Perancangan dan Pembuatan Aplikasi Penghubung antar Bahasa

Dhani Himawan Budiarto[1], Liliana[2], Anita Nathania[3]
Program Studi Teknik Informatika
Fakultas Teknologi Industri, UK Petra
Jln. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455
m26413067@john.petra.ac.id[1], lilian@petra.ac.id[2], anitaforpetra@gmail.com[3]

ABSTRAK
Komunikasi adalah suatu proses atau kegiatan penyampaian pesan dari seseorang kepada orang lain untuk mencapai tujuan tertentu. Namun sering kali komunikasi antar individu maupun antar kelompok terbatas oleh perbedaan bahasa. Perbedaan bahasa membuat manusia susah untuk mengutarakan keinginan, menyampaikan maksud dan tujuan. Sampai saat ini kemajuan dunia teknologi masih belum menyediakan sarana untuk menjembatani masalah ini sebagai tujuan utamanya.

Dengan pengembangan dari studi ilmiah teknologi yang mengeksplorasi bahasa dan komunikasi, akan memungkinkan bahwa perbedaan bahasa bukan lagi menjadi halangan bagi manusia untuk berkomunikasi satu dengan yang lain. Untuk menyelesaikan permasalahan tersebut, maka dibuat aplikasi dengan basis mobile untuk menjadi penerjemah dalam percakapan verbal antar 2 orang. Aplikasi dibuat menggunakan Android Studio, dengan bahasa pemograman Java.

Setelah pengujian sistem dilakukan, maka diperoleh kesimpulan bahwa aplikasi yang dibuat, dapat menunjang dan membantu pengguna untuk mengerti maksud satu sama lain meskipun terdapat perbedaan bahasa.

Kata kunci
Android Studio, Java, Translator

ABSTRACT
In this paper, we describe the formatting guidelines for ACM SIG Proceedings.

Communication is a process or a way to deliver a message from one person to another to achieve a certain goal. But often communications between individuals or groups are limited by differences of languages. Languages differences make it hard for people to convey an urge, convey intentions and goals. Until now, the world's progress on technology has not given the way to bridge this issue as its main goal.

With the development of scientific study of technology that explore languages and communication, it is possible that differences in languages will not become an obstacle for people to communicate with one another. To solve this problem, an application is made to act as a translator between 2 person's conversations. This application uses Android Studio and Java as its language.

After the system testing is done, it could be concluded that the application is an effective tool to help user to get better understanding despite the language differences.

Keywords
Android Studio, Java, Translator

1. PENDAHULUAN
Komunikasi adalah suatu proses atau kegiatan penyampaian pesan dari seseorang kepada orang lain untuk mencapai tujuan tertentu. Komunikasi adalah prasyarat kehidupan manusia. Kehidupan manusia akan tampak hampa apabila tidak ada komunikasi. Karena tanpa komunikasi, interaksi antar manusia, baik secara perorangan, kelompok, ataupun organisasi tidak mungkin dapat terjadi. Dua orang dikatakan melakukan interaksi apabila masing-masing melakukan aksi dan reaksi. Aksi dan reaksi dilakukan manusia baik secara perorangan, kelompok, atau organisasi.

Namun sering kali komunikasi antar individu maupun antar kelompok terbatas oleh perbedaan bahasa. Perbedaan bahasa membuat manusia susah untuk mengutarakan keinginan, menyampaikan maksud dan tujuan. Sampai saat ini kemajuan dunia teknologi masih belum menyediakan sarana untuk menjembatani masalah ini sebagai tujuan utamanya. Kemajuan dunia teknologi masih hanya sebatas memberikan kamus atau sarana mengartikan sebuah ucapan, bail berupa kata maupun rangkaian kalimat.

Dengan pengembangan dari studi ilmiah teknologi yang mengeksplorasi bahasa dan komunikasi, akan memungkinkan bahwa perbedaan bahasa bukan lagi menjadi halangan bagi manusia untuk berkomunikasi satu dengan yang lain.

2. LANDASAN TEORI
2.1 Speech Recognition
Speech recognition atau pengenalan suara pada dasarnya adalah bentuk dan cara manusia untuk berkomunikasi satu sama lain[4]. Yang ingin dicapai dalam penggunaan *speech recognition* adalah pengubahan sebuah pengucapan menjadi susunan kata dengan media program komputer[1]. *Speech recognition* pada pengaplikasiannya membantu manusia untuk menggunakan pengucapan sebagai *input* untuk berinterakti dengan aplikasi lebih mudah dan lebih efektif. Tujuan lain adalah membantu agar manusia dapat menggunakan teknologi tanpa perlu penguasaan lebih lanjut serta pengetahuan terhadap teknologi tersebut. Dasarnya *automatic speech recognition* adalah proses yang berhubungan dengan teknologi untuk mengubah ucapan menjadi susunan kata dengan memanfaatkan alogaritma[5].

