Error");
  }

  @Override
  public void onSuccess(LoginResult arg0) {
       // TODO Auto-generated method stub
       Log.d("FBLOGIN", "Success");
       if (!accessTokenTracker.isTracking()) {
            accessTokenTracker.startTracking();
       }
       if (!mIsPost) {
            LoginManager.getInstance().logInWithPublishPermissions(this,
                      Arrays.asList("publish_actions"));
            mIsPost = true;
       }
  }
}
```

Kode Sumber 4.6 Implementasi pengelolaan token Facebook

Hasil pendeteksian dari *speech recognition* Google akan diproses dan dilakukan pencocokan dengan algoritma pencocokan string, hasilnya adalah berupa persentase kemiripan (untuk *Soundex*) dan persentase besarnya jarak (untuk *LD*). Dengan syarat kecocokan sebesar lebih dari 90%. Pada implementasi ini setiap kata yang terdeteksi dan terproses akan dimunculkan dengan `Toast` pada layar perangkat. Setelah proses pencocokan selesai maka *speech recognizer* akan dilakukan *restart* untuk pendeteksian suara kembali.

Kode Sumber 4.8 adalah merupakan kode sumber dari perhitungan persentase tingkat kecocokan dari output *Levenshtein Distance.*

```
private void ProcessSpeechRecognitionResult(ArrayList<String> results) {
  String string = "";
  int maxMatchIdx = 0;
  int maxMatchPctg = 0;

  for (int i = 0; i < results.size(); i++) {
       string = results.get(i);
       int l1 = pecentageOfTextMatch(string.substring(1), "ekam");
       if (l1 > maxMatchPctg) {
            maxMatchPctg = l1;
            maxMatchIdx = i;
       }
       if (maxMatchPctg == 100) {
            break;
       }
  }

  string = results.get(maxMatchIdx);

  Log.e("Laferstein result: ", String.valueOf(maxMatchPctg) + "%");

  String soundexResult = soundex(string);

  Log.e("Soundex: ", soundexResult + "," + string);// REKAM=R250

  Toast toast = Toast.makeText(this, String.format("terdengar:\n \"%s\"",
  string), Toast.LENGTH_LONG);
  // toast.setGravity(Gravity.CENTER, 0, 0);
  toast.show();
  // matching string more than 90%
  if (maxMatchPctg > 90 && soundexResult.substring(1).equals("250")) {
       SpeechRecognitionService.this.stopSelf();
       // mSpeechRecognizer.stopListening();
       mIsRunning = false;

       Intent i = new Intent();
       i.setClass(this, VideoCaptureActivity.class);
       i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
       startActivity(i);
  } else {
       if (mNoSpeechCountDown == null) {
            mNoSpeechCountDown = new CountDownTimer(2000, 500) {
                 @Override
                 public void onTick(long l) {
                 }
                 @Override
                 public void onFinish() {
                      Log.d("Speech",
                 "Timer.onFinish: Timer Finished, Restart recognizer");
                      // mSpeechRecognizer.stopListening();
                      StartListening(true);
                 }
            };
       }
       mNoSpeechCountDown.start();
  }
}
```

Kode Sumber 4.7 Implementasi proses pencocokan perintah suara

```
private int pecentageOfTextMatch(String s0, String s1) {

      int percentage = 0;

      s0 = s0.trim().replaceAll("\\s+", " ");
      s1 = s1.trim().replaceAll("\\s+", " ");

      percentage = (int) (100 - ((float) LevenshteinDistance(s0, s1) * 100 /
      (float) (Math.max(s0.length(), s1.length()))));

      return percentage;
}
```

Kode Sumber 4.8 Implementasi perhitungan persentase Levensthein Distance

### 4.3.3. Implementasi Proses Perekaman Video

Implementasi proses perekaman video merupakan bagian inti dari aplikasi FaceTube, Kode Sumber 4.9 berisi kode sumber dari proses perekaman video. Kelas `mediaRecorder` merupakan kelas untuk melakukan perekaman video. Dan kelas `cameraPreview` untuk menampilkan preview gambar saat perekaman berjalan. Pada implementasi proses ini terdapat juga implementasi tombol rekam `btn_record` untuk perekaman langsung (tanpa perintah suara).

Dan Kode Sumber 4.10 adalah kode sumber dari implementasi pengelolaan dan perekaman video pada perangkat. Proses ini diawali dengan menentukan lokasi penyimpanan video yang akan direkam dan nama file dari video tersebut. Penamaan *file* video diambil dari kapan waktu video tersebut direkam, dengan format "video_TahunBulanTanggalJamMenitDetik". Penyimpanan video akan diarahkan ke folder bernama "FaceTube" pada *External Storage* perangkat (*.getExternalStorageDirectory*). Format dari video hasil perekaman adalah *.mp4*.

### 4.3.4. Implementasi Proses Menyimpan Video, Unggah Video, Pemberian Tag Reference dan Post Status

Pada bagian ini akan ditampilkan kode sumber implementasi dari proses menyimpan video, proses unggah video,

pemberian *tag reference* dan proses *post status*. Proses menyimpan video adalah proses penyimpanan video ke dalam memori *eksternal smartphone*. Sedangkan proses unggah dan pemberian *tag reference* adalah pengunggahan video ke akun YouTube pengguna yang disertai pemberian tag referensi berupa lokasi dan waktu. *Post status* adalah *post link* pada akun Facebook pengguna.

```
public void initialize() {
    mPreview = new CameraPreview(myContext, mCamera);
    mPreview.setOnSurfaceCreatedEventListener(new OnSurfaceCreatedEventListener()
    {
        @Override
        public void OnSurfaceCreated(SurfaceHolder holder) {
            if (!recording) {
                Surface s = holder.getSurface();
                record_start(holder);
                mediaRecorder.setPreviewDisplay(s);
                boolean isSuccessInit = false;
                try {
                    mediaRecorder.prepare();
                    isSuccessInit = true;
                }
                ...
            }
        }
    });

    cameraPreview.addView(mPreview);
    mPreview.SET_TEXT("FaceTube 1.0");

    SET_RECORD(0);

    btn_record = (Button) this.findViewById(R.id.btn_record);

    mHandler = new Handler();
    btn_record.setOnClickListener(this);
    refreshVoiceSettings();
}
```

Kode Sumber 4.9 Implementasi proses perekaman video

### 4.3.4.1. Menyimpan Video pada memori smartphone

Proses menyimpan video pada memori *smartphone* seperti yang sudah sedikit dijelaskan pada sub bab 4.3.3. Yaitu implementasi proses dari penyimpanan video yang telah terekam pada memori *eksternal* perangkat *smartphone*. Pada Kode Sumber 4.11 dapat dilihat bahwa penamaan dari *file* video yang hendak disimpan adalah dengan format "video_yyyyMMddhhmmss" yang urutannya adalah

TahunBulanHariJamMenitDetik dan diambil dari *kelas* `Calendar`. Video akan disimpan di dalam folder `FaceTube` dengan tujuan memori *eksternal* (`Environment.getExternalStorageDirectory`). Format video juga di-*set* untuk format *.mp4*.

```
private void record_start(SurfaceHolder h) {
 c = Calendar.getInstance();
 ds = new SimpleDateFormat("yyyyMMddhhmmss", Locale.getDefault());
 String DATE_X = ds.format(c.getTime());
 String path = String.format("%s/FaceTube/", Environment
         .getExternalStorageDirectory().getAbsolutePath());
 File dir = new File(path);
 if (!dir.exists()) {
     dir.mkdirs();
 }

 file_name = String.format("%svideo_%s", path, DATE_X);
 record_name = String.format("%s.mp4", file_name);

 // Camera.Size s = mPreview.getSize(screenWidth, screenHeight);
 CamcorderProfile profile =
     CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH);
 Camera.Parameters parameters = mCamera.getParameters();
 Parameters
 .setPreviewSize(profile.videoFrameWidth,profile.videoFrameHeight);

 mCamera.setParameters(parameters);

 try {
     mCamera.setPreviewDisplay(h);
     mCamera.startPreview();
 } catch (IOException e) {
     Log.d(TAG, e.getMessage());
 }

 mCamera.lock();
 mCamera.unlock();

 mediaRecorder.setOnErrorListener(this);
 mediaRecorder.setOnInfoListener(this);
 mediaRecorder.setCamera(mCamera);
 mediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
 mediaRecorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
 mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
 mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
 mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
 mediaRecorder.setOutputFile(record_name);
 mediaRecorder.setMaxDuration(5000);
 mediaRecorder.
 setVideoSize(profile.videoFrameWidth,profile.videoFrameHeight);
}
```

Kode Sumber 4.10 Implementasi pengelolaan serta perekaman video pada perangkat

```
private void record_start(SurfaceHolder h) {
    c = Calendar.getInstance();
    ds = new SimpleDateFormat("yyyyMMddhhmmss", Locale.getDefault());
    String DATE_X = ds.format(c.getTime());
    String path = String.format("%s/FaceTube/", Environment
            .getExternalStorageDirectory().getAbsolutePath());
    File dir = new File(path);
    if (!dir.exists()) {
        dir.mkdirs();
    }

    file_name = String.format("%svideo_%s", path, DATE_X);
    record_name = String.format("%s.mp4", file_name);
}
```

Kode Sumber 4.11 Implementasi penyimpanan video ke dalam memori smartphone

#### 4.3.4.2. Unggah Video ke YouTube

Sub bab ini berisikan implementasi dari proses pengunggahan video ke YouTube menggunakan YouTube API v3. Dengan proses awal yaitu *set upload type* menjadi *Resumable Upload* dimana pada proses pengunggahan ini tipe unggah yang dipakai adalah tipe unggah yang dapat di *resume* atau dilanjutkan prosesnya jika terjadi *error*. Jadi untuk `setDirectUploadEnabled` di *set* menjadi `false`. Kemudian ditambahkan *event listener* atau *progress listener* untuk meng-*handle* keadaan/*state* pengunggahan menggunakan `case`, berupa notifikasi pada perangkat seperti pengunggahan dimulai, sedang proses, sukses ataupun gagal. Setelah proses pengunggahan selesai dilakukan maka akan disimpan *id* dari *URL* video tersebut yang nantinya akan di post ke Facebook. Implementasi lebih lanjut dapat dilihat pada Kode Sumber 4.12.

#### 4.3.4.3. Pemberian Tag Reference

Untuk mendapatkan posisi (*longitude* dan *latitude*) dari pengguna pada saat perekaman, digunakan *GPS* dari perangkat pengguna. Kode Sumber 4.13 adalah potongan kode sumber yang mengakses *Location Service* dari *GPS* perangkat pengguna. Pertama-tama akan dilakukan pengecekan status dari *GPS*

perangkat. Kemudian pengecekan status dari penyedia layanan *network*, untuk selanjutnya melakukan *request* lokasi berupa `latitude` dan `longitude`.

```
  public static String upload(YouTube youtube, final InputStream
   fileInputStream,
       final long fileSize, final Uri mFileUri, final String path,
       final Context context) {
       …
   Video videoObjectDefiningMetadata = new Video();
       ...

   InputStreamContent mediaContent = new InputStreamContent(
           VIDEO_FILE_FORMAT, new BufferedInputStream(fileInputStream));
           mediaContent.setLength(fileSize);

   YouTube.Videos.Insert videoInsert = youtube.videos().insert(
               "snippet,statistics,status,recordingDetails",
                   videoObjectDefiningMetadata, mediaContent);

   // Set the upload type and add event listener.
   MediaHttpUploader uploader = videoInsert.getMediaHttpUploader();

   uploader.setDirectUploadEnabled(false);

   MediaHttpUploaderProgressListener progressListener =
               new MediaHttpUploaderProgressListener() {
           public void progressChanged(MediaHttpUploader uploader)
               throws IOException {
               switch (uploader.getUploadState()) {
                   case INITIATION_STARTED:
                                                       …
                       break;
                   case INITIATION_COMPLETE:
                                                       …
                       break;
                   case MEDIA_IN_PROGRESS:
                                                       …
                       break;
                   case MEDIA_COMPLETE:
                       …
                       break;
                   case NOT_STARTED:
                       …
                       break;
                   }
               }
           };
   uploader.setProgressListener(progressListener);

   // Execute upload.
   Video returnedVideo = videoInsert.execute();
   videoId = returnedVideo.getId();
       …

   return videoId;
}
```

Kode Sumber 4.12 Implementasi proses unggah video ke Youtube menggunakan Youtube API

```
public class GPSTracker extends Service implements LocationListener {
    ...
        public Location getLocation() {
        try {
            locationManager = (LocationManager) mContext
                    .getSystemService(LOCATION_SERVICE);

            // getting GPS status
            isGPSEnabled = locationManager
                    .isProviderEnabled(LocationManager.GPS_PROVIDER);

            // getting network status
            isNetworkEnabled = locationManager
            .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

            if (!isGPSEnabled && !isNetworkEnabled) {
                // no network provider is enabled
            } else {
                this.canGetLocation = true;
                // First get location from Network Provider
                if (isNetworkEnabled) {
                    locationManager.requestLocationUpdates(
                    LocationManager.NETWORK_PROVIDER,
                    MIN_TIME_BW_UPDATES,
                    MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

                                            Log.d("Network", "Network");
                    if (locationManager != null) {
                        location = locationManager
                        .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                        if (location != null) {
                            latitude =
                                                        location.getLatitude();
                            longitude =
                                                        location.getLongitude();
                        }
                    }
                }
                // if GPS Enabled get lat/long using GPS Services
                if (isGPSEnabled) {
                    if (location == null) {
                        locationManager.requestLocationUpdates(
                        LocationManager.GPS_PROVIDER,
                        MIN_TIME_BW_UPDATES,
                        MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                        Log.d("GPS Enabled", "GPS Enabled");
                        if (locationManager != null) {
                            location = locationManager
                            .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            if (location != null) {
                                latitude =
                                                        location.getLatitude();
                                longitude =
                                                        location.getLongitude();
                            }
                        }
                    }
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }

        return location;
    }
        ...
}
```

Kode Sumber 4.13 Implementasi proses mengakses Location Service GPS perangkat pengguna

Jika *GPS* pada perangkat menyala maka data yang diambil adalah *latitude* dan *longitude* dari penyedia layanan *GPS* untuk keakuratan yang lebih terjamin.

Sedangkan untuk mengetahui nama alamat dari lokasi yang didapatkan dari *GPS*, digunakan Google Maps API. Kode Sumber 4.14 adalah potongan kode sumber dari fungsi tersebut. Nama lokasi didapatkan dari *maps.google*. Data *latitude* dan *longitude* yang didapatkan dari *GPS* sebelumnya akan diinputkan untuk didapatkan nama lokasinya menggunakan *json*.

```
public static JSONObject getLocationInfo(double lat, double lng) {
  HttpGet httpGet = new HttpGet(
    "http://maps.googleapis.com/maps/api/geocode/json?latlng="
  + lat + "," + lng + "&sensor=true");
  HttpClient client = new DefaultHttpClient();
  HttpResponse response;
  StringBuilder stringBuilder = new StringBuilder();
  try {
       response = client.execute(httpGet);
       HttpEntity entity = response.getEntity();
       InputStream stream = entity.getContent();
       int b;
       while ((b = stream.read()) != -1) {
            stringBuilder.append((char) b);
       }
  } catch (ClientProtocolException e) {
  } catch (IOException e) {
  }
  JSONObject jsonObject = new JSONObject();
  try {
       jsonObject = new JSONObject(stringBuilder.toString());
  } catch (JSONException e) {
       e.printStackTrace();
  }
       return jsonObject;
}
```

Kode Sumber 4.14 Implementasi proses pengambilan nama lokasi dengan Google Maps API

### 4.3.4.4. Post Status ke Facebook

Proses penulisan status pada Facebook menggunakan Facebook Graph API pada Kode Sumber 4.15.

*Post* melalui Facebook adalah berupa *string* yang berisi *id link* video YouTube yang telah dimodifikasi menjadi sebuah *link* YouTube lengkap. *Post* status pada Facebook ini menggunakan `GraphRequest`.

```
public class FbManager {
        ...
        private void post(final String msg) {
                Log.d(LOGTAG, "facebook posting new message");
                @SuppressWarnings("unused")
                Set<String> permissions = AccessToken.getCurrentAccessToken()
                                .getPermissions();
                AccessToken accessToken = AccessToken.getCurrentAccessToken();

                Bundle postParams = new Bundle();
                postParams.putString("message", msg);

                GraphRequest request = new GraphRequest(accessToken, "me/feed",
                        postParams, HttpMethod.POST, new GraphRequest.Callback() {
                        @Override
                        public void onCompleted(GraphResponse graphResponse) {
                                Log.d("POST", "post page response::" + graphResponse);
                        }
                });
                GraphRequestAsyncTask asynTaskGraphRequest =
                                new GraphRequestAsyncTask(request);
                asynTaskGraphRequest.execute();
        }
        ...
}
```

Kode Sumber 4.15 Implementasi proses post status ke Facebook menggunakan Facebook Graph API

### 4.3.5. Implementasi Proses Pilih Kamera dan Kelola Service

Pada bagian ini akan ditampilkan kode sumber dari implementasi proses pilih kamera dan pengelolaan servis pengenalan suara.

#### 4.3.5.1. Proses Pilih Kamera Depan/Belakang

Implementasi proses pilih kamera diawali dengan *setting* dari *kelas* `MainActivity`. Pada *kelas* ini dilakukan pengaturan pada *toggle button* pilih kamera, apakah tombol tersebut aktif atau tidak. Jika tidak aktif itu berarti menandakan aplikasi memakai kamera belakang dan sebaliknya. Selanjutnya pada *kelas* `VideoCaptureActivity`, dilakukan persiapan kamera terlebih dahulu dan dilakukan pengecekan apakah terdapat kamera depan pada perangkat (`isFrontCamera ? findFrontFacingCamera()`).

Jika terdapat kamera depan maka aplikasi akan mengganti penggunaan kamera pada saat perekaman dimulai. Jadi disaat pengguna menekan tombol *toggle*, aplikasi tidak langsung mengganti penggunaan kamera akan tetapi melakukan *restart*

servis terlebih dahulu, setelah perekaman dimulai barulah aplikasi mengganti penggunaan kamera.

```
public class MainActivity extends Activity {
        ...
        Intent intent = new Intent(MainActivity.this, VideoCaptureActivi-
ty.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    intent.putExtra("isFrontCamera", btn_UseFrontCamera.isChecked());
    MainActivity.this.startActivity(intent);
        ...
}

public class VideoCaptureActivity extends Activity{
    public void onCreate(Bundle savedInstanceState) {
        ...
        if (callerIntent != null && (extras = callerIntent.getExtras()) != null
                && extras.containsKey("isFrontCamera")) {
            isFrontCamera = extras.getBoolean("isFrontCamera");
        }

        if (mCamera == null) {
            mCamera = Camera.open(isFrontCamera ? findFrontFacingCamera()
                    : findBackFacingCamera());
            mPreview.refreshCamera(mCamera);
        }
                ...
    }
    private int findFrontFacingCamera() {
        int cameraId = -1;
        int defaultCamera = -1;
        int numberOfCameras = Camera.getNumberOfCameras();
        for (int i = 0; i < numberOfCameras; i++) {
            CameraInfo info = new CameraInfo();
            Camera.getCameraInfo(i, info);
            defaultCamera = i;
            if (info.facing == CameraInfo.CAMERA_FACING_FRONT) {
                cameraId = i;
                cameraFront = true;
                break;
            }
        }
        return cameraId < 0 ? defaultCamera : cameraId;
    }

    private int findBackFacingCamera() {
        int cameraId = -1;
        int defaultCamera = -1;
        int numberOfCameras = Camera.getNumberOfCameras();
        for (int i = 0; i < numberOfCameras; i++) {
            CameraInfo info = new CameraInfo();
            Camera.getCameraInfo(i, info);
            defaultCamera = i;
            if (info.facing == CameraInfo.CAMERA_FACING_BACK) {
                cameraId = i;
                cameraFront = false;
                break;
            }
        }
        return cameraId < 0 ? defaultCamera : cameraId;
    }
}
```

Kode Sumber 4.16 Implementasi proses pilih kamera depan/belakang

Jika pada perangkat tidak terdapat kamera depan maka aplikasi akan tetap memakai kamera belakang untuk melakukan

perekaman (diimplementasikan pada fungsi `findBackFacingCamera()`). Implementasi pilih kamera ditunjukkan pada Kode Sumber 4.16.