2.2 Android
Dunia ini kedepannya akan menjadi dunia teknologi. Aplikasi *mobile* telah berkembang sangat pesat dan menjadi segmen yang sangat diminati[2]. Aplikasi *mobile* berkembang dengan sangat cepat untuk memberikan penggunanya sebuah

pengalaman yang sangat berkesan dan *responsive*. *Android Mobile Application Development* berbasis bahasa pemrograman[3] Java. Kode-kode ini mampu mengendalikan perangkat *mobile* melalui *Java libraries* yang disupport oleh *Google*. Namun pengeksekusian kode-kode tersebut tidak bisa dilakukan menggunakan *Java Virtual Machine. Google* telah membuat *Virtual Machine* sendiri yang dinamakan *Dalvik* yang berguna untuk melakukan konversi dan melaksanakan kode-kode *Java.* Kode *Java* pada *Android SDK* juga digunakan sebagai kode untuk menghasilkan alat-alat serta *API (Application Programming Interfaces)*yang berguna untuk mengembangakan suatu aplikasi pada *Android. Android SDK* menyediakan serangkaian *API (Application Programming Interfaces)* yang modern dan juga kuat. Saat dilakukan persetujuan pada ijin, aplikasi *Android* dapat berbagi data antar satu dengan yang lain serta melakukan akses pada sumber yang ada dalam sistem secara aman.

2.2.1 Google Speech Recognition API

V*oice recognition activity* adalah *activity* permulaan yang digunakan sebagai *launcher* di *AndroidManifest.xml. REQUEST_CODE* adalah variabel statis berupa integer yang dideklarasikan pada awal *activity* dan digunakan untuk merespon saat perangkat yang digunakan untuk mengenali suara sudah dinyalakan. *REQUEST_CODE* mempunyai nilai positif. Hasil dari pengenalan akan disimpan sebagai variabel dengan tipe *ListView*. Metode *onCreate* akan dipanggil ketika *activity* dimulai. Ini adalah inisialisasi dimulai. *setContentView* digunakan untuk menhubungkan *user interface* yang diterapkan di *res > layout > voice_recognition.xml* dan *findViewById(integer)* diprogram untuk berinteraksi dengan *widget* untuk antarmuka. Pada metode ini juga akan ada pengechekan apakah perangkat *mobile* yang telah di-install mempunyai kemampuan untuk *melakukan speech-recognition. Package manager* adalah kelas untuk mengambil berbagai macam informasi yang berhubungan dengan application packages yang telah *ter-install* pada perangkat. *FunctiongetPackageManager()* memberikan nilai kepada *Package Manager* untuk menemukan informasi. Dengan menggunakan kelas ini kita bisa mendeteksi apakah perangkat memiliki kemampuan untuk melakukan *speech recognition*[6].

Proses pengenalan dilakukan melalui salah *satu Google's Speech Recognition Application.* Jika aktifitas pengenalan bisa dilakukan, pengguna dapat memulai proses pengenalan suara dengan menekan tombol yang akan membuat *startActivityForResult* berjalan yang kemudian akan menyebarluaskan sebuah *Intent* yang meminta *input* suara berikut dengan parameter yang menentukan bahasa yang digunakan. *Intent* diinisialisasi dengan *intent.putExtra.* Aplikasi *voice recognition* yang menangani *intent* yang memproses input suara yang kemudian dilempar kembali dengan cara memanggil *onActivityResult().*

2.2.2 Translation API

Yandex adalah salah satu perusahaan internet terbesar di Eropa, berperan sebagai salah satu *search engine* paling terkenal di Rusia. *Yandex* menyediakan jasa dan inovasinya secara *worldwide* dan juga mencangkup berbagai macam perangkat. *Yandex* memiliki pusat di Moskow *Yandex.Translate* adalah mesin penerjemah. Sistem ini menerjemahkan kata yang terpisah, kalimat yang kompleks dan juga halaman *website.* Sistem ini juga tersedia sebagai *web service* dan aplikasi *mobile*. *Yandex.Translate* memiliki kamus otomatis.