```
public class MainActivity extends Activity{
    public void onCreate(Bundle savedInstanceState) {
    ...

        btn_stopspeechrecogsvc = (ToggleButton) this
                .findViewById(R.id.btn_StopSpeechRecogSvc);
        btn_stopspeechrecogsvc.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (SpeechRecognitionService.mIsRunning) {
                    SpeechRecognitionService
                    .stopContinuousListening(MainActivity.this);
                } else {
                    if (btn_UseFrontCamera == null) {
                        btn_UseFrontCamera = (ToggleButton) MainActivity.this
                        .findViewById(R.id.btn_UseFrontCamera);
                    }

                    SpeechRecognitionService.startContinuousListening(
                        MainActivity.this, btn_UseFrontCamera.isChecked());
                }
            }
        });
        ...
        }
}

public class SpeechRecognitionService extends Service{
    public static void startContinuousListening(Context context,
            boolean isFrontCamera) {
        if (!mIsRunning) {
            Intent service = new Intent(context,
                                    SpeechRecognitionService.class);
            service.putExtra("isFrontCamera", isFrontCamera);
            context.startService(service);
            mIsRunning = true;
        }
    }

    public static void stopContinuousListening(Context context) {
        if (mIsRunning) {
            Intent service = new Intent(context,
                                    SpeechRecognitionService.class);
            context.stopService(service);
            mIsRunning = false;
        }
    }
}
```

Kode Sumber 4.17 Implementasi proses menghidupkan/mematikan servis pengenalan suara

### 4.3.5.2. Proses Menghidupkan/Mematikan Servis Pengenalan Suara

Implementasi pengelolaan servis (menghidupkan dan mematikan servis pengenalan suara) juga mengimplementasikan

`ToggleButton` sebagai *switch*-nya. Pertama kali aplikasi dibuka servis pengenalan suara langsung menyala, sehingga *toggle button* berwarna hijau (menandakan servis sedang menyala), kemudian dilakukan pengecekan jika tombol ditekan pada saat servis sedang menyala (diimplementasikan dengan `onClickListener`) maka servis akan dimatikan. Dan sebaliknya, jika *toggle button* tidak menyala (menandakan servis sedang tidak berjalan) kemudian tombol ditekan maka *service* akan dinyalakan kembali. Implementasi proses menghidupkan/mematikan servis pengenalan suara ditunjukkan pada Kode Sumber 4.17.

## 4.4. Implementasi Antarmuka

Pada bagian ini dijelaskan mengenai implementasi tampilan antarmuka perangkat lunak. Implementasi tampilan antarmuka dibuat sesuai hasil perancangan antarmuka pada sub bab 3.2.5. Terdapat dua halaman implementasi antarmuka pada aplikasi FaceTube, yaitu antarmuka halaman utama dan halaman perekaman video. Kode Sumber lengkap dari implementasi antarmuka dapat dilihat pada LAMPIRAN A KODE SUMBER.

### 4.4.1. Antarmuka Halaman Utama

Antarmuka halaman utama untuk aplikasi FaceTube memiliki 3 *button* utama, yaitu tombol Gunakan Kamera Depan/Belakang (dengan indikator), tombol Nyalakan/Matikan Service Pengenalan Suara (dengan indikator), dan tombol manual untuk Rekam.

Untuk *login* Google menggunakan *pop up window*, sedangkan Facebook memakai tombol dan *pop up window*. Kode sumber dan hasil dari implementasi untuk halaman utama dapat dilihat pada Kode Sumber 4.18 dan Gambar 4.1.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/gray"
    android:orientation="horizontal" >
    <LinearLayout
        android:id="@+id/ln"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#000000"
        android:baselineAligned="false"
        android:orientation="horizontal" >
        <LinearLayout
            android:id="@+id/buttonsLayout"
            android:layout_width="0dp"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="4.5"
            android:background="@drawable/blue"
            android:orientation="vertical" >
            <com.facebook.login.widget.LoginButton
                android:id="@+id/login_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="30dp"
                android:layout_marginBottom="30dp" />
            <fragment
                android:id="@+id/user_fragment"
                android:name="com.video.facetube.UserFragment"
                android:layout_width="match_parent"
                android:layout_height="0dp"
                android:layout_weight="1"
                tools:layout="@layout/user_fragment" />
            <ToggleButton
                android:id="@+id/btn_UseFrontCamera"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textOn="@string/textBackCamera"
                android:textOff="@string/textFrontCamera"/>
            <ToggleButton
                android:id="@+id/btn_StopSpeechRecogSvc"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textOn="@string/button_StopSpeechRecogSvc"
                android:textOff="@string/button_StartSpeechRecogSvc"/>
            <Button
                android:id="@+id/button_StartRecording"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="@string/button_StartRecording"/>
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

Kode Sumber 4.18 Implementasi antarmuka halaman utama

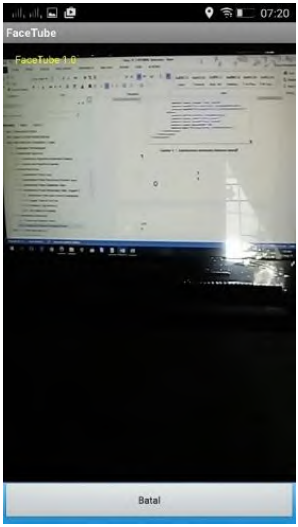
### 4.4.2. Antarmuka Halaman Perekaman Video

Antarmuka halaman perekaman video aplikasi FaceTube memiliki 1 *button* yaitu Batal, untuk membatalkan perekaman. Dan *preview screen* saat proses perekaman. Kode sumber untuk halaman perekaman video dapat dilihat pada Kode Sumber 4.19. Hasil dari implementasi dapat dilihat pada Gambar 4.2.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/gray"
    android:orientation="horizontal" >

    <LinearLayout
        android:id="@+id/ln"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#000000"
        android:baselineAligned="false"
        android:orientation="vertical" >

        <LinearLayout
            android:id="@+id/camera_preview"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:background="#000000"
            android:orientation="vertical" >
        </LinearLayout>

        <LinearLayout
            android:id="@+id/buttonsLayout"
            android:layout_width="fill_parent"
            android:layout_height="51dp"
            android:layout_gravity="center"
            android:background="@drawable/blue"
            android:orientation="vertical" >

            <Button
                android:id="@+id/btn_record"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/button_Record"
                android:textSize="12sp" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

Kode Sumber 4.19 Implementasi antarmuka halaman perekaman video


Gambar 4.1 Antarmuka halaman utama FaceTube




Gambar 4.2 Antarmuka halaman perekaman video

*[Halaman ini sengaja dikosongkan]*

# BAB 5
# UJI COBA DAN EVALUASI

Pada bab ini dibahas mengenai rangkaian pengujian dan evaluasi perangkat lunak yang dilakukan sesuai hasil implementasi. Pengujian dilakukan pada segi fungsionalitas, segi algoritma, dan kegunaan aplikasi. Pengujian fungsionalitas akan menggunakan skenario kasus penggunaan sebagai acuan kebutuhan pengguna. Pembahasan pada bab ini meliputi lingkungan pengujian fungsionalitas, dasar pengujian, skenario dan hasil pengujian, serta analisis hasil pengujian secara keseluruhan.

## 5.1. Lingkungan Pengujian

Uji coba aplikasi ini dilakukan pada sebuah perangkat. Berikut ini spesifikasi dari perangkat tersebut:

- Perangkat komunikasi bergerak Android yang memiliki *paket data* atau memiliki koneksi dengan internet untuk pengujian aplikasi berbasis perangkat bergerak. Perangkat komunikasi bergerak yang digunakan adalah ponsel pintar dengan spesifikasi sebagai berikut:
  1. Prosesor MT6752M 1.5 GHz, octa core (8 cores).
  2. Sistem operasi Android 6.0 Marshmallow (64-bit), SDK Version 22.
  3. RAM 2 GB.
  4. Internal Memory (ROM) 8 GB, External Memory (SD Card) 16 GB.
  5. Kamera belakang 8.0 mega pixel (3264x2448) dan kamera depan 5.0 mega pixel (2880x1728).
  6. Web browser Chrome pada *smartphone* untuk pengecekan video yang telah terunggah serta pengecekan post status atau dapat juga menggunakan aplikasi Facebook dan YouTube pada Android.

## 5.2. Skenario Pengujian

Pada sub bab ini akan dijelaskan skenario pengujian yang digunakan. Untuk pengujian fungsionalitas akan dilakukan dengan menggunakan metode pengujian kotak hitam (black-box). Untuk pengujian algoritma menggunakan unit tes yaitu *JUnit*. Sedangkan pengujian kegunaan akan dilakukan dengan memilih beberapa responden untuk mengadakan pengujian pada aplikasi.

### 5.2.1. Pengujian Fungsionalitas

Pengujian fungsionalitas merupakan pengujian yang dilakukan untuk memeriksa apakah fungsionalitas sistem sudah berjalan sebagaimana mestinya. Selain itu juga untuk mengetahui kesesuaian keluaran dari setiap tahapan atau langkah penggunaan fitur terhadap skenario yang dipersiapkan. Pengujian ini dilakukan dengan menggunakan metode black-box.

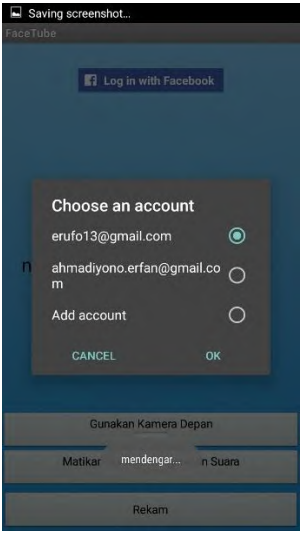
#### 5.2.1.1. Pengujian Login Youtube dan Facebook

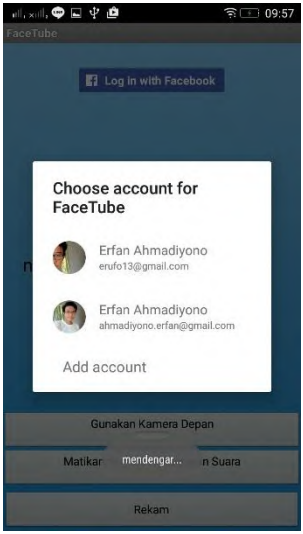
Pengujian ini bertujuan untuk menguji kasus penggunaan *login* YouTube dan Facebook. Untuk *login* Facebook, jika pengguna baru pertama kali melakukan *login* aplikasi maka akan muncul *consent screen* atau layar persetujuan aplikasi untuk melakukan *login* pada akun Facebook miliknya. Pengujian ini dimulai ketika pengguna telah membuka aplikasi baik untuk pertama kalinya dan berikutnya. Hasil pengujian dari kasus penggunaan *login* dijelaskan pada Tabel 5.1.

Tabel 5.1 Hasil pengujian kasus penggunaan login

| | |
|---|---|
| Kode | UC-001 |
| Nama | Pengujian *login* |
| Tujuan pengujian | Menguji fungsi *login* akun YouTube (Google) dan Facebook |

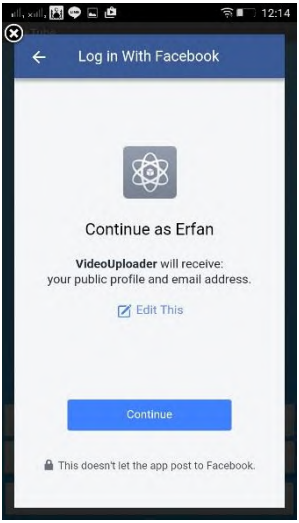
| | |
|---|---|
| | (*login* dan *consent screen*) pengguna aplikasi |
| *Skenario 1* | *Pengguna melakukan login kedua akun media sosial (Google dan Facebook)* |
| Masukan | Data *login* akun pengguna berupa (*username/email*) dan *password* |
| Keluaran yang diharapkan | *Login* sukses |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.1, Gambar 5.2, Gambar 5.3, dan Gambar 5.4 |
| *Skenario 2* | *Pengguna sudah pernah melakukan login akun Google dan Facebook pada aplikasi* |
| Masukan | - |
| Kondisi Awal | Pengguna membuka aplikasi |
| Keluaran yang diharapkan | Aplikasi melewati halaman *login* |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.5. |



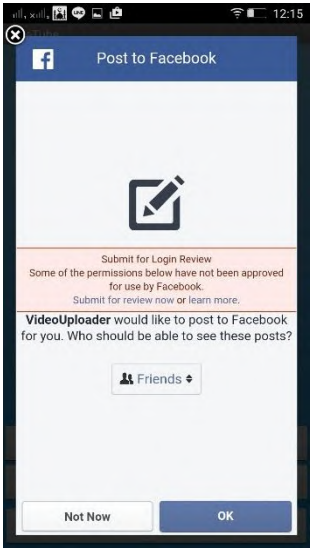




Gambar 5.1 Pop up window login Google untuk login akun Youtube

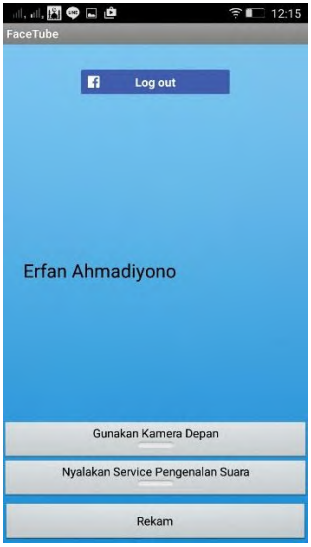

Gambar 5.2 Pop up window login Facebook




Gambar 5.3 Consent screen penggunaan aplikasi untuk pertama kali setelah login




Gambar 5.4 Halaman permission Facebook untuk melakukan post




Gambar 5.5 Halaman utama jika pengguna sudah pernah melakukan login

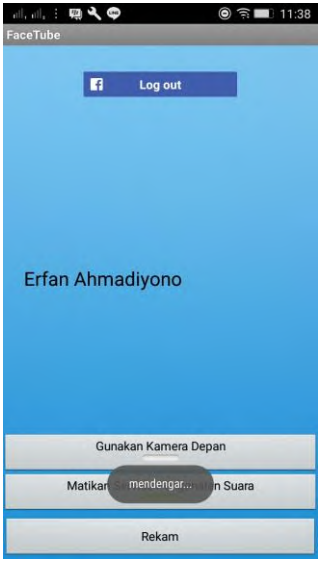
### 5.2.1.2. Pengujian Perintah Suara dan Perekaman Video

Pengujian ini bertujuan untuk menguji kasus penggunaan pengenalan perintah suara "rekam" untuk menjalankan perekaman video pada saat servis pengenalan suara berjalan. Perekaman video yang dilakukan secara *default* adalah otomatis dalam waktu rentang yang telah ditentukan. Hasil pengujian dari kasus penggunaan perintah suara dijelaskan pada Tabel 5.2.

Tabel 5.2 Hasil pengujian kasus penggunaan pengenalan perintah suara dan perekaman video

| | |
|---|---|
| Kode | UC-002 |
| Nama | Pengujian pengenalan perintah suara dan perekaman video |

| | |
|---|---|
| Tujuan pengujian | Menguji proses pengenalan perintah suara dan merekam video |
| *Skenario 1* | *Pengguna mengatakan kata perintah "rekam" ke arah smartphone* |
| Masukan | Suara pengguna |
| Kondisi awal | Aplikasi menunggu perintah suara pengguna |
| Keluaran yang diharapkan | Aplikasi mulai merekam video |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.6 dan Gambar 5.7 |
| Kondisi akhir | Aplikasi membuka halaman perekaman video dan mulai merekam |
| *Skenario 2* | *Aplikasi melakukan perekaman dalam rentang waktu yang telah ditentukan dalam sistem* |
| Masukan | - |
| Kondisi awal | Aplikasi sedang melakukan perekaman |
| Keluaran yang diharapkan | Aplikasi berhenti melakukan perekaman pada durasi yang telah ditentukan |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. |




Gambar 5.6 Aplikasi menunggu perintah suara pengguna


Gambar 5.7 Kata “rekam” terdeteksi dan aplikasi memulai perekaman video

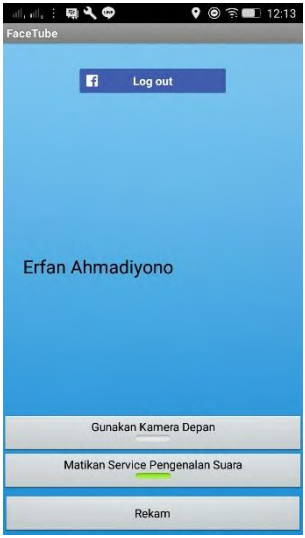
### 5.2.1.3. Pengujian Memilih Kamera

Pengujian ini bertujuan untuk menguji kasus penggunaan pemilihan kamera mana yang hendak digunakan untuk perekaman video. Hasil pengujian dari kasus penggunaan pemilihan kamera dijelaskan pada Tabel 5.3.

Tabel 5.3 Hasil pengujian kasus penggunaan memilih kamera

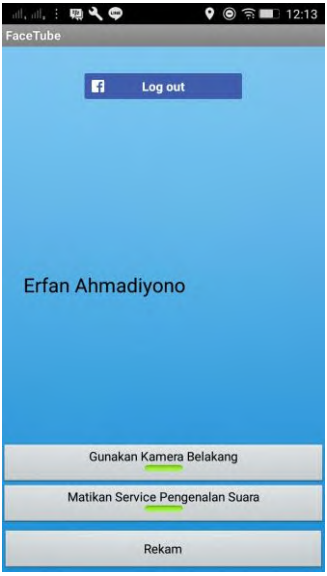
| | |
|---|---|
| Kode | UC-003 |
| Nama | Pengujian memilih kamera |
| Tujuan pengujian | Menguji pemilihan pemakaian kamera depan atau belakang pada perangkat |

| | |
|---|---|
| *Skenario 1* | *Pengguna menekan tombol Gunakan kamera depan/belakang* |
| Masukan | Pengguna menekan tombol Gunakan kamera depan/belakang |
| Kondisi awal | Pemakaian kamera belakang atau sebaliknya |
| Keluaran yang diharapkan | *Preview* tampilan kamera berganti ke kamera depan atau sebaliknya |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.8, Gambar 5.9, Gambar 5.10, Gambar 5.11 |




Gambar 5.8 Toggle button kamera tidak menyala


Gambar 5.9 Aplikasi menggunakan kamera belakang




Gambar 5.10 Toggle button kamera menyala


Gambar 5.11 Aplikasi menggunakan kamera depan

### 5.2.1.4. Pengujian Pembatalan Perekaman Video

Pengujian ini bertujuan untuk menguji kasus penggunaan pembatalan perekaman video yang terjadi jika pengguna menekan tombol Batal pada halaman perekaman. Hasil pengujian dari kasus penggunaan pembatalan perekaman video dijelaskan pada Tabel 5.4.

Tabel 5.4 Hasil pengujian kasus penggunaan pembatalan perekaman video

| | |
|---|---|
| Kode | UC-004 |
| Nama | Pengujian pembatalan rekam video |
| Tujuan pengujian | Menguji pembatalan perekaman video |

| | |
|---|---|
| *Skenario 1* | *Pengguna menekan tombol Batal pada halaman perekaman* |
| Masukan | Pengguna menekan tombol Batal |
| Kondisi awal | Perekaman video sedang berlangsung |
| Keluaran yang diharapkan | Aplikasi menghentikan perekaman dan menutup aplikasi |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. |
| Kondisi akhir | Aplikasi tidak mengolah video yang dibatalkan |

#### 5.2.1.5. Pengujian Unggah Video dan Pemberian Tag Reference

Pengujian ini bertujuan untuk menguji pengunggahan video dan pemberian tag referensi oleh aplikasi. Video yang sukses direkam akan secara otomatis diunggah ke akun Youtube pengguna. Hasil pengujian dari unggah video dijelaskan pada Tabel 5.5.

Tabel 5.5 Hasil pengujian unggah video dan pemberian tag reference

| | |
|---|---|
| Kode | UC-005 |
| Nama | Pengujian unggah video |
| Tujuan pengujian | Menguji proses pengunggahan video ke akun YouTube pengguna |
| *Skenario 1* | *Perekaman video telah selesai dan pengguna menunggu aplikasi mengunggah video pada akun YouTube miliknya* |

| | |
|---|---|
| Masukan | Video yang telah terekam |
| Keluaran yang diharapkan | Notifikasi bahwa video telah berhasil terunggah pada akun pengguna |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.12 dan Gambar 5.13 |
| *Skenario 2* | *Melakukan pengecekan video yang telah terunggah beserta tag reference pada videonya* |
| Masukan | - |
| Keluaran yang diharapkan | Video berhasil terunggah. Waktu video terdapat pada judul video di YouTube, sedangkan lokasi video terdapat pada deskripsinya |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.14 dan Gambar 5.15 |

#### 5.2.1.6. Pengujian Post Status berupa Link

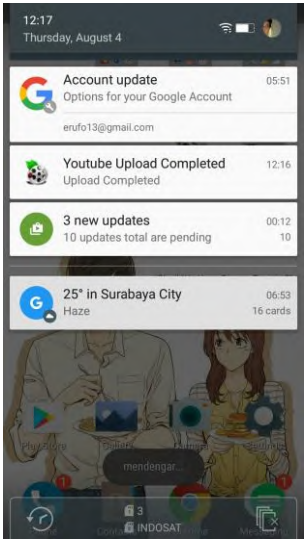
Pengujian ini bertujuan untuk menguji aplikasi dalam melakukan *post status* berupa *link* ke akun Facebook pengguna. Hasil pengujian dari post status dijelaskan pada Tabel 5.6.

Tabel 5.6 Hasil pengujian kasus post status berupa link

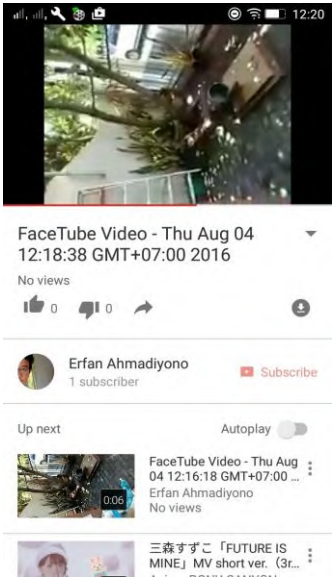
| | |
|---|---|
| Kode | UC-006 |
| Nama | Pengujian *post status* berupa *link* |

| | |
|---|---|
| Tujuan pengujian | Menguji proses *post status* berupa *link* pada akun Facebook pengguna |
| *Skenario 1* | *Setelah proses pengunggahan selesai pengguna mengecek akun Facebook miliknya* |
| Masukan | - |
| Keluaran yang diharapkan | *Status* berupa *link* video YouTube |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian pada Gambar 5.16 dan Gambar 3.17 |


Gambar 5.12 Notifikasi pengunggahan ke akun Youtube setelah perekaman video selesai




Gambar 5.13 Notifikasi bahwa pengunggahan video telah selesai dilakukan




Gambar 5.14 Video berhasil terunggah pada akun Youtube pengguna (Pengecekan menggunakan aplikasi Youtube)


Gambar 5.15 Judul video berupa waktu video direkam dan lokasi perekaman diletakkan pada Deskripsi video tersebut

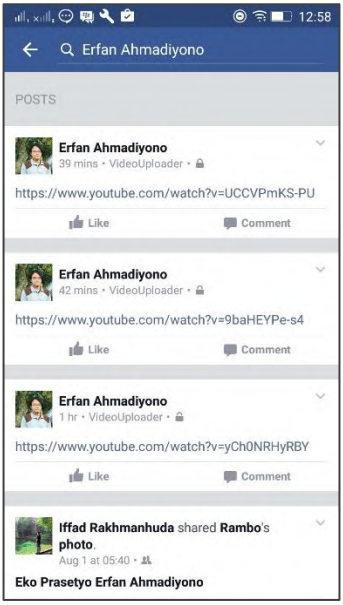
### 5.2.1.7. Pengujian Penyimpanan Video

Pengujian ini bertujuan untuk menguji penyimpanan video dalam memori perangkat *smartphone* pengguna. Hasil pengujian dari penyimpanan video dijelaskan pada Tabel 5.7.

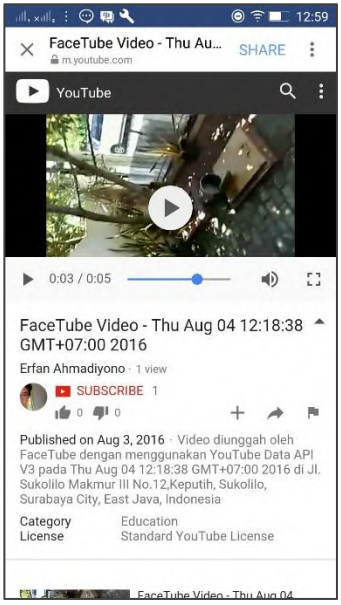
Tabel 5.7 Hasil pengujian penyimpanan video

| | |
|---|---|
| Kode | UC-007 |
| Nama | Pengujian penyimpanan video |
| Tujuan pengujian | Menguji apakah video tersimpan dalam memori *smartphone* pengguna |

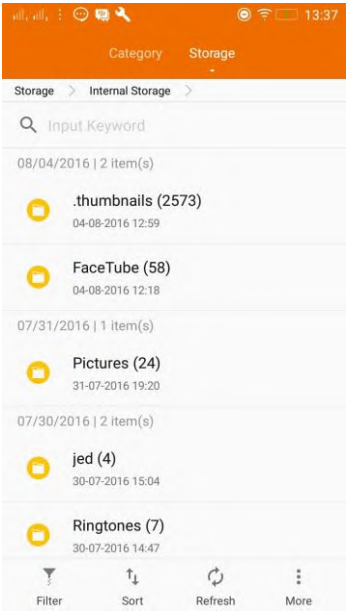
| | |
|---|---|
| *Skenario 1* | *Pengguna melakukan pengecekan pada memori internal/eksternal smartphone miliknya* |
| Masukan | - |
| Keluaran yang diharapkan | Video telah tersimpan dalam folder FaceTube |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. Hasil pengujian dapat dilihat pada Gambar 5.18 dan Gambar 5.19 |