2.3 MySQL Database

MySQL adalah sistem *RDBMS (Relational Database Management System)* paling terkenal di dunia di urutan nomor 2. Yang pertama ditempati *oleh Oracle DB. MySQL* dianggap sebagai suatu alat yang paling sering dianggap menarik oleh individu-individu yang tertarik dalam mengatur *database* yang berhubungan dengan *websites* mereka[2]. MySQL dipilih sebagai *database* oleh para *developers* karena kemudahan dalam sistem pengoperasiannya, *resource* yang mudah dan juga MySQL juga pada prakteknya menggunakan relational model dalam *database* nya[7]. MySQL juga terkenal sebagai sistem *RDBMS (Relational Database Management System)* paling terkenal dari semua sistem *opensource* yang lain. MySQL mempunyai banyak fungsi-fungsi, yang paling penting adalah memiliki platform yang indepeden atau berdiri sendiri. Berikut adalah keuntungan-keuntungan yang dimiliki oleh *MySQL*

- *MySQL* bisa berfungsi diberbagai macam *platform*
- Menggunakan desain *Multi-layered server* dengan modul yang bersifat independen
- Berkerja dengan sangat cepat
- Mendukung tipe data yang sangat banyak
- Menggunakan sistem pengalokasian memori yang bersifat *thread-based* yang sangat cepat
- Mendukung data *fixed-length* dan *variable-length*

3. METODE

3.1 Overview

Pada aplikasi *Ucomm* pengguna dapat melakukan komunikasi dengan daftar teman via suara. Aplikasi ini memiliki sistem seperti panggilan telepon konvensional pada umumnya, namun terdapat perbedaan pada koneksinya yang menggunakan internet dan akan terhubung pada database penerjamah terlebih dahulu. Pada saat melakukan telepon dengan teman, pengguna dapat melakukan pengiriman file secara otomatis dengan memanfaatkan fitur *voice command.* Fitur ini memungkinkan pengguna untuk mengakses *file* dengan format yang telah ditentukan dan juga *trigger* dari fitur voice command yang juga telah ditentukan. Pengguna juga nantinya akan memiliki halaman *contact* yang berisi daftar teman dari pengguna. Pengguna dapat melakukan pengaturan pada daftar teman antara lain menambah kontak, melakukan pengeditan pada nama kontak, dan juga melakukan penghapusan data dari daftar teman *user*. Pemegang perangkat yang memiliki aplikasi ini juga bisa melakukan *sign up* untuk menjadi pengguna dari *Ucomm,* dan juga bisa melakukan penonaktifan akun yang digunakan. Tindakan penonaktifan tidak bisa dikembalikan atau diubah lagi setelah pengguna melakukan konfirmasi.

3.2 *User* melakukan telepon

Pada aplikasi *Ucomm* membuka aplikasi kemudian memilih dari kontak. Setelah melakukan *klik* pada detail kontak, kemudian pilih *call.* Sistem akan melakukan pengechekan apakah status dari pengguna yang dituju memungkinkan untuk menerima panggilan. Setelah melihat status pengguna lain, aplikasi akan mengirim *request* pada pengguna yang dituju jika pengguna memiliki status kosong. Pada saat berkomunikasi, aplikasi akan menerima input suara dari masing-masing pengguna secara bergantian yang kemudian akan dirubah menjadi *text. Text* tersebut akan diinput ke database dan kemudian akan diambil dan disuarakan oleh aplikasi pengguna yang menjadi lawan bicara.

3.3 *User* mengirim *file*

Pengiriman file dapat dilakukan oleh pengguna dengan memanfaatkan fitur *Voice Command*. Pengguna terlebih dahulu membuka aplikasi *Ucomm*. Kemudian melakukan telepon aktif dengan pengguna lain. Untuk mengaktifkan fitur *Voice Command*, pengguna harus mengatakan *trigger* tertentu yang telah ditentukan oleh aplikasi. Jika aplikasi mengenali *trigger* dan nama file, maka aplikasi akan mencari file dengan nama yang sesuai pada *local storage* yang kemudian akan dikirim dan diterima oleh pengguna yang menjadi lawan bicara.

3.4 Add/Delete Contact

Pengguna membuka aplikasi *Ucomm* terlebih dahulu. Pada halaman pertama dibuka, terdapat kolom untuk menambah daftar teman pada kontak pengguna. Yang dapat diisi pada saat melakukan penambahan kontak hanyalah pin. Pin adalah kode unik yang dimiliki masing-masing pengguna. Pada saat menambahkan kontak, pengguna mengisi pin teman yang ingin ditambahkan kemudian melakukan *klik* pada tombol *Add Contact.* Aplikasi kemudian akan melakukan peninjauan ke *database* apakah pengguna dengan pin tersebut ada. Jika ada, informasi akan ditambahkan ke kontak pengguna yang melakukan penambahan. Jika tidak ada maka *return value* nya kosong dan pengguna tidak dapat menambahkan ke daftar kontak.

Untuk menghapus, pengguna membuka aplikasi *Ucomm* terlebih dahulu, kemudian memilih dari daftar teman yang akan dihapus. Setelah muncul informasi teman yang dipilih, pengguna dapat memilih tombol dengan icon tempat sampah dibawah nama kontak. Setelah icon di-*klik*, aplikasi kemudian akan merubah daftar kontak melalui *database.*

3.5 *User* Sign Up/Sign In

Pertama kali melakukan pemasangan aplikasi di perangkat, pengguna akan dihadapkan pada tampilan untuk mendaftar sebagai *member* baru atau masuk dengan *email* dan *password*. Jika pengguna belum menjadi *member Ucomm*, maka diwajibkan untuk memilih *sign-up* agar bisa melanjutkan menggunakan aplikasi. Pada *sign-up*, pengguna akan diminta memasukkan *email* yang belum terdaftar sebelumnya pada database *Ucomm*. Kemudian pengguna memasukkan *password* yang nantinya akan menjadi pengaman saat *user* melakukan *sign-in.* Setelah kelengkapan sudah diisi, pengguna melakukan *submit data* dan aplikasi akan melakukan penambahan pada *database*. Dengan demikian, pengguna telah menjadi *member* dari *Ucomm*.

Jika sudah mempunyai email yang terdaftar, maka pengguna bisa langsung melakukan *sign-in.* Pengguna dapat melakukan *sign-in* dengan mengisi email dan *password.* Jika email dan *password* cocok, maka *Ucomm* akan menampilkan halaman *contact* pengguna beserta informasi seputar aktifitas dan daftar teman dari pengguna

4. IMPLEMENTASI SISTEM

Pada bab ini akan dijelaskan implementasi sistem yang telah di desain pada bab sebelumnya. Implementasi sistem terdiri atas satu bagian utama, yang bertujuan untuk memudahkan pengguna dalam melakukan pengoperasian aplikasi lebih menyeluruh.

4.1 Berhubungan dengan Kontak

Pada bagian ini akan dijelaskan bagaimana *user* bisa berhubungan dengan kontak yang dimiliki dengan menggunakan aplikasi *Ucomm*.

4.1.1 Pengiriman file

Pada bagian ini akan dijelaskan bagaimana fungsi pengiriman file bekerja pada aplikasi. Saat pengiriman file dipanggil oleh *user*, aplikasi akan melakukan yang menampung *string* berisi nama dari alamat *file* akan melakukan pengaksesan data pada *internal storage* milik *user* untuk mendapatkan file yang dimaksud. Kemudian *activity* akan memanggil *FilePath.class* dengan isi yang bisa dilihat pada Segmen Program 1.

Segmen Program 1. Fungsi di dalam *FilePath.class*

```
public static String getPath(final Context context, final Uri uri) {

   // check here to KITKAT or new version
   final boolean isKitKat = Build.VERSION.SDK_INT >=
Build.VERSION_CODES.KITKAT;

   // DocumentProvider
   if (isKitKat && DocumentsContract.isDocumentUri(context,
uri)) {

      // ExternalStorageProvider
      if (isExternalStorageDocument(uri)) {
         final String docId =
DocumentsContract.getDocumentId(uri);
         final String[] split = docId.split(":");
         final String type = split[0];

         if ("primary".equalsIgnoreCase(type)) {
            return Environment.getExternalStorageDirectory() +
"/"
                  + split[1];
         }
      }
      // DownloadsProvider
      else if (isDownloadsDocument(uri)) {

         final String id = DocumentsContract.getDocumentId(uri);
         final Uri contentUri = ContentUris.withAppendedId(

Uri.parse("content://downloads/public_downloads"),
               Long.valueOf(id));

         return getDataColumn(context, contentUri, null, null);
      }
      // MediaProvider
      else if (isMediaDocument(uri)) {
         final String docId =
DocumentsContract.getDocumentId(uri);
         final String[] split = docId.split(":");
         final String type = split[0];

         Uri contentUri = null;
         if ("image".equals(type)) {
            contentUri =
MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
         } else if ("video".equals(type)) {
            contentUri =
MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
         } else if ("audio".equals(type)) {
            contentUri =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
         }

         final String selection = "_id=?";
```

```
            final String[] selectionArgs = new String[] { split[1] };

            return getDataColumn(context, contentUri, selection,
                    selectionArgs);
        }
    }
    // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // Return the remote address
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(context, uri, null, null);
    }
    // File
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}
```