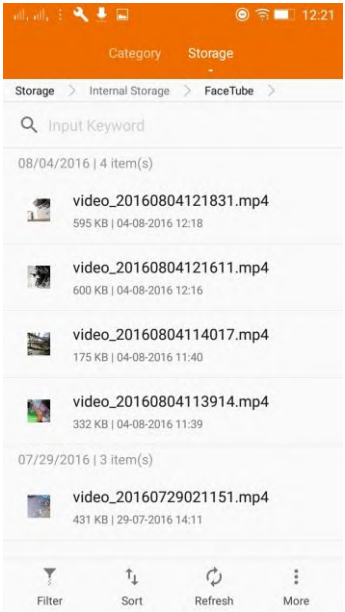

Gambar 5.16 Status berupa link video Youtube yang sebelumnya telah berhasil terunggah




Gambar 5.17 Pengujian video dengan *in-app browser* aplikasi Facebook




Gambar 5.18 Pengecekan pembuatan folder FaceTube pada internal storage perangkat




Gambar 5.19 Pengecekan video yang tersimpan di dalam memori smartphone

### 5.2.1.8. Pengujian Menghidupkan/Mematikan Service Pengenalan Suara

Pengujian ini bertujuan untuk menguji pengelolaan *service* pengenalan suara (menghidupkan/mematikannya). Hasil pengujian dari penyimpanan video dijelaskan pada Tabel 5.8.

Tabel 5.8 Hasil pengujian pengelolaan service pengenalan suara

| | |
|---|---|
| Kode | UC-008 |
| Nama | Pengujian menghidupkan/mematikan *service* pengenalan suara |

| | |
|---|---|
| Tujuan pengujian | Menguji tombol hidupkan/matikan *service* pengenalan suara yang disertai indikator |
| *Skenario 1* | *Pengguna menekan tombol Nyalakan/Matikan Service Pengenalan Suara* |
| Masukan | Pengguna menekan tombol Nyalakan/Matikan Service Pengenalan Suara |
| Keluaran yang diharapkan | Service pengenalan suara mati atau sebaliknya |
| Hasil pengujian | Kasus penggunaan berhasil diimplementasikan. |

### 5.2.2. Pengujian Kegunaan

Selain melakukan pengujian fungsionalitas dengan metode black box, juga dilakukan pengujian kegunaan untuk menilai kegunaan aplikasi secara langsung kepada pengguna. Pengujian kegunaan dilakukan untuk mengetahui penilaian dan tanggapan dari pengguna terhadap sejumlah aspek dari aplikasi ini.

#### 5.2.2.1. Kriteria Responden

Pengujian kegunaan dilakukan dengan melibatkan beberapa pengguna dari aplikasi ini. Para pengguna yang terlibat dalam pengujian ini selanjutnya disebut sebagai responden. Para responden diberikan keleluasaan untuk menjalankan aplikasi pada perangkat komunikasi bergerak Android.

#### 5.2.2.2. Skenario Pengujian Kegunaan

Dalam melakukan pengujian kegunaan aplikasi, responden diminta untuk menjalankan aplikasi FaceTube. Dalam memberikan penilaian terhadap aplikasi, responden mengisi

formulir penilaian yang telah disediakan untuk pengujian ini. Formulir penilaian memiliki beberapa aspek penilaian secara kualitatif. Formulir penilaian pengujian kegunaan aplikasi yang digunakan adalah sebagaimana pada LAMPIRAN B KUESIONER PENGUJIAN KEGUNAAN.

#### 5.2.2.3. Daftar Responden

Pada pengujian kegunaan ini terdapat 10 orang yang menjadi responden. Responden tersebut merupakan orang yang senang melakukan perekaman video menggunakan *smartphone* mereka di berbagai kesempatan dan termasuk orang-orang yang aktif dalam menggunakan akun sosial medianya. Daftar responden beserta keterangan perangkat bergerak yang digunakan terdapat pada Tabel 5.9.

Tabel 5.9 Daftar Responden Pengujian Kegunaan

| No | Nama | Tipe Perangkat Bergerak | Versi Android | Keterangan |
|---|---|---|---|---|
| 1 | Laksmono W. | Evercoss One-X | MM (6.0) | |
| 2 | Dimas R. | Zenfone 2 | Lollipop (5.0) | |
| 3 | Imam F. | Oppo | 4.4.1 | |
| 4 | Syamsul A. | Zenfone 2 Laser | 5.0.1 | |
| 5 | Dimas N. D. | Sony Xperia | 5.0 | |
| 6 | Jonathan I. K. | Samsung Galaxy S6 | 5.0 | |
| 7 | Satrio A. W. | Evercoss | 4.4 | |

| No | Nama | Tipe Perangkat Bergerak | Versi Android | Keterangan |
|---|---|---|---|---|
| 8 | Erlangga W. | Samsung Galaxy Ace | 5.0 | |
| 9 | Rahayu K. | Sony Xperia | 5.1.1 | |
| 10 | Wachid N. | Zenfone 5 | 5.0 | |

#### 5.2.2.4. Hasil Pengujian Kegunaan

Pada penilaian pengujian kegunaan, terdapat empat pilihan nilai untuk setiap pertanyaan yaitu tidak baik, kurang baik, cukup baik, dan sangat baik. Untuk menghitung hasil pengujian, maka dilakukan pengubahan nilai tersebut menjadi nilai-nilai angka dan persentase seperti pada Tabel 5.10.

Tabel 5.10 Daftar Perubahan Penilaian pada Pengujian Kegunaan

| No | Nilai pada Kuisioner | Nilai Angka | Persentase |
|---|---|---|---|
| 1 | Tidak Baik | 1 | 25% |
| 2 | Kurang Baik | 2 | 50% |
| 3 | Cukup Baik | 3 | 75% |
| 4 | Sangat Baik | 4 | 100% |

Pengujian dibagi menjadi tiga aspek yaitu antarmuka pengguna, pemberian rekomendasi, dan manfaat. Berikut ini adalah rekapitulasi penilaian dari hasil pengujian kegunaan.

##### 5.2.2.4.1. Rekapitulasi Penilaian Antarmuka Pengguna

Antarmuka pengguna merupakan salah satu aspek yang dinilai pada pengujian kegunaan aplikasi ini. Rekapitulasi

penilaian pengujian terhadap antarmuka pengguna terdapat pada Tabel 5.11.

Tabel 5.11 Penilaian Antarmuka Pengguna

| No | Penilaian | | | | Rata-rata |
|---|---|---|---|---|---|
| | Tidak | Kurang | Cukup | Sangat | |
| 1 | 0 | 0 | 7 | 3 | 3,3 |
| 2 | 0 | 0 | 2 | 8 | 3,8 |
| 3 | 0 | 0 | 4 | 6 | 3,6 |
| Nilai Akhir | | | | | 3,57 |

Tabel 5.11 merupakan hasil dari penilaian pengujian antarmuka. Rata-rata yang didapatkan dari penilaian antarmuka ini adalah 3,57. Nilai tersebut menandakan bahwa aspek ini sudah lebih dari cukup baik.

#### 5.2.2.4.2. Rekapitulasi Penilaian Pendeteksian Perintah Suara

Pengujian kegunaan aplikasi juga menilai keberhasilan pendeteksian perintah suara. Rekapitulasi penilaian pengujian terhadap pendeteksian perintah suara terdapat pada Tabel 5.12.

Tabel 5.12 Penilaian Pendeteksian Perintah Suara

| No | Penilaian | | | | Rata-rata |
|---|---|---|---|---|---|
| | Tidak | Kurang | Cukup | Sangat | |
| 1 | 0 | 0 | 10 | 0 | 3 |
| 2 | 0 | 0 | 0 | 10 | 4 |
| Nilai Akhir | | | | | 3,5 |

Berdasarkan pada Tabel 5.12, pengujian kegunaan pada aspek penilaian pendeteksian perintah suara memiliki rata-rata 3,5 yang berarti aspek ini sudah lebih dari cukup baik.

#### 5.2.2.4.3. Rekapitulasi Penilaian Manfaat Aplikasi

Aspek yang dinilai dalam pengujian kegunaan adalah penilaian manfaat aplikasi untuk pengguna. Rekapitulasi penilaian manfaat aplikasi ini terdapat pada Tabel 5.13.

Tabel 5.13 Penilaian Manfaat Aplikasi

| No | Penilaian | | | | Rata-rata |
|---|---|---|---|---|---|
| | Tidak | Kurang | Cukup | Sangat | |
| 1 | 0 | 0 | 1 | 9 | 3,9 |
| 2 | 0 | 0 | 1 | 9 | 3,9 |
| 3 | 0 | 0 | 1 | 9 | 3,9 |
| Nilai Akhir | | | | | 3,9 |

Sesuai dengan rekapitulasi penilaian pada Tabel 5.13, dapat diketahui bahwa manfaat aplikasi untuk pengguna memiliki nilai rata-rata 3,9, yang berarti aspek ini sudah lebih dari cukup baik.

### 5.2.3. Pengujian Algoritma

Pengujian algoritma pada aplikasi FaceTube ini dilakukan pada fungsi pencocokan teks yang didapatkan dari hasil proses pengenalan suara oleh *Google Speech Recognition*. Pada fungsi pencocokan teks terdapat 2 algoritma yaitu *Levenshtein Distance* dan *Soundex*.

*Tools* yang digunakan pada pengujian algoritma ini adalah *JUnit* pada *Eclipse*, untuk melakukan *Unit Test* pada kedua

algoritma tersebut dan juga *Unit Test* secara keseluruhan dari fungsi pencocokan teks pada FaceTube. Karena proses pencocokan teks ini berlangsung relatif cepat, maka penggunaan satuan *second* atau *millisecond*, untuk mengukur lamanya proses, kurang tepat. Dengan penggunaan kedua satuan tersebut tidak didapatkan perbedaan waktu yang signifikan (cenderung mendekati 0). Oleh karena itu pada pengujian kali ini digunakan satuan *nanosecond.*

Pada pengujian kedua algoritma ini digunakan beberapa kata uji yang mirip dengan kata "rekam", baik dari segi suara pengucapannya, maupun dari segi kemungkinan terdeteksi kata-nya. Daftar input kata uji dan tabel pengujian yang akan dilakukan terdapat pada Tabel 5.14.

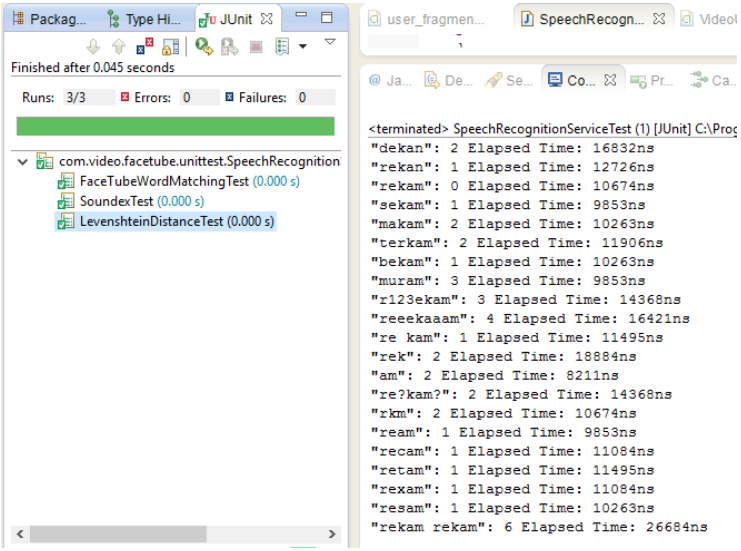
### 5.2.3.1. Pengujian Algoritma Levenshtein Distance

Gambar 5.20 adalah pengujian algoritma yang dilakukan dengan *JUnit* pada *Eclipse* dan Tabel 5.14 adalah hasil dari pengujian algoritma *Levenshtein Distance* dengan menggunakan beberapa variasi inputan.

Tabel 5.14 Hasil pengujian algoritma Levensthein Distance dengan berbagai variasi input

| No | Input | Output Levenshtein Distance | Waktu Proses (ns) |
|---|---|---|---|
| 1 | dekan | 2 | 16.832 |
| 2 | rekan | 1 | 12.726 |
| 3 | rekam | 0 | 10.674 |
| 4 | sekam | 1 | 9.853 |
| 5 | makam | 2 | 10.263 |
| 6 | terkam | 2 | 11.906 |
| 7 | bekam | 1 | 10.263 |
| 8 | muram | 3 | 9.853 |
| 9 | r123ekam | 3 | 14.368 |

| No | Input | Output Levenshtein Distance | Waktu Proses (ns) |
|---|---|---|---|
| 10 | reeekaaam | 4 | 16.421 |
| 11 | re kam | 1 | 11.495 |
| 12 | rek | 2 | 18.884 |
| 13 | am | 2 | 8.211 |
| 14 | re?kam? | 2 | 14.368 |
| 15 | rkm | 2 | 10.674 |
| 16 | ream | 1 | 9.853 |
| 17 | recam | 1 | 11.084 |
| 18 | retam | 1 | 11.495 |
| 19 | rexam | 1 | 11.084 |
| 20 | resam | 1 | 10.263 |
| 21 | rekam rekam | 6 | 26.684 |




Gambar 5.20 Hasil pengujian algoritma Levensthein Distance

Dari hasil pengujian diatas dapat dilihat bahwa semakin banyak perbedaan suatu kata dengan kata “rekam” (dari segi huruf) maka nilai *Levenshtein*-nya semakin besar yang berarti kata tersebut dinyatakan tidak sama atau tidak mirip. Nilai tersebut didapatkan dari seberapa banyak proses yang dilakukan untuk mengubah kata uji menjadi kata tujuan yaitu “rekam”. Dari beberapa kata uji yang dilakukan pencocokan hanya 1 yang bernilai 0 yaitu kata “rekam”. Maka dari itu implementasi algoritma ini pada aplikasi FaceTube perlu dilakukan penyesuaian berupa persentase perbandingan banyaknya proses dengan panjang *string* kata. Dapat dilihat juga waktu proses pengecekan yang dilakukan algoritma sangat cepat (satuan nano detik). Sehingga dapat disimpulkan bahwa pengujian algoritma *Levensthein Distance* berhasil diimplementasikan dengan baik untuk pencocokan sebuah *string*.

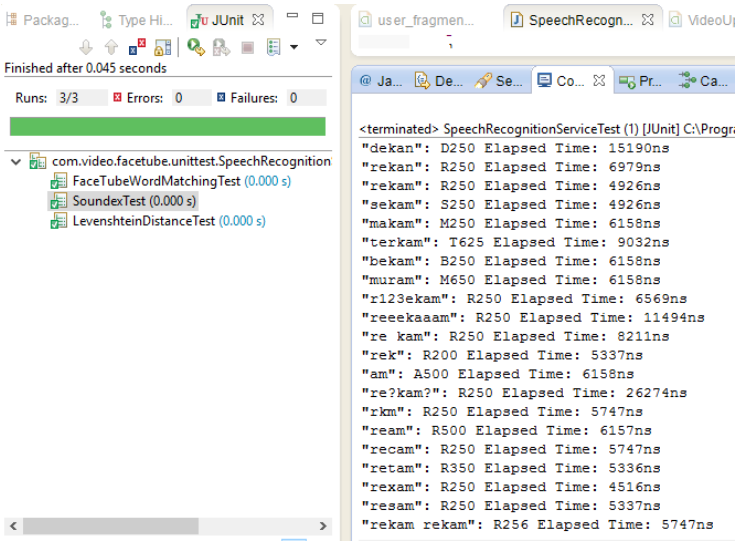
#### 5.2.3.2. Pengujian Algoritma Soundex

Gambar 5.21 merupakan pengujian algoritma yang dilakukan dengan *JUnit* pada *Eclipse* dan Tabel 5.15 adalah hasil dari pengujian algoritma *Soundex* dengan menggunakan beberapa variasi inputan.

Tabel 5.15 Hasil pengujian algoritma Soundex dengan berbagai input

| No | Input | Output Soundex | Waktu Proses (ns) |
|---|---|---|---|
| 1 | dekan | D250 | 15.190 |
| 2 | rekan | R250 | 6.979 |
| 3 | rekam | R250 | 4.926 |
| 4 | sekam | S250 | 4.926 |
| 5 | makam | M250 | 6.158 |
| 6 | terkam | T625 | 9.032 |
| 7 | bekam | B250 | 6.158 |
| 8 | muram | M650 | 6.158 |
| 9 | r123ekam | R250 | 6.569 |

| No | Input | Output Soundex | Waktu Proses (ns) |
|---|---|---|---|
| 10 | reeekaaam | R250 | 11.494 |
| 11 | re kam | R250 | 8.211 |
| 12 | rek | R200 | 5.337 |
| 13 | am | A500 | 6.158 |
| 14 | re?kam? | R250 | 26.274 |
| 15 | rkm | R250 | 5.747 |
| 16 | ream | R500 | 6.157 |
| 17 | recam | R250 | 5.747 |
| 18 | retam | R350 | 5.336 |
| 19 | rexam | R250 | 4.516 |
| 20 | resam | R250 | 5.337 |
| 21 | rekam rekam | R256 | 5.747 |




Gambar 5.21 Hasil pengujian algoritma Soundex

Dari hasil pengujian diatas dapat dilihat perubahan atau konversi kata uji menjadi kode fonetis. Dari hasil output yang didapat dari pengecekan *Soundex*, lebih banyak kata yang dinyatakan sama dengan kode fonetis "rekam" (R250) yaitu 12 kata. Hal ini dikarenakan huruf vokal dalam pemetaaan fonetis dinilai 0. Dan dapat dilihat bahwa waktu dari proses pengecekan yang dilakukan algoritma *Soundex* lebih cepat dibandingkan *Levensthein* dikarenakan prosesnya tidak sepanjang *Levensthein* yang memakai *array* dan dilakukan pengecekan tiap proses perubahannya. Dengan demikian, algoritma *Soundex* juga telah berjalan dengan baik

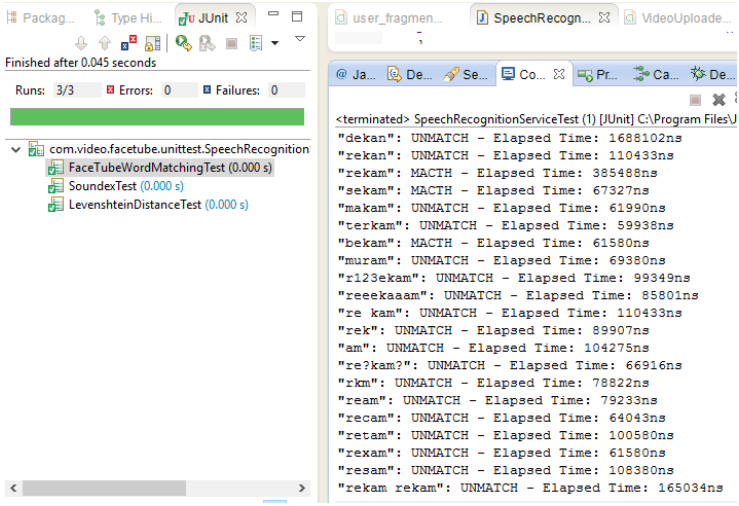
#### 5.2.3.3. Pengujian Algoritma Pencocokan Kata secara keseluruhan

Gambar 5.22 merupakan pengujian pencocokan kata secara keseluruhan (menggunakan 2 algoritma) dan Tabel 5.16 adalah hasil dari pengujian algoritma pencocokan kata secara keseluruhan dengan menggunakan beberapa variasi inputan. Tingkat kecocokan yang digunakan pada Algoritma *Levenshtein Distance* adalah diatas 90% dengan mengabaikan karakter pertama dari kata "rekam", yaitu "ekam". Sedangkan hasil yang diharapkan dari Algoritma *Soundex* adalah "R250", dengan mengabaikan karakter pertama menjadi "250".

Tabel 5.16 Hasil pengujian algoritma pencocokan kata dengan berbagai macam inputan

| No | Input | Output (Match/Unmatch) | Waktu Proses (ns) |
|---|---|---|---|
| 1 | dekan | Unmatch | 1.688.102 |
| 2 | rekan | Unmatch | 110.433 |
| 3 | rekam | Match | 385.488 |
| 4 | sekam | Match | 67.327 |
| 5 | makam | Unmatch | 61.990 |
| 6 | terkam | Unmatch | 59.938 |

| No | Input | Output (Match/Unmatch) | Waktu Proses (ns) |
|---|---|---|---|
| 7 | bekam | Match | 61.580 |
| 8 | muram | Unmatch | 69.380 |
| 9 | r123ekam | Unmatch | 99.349 |
| 10 | reeekaaam | Unmatch | 85.801 |
| 11 | re kam | Unmatch | 110.433 |
| 12 | rek | Unmatch | 89.907 |
| 13 | am | Unmatch | 104.275 |
| 14 | re?kam? | Unmatch | 66.916 |
| 15 | rkm | Unmatch | 78.822 |
| 16 | ream | Unmatch | 79.233 |
| 17 | recam | Unmatch | 64.043 |
| 18 | retam | Unmatch | 100.580 |
| 19 | rexam | Unmatch | 61.580 |
| 20 | resam | Unmatch | 108.380 |
| 21 | rekam rekam | Unmatch | 165.034 |




Gambar 5.22 Hasil pengujian algoritma pencocokan kata secara keseluruhan

Dari Tabel 5.16 dapat dilihat bahwa terdapat 3 kata uji yang dinyatakan *Match* atau cocok, yaitu kata rekam, sekam dan bekam. Jika kita perhatikan pada pengujian pada masing-masing algoritma, ketiga kata tersebut sama-sama memiliki kode fonetis yg sama "250" dengan menghiraukan huruf pertama dari string ("R", "S", "K"). Dan memiliki nilai *Levensthein* paling kecil yaitu sebesar rekam=0, sekam=1, dan bekam=1. Sehingga ketiga kata tersebut lolos dari uji kedua algoritma pencocokan string dan termasuk perintah suara untuk perekaman pada aplikasi FaceTube.

## 5.3. Evaluasi Pengujian

Pada sub bab ini akan dibahas evaluasi pengujian berdasarkan hasil pengujian yang telah dilakukan. Evaluasi meliputi semua pengujian yang telah dilakukan, yaitu pengujian fungsionalitas, pengujian kegunaan dan algoritma.

### 5.3.1. Evaluasi Pengujian Fungsionalitas

Rincian rangkuman hasil uji coba fungsionalitas dapat dilihat pada Tabel 5.17. Berdasarkan data pada tabel tersebut, semua skenario pengujian fungsionalitas berhasil. Sehingga dapat disimpulkan bahwa fungsionalitas sistem dapat berjalan dengan baik dan sesuai harapan.

Tabel 5.17 Hasil uji coba fungsionalitas

| Kode | Nama | Skenario | Hasil |
|---|---|---|---|
| UC-001 | Pengujian Login Youtube dan Facebook | Skenario 1 | Berhasil |
| | | Skenario 2 | Berhasil |
| UC-002 | Pengujian Perintah Suara dan Perekaman Video | Skenario 1 | Berhasil |
| | | Skenario 2 | Berhasil |

| Kode | Nama | Skenario | Hasil |
|---|---|---|---|
| UC-003 | Pengujian Memilih Kamera | Skenario 1 | Berhasil |
| UC-004 | Pengujian Pembatalan Perekaman Video | Skenario 1 | Berhasil |
| UC-005 | Pengujian Unggah Video | Skenario 1 | Berhasil |
| | | Skenario 2 | Berhasil |
| UC-006 | Pengujian Post Status | Skenario 1 | Berhasil |
| UC-007 | Pengujian Penyimpanan Video | Skenario 1 | Berhasil |
| UC-008 | Pengujian Pengelolaan Service | Skenario 1 | Berhasil |

### 5.3.2. Evaluasi Pengujian Kegunaan

Untuk menganalisis pengujian kegunaan aplikasi, maka dilakukan rekapitulasi akhir. Rekapitulasi akhir menghasilkan nilai persentase tehadap aspek-aspek yang dinilai dalam pengujian kegunaan. Rekapitulasi akhir ini ditunjukkan dalam Tabel 5.18.

Tabel 5.18 Rekapitulasi Akhir Pengujian Kegunaan

| No | Aspek Pengujian | Nilai Akhir | Persentase |
|---|---|---|---|
| 1 | Antarmuka Pengguna | 3,57 | 89,2% |

| No | Aspek Pengujian | Nilai Akhir | Persentase |
|---|---|---|---|
| 2 | Pendeteksian Perintah Suara | 3,5 | 87,5% |
| 3 | Manfaat Aplikasi | 3,9 | 97,5% |

Dari Tabel 5.18, dapat dilihat bahwa aspek-aspek yang dinilai menghasilkan nilai-nilai persentase. Penilaian terhadap antarmuka pengguna menghasilkan persentase sebesar 89,2%. Penilaian terhadap pendeteksian perintah suara menghasilkan persentase sebesar 87,5%. Sedangkan penilaian terhadap manfaat aplikasi menghasilkan persentase sebesar 97,5%. Ketiga aspek menunjukkan persentase nilai yang lebih dari 75%. Sehingga dapat disimpulkan bahwa dari segi kegunaan, perangkat lunak yang dibuat pada Tugas Akhir ini memberikan hasil yang lebih dari cukup baik.

### 5.3.3. Evaluasi Pengujian Algoritma

Dari hasil pengujian algoritma, terdapat beberapa poin–poin penting yang dapat disimpulkan. Berikut ini adalah penjabarannya.

1. Penggunaan algoritma *Levensthein Distance* pada aplikasi adalah untuk mencocokkan *string* berdasarkan perbedaan penulisannya.
2. Pencocokan algoritma *Levensthein* sangat akurat karena memproses setiap huruf dalam *string* satu per satu, sehingga 1 perbedaan saja mengubah nilai jarak proses *transformasinya*. Akan tetapi hal tersebut sedikit berpengaruh pada lamanya waktu proses algoritma.
3. Penggunaan algoritma *Soundex* adalah untuk mencocokkan *string* berdasarkan kemiripan bunyi pengucapannya.
4. Pencocokan yang dilakukan algoritma *Soundex* tidak seakurat *Levensthein* dikarenakan pencocokan yang diproses adalah

berdasarkan pemetaan kode fonetis. Sehingga misalnya jika terdapat sebuah kata “rikom”, akan memiliki kode fonetis yang sama dengan “rekam” yakni R250. Padahal secara penulisan 2 kata tersebut jelas berbeda. Hal ini dikarenakan huruf vokal bernilai 0 dalam pemetaan algoritma *Soundex*. Hal tersebut yang menjadikan pentingnya menggunakan algoritma pencocokan *string* lain pada aplikasi FaceTube, agar pencocokan perintah suara semakin akurat.

*5.* Pengulangan kata atau pendeteksian dalam bentuk kalimat tidak termasuk kasus *Match* walaupun kata yang terdeteksi terdapat kata “rekam”. Seperti pada contoh kata uji yang dipakai pada pengujian algoritma, terdapat kata uji “rekam rekam”. “rekam rekam” tidak sama dengan “rekam” dikarenakan termasuk sebuah kalimat sehingga perbedaannya bernilai besar jika dibandingkan dengan kata “rekam”.

*6.* Untuk kata “sekam” dan “bekam” dinyatakan *Match* dikarenakan pada pengujian algoritma ini huruf pertama dihiraukan. Hal tersebut bertujuan untuk mengurangi kesalahan pendeteksian.

7. Kedua algoritma yang digunakan berhasil diimplementasikan dengan cukup baik untuk pencocokan *string* kata.

# LAMPIRAN A KODE SUMBER

Kelas Auth.java

```
package com.video.facetube;

import com.google.android.gms.common.Scopes;
import com.google.api.services.youtube.YouTubeScopes;

public class Auth {
    // Register an API key here:https://console.developers.google.com
    public static final String KEY = "AIzaSyAwa1zhQfeNbaWGgmRB-
bhxd4frbgm5zVI";

    public static final String[] SCOPES = { Scopes.PROFILE,
            YouTubeScopes.YOUTUBE, YouTubeScopes.YOUTUBE_UPLOAD };

}
```

Kelas CameraPreview.java

```
package com.video.facetube;

import java.io.IOException;
import java.util.List;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.util.Log;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class CameraPreview extends SurfaceView implements
        SurfaceHolder.Callback {

    public OnSurfaceCreatedEventListener OnSurfaceCreated;

    private SurfaceHolder mHolder;
    private Camera mCamera;
    private String TEXT = "FIRST";
    private Paint textPaint = new Paint();
    private List<Camera.Size> mSupportedPreviewSizes;
    private Camera.Size mPreviewSize;

    public void setOnSurfaceCreatedEventListener(
            OnSurfaceCreatedEventListener listener) {
        OnSurfaceCreated = listener;
    }

    public Camera get_cam() {
        return this.mCamera;
```

```
    }

    public Surface get_surf() {
        return this.mHolder.getSurface();
    }

    public void SET_TEXT(String txt) {
        this.TEXT = txt;
        invalidate();
    }

        public String GET_TEXT() {
        return this.TEXT;
    }


    public CameraPreview(Context context, Camera camera) {
        super(context);
        textPaint.setARGB(255, 255, 255, 0);
        textPaint.setTextSize(25);
        setWillNotDraw(false);
        mCamera = camera;
        mHolder = getHolder();
        mHolder.addCallback(this);
    }


    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            if (!VideoCaptureActivity.recording && OnSurfaceCreated
!= null) {
                OnSurfaceCreated.OnSurfaceCreated(holder);
            }
            if (mCamera != null) {
                try {
                    mCamera.setPreviewDisplay(holder);
                    mCamera.setDisplayOrientation(90);
                    mCamera.startPreview();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        } catch (Exception e) {
            Log.d(VIEW_LOG_TAG,
                    "Error setting camera preview: " +
e.getMessage());
        }
    }

    public void refreshCamera(Camera camera) {
        if (mHolder.getSurface() == null) {
            return;
        }
        try {
            mCamera.stopPreview();
        } catch (Exception e) {
```

```
            }
            setCamera(camera);
            try {
                mCamera.setPreviewDisplay(mHolder);
                mCamera.startPreview();
                // mCamera.setDisplayOrientation(90);
            } catch (Exception e) {
                Log.d(VIEW_LOG_TAG,
                        "Error starting camera preview: " +
e.getMessage());
            }
        }

        @Override
        public void onDraw(Canvas canvas) {
            canvas.drawText(GET_TEXT(), 30, 50, textPaint);
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int
w, int h) {
            if (!VideoCaptureActivity.recording) {
                refreshCamera(mCamera);
            }
        }

        public Camera.Size getSize(int width, int height) {
            this.measureSize(width, height);// (1920, 1088);

            return mPreviewSize;
        }

        public void setCamera(Camera camera) {
            mCamera = camera;
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            mCamera.release();
        }
        private void measureSize(int width, int height) {
            if (mSupportedPreviewSizes == null) {
                Parameters params = mCamera.getParameters();

                if (params != null) {
                    mSupportedPreviewSizes =
params.getSupportedPreviewSizes();
                }
            }

            if (mSupportedPreviewSizes != null) {
                mPreviewSize =
getOptimalPreviewSize(mSupportedPreviewSizes, width, height);
            }
        }
```

```
    private Camera.Size getOptimalPreviewSize(List<Camera.Size>
sizes, int w, int h) {
        final double ASPECT_TOLERANCE = 0.1;
        double targetRatio = (double) h / w;

        if (sizes == null)
                return null;


        Camera.Size optimalSize = null;
        double minDiff = Double.MAX_VALUE;

        int targetHeight = h;

        for (Camera.Size size : sizes) {
            double ratio = (double) size.width / size.height;
            if (Math.abs(ratio - targetRatio) > ASPECT_TOLERANCE)
                continue;
            if (Math.abs(size.height - targetHeight) < minDiff) {
                optimalSize = size;
                minDiff = Math.abs(size.height - targetHeight);
            }
        }

        if (optimalSize == null) {
            minDiff = Double.MAX_VALUE;
            for (Camera.Size size : sizes) {
                if (Math.abs(size.height - targetHeight) < minDiff){
                    optimalSize = size;
                    minDiff = Math.abs(size.height - targetHeight);
                }
            }
        }
        return optimalSize;

    }
}
```

Kelas FbManager.java

```
package com.video.facetube;

import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Set;
import android.app.Activity;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.Signature;
import android.os.Bundle;
import android.util.Base64;
import android.util.Log;
import com.facebook.AccessToken;
import com.facebook.CallbackManager;
import com.facebook.FacebookCallback;
```

```
import com.facebook.FacebookException;
import com.facebook.GraphRequest;
import com.facebook.GraphRequestAsyncTask;
import com.facebook.GraphResponse;
import com.facebook.HttpMethod;
import com.facebook.login.LoginManager;
import com.facebook.login.LoginResult;

public class FbManager {
    private Context context;
    private static final String LOGTAG = "FbManager";
    private CallbackManager callbackManager;

    public FbManager(Context context, CallbackManager
callbackManager) {
        this.context = context;
        this.callbackManager = callbackManager;
    }

        public static void traceKeyHash(Activity activity) {
        try {
            PackageInfo info =
activity.getPackageManager().getPackageInfo(
                "com.video.facetube",
PackageManager.GET_SIGNATURES);
            for (Signature signature : info.signatures) {
                MessageDigest md = MessageDigest.getInstance("SHA");
                md.update(signature.toByteArray());
                Log.i(LOGTAG, "Share - KeyHash: "
    + Base64.encodeToString(md.digest(),
    Base64.DEFAULT));         }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void share(final String msg) {
        if (isLoggedIn()) {
            post(msg);
        } else {

    LoginManager.getInstance().registerCallback(callbackManager,
                    new FacebookCallback<LoginResult>() {

                        @Override
                        public void onSuccess(LoginResult
loginResult) {
                            Log.d(LOGTAG, "facebook login
success");
                            post(msg);
                        }

                        @Override
                        public void onCancel() {
                            Log.w(LOGTAG, "facebook login
canceled");
                        }
```

```
                        @Override
                        public void onError(FacebookException
exception) {
                            Log.e(LOGTAG, "facebook login error");
                            exception.printStackTrace();
                        }
                    });

            LoginManager.getInstance().logInWithPublishPermissions(
                    (Activity) context,
Arrays.asList("publish_actions"));
        }
    }

    public boolean isLoggedIn() {
        AccessToken accessToken =
AccessToken.getCurrentAccessToken();
        return accessToken != null;
    }


    private void post(final String msg) {
            Log.d(LOGTAG, "facebook posting new message");
            @SuppressWarnings("unused")
            Set<String> permissions =
AccessToken.getCurrentAccessToken()
                    .getPermissions();
            AccessToken accessToken =
AccessToken.getCurrentAccessToken();

            Bundle postParams = new Bundle();
            postParams.putString("message", msg);
            GraphRequest request = new GraphRequest(accessToken,
"me/feed",
                postParams, HttpMethod.POST, new
GraphRequest.Callback() {
                    @Override
                    public void onCompleted(GraphResponse
graphResponse) {
                        Log.d("POST", "post page response::" +
graphResponse);
                    }
                });
        GraphRequestAsyncTask asynTaskGraphRequest = new
GraphRequestAsyncTask(
                request);
        asynTaskGraphRequest.execute();
    }
}
```

Kelas GPSTracker.java

```
package com.video.facetube;

import android.app.AlertDialog;
import android.app.Service;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.provider.Settings;
import android.util.Log;

public class GPSTracker extends Service implements LocationListener {
    private final Context mContext;

    // flag for GPS status
    boolean isGPSEnabled = false;

    // flag for network status
    boolean isNetworkEnabled = false;

    boolean canGetLocation = false;

    Location location; // location
    double latitude; // latitude
    double longitude; // longitude

    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10;
// 10 meters

    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; //
1 minute

    // Declaring a Location Manager
    protected LocationManager locationManager;

    public GPSTracker(Context context) {
        this.mContext = context;
        getLocation();
    }

    public Location getLocation() {
        try {
            locationManager = (LocationManager) mContext
                    .getSystemService(LOCATION_SERVICE);

            // getting GPS status
            isGPSEnabled = locationManager

    .isProviderEnabled(LocationManager.GPS_PROVIDER);
```

```
            // getting network status
            isNetworkEnabled = locationManager

    .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

            if (!isGPSEnabled && !isNetworkEnabled) {
                // no network provider is enabled
            } else {
                this.canGetLocation = true;
            // First get location from Network Provider
                if (isNetworkEnabled) {
                    locationManager.requestLocationUpdates(
                            LocationManager.NETWORK_PROVIDER,
                            MIN_TIME_BW_UPDATES,
                            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    Log.d("Network", "Network");
                    if (locationManager != null) {
                        location = locationManager

    .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                        if (location != null) {
                            latitude = location.getLatitude();
                            longitude = location.getLongitude();
                        }
                    }
                }

// if GPS Enabled get lat/long using GPS Services
                if (isGPSEnabled) {
                    if (location == null) {
                        locationManager.requestLocationUpdates(
                                LocationManager.GPS_PROVIDER,
                                MIN_TIME_BW_UPDATES,
                                MIN_DISTANCE_CHANGE_FOR_UPDATES,
this);
                        Log.d("GPS Enabled", "GPS Enabled");
                        if (locationManager != null) {
                            location = locationManager

    .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            if (location != null) {
                                latitude = location.getLatitude();
                                longitude =
location.getLongitude();
                            }
                        }
                    }
                }

        } catch (Exception e) {
            e.printStackTrace();
        }

        return location;
```

```
    }

    /**
     * Function to get latitude
     * */
    public double getLatitude() {
        if (location != null) {
            latitude = location.getLatitude();
        }

        // return latitude
        return latitude;
    }

    /**
     * Function to get longitude
     * */
    public double getLongitude() {
        if (location != null) {
            longitude = location.getLongitude();
        }

        // return longitude
        return longitude;
    }
    /**
     * Function to check if best network provider
     *
     * @return boolean
     * */
    public boolean canGetLocation() {
        return this.canGetLocation;
    }

    /**
     * Function to show settings alert dialog
     * */
    public void showSettingsAlert() {
        AlertDialog.Builder alertDialog = new
AlertDialog.Builder(mContext);

        // Setting Dialog Title
        alertDialog.setTitle("GPS is settings");

        // Setting Dialog Message
        alertDialog
                .setMessage("GPS is not enabled. Do you want to go
to settings menu?");

        // Setting Icon to Dialog
        // alertDialog.setIcon(R.drawable.delete);

        // On pressing Settings button
        alertDialog.setPositiveButton("Settings",
                new DialogInterface.OnClickListener() {
```

```
                    public void onClick(DialogInterface dialog, int
which) {
                        Intent intent = new Intent(

    Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                        mContext.startActivity(intent);
                    }
                });

        // on pressing cancel button
        alertDialog.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int
which) {
                        dialog.cancel();
                    }
                });

        // Showing Alert Message
        alertDialog.show();
    }

    @Override
    public void onLocationChanged(Location location) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle
extras) {
    }

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
}
```

Kelas MainActivity.java

```
package com.video.facetube;

import java.util.Arrays;

import android.accounts.AccountManager;
import android.app.Activity;
import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.Context;
```

```
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.facebook.AccessToken;
import com.facebook.AccessTokenTracker;
import com.facebook.CallbackManager;
import com.facebook.FacebookCallback;
import com.facebook.FacebookException;
import com.facebook.FacebookSdk;
import com.facebook.login.LoginManager;
import com.facebook.login.LoginResult;
import com.facebook.login.widget.LoginButton;
import com.google.android.gms.common.GoogleApiAvailability;
import
com.google.api.client.googleapis.extensions.android.gms.auth.GoogleAc
countCredential;
import com.google.api.client.util.ExponentialBackOff;

public class MainActivity extends Activity implements
UserFragment.Callbacks,
        FacebookCallback<LoginResult> {
    public static final String TAG = "VIDEOCAPTURE";
    private static final String REQUEST_AUTHORIZATION_INTENT =
"com.google.example.yt.RequestAuth";
    private static final String REQUEST_AUTHORIZATION_INTENT_PARAM =
"com.google.example.yt.RequestAuth.param";
    private static final int REQUEST_GOOGLE_PLAY_SERVICES = 0;
    private static final int REQUEST_ACCOUNT_PICKER = 2;
    private static final int REQUEST_AUTHORIZATION = 3;
    private static final int REQUEST_FACEBOOK_LOGIN = 64206;

    private static CallbackManager callbackManager;

    private UploadBroadcastReceiver broadcastReceiver;
    private Button btn_startrecord;
    private ToggleButton btn_stopspeechrecogsvc;
    private ToggleButton btn_UseFrontCamera;
    private LoginButton loginButton;
    private UserFragment mUserFragment;
    private GoogleAccountCredential credential;
    private AccessTokenTracker accessTokenTracker;
    private Intent callerIntent = null;
    private String mChosenAccountName;
    private boolean mIsPost = false;

        /** Called when the activity is first created. */
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Context context = getApplicationContext();

        FacebookSdk.sdkInitialize(context);
        callbackManager = CallbackManager.Factory.create();
    mUserFragment = new UserFragment(context);


    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT)
;

        setContentView(R.layout.main_layout);

        btn_UseFrontCamera = (ToggleButton) this
                .findViewById(R.id.btn_UseFrontCamera);
        btn_UseFrontCamera.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (SpeechRecognitionService.mIsRunning) {
                    SpeechRecognitionService

    .stopContinuousListening(MainActivity.this);
                }
                SpeechRecognitionService.startContinuousListening(
                        MainActivity.this,
btn_UseFrontCamera.isChecked());
            }
        });

        btn_stopspeechrecogsvc = (ToggleButton) this
                .findViewById(R.id.btn_StopSpeechRecogSvc);
        btn_stopspeechrecogsvc.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (SpeechRecognitionService.mIsRunning) {
                    SpeechRecognitionService

    .stopContinuousListening(MainActivity.this);
                } else {
                    if (btn_UseFrontCamera == null) {
                        btn_UseFrontCamera = (ToggleButton)
MainActivity.this

    .findViewById(R.id.btn_UseFrontCamera);
                    }

    SpeechRecognitionService.startContinuousListening(
                            MainActivity.this,
btn_UseFrontCamera.isChecked());
                }
            }
        });
```

```
        btn_startrecord = (Button) this
                .findViewById(R.id.button_StartRecording);
        btn_startrecord.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SpeechRecognitionService

    .stopContinuousListening(MainActivity.this);

                Intent intent = new Intent(MainActivity.this,
                        VideoCaptureActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                intent.putExtra("isFrontCamera",
btn_UseFrontCamera.isChecked());
                MainActivity.this.startActivity(intent);
            }
        });

        // Check to see if the proper keys and play-list IDs have
been set up
        if (isCorrectlyConfigured()) {
            credential = GoogleAccountCredential.usingOAuth2(
                    getApplicationContext(),
Arrays.asList(Auth.SCOPES));
            // set exponential back-off policy
            credential.setBackOff(new ExponentialBackOff());

            loadAccount();

            if (mChosenAccountName == null) {
                chooseAccount();
            }
            credential.setSelectedAccountName(mChosenAccountName);

            mUserFragment = (UserFragment) getFragmentManager()
                    .findFragmentById(R.id.user_fragment);

            if (btn_UseFrontCamera == null) {
                btn_UseFrontCamera = (ToggleButton) this
                        .findViewById(R.id.btn_UseFrontCamera);
            }

            SpeechRecognitionService.startContinuousListening(
                    MainActivity.this,
btn_UseFrontCamera.isChecked());
            btn_stopspeechrecogsvc.setChecked(true);
        }

        loginButton = (LoginButton)
this.findViewById(R.id.login_button);
        loginButton.setReadPermissions("user_friends");
        loginButton.setReadPermissions("public_profile");
        loginButton.setReadPermissions("email");
        // loginButton.setReadPermissions("publish_actions");
        // Callback registration
```

```
        loginButton.registerCallback(callbackManager, this);

        mIsPost = false;

        accessTokenTracker = new AccessTokenTracker() {
            @Override
            protected void onCurrentAccessTokenChanged(AccessToken
arg0,
                    AccessToken arg1) {
                if (arg1 == null) {
                    // what to do when user logout
                    mIsPost = false;
                }
            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();