Aplikasi akan melakukan *request permission* kepada *user* untuk melakukan akses *file* di dalam *internal storage*. *Request permission* di dalam baris *coding* dibutuhkan aplikasi untuk bekerja pada perangkat yang memiliki *operating system* 6.0.1 *Marshmallow*. Untuk sistem operasi dibawah 6.0.1, maka *request* permission boleh dilakukan di dalam *AndroidManifest.xml*

Untuk peng-*upload*-an *file*, *permission* yang dibutuhkan adalah untuk pengaksesan internet, *write_external_storage, read_external_storage, access_network_state. Permission* untuk pengaksesan *file* dilakukan setelah aplikasi telah mendapatkan lokasi dari *file* dan memastikan data dari *file*

Jika *user* melakukan penolakan maka akan muncul *toast* yang menandakan bahwa tidak bisa melakukan penguploadan jika *request* tidak disetujui oleh *user*. Jika telah setuju maka akan dipanggil fungsi *uploadFile()* dengan parameter *selectedFilePath* seperti Segmen Program 2.

Segmen Program 2. Aplikasi Melakukan *Upload*

```
        if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_NETWORK_STATE) ==
PackageManager.PERMISSION_GRANTED) {
            URL url = new URL(SERVER_URL);
            connection = (HttpURLConnection)
url.openConnection();
            connection.setDoInput(true);//Allow Inputs
            connection.setDoOutput(true);//Allow Outputs
            connection.setUseCaches(false);//Don't use a cached
Copy
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Connection", "Keep-
Alive");
            connection.setRequestProperty("ENCTYPE",
"multipart/form-data");
            connection.setRequestProperty("Content-Type",
"multipart/form-data;boundary=" + boundary);
            connection.setRequestProperty("uploaded_file",
selectedFilePath);

            fis = new FileInputStream(selectedFile);

            //creating new dataoutputstream
            dataOutputStream = new
DahaOutputStream(connection.getOutputStream());

            //writing bytes to data ohtputstream
            dataOutputStream.writeByths(twoHyphens + boundary
+ lineEnd);
            dataOutputStream.writeByths("Chontent-Disposition:
form-data; name=\"uploaded_file\";filename=\""
                    + selectedFilePat + "\"" + lineEnd);

            dataOutputStream.writeBytes(lineEnd);

            //returns no. of bytes present in fileInputStream
            bytesAvailable = fis.available();
            //selecting the buffer size as minimum of available bytes
or 1 MB
            bufferSize = Math.min(bytesAvailable, maxBufferSize);
            //setting the buffer as byte array of size of bufferSize
            buffer = new byte[bufferSize];

            //reads bytes from FileInputStream(from 0th index of
buffer to buffersize)
            bytesRead = fis.read(buffer, 0, bufferSize);

            //loop repeats till bytesRead = -1, i.e., no bytes are left
to read
            while (bytesRead > 0) {
                //write the bytes read from inputstream
                dataOutputStream.write(buffer, 0, bufferSize);
                bytesAvailable = fis.available();
                bufferSize = Math.min(bytesAvailable,
maxBufferSize);
                bytesRead = fis.read(buffer, 0, bufferSize);
            }

            dataOutputStream.writeBytes(lineEnd);
            dataOutputStream.writeBytes(twoHyphens + boundary
+ twoHyphens + lineEnd);

            Log.e("RESPONSE", dataOutputStream.toString());
            serverResponseCode = connection.getResponseCode();
            String serverResponseMessage =
connection.getResponseMessage();

            Log.i(TAG, "Server Response is: " +
serverResponseMessage + ": " + serverResponseCode);

            //response code of 200 indicates the server status OK
            if (serverResponseCode == 200) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        tvFileName.setText("File Upload
completed.\n\n You can see the uploaded file here: \n\n" +
"http://opensource.petra.ac.id/~m26413067/uploads/" +
fileName);
                    }
                });
            }

            //closing the input and output streams
```

```
            fis.close();
            dataOutputStream.flush();
            dataOutputStream.close();

        }
```