        // if (broadcastReceiver == null)
        // broadcastReceiver = new UploadBroadcastReceiver();
        // IntentFilter intentFilter = new IntentFilter(
        // REQUEST_AUTHORIZATION_INTENT);
        // LocalBroadcastManager.getInstance(this).registerReceiver(
        // broadcastReceiver, intentFilter);

        callerIntent = getIntent();

        if (callerIntent != null) {
            if (callerIntent.getBooleanExtra("EXIT", false)) {
                finish();
            } else {
                if (callerIntent.getBooleanExtra("UPLOAD", false)) {
                    uploadVideo(callerIntent.getData());
                }
            }
        }
    }

    private boolean isGooglePlayServicesAvailable() {
        GoogleApiAvailability gAPIAvail =
GoogleApiAvailability.getInstance();
        final int connectionStatusCode = gAPIAvail
                .isGooglePlayServicesAvailable(this);
        if (gAPIAvail.isUserResolvableError(connectionStatusCode)) {

    showGooglePlayServicesAvailabilityErrorDialog(connectionStatusCod
e);
            return false;
            }

        return true;
    }

    private void showGooglePlayServicesAvailabilityErrorDialog(
```

```
            final int connectionStatusCode) {
        runOnUiThread(new Runnable() {
            public void run() {
                GoogleApiAvailability gAPIAvail =
GoogleApiAvailability
                        .getInstance();
                Dialog dialog =
gAPIAvail.getErrorDialog(MainActivity.this,
                        connectionStatusCode,
REQUEST_GOOGLE_PLAY_SERVICES);
                dialog.show();
            }
        });
    }

    private void checkGooglePlayServices() {
        // check if there is already an account selected
        if (credential.getSelectedAccountName() == null) {
            // ask user to choose account
            chooseAccount();
        }
    }

    private void chooseAccount() {
        startActivityForResult(credential.newChooseAccountIntent(),
                REQUEST_ACCOUNT_PICKER);
    }

    public void uploadVideo(Uri recordedFile) {
        if (mChosenAccountName == null) {
            return;
        }
        // if a video is picked or recorded.
        if (recordedFile != null) {
            String[] projection = { MediaStore.Images.Media.DATA };
            @SuppressWarnings("deprecation")
            Cursor cursor = managedQuery(recordedFile, projection,
null, null,
                    null);
            if (cursor.getCount() == 0) {
                // throw new
                // IOException(String.format("cannot find data from
%s",
                // recordedFile.toString()));
            } else {
                cursor.moveToFirst();
            }

            String filePath = cursor.getString(cursor

    .getColumnIndex(MediaStore.Video.VideoColumns.DATA));

            Intent uploadIntent = new Intent(this,
UploadService.class);
            uploadIntent.setData(recordedFile);
            uploadIntent.putExtra(VideoUploadActivity.ACCOUNT_KEY,
                    mChosenAccountName);
```

```
            uploadIntent.putExtra("Path", filePath);
            startService(uploadIntent);
            Toast.makeText(this, "Upload Mulai...",
Toast.LENGTH_LONG).show();
        }

        finish();
    }

    private void saveAccount() {
        SharedPreferences sp = PreferenceManager
                .getDefaultSharedPreferences(this);
        sp.edit().putString(VideoUploadActivity.ACCOUNT_KEY,
mChosenAccountName)
                .commit();
    }

    private void loadAccount() {
        SharedPreferences sp = PreferenceManager
                .getDefaultSharedPreferences(this);
        mChosenAccountName = sp
                .getString(VideoUploadActivity.ACCOUNT_KEY, null);
        invalidateOptionsMenu();
    }

    private boolean isCorrectlyConfigured() {
        if (Auth.KEY.length() == 0) {
            return false;
        }
        return true;
    }

    private class UploadBroadcastReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            if
(intent.getAction().equals(REQUEST_AUTHORIZATION_INTENT)) {
                Log.d(TAG, "Request auth received - executing the
intent");
                Intent toRun = intent

    .getParcelableExtra(REQUEST_AUTHORIZATION_INTENT_PARAM);
                startActivityForResult(toRun,
REQUEST_AUTHORIZATION);
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
        case REQUEST_GOOGLE_PLAY_SERVICES:
            if (resultCode == Activity.RESULT_OK) {
                checkGooglePlayServices();
            } else {
```

```
                    isGooglePlayServicesAvailable();
                }
                break;
            case REQUEST_AUTHORIZATION:
                if (resultCode != Activity.RESULT_OK) {
                    chooseAccount();
                }
                break;
            case REQUEST_ACCOUNT_PICKER:
                if (resultCode == Activity.RESULT_OK && data != null
                        && data.getExtras() != null) {
                    String accountName = data.getExtras().getString(
                            AccountManager.KEY_ACCOUNT_NAME);
                    if (accountName != null) {
                        mChosenAccountName = accountName;
                        credential.setSelectedAccountName(accountName);
                        saveAccount();
                        mUserFragment.setProfileInfo(accountName);
                    }
                }
                break;
            case REQUEST_FACEBOOK_LOGIN:
                callbackManager.onActivityResult(requestCode,
resultCode, data);
                break;
            }
        }

        @Override
        public void onCancel() {
            // TODO Auto-generated method stub
            Log.d("FBLOGIN", "Cancel");
        }

        @Override
        public void onError(FacebookException arg0) {
            // TODO Auto-generated method stub
            Log.d("FBLOGIN", "Error");
        }

        @Override
        public void onSuccess(LoginResult arg0) {
            // TODO Auto-generated method stub
            Log.d("FBLOGIN", "Success");
            if (!accessTokenTracker.isTracking()) {
                accessTokenTracker.startTracking();
            }
            if (!mIsPost) {

        LoginManager.getInstance().logInWithPublishPermissions(this,
                        Arrays.asList("publish_actions"));
                mIsPost = true;
            }
        }

        @Override
        public void onConnected(String connectedAccountName) {
```

```
        // mChosenAccountName = connectedAccountName;
        // saveAccount();
    }
}
```

Kelas MyApplication.java

```
package com.video.facetube;

import android.app.Application;
import android.content.Context;
import android.support.multidex.MultiDex;

public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

Kelas OnSurfaceCreatedEventListener.java

```
package com.video.facetube;

import java.util.EventListener;

import android.view.SurfaceHolder;

public interface OnSurfaceCreatedEventListener extends EventListener
{
    public void OnSurfaceCreated(SurfaceHolder holder);
}
```

Kelas ResumableUpload.java

```
package com.video.facetube;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
```

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.ThumbnailUtils;
import android.net.Uri;
import android.provider.MediaStore.Video.Thumbnails;
import android.support.v4.app.NotificationCompat;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;

import com.facebook.CallbackManager;
import
com.google.api.client.googleapis.extensions.android.gms.auth.GooglePl
ayServicesAvailabilityIOException;
import
com.google.api.client.googleapis.extensions.android.gms.auth.UserReco
verableAuthIOException;
import
com.google.api.client.googleapis.json.GoogleJsonResponseException;
import com.google.api.client.googleapis.media.MediaHttpUploader;
import
com.google.api.client.googleapis.media.MediaHttpUploaderProgressListe
ner;
import com.google.api.client.http.InputStreamContent;
import com.google.api.client.util.DateTime;
import com.google.api.services.youtube.YouTube;
import com.google.api.services.youtube.model.GeoPoint;
//import
com.google.api.services.youtube.model.ActivityContentDetails.Upload;
import com.google.api.services.youtube.model.Video;
import com.google.api.services.youtube.model.VideoListResponse;
import com.google.api.services.youtube.model.VideoRecordingDetails;
import com.google.api.services.youtube.model.VideoSnippet;
import com.google.api.services.youtube.model.VideoStatus;

/**
 * @author Ibrahim Ulukaya <ulukaya@google.com>
 *         <p/>
 *         YouTube Resumable Upload controller class.
 */
public class ResumableUpload {
    /**
     * Indicates that the video is fully processed, see
     * https://www.googleapis.com/discovery/v1/apis/youtube/v3/rpc
```

```
     */
    private static final String SUCCEEDED = "succeeded";
    private static final String TAG = "UploadingActivity";
    private static int UPLOAD_NOTIFICATION_ID = 1001;
    private static int PLAYBACK_NOTIFICATION_ID = 1002;
    /*
     * Global instance of the format used for the video being
uploaded (MIME
     * type).
     */
    private static String VIDEO_FILE_FORMAT = "video/*";

    private static String generateKeywordFromPlaylistId(String
playlistId) {
        if (playlistId == null)
            playlistId = "";
        if (playlistId.indexOf("PL") == 0) {
            playlistId = playlistId.substring(2);
        }
        playlistId = playlistId.replaceAll("\\W", "");
        String keyword = "VideoUploader".concat(playlistId);
        if (keyword.length() > 38) {
            keyword = keyword.substring(0, 38);
        }
        return keyword;
    }

    /**
     * Uploads user selected video in the project folder to the
user's YouTube
     * account using OAuth2 for authentication.
     */
    public static String upload(YouTube youtube,
            final InputStream fileInputStream, final long fileSize,
            final Uri mFileUri, final String path, final Context
context) {
        final NotificationManager notifyManager =
(NotificationManager) context
                .getSystemService(Context.NOTIFICATION_SERVICE);
        final NotificationCompat.Builder builder = new
NotificationCompat.Builder(
                context);

        Intent notificationIntent = new Intent(context,
                VideoUploadActivity.class);
        notificationIntent.setData(mFileUri);
        notificationIntent.setAction(Intent.ACTION_VIEW);
        Bitmap thumbnail = ThumbnailUtils.createVideoThumbnail(path,
                Thumbnails.MICRO_KIND);
        PendingIntent contentIntent =
PendingIntent.getActivity(context, 0,
                notificationIntent,
PendingIntent.FLAG_CANCEL_CURRENT);
        builder.setContentTitle("Unggah Youtube")
                .setContentText("Mulai unggah ke Youtube")
                .setSmallIcon(R.drawable.movieicon)
                .setContentIntent(contentIntent)
```

```
                .setStyle(
                        new NotificationCompat.BigPictureStyle()
                                .bigPicture(thumbnail));
        notifyManager.notify(UPLOAD_NOTIFICATION_ID,
builder.build());

        String videoId = null;
        try {
            Video videoObjectDefiningMetadata = new Video();

            VideoStatus status = new VideoStatus();
            status.setPrivacyStatus("public");
            videoObjectDefiningMetadata.setStatus(status);

            VideoSnippet snippet = new VideoSnippet();

            Calendar cal = Calendar.getInstance();
            Date dt = cal.getTime();
            String address = null;
            GPSTracker gps = new GPSTracker(context);
            if (gps.canGetLocation()) {
                double latitude = gps.getLatitude();
                double longitude = gps.getLongitude();

                VideoRecordingDetails details = new
VideoRecordingDetails();
                GeoPoint geoPoint = new GeoPoint();
                geoPoint.setLatitude(latitude);
                geoPoint.setLongitude(longitude);
                details.setLocation(geoPoint);
                details.setRecordingDate(new DateTime(dt));

                address = getCurrentLocationViaJSON(latitude,
longitude);

    videoObjectDefiningMetadata.setRecordingDetails(details);
            }

            snippet.setTitle("FaceTube Video - " + dt);

            snippet.setDescription("Video diunggah oleh FaceTube
dengan menggunakan YouTube Data API V3 "
                    + "pada " + dt + (address != null ? " di " +
address : ""));

            // Set your keywords.
            snippet.setTags(Arrays.asList("VideoUploader",

    ResumableUpload.generateKeywordFromPlaylistId("")));

            // Set completed snippet to the video object.
            videoObjectDefiningMetadata.setSnippet(snippet);
            InputStreamContent mediaContent = new
InputStreamContent(
                    VIDEO_FILE_FORMAT, new
BufferedInputStream(fileInputStream));
            mediaContent.setLength(fileSize);
```

```
            /*
             * The upload command includes: 1. Information we want
returned
             * after file is successfully uploaded. 2. Metadata we
want
             * associated with the uploaded video. 3. Video file
itself.
             */
            YouTube.Videos.Insert videoInsert =
youtube.videos().insert(
                    "snippet,statistics,status,recordingDetails",
                    videoObjectDefiningMetadata, mediaContent);

            // Set the upload type and add event listener.
            MediaHttpUploader uploader =
videoInsert.getMediaHttpUploader();

            /*
             * Sets whether direct media upload is enabled or
disabled. True =
             * whole media content is uploaded in a single request.
False
             * (default) = resumable media upload protocol to upload
in data
             * chunks.
             */
            uploader.setDirectUploadEnabled(false);

            MediaHttpUploaderProgressListener progressListener = new
MediaHttpUploaderProgressListener() {
                public void progressChanged(MediaHttpUploader
uploader)
                        throws IOException {
                    switch (uploader.getUploadState()) {
                    case INITIATION_STARTED:
                        builder.setContentText("Init
Started").setProgress(
                                (int) fileSize,
                                (int)
uploader.getNumBytesUploaded(), false);

    notifyManager.notify(UPLOAD_NOTIFICATION_ID,
                                builder.build());
                        break;
                    case INITIATION_COMPLETE:
                        builder.setContentText("Init
Completed").setProgress(
                        (int) fileSize,
                        (int) upload
er.getNumBytesUploaded(), false);

    notifyManager.notify(UPLOAD_NOTIFICATION_ID,
                                builder.build());
                        break;
                    case MEDIA_IN_PROGRESS:
                        builder.setContentTitle(
```

```
                                    "Unggah Youtube"
                                            + (int)
(uploader.getProgress() * 100)
                                            + "%")
                                    .setContentText("Mulai Mengunggah")
                                    .setProgress((int) fileSize,
                                            (int)
uploader.getNumBytesUploaded(),
                                            false);

    notifyManager.notify(UPLOAD_NOTIFICATION_ID,
                                    builder.build());
                            break;
                        case MEDIA_COMPLETE:
                            builder.setContentTitle("Youtube Upload
Completed")
                                    .setContentText("Upload Completed")
                                    // Removes the progress bar
                                    .setProgress(0, 0, false);

    notifyManager.notify(UPLOAD_NOTIFICATION_ID,
                                    builder.build());
                            // Toast.makeText(context, "Upload
Selesai",
                            // Toast.LENGTH_LONG).show();
                            break;
                        case NOT_STARTED:
                            // Log.d(this.getClass().getSimpleName(),
                            // "Upload Not Started");
                            break;
                        }
                    }
                };
                uploader.setProgressListener(progressListener);

                // Execute upload.
                Video returnedVideo = videoInsert.execute();
                Log.d(TAG, "Video upload completed");
                // Toast.makeText(context, "Upload Selesai",
                // Toast.LENGTH_LONG).show();
                videoId = returnedVideo.getId();
                Log.d(TAG, String.format("videoId = [%s]", videoId));
                FbManager m = new FbManager(context,
                        CallbackManager.Factory.create());
                m.share("https://www.youtube.com/watch?v=" + videoId);
            } catch (final GooglePlayServicesAvailabilityIOException
availabilityException) {
                Log.e(TAG, "GooglePlayServicesAvailabilityIOException",
                        availabilityException);
                notifyFailedUpload(context, "Cannot Access Google Play
Service",
                        notifyManager, builder);
            } catch (UserRecoverableAuthIOException
userRecoverableException) {
                Log.i(TAG,
String.format("UserRecoverableAuthIOException: %s",
                        userRecoverableException.getMessage()));
```

```
                requestAuth(context, userRecoverableException);
            } catch (IOException e) {
                Log.e(TAG, "IOException", e);
                notifyFailedUpload(context, "Please Try Again",
notifyManager,
                        builder);
            } catch (Throwable t) {
                // Log.e(TAG, "Exception", t);
                notifyFailedUpload(context, "Please Try Again",
notifyManager,
                        builder);
            }
            return videoId;
        }

        private static void requestAuth(Context context,
                UserRecoverableAuthIOException userRecoverableException)
{
            LocalBroadcastManager manager = LocalBroadcastManager
                    .getInstance(context);
            Intent authIntent = userRecoverableException.getIntent();
            Intent runReqAuthIntent = new Intent(
                    VideoUploadActivity.REQUEST_AUTHORIZATION_INTENT);
            runReqAuthIntent.putExtra(

        VideoUploadActivity.REQUEST_AUTHORIZATION_INTENT_PARAM,
                    authIntent);
            manager.sendBroadcast(runReqAuthIntent);
            Log.d(TAG, String.format("Sent broadcast %s",
                    VideoUploadActivity.REQUEST_AUTHORIZATION_INTENT));
        }

        private static void notifyFailedUpload(Context context, String
message,
                NotificationManager notifyManager,
                NotificationCompat.Builder builder) {
            builder.setContentTitle("Youtube Upload Failed")
                    .setContentText(message);
            notifyManager.notify(UPLOAD_NOTIFICATION_ID,
builder.build());
            Log.e(ResumableUpload.class.getSimpleName(), message);
        }

        public static void showSelectableNotification(String videoId,
                Context context) {
            Log.d(TAG, String.format(
                    "Posting selectable notification for video ID [%s]",
videoId));
            final NotificationManager notifyManager =
(NotificationManager) context
                    .getSystemService(Context.NOTIFICATION_SERVICE);
            final NotificationCompat.Builder builder = new
NotificationCompat.Builder(
                    context);
            Intent notificationIntent = new Intent(context,
                    VideoUploadActivity.class);
```

```
        notificationIntent.putExtra(VideoUploadActivity.YOUTUBE_ID,
videoId);
        notificationIntent.setAction(Intent.ACTION_VIEW);

        URL url;
        try {
            url = new URL("https://i1.ytimg.com/vi/" + videoId
                    + "/mqdefault.jpg");
            Bitmap thumbnail =
BitmapFactory.decodeStream(url.openConnection()
                    .getInputStream());
            PendingIntent contentIntent =
PendingIntent.getActivity(context, 0,
                    notificationIntent,
PendingIntent.FLAG_CANCEL_CURRENT);
            builder.setContentTitle("Watch Your Video")
                    .setContentText("see the newly uploaded video")
                    .setContentIntent(contentIntent)
                    //
.setSmallIcon(R.drawable.ic_stat_device_access_video)
                    .setStyle(
                            new
NotificationCompat.BigPictureStyle()
                                    .bigPicture(thumbnail));
            notifyManager.notify(PLAYBACK_NOTIFICATION_ID,
builder.build());
            Log.d(TAG, String
            .format("Selectable notification for video ID [%s]
posted",
                            videoId));
        } catch (MalformedURLException e) {
            Log.e(TAG, e.getMessage());
        } catch (IOException e) {
            Log.e(TAG, e.getMessage());
        }
    }

    /**
     * @return url of thumbnail if the video is fully processed
     */
    public static boolean checkIfProcessed(String videoId, YouTube
youtube) {
        try {
            YouTube.Videos.List list = youtube.videos().list(
                    "processingDetails");
            list.setId(videoId);
            VideoListResponse listResponse = list.execute();
            List<Video> videos = listResponse.getItems();
            if (videos.size() == 1) {
                Video video = videos.get(0);
                String status = video.getProcessingDetails()
                        .getProcessingStatus();
                Log.e(TAG, String.format("Processing status of [%s]
is [%s]",
                        videoId, status));
                if (status.equals(SUCCEEDED)) {
                    return true;
```

```
                }
            } else {
                // can't find the video
                Log.e(TAG,
                        String.format("Can't find video with ID
[%s]", videoId));
                return false;
            }
        } catch (IOException e) {
            Log.e(TAG, "Error fetching video metadata", e);
        }
        return false;
    }

    private static int NOTIFICATION_ID = 1001;

    public static String upload(YouTube youtube,
            final InputStream fileInputStream, final long fileSize,
            Context context) {
        final NotificationManager mNotifyManager =
(NotificationManager) context
                .getSystemService(Context.NOTIFICATION_SERVICE);
        final NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(
                context);
        mBuilder.setContentTitle("YouTube Upload").setContentText(
                "Direct Lite upload started");
        // .setSmallIcon(R.drawable.icon);
        String videoId = null;
        try {
            // Add extra information to the video before uploading.
            Video videoObjectDefiningMetadata = new Video();

            /*
             * Set the video to public, so it is available to
everyone (what
             * most people want). This is actually the default, but
I wanted you
             * to see what it looked like in case you need to set it
to
             * "unlisted" or "private" via API.
             */
            VideoStatus status = new VideoStatus();
            status.setPrivacyStatus("public");
            videoObjectDefiningMetadata.setStatus(status);

            // We set a majority of the metadata with the
VideoSnippet object.
            VideoSnippet snippet = new VideoSnippet();
            /*
             * The Calendar instance is used to create a unique name
and
             * description for test purposes, so you can see
multiple files
             * being uploaded. You will want to remove this from
your project
             * and use your own standard names.
```

```
             */
            Calendar cal = Calendar.getInstance();
            snippet.setTitle("FaceTube Video - " + cal.getTime());
            snippet.setDescription("Video diunggah oleh FaceTube
dengan menggunakan YouTube Data API V3 "
                    + "pada " + cal.getTime());

            // Set your keywords.
            List<String> tags = new ArrayList<String>();
            tags.add("ytdl");
            // tags.add(Upload
            //
.generateKeywordFromPlaylistId(Constants.UPLOAD_PLAYLIST));
            snippet.setTags(tags);

            // Set completed snippet to the video object.
            videoObjectDefiningMetadata.setSnippet(snippet);

            InputStreamContent mediaContent = new
InputStreamContent(
                    VIDEO_FILE_FORMAT, new
BufferedInputStream(fileInputStream));
            mediaContent.setLength(fileSize);

            /*
             * The upload command includes: 1. Information we want
returned
             * after file is successfully uploaded. 2. Metadata we
want
             * associated with the uploaded video. 3. Video file
itself.
             */
            YouTube.Videos.Insert videoInsert =
youtube.videos().insert(
                    "snippet,statistics,status",
videoObjectDefiningMetadata,
                    mediaContent);

            // Set the upload type and add event listener.
            MediaHttpUploader uploader =
videoInsert.getMediaHttpUploader();

            /*
             * Sets whether direct media upload is enabled or
disabled. True =
             * whole media content is uploaded in a single request.
False
             * (default) = resumable media upload protocol to upload
in data
             * chunks.
             */
            uploader.setDirectUploadEnabled(false);

            MediaHttpUploaderProgressListener progressListener = new
MediaHttpUploaderProgressListener() {
                public void progressChanged(MediaHttpUploader
uploader)
```

```
                        throws IOException {
                    switch (uploader.getUploadState()) {
                    case INITIATION_STARTED:
                        mBuilder.setContentText("Initiation
Started")
                                .setProgress((int) fileSize,
                                        (int)
uploader.getNumBytesUploaded(),
                                        false);
                        mNotifyManager
                                .notify(NOTIFICATION_ID,
mBuilder.build());
                        break;
                    case INITIATION_COMPLETE:
                        mBuilder.setContentText("Initiation
Completed")                     .setProgress((int) fileSize,
                (int) uploader.getNumBytesUploaded(),false);
                        mNotifyManager
                        .notify(NOTIFICATION_ID, mBuilder.build());
                        break;
                    case MEDIA_IN_PROGRESS:
                        mBuilder.setContentTitle(
                                "YouTube Upload "
                                        + (int)
(uploader.getProgress() * 100)
                                        + "%")
                                .setContentText(
                                        "Direct Lite upload in
progress")
                                .setProgress((int) fileSize,
                                        (int)
uploader.getNumBytesUploaded(),
                                        false);
                        mNotifyManager
                                .notify(NOTIFICATION_ID,
mBuilder.build());
                        break;
                    case MEDIA_COMPLETE:
                        mBuilder.setContentTitle("YouTube Upload
Completed")
                                .setContentText("Upload complete")
                                // Removes the progress bar
                                .setProgress(0, 0, false);
                        mNotifyManager
                                .notify(NOTIFICATION_ID,
mBuilder.build());
                    case NOT_STARTED:
                        Log.d(this.getClass().getSimpleName(),
                                "Upload Not Started!");
                        break;
                    }
                }
            };
            uploader.setProgressListener(progressListener);

            // Execute upload.
            Video returnedVideo = videoInsert.execute();
```

```
            videoId = returnedVideo.getId();

        } catch (final GoogleJsonResponseException e) {
            if (401 == e.getDetails().getCode()) {
                Log.e(ResumableUpload.class.getSimpleName(),
e.getMessage());
                @SuppressWarnings("unused")
                LocalBroadcastManager manager =
LocalBroadcastManager
                        .getInstance(context);
                // manager.sendBroadcast(new Intent(
                // MainActivity.INVALIDATE_TOKEN_INTENT));
            }
        } catch (IOException e) {
            Log.e("IOException", e.getMessage());
        } catch (Throwable t) {
            Log.e("Throwable", t.getMessage());
        }
        return videoId;
    }

    public static JSONObject getLocationInfo(double lat, double lng)
{
        HttpGet httpGet = new HttpGet(
        "http://maps.googleapis.com/maps/api/geocode/json?latlng="
                    + lat + "," + lng + "&sensor=true");
        HttpClient client = new DefaultHttpClient();
        HttpResponse response;
        StringBuilder stringBuilder = new StringBuilder();

        try {
            response = client.execute(httpGet);
            HttpEntity entity = response.getEntity();
            InputStream stream = entity.getContent();
            int b;
            while ((b = stream.read()) != -1) {
                stringBuilder.append((char) b);
            }
        } catch (ClientProtocolException e) {
        } catch (IOException e) {
        }

        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject = new JSONObject(stringBuilder.toString());
        } catch (JSONException e) {
            e.printStackTrace();
        }

        return jsonObject;
    }

    public static String getCurrentLocationViaJSON(double lat, double
lng) {

        JSONObject jsonObj = getLocationInfo(lat, lng);
        Log.i("JSON string =>", jsonObj.toString());
```

```
        String currentLocation = "";
        String street_address = null;
        String postal_code = null;

        try {
            String status = jsonObj.getString("status").toString();
            Log.i("status", status);

            if (status.equalsIgnoreCase("OK")) {
                JSONArray results = jsonObj.getJSONArray("results");
                int i = 0;
                Log.i("i", i + "," + results.length()); // TODO
delete this
                do {

                    JSONObject r = results.getJSONObject(i);
                    JSONArray typesArray = r.getJSONArray("types");
                    String types = typesArray.getString(0);

                    if (types.equalsIgnoreCase("street_address")) {
                        street_address =
r.getString("formatted_address")
                                .split(",")[0];
                        Log.i("street_address", street_address);
                    } else if
(types.equalsIgnoreCase("postal_code")) {
                        postal_code =
r.getString("formatted_address");
                        Log.i("postal_code", postal_code);
                    }

                    if (street_address != null && postal_code !=
null) {
                        currentLocation = street_address + "," +
postal_code;
                        Log.i("Current Location =>",
currentLocation); // Delete
                                                                //
this
                        i = results.length();
                    }

                    i++;
                } while (i < results.length());

                Log.i("JSON Geo Locatoin =>", currentLocation);
                return currentLocation;
            }

        } catch (JSONException e) {
            Log.e("testing", "Failed to load JSON");
            e.printStackTrace();
        }
        return null;
    }
```

```
}
```

Kelas SpeechRecognitionService.java

```
package com.video.facetube;

import java.util.ArrayList;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.os.Build;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.IBinder;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.util.Log;
import android.widget.Toast;

import com.video.facetube.matching.FaceTubeMatching;
import com.video.facetube.matching.FaceTubeMatching.MatchingCallback;

public class SpeechRecognitionService extends Service implements
        MatchingCallback {

    private static final String TAG = "SpeechRecognitionSvc";
    protected static AudioManager mAudioManager;
    protected SpeechRecognizer mSpeechRecognizer;
    protected Intent mSpeechRecognizerIntent;
    protected boolean mIsListening;
    protected volatile boolean mIsCountDownOn;
    private static boolean mIsStreamSolo;
    public static boolean mIsRunning = false;

    public static void startContinuousListening(Context context,
            boolean isFrontCamera) {
        if (!mIsRunning) {
            Intent service = new Intent(context,
SpeechRecognitionService.class);
            service.putExtra("isFrontCamera", isFrontCamera);
            context.startService(service);
            mIsRunning = true;
        }
    }
    public static void stopContinuousListening(Context context) {
        if (mIsRunning) {
            Intent service = new Intent(context,
SpeechRecognitionService.class);
            context.stopService(service);
            mIsRunning = false;
        }
    }
```

```
    private boolean mIsFrontCamera = false;

    @Override
    public void onCreate() {
        super.onCreate();
        mAudioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
        this.initVoiceRecognizer();
    }

    private void initVoiceRecognizer() {
        if (mSpeechRecognizer == null
                && SpeechRecognizer

    .isRecognitionAvailable(SpeechRecognitionService.this)) {
            mSpeechRecognizer =
SpeechRecognizer.createSpeechRecognizer(this);
            mSpeechRecognizer
                    .setRecognitionListener(new
SpeechRecognitionListener());
            mSpeechRecognizerIntent = new Intent(
                    RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
            mSpeechRecognizerIntent.putExtra(
                    RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
            mSpeechRecognizerIntent.putExtra(
                    RecognizerIntent.EXTRA_CALLING_PACKAGE,
getPackageName());
            mSpeechRecognizerIntent.putExtra(
                    RecognizerIntent.EXTRA_MAX_RESULTS, 500);

    mSpeechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
                    "id-ID");
            mSpeechRecognizerIntent
                    .putExtra(

    RecognizerIntent.EXTRA_SPEECH_INPUT_COMPLETE_SILENCE_LENGTH_MILLI
S,
                            100);
        }
    }

    protected void CancelListening() {
        UnMuteSound();
        mSpeechRecognizer.cancel();
        Log.d(TAG, "message canceled recognizer"); //$NON-NLS-1$
    }

    private void UnMuteSound() {
        if (mIsStreamSolo) {

    mAudioManager.setStreamSolo(AudioManager.STREAM_VOICE_CALL,
false);
            mAudioManager.setMode(AudioManager.MODE_NORMAL);
            mIsStreamSolo = false;
        }
    }
```

```
    private void MuteSound() {
        if (Build.VERSION.SDK_INT >= 16) {
            // turn off beep sound
            if (!mIsStreamSolo) {
                mAudioManager.setMode(AudioManager.MODE_IN_CALL);

    mAudioManager.setStreamSolo(AudioManager.STREAM_VOICE_CALL,
true);
                mIsStreamSolo = true;
            }
        }
    }
    protected void StartListening(boolean cancelExisting) {
        if (cancelExisting) {
            this.CancelListening();
        }

        MuteSound();

        try {

    mSpeechRecognizer.startListening(mSpeechRecognizerIntent);
        } catch (Exception x) {
            Log.e(TAG, x.getMessage());
        }
        Log.d(TAG, "message start listening"); //$NON-NLS-1$
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
{
        try {
            Toast.makeText(this, "mendengar...",
Toast.LENGTH_SHORT).show();
            if (intent != null) {
                Bundle extras = intent.getExtras();
                if (extras != null) {
                    if (extras.containsKey("isFrontCamera")) {
                        mIsFrontCamera =
extras.getBoolean("isFrontCamera");
                    }
                }
            }
            this.StartListening(false);
        } catch (Exception x) {
            Log.d(TAG, x.getMessage());
        }
        return super.onStartCommand(intent, flags, startId);
    }

    // Count down timer for Jelly Bean work around
    protected CountDownTimer mNoSpeechCountDown;

    /*
     * new CountDownTimer(5000, 5000) {
     *
```

```
     * @Override public void onTick(long millisUntilFinished) { }
     *
     * @Override public void onFinish() { mIsCountDownOn = false;
     *
     * try { mSpeechRecognizer.stopListening(); mIsListening = false;
     * StartListening(true); } catch (Exception e) { } } };
     */

    @Override
    public void onDestroy() {
        super.onDestroy();

        // UnMuteSound();

        if (mIsCountDownOn) {
            mNoSpeechCountDown.cancel();
        }
        if (mSpeechRecognizer != null) {
            mSpeechRecognizer.destroy();
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        Log.d(TAG, "onBind"); //$NON-NLS-1$
        return null;
    }

    protected class SpeechRecognitionListener implements
RecognitionListener {

        @Override
        public void onBeginningOfSpeech() {
            // speech input will be processed, so there is no need
for count
            // down anymore
    protected void StartListening(boolean cancelExisting) {
        if (cancelExisting) {
            this.CancelListening();
        }

        MuteSound();

        try {

    mSpeechRecognizer.startListening(mSpeechRecognizerIntent);
        } catch (Exception x) {
            Log.e(TAG, x.getMessage());
        }
        Log.d(TAG, "message start listening"); //$NON-NLS-1$
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
{
        try {
```

```
            Toast.makeText(this, "mendengar...",
Toast.LENGTH_SHORT).show();
            if (intent != null) {
                Bundle extras = intent.getExtras();
                if (extras != null) {
                    if (extras.containsKey("isFrontCamera")) {
                        mIsFrontCamera =
extras.getBoolean("isFrontCamera");
                    }
                }
            }
            this.StartListening(false);
        } catch (Exception x) {
            Log.d(TAG, x.getMessage());
        }
        return super.onStartCommand(intent, flags, startId);
    }

    // Count down timer for Jelly Bean work around
    protected CountDownTimer mNoSpeechCountDown;

    /*
     * new CountDownTimer(5000, 5000) {
     *
     * @Override public void onTick(long millisUntilFinished) { }
     *
     * @Override public void onFinish() { mIsCountDownOn = false;
     *
     * try { mSpeechRecognizer.stopListening(); mIsListening = false;
     * StartListening(true); } catch (Exception e) { } } };
     */

    @Override
    public void onDestroy() {
        super.onDestroy();

        // UnMuteSound();

        if (mIsCountDownOn) {
            mNoSpeechCountDown.cancel();
        }
        if (mSpeechRecognizer != null) {
            mSpeechRecognizer.destroy();
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        Log.d(TAG, "onBind"); //$NON-NLS-1$
        return null;
    }

    protected class SpeechRecognitionListener implements
RecognitionListener {

        @Override
        public void onBeginningOfSpeech() {
```

```
            // speech input will be processed, so there is no need
for count
            // down anymore
            /*
             * if (mIsCountDownOn) { mIsCountDownOn = false;
             * mNoSpeechCountDown.cancel(); }
             */
            Log.d(TAG, "onBeginingOfSpeech"); //$NON-NLS-1$
        }

        @Override
        public void onBufferReceived(byte[] buffer) {
            Log.d(TAG, "onBufferReceived"); //$NON-NLS-1$
        }

        @Override
        public void onEndOfSpeech() {
            Log.d(TAG, "onEndOfSpeech"); //$NON-NLS-1$
        }

        @Override
        public void onError(int error) {
            mSpeechRecognizer.cancel();
            stopSelf();
            Intent i = new Intent(SpeechRecognitionService.this,
                    SpeechRecognitionService.class);
            i.putExtra("isFrontCamera",
                    SpeechRecognitionService.this.mIsFrontCamera);
            startService(i);

            Log.d(TAG, "error = " + error); //$NON-NLS-1$
        }

        @Override
        public void onEvent(int eventType, Bundle params) {
            Log.d(TAG, "onEvent");
        }

        @Override
        public void onPartialResults(Bundle partialResults) {
            onResultsHandler(partialResults);
        }

        @Override
        public void onReadyForSpeech(Bundle params) {
            Log.d(TAG, "onReadyForSpeech: Cancel Timer");

            if (mNoSpeechCountDown != null) {
                mNoSpeechCountDown.cancel();
            }
        }

        @Override
        public void onResults(Bundle results) {
            onResultsHandler(results);
        }
```

```
        @Override
        public void onRmsChanged(float rmsdB) {
            Log.d(TAG, mNoSpeechCountDown != null ? "count down" :
"null");
        }

        private void onResultsHandler(Bundle results) {
            if (mNoSpeechCountDown != null) {
                mNoSpeechCountDown.cancel();
            }
            Log.d(TAG, "onResults");
            ArrayList<String> data = results

    .getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
            ProcessSpeechRecognitionResult(data);
        }
    }

    private void ProcessSpeechRecognitionResult(ArrayList<String>
results) {
        FaceTubeMatching ftm = new FaceTubeMatching(this);
        if (ftm.Run(results)) {
            SpeechRecognitionService.this.stopSelf();
            // mSpeechRecognizer.stopListening();
            mIsRunning = false;

            Intent i = new Intent();
            i.setClass(this, VideoCaptureActivity.class);
            i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            i.putExtra("isFrontCamera", mIsFrontCamera);
            startActivity(i);
        } else {
            if (mNoSpeechCountDown == null) {
                mNoSpeechCountDown = new CountDownTimer(2000, 500) {
                    @Override
                    public void onTick(long l) {
                    }

                    @Override
                    public void onFinish() {
                        Log.d("Speech",
                                "Timer.onFinish: Timer Finished,
Restart recognizer");
                        // mSpeechRecognizer.stopListening();
                        StartListening(true);
                    }
                };
            }
            mNoSpeechCountDown.start();
        }
    }

    @Override
    public void call(String result) {
        Toast toast = Toast
                .makeText(this, String.format("terdengar:\n \"%s\"",
result),
```

```
                        Toast.LENGTH_LONG);
        // toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    }
}
```

Kelas SplashActivity.java

```
package com.video.facetube;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ProgressBar;

public class SplashActivity extends Activity {

    private ProgressBar progressBar1;
    private boolean mIsBackButtonPressed;
    private Handler handler = new Handler();
    private int progressStatus = 0;

    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash_layout);
        progressBar1 = (ProgressBar)
findViewById(R.id.progressBar1);
        progressBar1.setProgress(0);
        new Thread(new Runnable() {
            public void run() {
                while (progressStatus < 100) {
                    progressStatus += 1;
                    handler.post(new Runnable() {
                        public void run() {

    progressBar1.setProgress(progressStatus);
                        }
                    });
                    try {
                        Thread.sleep(25);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                finish();
                if (!mIsBackButtonPressed) {
                    Intent intent = new Intent(SplashActivity.this,
                            MainActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    SplashActivity.this.startActivity(intent);
                }
            }
```

```
        }).start();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Kelas UploadService.java

```
package com.video.facetube;

import java.io.BufferedInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import android.app.IntentService;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.AsyncTask;
import android.provider.OpenableColumns;
import android.util.Log;
import com.google.api.client.extensions.android.http.AndroidHttp;
import
com.google.api.client.googleapis.extensions.android.gms.auth.GoogleAc
countCredential;
import
com.google.api.client.googleapis.extensions.android.gms.auth.GoogleAu
thIOException;
import
com.google.api.client.googleapis.extensions.android.gms.auth.GooglePl
ayServicesAvailabilityIOException;
import
com.google.api.client.googleapis.extensions.android.gms.auth.UserReco
verableAuthIOException;
import com.google.api.client.googleapis.media.MediaHttpUploader;
import
com.google.api.client.googleapis.media.MediaHttpUploaderProgressListe
ner;
```

```
import com.google.api.client.http.HttpTransport;
import com.google.api.client.http.InputStreamContent;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.gson.GsonFactory;
import com.google.api.services.youtube.YouTube;
import com.google.api.services.youtube.model.Video;
import com.google.api.services.youtube.model.VideoSnippet;
import com.google.api.services.youtube.model.VideoStatus;
import com.google.common.collect.Lists;

/**
 * @author Ibrahim Ulukaya <ulukaya@google.com>
 *         <p/>
 *         Intent service to handle uploads.
 */
public class UploadService extends IntentService {

    /**
     * defines how long we'll wait for a video to finish processing
     */
    private static final int PROCESSING_TIMEOUT_SEC = 60 * 20; // 20
minutes

    /**
     * controls how often to poll for video processing status
     */
    private static final int PROCESSING_POLL_INTERVAL_SEC = 60;
    /**
     * how long to wait before re-trying the upload
     */
    private static final int UPLOAD_REATTEMPT_DELAY_SEC = 60;
    /**
     * max number of retry attempts
     */
    private static final int MAX_RETRY = 3;
    private static final String TAG = "UploadService";
    /**
     * processing start time
     */
    private static long mStartTime;
    final HttpTransport transport =
AndroidHttp.newCompatibleTransport();
    final JsonFactory jsonFactory = new GsonFactory();
    GoogleAccountCredential credential;
    /**
     * tracks the number of upload attempts
     */
    private int mUploadAttemptCount;

    public UploadService() {
        super("YTUploadService");
    }

    private static void zzz(int duration) throws InterruptedException
{
        Log.d(TAG, String.format("Sleeping for [%d] ms ...",
duration));
```

```
            Thread.sleep(duration);
            Log.d(TAG, String.format("Sleeping for [%d] ms ... done",
duration));
        }

        private static boolean timeoutExpired(long startTime, int
timeoutSeconds) {
            long currTime = System.currentTimeMillis();
            long elapsed = currTime - startTime;
            if (elapsed >= timeoutSeconds * 1000) {
                return true;
            } else {
                return false;
            }
        }
        private String filePath;

        @Override
        protected void onHandleIntent(Intent intent) {
            Uri fileUri = intent.getData();
            String chosenAccountName = intent
                    .getStringExtra(VideoUploadActivity.ACCOUNT_KEY);
            filePath = intent.getStringExtra("Path");

            credential = GoogleAccountCredential.usingOAuth2(
                    getApplicationContext(),
Lists.newArrayList(Auth.SCOPES));
            credential.setSelectedAccountName(chosenAccountName);

            // credential.setBackOff(new ExponentialBackOff());

            String appName =
getResources().getString(R.string.app_name);
            final YouTube youtube = new YouTube.Builder(transport,
jsonFactory,
                    credential).setApplicationName(appName).build();

            try {
                // uploadVideo(fileUri, youtube);
                // uploadMyVideo(fileUri, youtube);
                tryUploadAndShowSelectableNotification(fileUri,
youtube);
                // } catch (InterruptedException e) {
                // ignore
            } catch (Exception e) {

            }
        }

        private String uploadMyVideo(Uri videoUri, YouTube youtube) {
            String VIDEO_FILE_FORMAT = "video/*";
            // File videoFile = null;// getFileFromUri(videoUri);

            long fileSize = 0;
            InputStream fileInputStream = null;

            try {
```

```
            fileSize =
getContentResolver().openFileDescriptor(videoUri, "r")
                    .getStatSize();
            fileInputStream =
getContentResolver().openInputStream(videoUri);
        } catch (Exception x) {
        }

        // Log.d(TAG, "You chose " + videoFile + " to upload.");
        Video videoObjectDefiningMetadata = new Video();
        VideoStatus status = new VideoStatus();
        status.setPrivacyStatus("unlisted");
        videoObjectDefiningMetadata.setStatus(status);
        VideoSnippet snippet = new VideoSnippet();
        Calendar cal = Calendar.getInstance();
        snippet.setTitle("Test upload video " + cal.getTime());
        snippet.setDescription("Video uploaded api V3 on " +
cal.getTime());

        List<String> tags = new ArrayList<String>();
        tags.add("Test");
        tags.add("video");
        snippet.setTags(tags);
        videoObjectDefiningMetadata.setSnippet(snippet);
        InputStreamContent mediaContent;
        try {
            mediaContent = new InputStreamContent(VIDEO_FILE_FORMAT,
                    new BufferedInputStream(fileInputStream));// new

    // FileInputStream(videoFile)));
            mediaContent.setLength(fileSize);// videoFile.length());
            Log.d(TAG, "videoFile.length()=" + fileSize);//
videoFile.length());
            YouTube.Videos.Insert videoInsert =
youtube.videos().insert(
            "snippet,statistics,status",
videoObjectDefiningMetadata,
                    mediaContent);

            // YouTube.Activities.Insert activitinsert =
            // youtube.activities().insert(arg0, arg1)

            MediaHttpUploader uploader =
videoInsert.getMediaHttpUploader();
            uploader.setDirectUploadEnabled(false);
            uploader.setChunkSize(1024 * 1024);
            MediaHttpUploaderProgressListener progressListener = new
MediaHttpUploaderProgressListener() {
                public void progressChanged(MediaHttpUploader
uploader)
                        throws IOException {
                    switch (uploader.getUploadState()) {
                    case INITIATION_STARTED:
                        Log.d(TAG, "Initiation Started");
                        break;
                    case INITIATION_COMPLETE:
                        Log.d(TAG, "Initiation Completed");
```

```
                        break;
                    case MEDIA_IN_PROGRESS:
                        Log.d(TAG, "Upload in progress");
                        Log.d(TAG,
                                "Upload percentage: " +
uploader.getProgress());
                        // publishProgress((int)
(uploader.getProgress()
                        // 100)); break; case MEDIA_COMPLETE:
Log.d(TAG,
                        // "Upload Completed!");
                        break;
                    case NOT_STARTED:
                        Log.d(TAG, "Upload Not Started!");
                        break;
                    }
                }
            };
            uploader.setProgressListener(progressListener);

            // Execute upload.
            Video returnedVideo = videoInsert.execute();
            return returnedVideo.getId();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }

    private void tryUploadAndShowSelectableNotification(final Uri
fileUri,
            final YouTube youtube) throws InterruptedException {
        while (true) {
            Log.i(TAG,
                    String.format("Uploading [%s] to YouTube",
                            fileUri.toString()));
            String videoId = tryUpload(fileUri, youtube);
            if (videoId != null) {
                Log.i(TAG, String.format("Uploaded video with ID:
%s", videoId));
                tryShowSelectableNotification(videoId, youtube);
                return;
            } else {
                Log.e(TAG, String.format("Failed to upload %s",
                        fileUri.toString()));
                if (mUploadAttemptCount++ < MAX_RETRY) {
                    Log.i(TAG,
                            String.format(
                                    "Will retry to upload the
video ([%d] out of [%d] reattempts)",
                            mUploadAttemptCount, MAX_RETRY));
                    zzz(UPLOAD_REATTEMPT_DELAY_SEC * 100);
```

```
                } else {
                    Log.e(TAG,
                            String.format(
                                    "Giving up on trying to upload
%s after %d attempts",
                                    fileUri.toString(),
mUploadAttemptCount));
                    return;
                }
            }
        }
    }

    private void tryShowSelectableNotification(final String videoId,
            final YouTube youtube) throws InterruptedException {
        mStartTime = System.currentTimeMillis();
        boolean processed = false;
        while (!processed) {
            processed = ResumableUpload.checkIfProcessed(videoId,
youtube);
            if (!processed) {
                // wait a while
                Log.d(TAG,
                        String.format(
                                "Video [%s] is not processed yet,
will retry after [%d] seconds",
                                videoId,
PROCESSING_POLL_INTERVAL_SEC));
                if (!timeoutExpired(mStartTime,
PROCESSING_TIMEOUT_SEC)) {
                    zzz(PROCESSING_POLL_INTERVAL_SEC * 1000);
                } else {
                    Log.d(TAG,
                            String.format(
                                    "Bailing out polling for
processing status after [%d] seconds",
                                    PROCESSING_TIMEOUT_SEC));
                    return;
                }
            } else {
                ResumableUpload.showSelectableNotification(videoId,
                        getApplicationContext());
                return;
            }
        }
    }

    private String tryUpload(Uri mFileUri, YouTube youtube) {
        long fileSize;
        InputStream fileInputStream = null;
        String videoId = null;
        try {
            fileSize =
getContentResolver().openFileDescriptor(mFileUri, "r")
                    .getStatSize();
            fileInputStream =
getContentResolver().openInputStream(mFileUri);
```

```
            // String[] proj = { MediaStore.Images.Media.DATA };
            // Cursor cursor = getContentResolver().query(mFileUri,
proj, null,
            // null, null);
            // int column_index = cursor
            // .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
            // cursor.moveToFirst();

            // File f = new File(mFileUri.getPath());
            // String absolutePath = f.getAbsolutePath();
            // String filePath = absolutePath.substring(0,
            // absolutePath.lastIndexOf(File.separator));

            videoId = ResumableUpload.upload(youtube,
fileInputStream,
                    fileSize, mFileUri, filePath,//
                    // cursor.getString(column_index),
                    getApplicationContext());

        } catch (Exception e) {
            Log.e(getApplicationContext().toString(),
e.getMessage());
        } finally {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                // ignore
            }
        }
        return videoId;
    }

    private class VideoUploadAsyncTask extends
            AsyncTask<YouTube.Videos.Insert, Integer, String> {

        @Override
        protected String doInBackground(YouTube.Videos.Insert...
inserts) {
            try {
                YouTube.Videos.Insert videoInsert = inserts[0];

                MediaHttpUploader uploader =
videoInsert.getMediaHttpUploader();

                MediaHttpUploaderProgressListener progressListener =
new MediaHttpUploaderProgressListener() {

                    @Override
                    public void progressChanged(MediaHttpUploader
uploader)
                            throws IOException {

                        switch (uploader.getUploadState()) {
                        case INITIATION_STARTED:
                            Log.d(TAG, "# INITIATION_STARTED ");
                            break;
```

```
                            case INITIATION_COMPLETE:
                                Log.d(TAG, "# INITIATION_COMPLETE ");
                                break;

                            case MEDIA_IN_PROGRESS:
                                int progress = (int)
Math.round(uploader
                                        .getProgress() * 100);
                                Log.d(TAG, "# MEDIA_IN_PROGRESS :
progress = "
                                        + progress + "%");

                                publishProgress(progress);
                                break;

                            case MEDIA_COMPLETE:
                                Log.d(TAG, "# MEDIA_COMPLETE ");
                                publishProgress(100);
                                break;

                            case NOT_STARTED:
                                Log.d(TAG, "# NOT_STARTED ");
                                break;

                            default:
                                break;
                            }
                        }
                    };

                    uploader.setProgressListener(progressListener);

                    Video returnedVideo = videoInsert.execute();

                    return returnedVideo.getId();

                } catch (GooglePlayServicesAvailabilityIOException
gpsaioe) {

                } catch (UserRecoverableAuthIOException uraioe) {

                } catch (GoogleAuthIOException gaioe) {

                } catch (IOException ioe) {

                }
                return null;
            }
        }

        private void uploadVideo(Uri mFile, YouTube youtube) {
            YouTube.Videos.Insert videoInsert = prepareUpload(mFile,
youtube);

            new
VideoUploadAsyncTask().executeOnExecutor(AsyncTask.SERIAL_EXECUTOR,
                    videoInsert);
```

```
    }

    public YouTube.Videos.Insert prepareUpload(Uri videoUri, YouTube
youtube) {

        try {
            long fileSize = 0;
            InputStream fileInputStream = null;
            String fileName = "";

            if (videoUri.getScheme().equals("content")) {
                Cursor cursor = getContentResolver().query(videoUri,
null,
                        null, null, null);
                try {
                    if (cursor != null && cursor.moveToFirst()) {
                        fileName = cursor.getString(cursor

    .getColumnIndex(OpenableColumns.DISPLAY_NAME));
                    }
                } finally {
                    cursor.close();
                }
            }
            if (fileName == null) {
                fileName = videoUri.getPath();
                int cut = fileName.lastIndexOf('/');
                if (cut != -1) {
                    fileName = fileName.substring(cut + 1);
                }
            }

            try {
                fileSize =
getContentResolver().openFileDescriptor(videoUri,
                        "r").getStatSize();
                fileInputStream = getContentResolver()
                        .openInputStream(videoUri);
            } catch (Exception x) {
            }

            Video videoObjectDefiningMetadata = new Video();

            VideoStatus status = new VideoStatus();
            status.setPrivacyStatus("public");
            videoObjectDefiningMetadata.setStatus(status);

            VideoSnippet snippet = new VideoSnippet();
            snippet.setTitle(fileName);
            snippet.setDescription("Uploaded via FaceTube");

            List<String> tags = new ArrayList<String>();
            tags.add("FaceTube");
            snippet.setTags(tags);

            videoObjectDefiningMetadata.setSnippet(snippet);
```

```
            InputStreamContent mediaContent = new
InputStreamContent(

    URLConnection.guessContentTypeFromName(fileName),
                    new BufferedInputStream(fileInputStream));
            mediaContent.setLength(fileSize);

            YouTube.Videos.Insert videoInsert = youtube.videos()
                    .insert("snippet,status",
videoObjectDefiningMetadata,
                            mediaContent);

            MediaHttpUploader uploader =
videoInsert.getMediaHttpUploader();

            uploader.setDirectUploadEnabled(false);

    uploader.setChunkSize(MediaHttpUploader.MINIMUM_CHUNK_SIZE);

            return videoInsert;
        } catch (FileNotFoundException e) {
            Log.e(TAG, e.getMessage());
            return null;
        } catch (IOException ex) {
            Log.e(TAG, ex.getMessage());
            return null;
        }
    }
}
```

Kelas UserFragment.java

```
package com.video.facetube;
/* Copyright (c) 2013 Google Inc.
 * Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You
may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0 Unless required by
applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License
for the specific language governing permissions and limitations under
the License. */
import android.app.Activity;
import android.app.Fragment;
import android.content.Context;
import android.content.Intent;
import android.content.IntentSender.SendIntentException;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
```

```
import
com.google.android.gms.common.api.GoogleApiClient.ConnectionCallbacks
;
import
com.google.android.gms.common.api.GoogleApiClient.OnConnectionFailedL
istener;
import com.google.android.gms.plus.Plus;
import com.google.android.gms.plus.model.people.Person;
/**
 * @author Ibrahim Ulukaya <ulukaya@google.com>
 *         <p/>
 *         Left side fragment showing user's uploaded YouTube videos.
 */
public class UserFragment extends Fragment implements
ConnectionCallbacks,
        OnConnectionFailedListener {

    private static final String TAG = UserFragment.class.getName();
    private static Context mContext;
    private Callbacks mCallbacks;
    private GoogleApiClient mGoogleApiClient;
    private boolean mResolvingError = false;
    private static final int REQUEST_RESOLVE_ERROR = 1;

    // private ImageLoader mImageLoader;

    public UserFragment() {
    }

    public UserFragment(Context context) {
        mContext = context;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            mGoogleApiClient = new GoogleApiClient.Builder(mContext)
                    .addConnectionCallbacks(this)
                    .addOnConnectionFailedListener(this)
                    .addApi(Plus.API,
Plus.PlusOptions.builder().build())
                    // .addApi(LocationServices.API)
                    .addScope(Plus.SCOPE_PLUS_PROFILE).build();

            mResolvingError = savedInstanceState != null
                    &&
savedInstanceState.getBoolean(STATE_RESOLVING_ERROR,
                            false);
        } catch (Exception x) {

        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
```

```
            Bundle savedInstanceState) {
        View listView = inflater.inflate(R.layout.user_fragment,
container,
                false);
        // TextView emptyView = (TextView) listView
        // .findViewById(android.R.id.empty);
        return listView;
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        setProfileInfo();
    }

    public void setProfileInfo() {
        setProfileInfo(null);
    }

    @SuppressWarnings({ "deprecation" })
    public void setProfileInfo(String accountName) {
        if (accountName != null && accountName.length() > 0) {
            ((TextView) getView().findViewById(R.id.display_name))
                    .setText(accountName);
            return;
        }
        if (!mGoogleApiClient.isConnected()) {
            ((TextView) getView().findViewById(R.id.display_name))
                    .setText("not signed in");
        } else {
            Person currentPerson = Plus.PeopleApi
                    .getCurrentPerson(mGoogleApiClient);
            ((TextView) getView().findViewById(R.id.display_name))
                    .setText(currentPerson.getDisplayName());//
Plus.AccountApi.getAccountName(mGoogleApiClient));
        }
    }

    @Override
    public void onStart() {
        super.onStart();
        if (!mResolvingError) { // more about this later
            mGoogleApiClient.connect();
        }
    }

    @Override
    public void onStop() {
        mGoogleApiClient.disconnect();
        super.onStop();
    }

    @Override
    public void onPause() {
        super.onPause();
        mGoogleApiClient.disconnect();
    }
```

```
    @SuppressWarnings("deprecation")
    @Override
    public void onConnected(Bundle bundle) {
        setProfileInfo();
        mCallbacks

    .onConnected(Plus.AccountApi.getAccountName(mGoogleApiClient));
    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (requestCode == REQUEST_RESOLVE_ERROR) {
            mResolvingError = false;
            if (resultCode == Activity.RESULT_OK) {
                // Make sure the app is not already connected or
attempting to
                // connect
                if (!mGoogleApiClient.isConnecting()
                        && !mGoogleApiClient.isConnected()) {
                    mGoogleApiClient.connect();
                }
            }
        }
    }

    private static final String STATE_RESOLVING_ERROR =
"resolving_error";

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putBoolean(STATE_RESOLVING_ERROR, mResolvingError);
    }

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult)
{
        if (mResolvingError) {
            // Already attempting to resolve an error.
            return;
        } else if (connectionResult.hasResolution()) {
            try {
                mResolvingError = true;

    connectionResult.startResolutionForResult(getActivity(),
                        REQUEST_RESOLVE_ERROR);
            } catch (SendIntentException e) {
                // There was an error with the resolution intent.
Try again.
                Log.e(TAG, e.toString(), e);
```

```
                mGoogleApiClient.connect();
            }
        } else {
            // Show dialog using
GoogleApiAvailability.getErrorDialog()
            // showErrorDialog(connectionResult.getErrorCode());
            Log.i(TAG,
Integer.toString(connectionResult.getErrorCode()));
            mResolvingError = true;
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        if (!(activity instanceof Callbacks)) {
            throw new ClassCastException("Activity must implement
callbacks.");
        }

        mCallbacks = (Callbacks) activity;
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mCallbacks = null;
    }

    public interface Callbacks {
        public void onConnected(String connectedAccountName);
    }
}
```

Kelas VideoCaptureActivity.java

```
package com.video.facetube;

import java.io.File;
import java.io.IOException;
import java.lang.reflect.Method;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.hardware.Camera.PictureCallback;
```

```
import android.hardware.Camera.ShutterCallback;
//import android.hardware.Camera.Size;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.media.CamcorderProfile;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.StrictMode;
import android.provider.MediaStore;
//import android.provider.MediaStore.Video;
import android.speech.RecognizerIntent;
import android.util.Log;
import android.view.Display;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.LinearLayout;

public class VideoCaptureActivity extends Activity implements
OnClickListener,
        LocationListener, MediaRecorder.OnInfoListener,
        MediaRecorder.OnErrorListener {
    private Camera mCamera;
    private CameraPreview mPreview;
    private MediaRecorder mediaRecorder;
    // private Button switchCamera;
    private Context myContext;
    private LinearLayout cameraPreview;
    private boolean cameraFront = false;
    private static final String TAG = "VoiceRecognition";
    public static boolean recording = false;
    // private static final int VOICE_RECOGNITION_REQUEST_CODE =
1234;
    private Handler mHandler;
    // private ImageView img_record, img_stop;
    SurfaceView surfaceView;
    SurfaceHolder surfaceHolder;
    PictureCallback rawCallback;
    ShutterCallback shutterCallback;
    PictureCallback jpegCallback;
    Button btn_record;
    // private long time, last_time;// ,time_x;
    // private int index_srt;
    Calendar c;
    SimpleDateFormat ds;
    String file_name;
    // String SRT = "";
```

```
    // **
    private LocationManager lm;
    // private double latitude, longitude;

    // private String LOCATION_ADDRESS;
    private int screenWidth = 640;
    private int screenHeight = 480;
    private boolean isFrontCamera = false;
    private boolean ready;
    private String LAST_TIME = "", THIS_TIME = "";

    @SuppressLint("NewApi")
    @SuppressWarnings("deprecation")
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent callerIntent = getIntent();

        Bundle extras;
        if (callerIntent != null && (extras =
callerIntent.getExtras()) != null
                && extras.containsKey("isFrontCamera")) {
            isFrontCamera = extras.getBoolean("isFrontCamera");
        }

        myContext = this;
        mediaRecorder = new MediaRecorder();
        if (android.os.Build.VERSION.SDK_INT > 9) {
            StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder()
                    .permitAll().build();
            StrictMode.setThreadPolicy(policy);
        }
        initialize();
        Display display = getWindowManager().getDefaultDisplay();

        screenWidth = display.getWidth();
        screenHeight = display.getHeight();

        lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        Criteria criteria = new Criteria();
        criteria.setPowerRequirement(Criteria.POWER_HIGH);
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setSpeedRequired(true);
        criteria.setAltitudeRequired(false);
        criteria.setBearingRequired(false);
        criteria.setCostAllowed(false);
        lm.getBestProvider(criteria, true);
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
1000, 0.0f,
                this);

        if (!hasCamera(myContext)) {
            finish();
        }
```

```
        if (mCamera == null) {
            mCamera = Camera.open(isFrontCamera ?
findFrontFacingCamera()
                    : findBackFacingCamera());
            mPreview.refreshCamera(mCamera);
        }

        if (!recording) {
            SET_RECORD(1);
        }
    }

    private int findFrontFacingCamera() {
        int cameraId = -1;
        int defaultCamera = -1;
        int numberOfCameras = Camera.getNumberOfCameras();
        for (int i = 0; i < numberOfCameras; i++) {
            CameraInfo info = new CameraInfo();
            Camera.getCameraInfo(i, info);
            defaultCamera = i;
            if (info.facing == CameraInfo.CAMERA_FACING_FRONT) {
                cameraId = i;
                cameraFront = true;
                break;
            }
        }
        return cameraId < 0 ? defaultCamera : cameraId;
    }

    private int findBackFacingCamera() {
        int cameraId = -1;
        int defaultCamera = -1;
        int numberOfCameras = Camera.getNumberOfCameras();
        for (int i = 0; i < numberOfCameras; i++) {
            CameraInfo info = new CameraInfo();
            Camera.getCameraInfo(i, info);
            defaultCamera = i;
            if (info.facing == CameraInfo.CAMERA_FACING_BACK) {
                cameraId = i;
                cameraFront = false;
                break;
            }
        }
        return cameraId < 0 ? defaultCamera : cameraId;
    }

    private void SET_RECORD(int x) {
    }
    public void initialize() {
        setContentView(R.layout.video_layout);

    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_
ON);
        cameraPreview = (LinearLayout)
findViewById(R.id.camera_preview);
        mPreview = new CameraPreview(myContext, mCamera);
```

```
        mPreview.setOnSurfaceCreatedEventListener(new
OnSurfaceCreatedEventListener() {
            @Override
            public void OnSurfaceCreated(SurfaceHolder holder) {
                if (!recording) {
                    Surface s = holder.getSurface();
                    record_start(holder);
                    mediaRecorder.setPreviewDisplay(s);

                    boolean isSuccessInit = false;
                    try {
                        mediaRecorder.prepare();
                        isSuccessInit = true;
                    } catch (IllegalStateException e) {
                        Log.e("Prepare error 1:", e.toString());
                        releaseMediaRecorder();
                    } catch (IOException e) {
                        Log.e("Prepare error 2:", e.toString());
                        releaseMediaRecorder();
                    } catch (Exception e) {
                        Log.e("Prepare error 3:", e.toString());
                        releaseMediaRecorder();
                    }

                    if (!isSuccessInit) {
                        finish();
                    } else {
                        runOnUiThread(new Runnable() {
                            public void run() {
                                try {
                                    mediaRecorder.start();
                                } catch (final Exception ex) {

                                }
                                recording = true;
                            }
                        });
                    }
                }
            }
        });

        cameraPreview.addView(mPreview);
        mPreview.SET_TEXT("FaceTube 1.0");

        SET_RECORD(0);

        btn_record = (Button) this.findViewById(R.id.btn_record);

        mHandler = new Handler();
        btn_record.setOnClickListener(this);
        refreshVoiceSettings();
    }

    private void refreshVoiceSettings() {
        Log.i(TAG, "Sending broadcast");
```

```
    VideoCaptureActivity.this.sendOrderedBroadcast(RecognizerIntent
                .getVoiceDetailsIntent(VideoCaptureActivity.this),
null,
                new SupportedLanguageBroadcastReceiver(), null,
                Activity.RESULT_OK, null, null);
    }
    private class SupportedLanguageBroadcastReceiver extends
BroadcastReceiver {
        @Override
        public void onReceive(Context context, final Intent intent)
{
            Log.i(TAG, "Receiving broadcast " + intent);
            final Bundle extra = getResultExtras(false);
            if (getResultCode() != Activity.RESULT_OK) {
                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        showToast("Error code:" + getResultCode());
                    }
                });
            }
            if (extra == null) {
                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        showToast("No extra");
                    }
                });
            }

            if
(extra.containsKey(RecognizerIntent.EXTRA_SUPPORTED_LANGUAGES)) {
                mHandler.post(new Runnable() {

                    @Override
                    public void run() {
                        //
updateSupportedLanguages(extra.getStringArrayList(RecognizerIntent.EX
TRA_SUPPORTED_LANGUAGES));
                    }
                });
            }
            if
(extra.containsKey(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE)) {
                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        //
updateLanguagePreference(extra.getString(RecognizerIntent.EXTRA_LANGU
AGE_PREFERENCE));
                    }
                });
            }
        }

        public void showToast(String text) {
```

```
        }
    }

    OnClickListener switchCameraListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            if (!recording) {
                int camerasNumber = Camera.getNumberOfCameras();
                if (camerasNumber > 1) {
                    releaseCamera();
                    chooseCamera();
                } else {
                    // Toast toast = Toast.makeText(myContext, "No
camera!",
                    // Toast.LENGTH_LONG);
                    // toast.show();
                }
            }
        }
    };

    public void chooseCamera() {
        if (cameraFront) {
            int cameraId = findBackFacingCamera();
            if (cameraId >= 0) {
                mCamera = Camera.open(cameraId);
                mCamera.setDisplayOrientation(0);
                mPreview.refreshCamera(mCamera);
            }
        } else {
            int cameraId = findFrontFacingCamera();
            if (cameraId >= 0) {
                mCamera = Camera.open(cameraId);
                mPreview.refreshCamera(mCamera);
            }
        }
    }

    private void record_start(SurfaceHolder h) {
        c = Calendar.getInstance();
        ds = new SimpleDateFormat("yyyyMMddhhmmss",
Locale.getDefault());
        String DATE_X = ds.format(c.getTime());
        String path = String.format("%s/FaceTube/", Environment
                .getExternalStorageDirectory().getAbsolutePath());
        File dir = new File(path);
        if (!dir.exists()) {
            dir.mkdirs();
        }

        file_name = String.format("%svideo_%s", path, DATE_X);

        record_name = String.format("%s.mp4", file_name);

        // Camera.Size s = mPreview.getSize(screenWidth,
screenHeight);
        CamcorderProfile profile = CamcorderProfile
```

```
                .get(CamcorderProfile.QUALITY_HIGH);

        Camera.Parameters parameters = mCamera.getParameters();
        parameters.setPreviewSize(profile.videoFrameWidth,
                profile.videoFrameHeight);

        mCamera.setParameters(parameters);

        try {
            mCamera.setPreviewDisplay(h);
            mCamera.startPreview();
        } catch (IOException e) {
            Log.d(TAG, e.getMessage());
        }

        mCamera.lock();
        mCamera.unlock();

        mediaRecorder.setOnErrorListener(this);
        mediaRecorder.setOnInfoListener(this);

        mediaRecorder.setCamera(mCamera);

    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);

    mediaRecorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);

    mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);

    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT)
;

    mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_S
P);
        mediaRecorder.setOutputFile(record_name);
        mediaRecorder.setMaxDuration(5000);
        mediaRecorder.setVideoSize(profile.videoFrameWidth,
                profile.videoFrameHeight);
    }

    public void onInfo(MediaRecorder arg0, int arg1, int arg2) {
        Log.i("info", "info");
        if (arg1 ==
MediaRecorder.MEDIA_RECORDER_INFO_MAX_DURATION_REACHED) {
            record_stop(false);
        }
    }

    public void onError(MediaRecorder mr, int what, int extra) {
        switch (what) {
        case MediaRecorder.MEDIA_RECORDER_ERROR_UNKNOWN:
        case MediaRecorder.MEDIA_ERROR_SERVER_DIED:
            try {
                mediaRecorder.stop(); // stop
            } catch (IllegalStateException x) {
                Log.e("stop", x.getMessage());
            } catch (Exception x) {
```

```
                Log.e("stop", x.getMessage());
            }
            releaseMediaRecorder(); // release
            break;
        default:
            break;
        }
    }

    private void record_stop(boolean cancel) {
        SET_RECORD(0);
        if (recording) {
            try {
                if (mediaRecorder != null) {
                    mediaRecorder.stop(); // stop
                }
            } catch (IllegalStateException x) {
                Log.e("stop", x.getMessage());
            } catch (Exception x) {
                Log.e("stop", x.getMessage());
            }
            releaseMediaRecorder(); // release
            // Toast.makeText(VideoCaptureActivity.this, "Rekam
selesai!",
            // Toast.LENGTH_LONG).show();
            recording = false;

            SpeechRecognitionService.startContinuousListening(this,
                    isFrontCamera);

            Intent intent = new Intent(getApplicationContext(),
                    MainActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

            File file = new File(record_name);
            if (file.exists()) {
                if (cancel) {
                    file.delete();
                    intent.putExtra("EXIT", true);
                } else {
                    // UPLOAD_VIDEO();
                    ContentValues content = new ContentValues(1);
                    content.put(MediaStore.Video.Media.DATA,
record_name);
                    ContentResolver resolver = getContentResolver();
                    Uri uri = resolver.insert(

    MediaStore.Video.Media.EXTERNAL_CONTENT_URI,
                            content);

                    intent.setData(uri);
                    intent.putExtra("EXIT", false);
                    intent.putExtra("UPLOAD", true);
                }
            } else {
                intent.putExtra("EXIT", true);
            }
```

```
            startActivity(intent);

            finish();
        }
    }

    @Override
    public void onBackPressed() {
    }

    @Override
    protected void onPause() {
        super.onPause();
        releaseCamera();
    }
    private boolean hasCamera(Context context) {
        if (context.getPackageManager().hasSystemFeature(
                PackageManager.FEATURE_CAMERA)) {
            return true;
        } else {
            return false;
        }
    }

    private void releaseMediaRecorder() {
        if (mediaRecorder != null) {
            mediaRecorder.reset(); //
            mediaRecorder.release(); //
            mediaRecorder = null;
            mCamera.lock(); // lock camera for later use
        }
    }

    String record_name = "";

    protected void setDisplayOrientation(Camera camera, int angle) {
        Method downPolymorphic;
        try {
            downPolymorphic = camera.getClass().getMethod(
                    "setDisplayOrientation", new Class[] { int.class
});
            if (downPolymorphic != null)
                downPolymorphic.invoke(camera, new Object[] { angle
});
        } catch (Exception e1) {
            Log.e("Error Camera!,", e1.toString());
        }
    }

    private void releaseCamera() {
        if (mCamera != null) {
            mCamera.release();
            mCamera = null;
        }
    }

    @Override
```

```
    public void onClick(View v) {
        if (v.getId() == R.id.btn_record && recording) {
            record_stop(true);
        }
    }

    @Override
    public void onLocationChanged(Location location) {
        if (location.getAccuracy() < 100) {
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        try {
            if (mediaRecorder != null) {
                mediaRecorder.release();
            }
        } catch (Exception ex) {
            Log.e(TAG, ex.getMessage());
        }
    }

    @Override
    public void onProviderDisabled(String arg0) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onProviderEnabled(String arg0) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
        // TODO Auto-generated method stub
    }
}
```

Kelas VideoData.java

```
package com.video.facetube;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import com.google.api.services.youtube.model.Video;
import com.google.api.services.youtube.model.VideoSnippet;

/**
 * @author Ibrahim Ulukaya <ulukaya@google.com>
 *         <p/>
 *         Helper class to handle YouTube videos.
 */
```

```
public class VideoData {
    private Video mVideo;

    public Video getVideo() {
        return mVideo;
    }

    public void setVideo(Video video) {
        mVideo = video;
    }

    public String getYouTubeId() {
        return mVideo.getId();
    }

    public String getTitle() {
        return mVideo.getSnippet().getTitle();
    }

    public VideoSnippet addTags(Collection<? extends String> tags) {
        VideoSnippet mSnippet = mVideo.getSnippet();
        List<String> mTags = mSnippet.getTags();
        if (mTags == null) {
            mTags = new ArrayList<String>(2);
        }
        mTags.addAll(tags);
        return mSnippet;
    }

    public String getThumbUri() {
        return
mVideo.getSnippet().getThumbnails().getDefault().getUrl();
    }

    public String getWatchUri() {
        return "http://www.youtube.com/watch?v=" + getYouTubeId();
    }
}
```

Kelas VideoUploadActivity.java

```
package com.video.facetube;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.widget.Toast;

public class VideoUploadActivity extends Activity {
    public static final String ACCOUNT_KEY = "accountName";
    public static final String MESSAGE_KEY = "message";
    public static final String YOUTUBE_ID = "youtubeId";
```

```
    public static final String YOUTUBE_WATCH_URL_PREFIX =
"http://www.youtube.com/watch?v=";
    static final String REQUEST_AUTHORIZATION_INTENT =
"com.google.example.yt.RequestAuth";
    static final String REQUEST_AUTHORIZATION_INTENT_PARAM =
"com.google.example.yt.RequestAuth.param";

    private String mChosenAccountName;
    private Uri mFileUri;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        mFileUri = intent.getData();
        loadAccount();
    }

    private void loadAccount() {
        SharedPreferences sp = PreferenceManager
                .getDefaultSharedPreferences(this);
        mChosenAccountName = sp.getString(ACCOUNT_KEY, null);
    }

    public void uploadVideo(View view) {
        if (mChosenAccountName == null) {
            return;
        }
        // if a video is picked or recorded.
        if (mFileUri != null) {
            Intent uploadIntent = new Intent(this,
UploadService.class);
            uploadIntent.setData(mFileUri);
            uploadIntent.putExtra(ACCOUNT_KEY, mChosenAccountName);
            startService(uploadIntent);
            Toast.makeText(this, "youtube upload started",
Toast.LENGTH_LONG)
                    .show();
            // Go back to MainActivity after upload
            // finish();
        }
    }
}
```

Kelas FaceTubeMatching.java

```
package com.video.facetube.matching;

import java.util.ArrayList;

public class FaceTubeMatching {

    public interface MatchingCallback {
        void call(String result);
    }
```

```
    public MatchingCallback callback;

    public FaceTubeMatching(MatchingCallback matchingCallback) {
        this.callback = matchingCallback;
    }

    private int PecentageOfTextMatch(String s0, String s1) {
        int percentage = 0;
        s0 = s0.trim().replaceAll("\\s+", " ");
        s1 = s1.trim().replaceAll("\\s+", " ");
        LevenshteinDistance ld = new LevenshteinDistance(s1);
        percentage = (int) (100 - ((float) ld.Run(s0) * 100 /
(float) (Math
                .max(s0.length(), s1.length()))));
        return percentage;
    }

    public boolean Run(ArrayList<String> results) {
        String string = "";
        int maxMatchIdx = 0;
        int maxMatchPctg = 0;

        for (int i = 0; i < results.size(); i++) {
            string = results.get(i);
            int l1 = PecentageOfTextMatch(string.substring(1),
"ekam");
            if (l1 > maxMatchPctg) {
                maxMatchPctg = l1;
                maxMatchIdx = i;
            }
            if (maxMatchPctg == 100) {
                break;
            }
        }

        string = results.get(maxMatchIdx);

        this.callback.call(string);

        Soundex sndx = new Soundex("250");

        // matching string more than 90%
        return maxMatchPctg > 90 && sndx.IsMatch(string);
    }
}
```

Kelas LevenshteinDistance.java

```
package com.video.facetube.matching;

public class LevenshteinDistance {

    private String destinationWord;

    public LevenshteinDistance(String destination) {
        this.destinationWord = destination;
    }

    public int Run(String s0) {
        int len0 = s0.length() + 1;
        int len1 = destinationWord.length() + 1;

        // array jarak
        int[] cost = new int[len0];
        int[] newcost = new int[len0];

        // nilai awal pada s0
        for (int i = 0; i < len0; i++) {
            cost[i] = i;
        }

        // komputasi array jarak
        // nilai transformasi untuk setiap huruf dalam s1
        for (int j = 1; j < len1; j++) {
            // nilai awal pada s1
            newcost[0] = j - 1;

            // nilai transformasi untuk setiap huruf dalam s0
            for (int i = 1; i < len0; i++) {
                // pencocokan huruf pada kedua string
                int match = (s0.charAt(i - 1) ==
destinationWord.charAt(j - 1)) ? 0
                        : 1;

                // nilai komputasi untuk setiap transformasi
                int cost_replace = cost[i - 1] + match;
                int cost_insert = cost[i] + 1;
                int cost_delete = newcost[i - 1] + 1;

                // menjaga nilai minimum
                newcost[i] = Math.min(Math.min(cost_insert,
cost_delete),
                        cost_replace);
            }

            // menukar nilai/array nilai baru
            int[] swap = cost;
            cost = newcost;
            newcost = swap;
        }
        // jarak adalah nilai untuk trasnformasi semua huruf pada
kedua string
        return cost[len0 - 1];
```

```
    }

    public boolean IsMatch(String s1) {
        return this.Run(s1) == 0;
    }
}
```

Kelas Soundex.java

```
package com.video.facetube.matching;

import java.util.Locale;

public class Soundex {

    private String expectedSoundexResult;

    /*
     * Implements mapping from: AEHIOUWYBFPVCGJKQSXZDTLMNR to:
     * 00000000111122222222334556
     */
    public static final char[] MAP = {
            // A B C D E F G H I J K L M
            '0', '1', '2', '3', '0', '1', '2', '0', '0', '2', '2',
'4', '5',
            // N O P W R S T U V W X Y Z
            '5', '0', '1', '2', '6', '2', '3', '0', '1', '0', '2',
'0', '2' };

    public Soundex() {

    }

    public Soundex(String expectedSoundexRes) {
        this.expectedSoundexResult = expectedSoundexRes;
    }

    public String Run(String s) {
        // Algorithm works on upper case (mainframe era).
        String t = s.toUpperCase(Locale.getDefault());

        StringBuffer res = new StringBuffer();
        char c, prev = '?';

        // Main loop: find up to 4 chars that map.
        for (int i = 0; i < t.length() && res.length() < 4
                && (c = t.charAt(i)) != ','; i++) {

            // Check character = alphabetic.
            // Already converted to upper case. Algorithm
            // only handles ASCII letters, do NOT use
Character.isLetter()!
            // Also, skip double letters.
            if (c >= 'A' && c <= 'Z' && c != prev) {
                prev = c;
                // First char is installed unchanged, for sorting.
```

```
                if (i == 0)
                    res.append(c);
                else {
                    char m = MAP[c - 'A'];
                    if (m != '0')
                        res.append(m);
                }
            }
        }

        if (res.length() == 0)
            return "";

        for (int i = res.length(); i < 4; i++)
            res.append('0');

        return res.toString();
    }

    public boolean IsMatch(String s) {
        return
this.Run(s).substring(1).equalsIgnoreCase(expectedSoundexResult);
    }
}
```

main_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/gray"
    android:orientation="horizontal" >

    <LinearLayout
        android:id="@+id/ln"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#000000"
        android:baselineAligned="false"
        android:orientation="horizontal" >

        <LinearLayout
            android:id="@+id/buttonsLayout"
            android:layout_width="0dp"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="4.5"
            android:background="@drawable/blue"
            android:orientation="vertical" >

            <com.facebook.login.widget.LoginButton
                android:id="@+id/login_button"
                android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="30dp"
                android:layout_marginBottom="30dp" />

            <fragment
                android:id="@+id/user_fragment"
                android:name="com.video.facetube.UserFragment"
                android:layout_width="match_parent"
                android:layout_height="0dp"
                android:layout_weight="1"
                tools:layout="@layout/user_fragment" />

            <ToggleButton
                android:id="@+id/btn_UseFrontCamera"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textOn="@string/textBackCamera"
                android:textOff="@string/textFrontCamera"/>

            <ToggleButton
                android:id="@+id/btn_StopSpeechRecogSvc"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textOn="@string/button_StopSpeechRecogSvc"

android:textOff="@string/button_StartSpeechRecogSvc"/>

            <Button
                android:id="@+id/button_StartRecording"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="@string/button_StartRecording"/>
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

splash_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/black1"
    android:gravity="center_horizontal"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/imageView5"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:layout_marginTop="5dp"
```

```
        android:layout_weight="2"
        android:contentDescription="@string/icon_VideoDesc"
        android:scaleType="fitCenter"
        android:src="@drawable/videoicon" />

    <ProgressBar
        android:id="@+id/progressBar2"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp" />

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="center_horizontal"
        android:alwaysDrawnWithCache="true"
        android:animationCache="true"
        android:background="@drawable/gray"
        android:gravity="center_horizontal|center_vertical"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="2dp"
            android:text="@string/text_Loading"

android:textAppearance="?android:attr/textAppearanceMedium"
            android:textColor="#ffffff"
            android:textSize="13sp" />

        <ProgressBar
            android:id="@+id/progressBar1"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="5dp"
            android:progress="100" />
    </LinearLayout>

</LinearLayout>
```

user_fragment.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:gravity="center_vertical"
     android:orientation="horizontal"
     android:padding="16dp">
```

```
    <TextView
        android:id="@+id/display_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:textAppearance="?android:textAppearanceLarge" />
</LinearLayout>
```

video_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/gray"
    android:orientation="horizontal" >

    <LinearLayout
        android:id="@+id/ln"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#000000"
        android:baselineAligned="false"
        android:orientation="vertical" >

        <LinearLayout
            android:id="@+id/camera_preview"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:background="#000000"
            android:orientation="vertical" >
        </LinearLayout>

        <LinearLayout
            android:id="@+id/buttonsLayout"
            android:layout_width="fill_parent"
            android:layout_height="51dp"
            android:layout_gravity="center"
            android:background="@drawable/blue"
            android:orientation="vertical" >

            <Button
                android:id="@+id/btn_record"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/button_Record"
                android:textSize="12sp" />
        </LinearLayout>
    </LinearLayout>

</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.video.facetube"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="24" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.USE_CREDENTIALS" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS"
/>
    <uses-permission android:name="android.permission.RECORD_VIDEO"
/>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO"
/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.STORAGE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-feature
        android:name="android.hardware.camera.front"
        android:required="false" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/movieicon"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:name="com.video.facetube.MyApplication">
        <activity
            android:name="com.video.facetube.SplashActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
android:name="com.video.facetube.VideoCaptureActivity"
            android:screenOrientation="portrait">
        </activity>
        <activity android:name="com.video.facetube.MainActivity" >
        </activity>
```

```
        <activity android:name="com.facebook.FacebookActivity"
          android:configChanges=

"keyboard|keyboardHidden|screenLayout|screenSize|orientation"
          android:theme="@android:style/Theme.Translucent.NoTitleBar"
          android:label="@string/app_name" />
        <service
            android:name=".FetchAddressIntentService"
            android:exported="false"/>
        <service
android:name="com.video.facetube.SpeechRecognitionService"
         android:label="SpeechRecognitionService" >
         </service>
         <service
            android:name="com.video.facetube.UploadService"
            android:label="UploadService" />
        <meta-data android:name="com.google.android.gms.version"
         android:value="@integer/google_play_services_version" />
        <meta-data android:name="com.facebook.sdk.ApplicationId"
            android:value="@string/facebook_app_id"/>
        <uses-library android:name="android.test.runner" />
    </application>
    <instrumentation
android:name="android.test.InstrumentationTestRunner"
         android:targetPackage="com.video.facetube"
         android:label="SpeechRecognitionServiceTest" />
</manifest>
```

*[Halaman ini sengaja dikosongkan]*

# LAMPIRAN B KUESIONER PENGUJIAN KEGUNAAN

## Kuesioner Pengujian Kegunaan Aplikasi Facetube

**Data Responden**

Nama : ……..…….

Tipe Ponsel : ……

Versi Android : ……

**A. Antarmuka Pengguna**

1. Apakah aplikasi ini memiliki antarmuka pengguna yang menarik?
   a. Tidak b. Kurang c. Cukup d.Sangat
2. Apakah antarmuka aplikasi ini jelas dan mudah dipahami?
   a. Tidak b. Kurang c. Cukup d.Sangat
3. Apakah aplikasi ini sudah memiliki tata letak yang baik?
   a. Tidak b. Kurang c. Cukup d.Sangat

**B. Pendeteksian Perintah Suara**

1. Apakah aplikasi ini sudah dapat mendeteksi perintah suara Anda dengan baik?
   a. Tidak b. Kurang c. Cukup d.Sangat
2. Apakah aplikasi ini memberikan respons yang sesuai dengan yang Anda inginkan?
   a. Tidak b. Kurang c. Cukup d.Sangat

**C. Manfaat**

1. Apakah aplikasi ini dapat memberikan manfaat dalam

perekaman dan publikasi video?
a. Tidak b. Kurang c. Cukup d.Sangat

2. Apakah aplikasi ini sudah memberikan kemudahan dalam hal perekaman video?
a. Tidak b. Kurang c. Cukup d.Sangat

3. Apakah aplikasi ini sudah memberikan kemudahan dalam membantu publikasi video Anda?
a. Tidak b. Kurang c. Cukup d.Sangat

# BAB 6
# KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan yang didapat selama proses pengembangan aplikasi pada Tugas Akhir ini. Selain itu, juga terdapat beberapa saran terkait pengembangan lebih lanjut dari aplikasi.

## 6.1. Kesimpulan

Berdasarkan pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak di dalam Tugas Akhir ini, dapat diambil kesimpulan sebagai berikut:

1. Proses otomatisasi unggah video pada akun Youtube dapat diimplementasikan dengan menggunakan Youtube Data API, dan *post status* berupa *link* pada akun Facebook juga dapat diimplementasikan dengan menggunakan Facebook Graph API, sehingga pemublikasian video lebih hemat waktu dan efisien
2. Fungsionalitas aplikasi dapat berjalan dengan baik. Hal ini dibuktikan dengan hasil pengujian fungsionalitas yang menyatakan bahwa seluruh fungsionalitas berhasil diimplementasikan dengan baik.
3. Aplikasi FaceTube dapat dibangun dengan mengimplementasikan *Google Speech Recognition* untuk mendeteksi perintah suara.
4. Algoritma pencocokan *string Levenshtein Distance* dan *Soundex* dapat diimplementasikan untuk proses pencocokan *string* perintah suara.
5. Dengan menerapkan Youtube API dan Google Maps API, aplikasi dapat memberikan *tag reference* berupa waktu dan lokasi pada video yang diunggah.
6. Penggantian kamera depan atau belakang dapat diimplementasikan untuk perekaman video dalam

aplikasi FaceTube.

7. Aplikasi ini dapat berjalan pada perangkat komunikasi bergerak berbasis Android untuk melakukan perekaman video dan otomatisasi pembagian video.
8. Aplikasi FaceTube yang dibangun pada Tugas Akhir ini dapat sedikit membantu masyarakat dalam perihal perekaman video dan pemublikasiannya. Dengan adanya fitur pemberian tag reference dan perintah suara, video akan semakin bersifat informatif sehingga tujuan awal dibangunnya aplikasi yaitu pemublikasian serta perekaman video pada saat darurat dapat tercapai.

## 6.2. Saran

Dalam pembuatan Tugas Akhir ini, terdapat beberapa saran untuk perbaikan serta pengembangan dari aplikasi yang telah dikerjakan untuk kedepannya, yakni sebagai berikut:

1. Diperlukannya rancangan antarmuka yang lebih baik, dengan menggunakan *material design* yang banyak diimplementasikan pada aplikasi berbasis Android saat ini.
2. Pengembangan fitur aplikasi yang lebih beragam, terutama opsi-opsi dari penggunaan aplikasi sehingga aplikasi lebih dapat memenuhi keinginan pengguna.
3. Pengembangan fitur aplikasi untuk dapat menghadapi kasus dimana tidak tersedianya *paket data* atau koneksi internet (*airplane mode*).
4. Pengembangan penambahan perintah suara serta fitur penggantian bahasa, sehingga aplikasi dapat digunakan dalam banyak perintah bahasa.
5. Pengembangan publikasi video agar tidak hanya dapat dipublikasikan di YouTube dan Facebook saja, akan tetapi media sosial lainnya.

# DAFTAR PUSTAKA


[1] Syaroni, Mokhamad dan Munir, Rinaldi. *Pencocokan String Berdasarkan Kemiripan Ucapan (Phonetic String Matching) Dalam Bahasa Inggris.* Departemen Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Bandung, 2004.

[2] Purnamasari, Tryas Ayu dan Luthfi, Emha Taufiq. *Membangun Aplikasi Pencocokan String Berdasarkan Penulisan dan Kemiripan Pengucapan.* Jurusan Teknik Informatika, STMIK AMIKOM Yogyakarta, 2012.

[3] Creativyst, Inc. Understanding Classic Soundex Algorithms [Online]. Available: http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm. [Diakses pada Agustus 2014].

[4] *Soundex Coding Rules* [Online]. Available: http://www.genealogyintime.com/GenealogyResources/Articles/what_is_soundex_and_how_does_soundex_work_page2.html. [Diakses pada September 2014].

[5] *Vladimir I. Levensthein biography* [Online]. Available: http://www.ieeeghn.org/wiki/index.php/Vladimir_I._Levensthein. [Diakses pada September 2014].

[6] Pinzon, Yoan. *Algorithms for Approximate String Matching.* Universidad Nacional de Colombia, 2006.

[7] *API (Application Programming Interface)* [Online]. Available: http://botoykoma.blogspot.co.id/2013/01/api-aplikasi-programing-interface.html. [Diakses pada September 2014].

[8] Hopkins, Jim. "Surprise! There's a third YouTube co-founder" [Online]. Available: http://usatoday30.usatoday.com/tech/news/2006-10-11-youtube-karim_x.htm. [Diakses pada November 2014].

[9] Weber, Tim. "BBC strikes Google-YouTube deal" [Online]. Available:

http://news.bbc.co.uk/2/hi/business/6411017.stm. [Diakses pada November 2014].

[10] *The Wall Street Journal* (Dow Jones). "Facebook Tops Billion-User Mark" [Online]. Available: http://www.wsj.com/articles/SB10000872396390443635404578036164027386112. [Diakses pada November 2014].

[11] Sengupta, Somini. "Facebook's Prospects May Rest on Trove of Data" [Online]. Available: http://www.nytimes.com/2012/05/15/technology/facebook-needs-to-turn-data-trove-into-investor-gold.html?_r=0:. [Diakses pada November 2014].

[12] Carlson, Nicholas. "At Last – The Full Story Of How Facebook Was Founded" [Online]. Available: http://www.businessinsider.com/how-facebook-was-founded-2010-3?IR=T&r=US&IR=T#we-can-talk-about-that-after-i-get-all-the-basic-functionality-up-tomorrow-night-1. [Diakses pada November 2014].

[13] "Using OAuth 2.0 to Access Google APIs" [Online]. Available: https://developers.google.com/identity/protocols/OAuth2. [Diakses pada Februari 2015].

[14] "Google APIs Client Libraries" [Online]. Available: https://developers.google.com/discovery/libraries. [Diakses pada Februari 2015].

[15] "YouTube Data API Client Library for Java" [Online]. Available: https://developers.google.com/api-client-library/java/apis/youtube/v3. [Diakses pada Februari 2015].

[16] "Getting Started Android SDK" [Online]. Available: https://developers.facebook.com/docs/android/getting-started/. [Diakses pada Februari 2015].

[17] "Using the Google API Client Library for Java on Android" [Online]. Available: https://developers.google.com/api-client-

library/java/google-api-java-client/android. [Diakses pada Februari 2015].
[18] "Setting Up Google Play Services" [Online]. Available: https://developers.google.com/android/guides/setup. [Diakses pada Februari 2015].

*[Halaman ini sengaja dikosongkan]*

# BIODATA PENULIS

Penulis bernama lengkap Erfan Ahmadiyono. Lahir di Surabaya, pada tanggal 13 Agustus 1991. Penulis menyelesaikan pendidikan dasar di SD Muhammadiyah 4 Pucang Surabaya. Kemudian pendidikan menengah pertama, penulis tempuh di SMP Negeri 6 Surabaya. Selanjutnya penulis melanjutkan pendidikan menengah atas di SMA Negeri 9 Surabaya. Pada tingkat perguruan tinggi, penulis mengenyam pendidikan sarjana di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Rekayasa Perangkat Lunak (Software Engineering).

Penulis dapat dihubungi melalui e-mail di alamat sebagai berikut: ahmadiyono.erfan@gmail.com.