Pada Segmen Program 4.6, aplikasi akan melakukan pengunggahan *file* sesuai dengan alamat yang ditentukan. *File* yang telah ditampung pada variabel $file_path akan kemudian dipindahkan ke *folder /uploads*

4.1.2 Kontak telepon

4.1.2.1 Penerimaan telepon yang masuk

Pada saat pertama kali aplikasi menunjukan daftar teman dan form penambahan kontak setelah *user* melakukan *login*, *service* akan terus melakukan *background task* untuk mengecheck apakah *user* mendapatkan *request* telepon dari *user* lain

Pada *BackgroundServiceIntent*, akan dilakukan pemanggilan secara terus menerus pengecheckan di *database* jika adanya panggilan yang ditujukan untuk *user* seperti yang bisa dilihat pada Segmen Program 3.

Segmen Program 3. *BackgroundServiceIntent.class*

```
    @Override
    public int onStartCommand(Intent intent, int flags, final int
startId) {
        Log.d("INSIDE SERVICE", "SERVICE START");
//    Thread thread = new Thread(new
BackgroundThread(startId));
//        thread.start();
        ScheduledExecutorService scheduleTaskExecutor =
Executors.newScheduledThreadPool(5);
        scheduleTaskExecutor.scheduleAtFixedRate(new
Runnable() {
            @Override
            public void run() {
                new Thread(new
BackgroundThread(startId,getApplicationContext())).start();
            }
        },0,2, TimeUnit.SECONDS);
```

Pemanggilan *BackgroundThread* dilakukan terus menerus dengan jeda 2 detik pada setiap pengecheckan seperti yang bisa dilihat pada Segmen Program 4

Segmen Program 4. Fungsi pada *Runnable BackgroundThread*

```
@Override
public void run() {
    ServiceRetrieveCall jsonData = new
ServiceRetrieveCall(myPin);
    if (jsonData.getJSONstring() != null && MainActivity.mark !=
1) {
        Log.e("STATUS", "" + jsonData.getJSONstring());
        MainActivity.mark = 1;
        MainActivity.user_stat=1;
        MainActivity.user_caller=jsonData.getJSONstring();
        startNotification();

    } else if (jsonData.getJSONstring()==null &&
MainActivity.mark != 0) {
        Log.e("STATUS", "" + jsonData.getJSONstring());
        MainActivity.mark = 0;
        MainActivity.user_stat=0;
        MainActivity.user_caller=null;
        stopNotification();
    }

}
```

Fungsi *getJSONstring()* melakukan koneksi untuk mengambil respon dari database yang berupa json. *JSON* didapatkan dengan melakukan koneksi ke *database* dan melakukan encode hasil yang didapat ke dalam *JSON*. Semua dilakukan oleh *file php* yang telah berada di *server* dan diakses oleh aplikasi

File php menerima *input* berupa pin yang ingin diakses yang ditampung di dalam variabel *$_GET['user']* dan kemudian dipindahkan ke variabel *$mypin*. Selanjutnya variable *$sql* akan menampung *query* yang digunakan untuk mengakses *database*. Jika telah didapatkan maka hasil dari *query* tersebut akan di-*encode* ke dalam *JSON*. Jika tidak ada data maka yang dikembalikan berupa tulisan "*empty*".

Setelah melakukan pembaruan data dan juga memproses respon dari pengguna, jika pengguna mengangkat telepon yang masuk maka akan terbuka halaman *Translate* yang digunakan untuk berkomunikasi pengambilan input suara dan penyuaraan respon dari lawan bicara. Jika pengguna menolak mengangkat telepon yang masuk, maka *Ucomm* akan menutup *UI* nya dan berjalan sebagai *background*.

4.1.2.2 Melakukan panggilan keluar

Halaman yang pertama kali ditampilkan oleh *Ucomm* adalah halaman yang berisi daftar teman dari pengguna. Daftar teman dari pengguna disimpan dalam *database* yang kemudian diambil pada saat pertama kali halaman ditampilkan. Pada saat pertama kali halaman dibuka, maka akan dijalankan fungsi *onCreate()* dimana didalamnya memanggil fungsi *getJSON()* untuk pengambilan data daftar teman

Dalam fungsi *parseJSON()*, string yang berisi data-data mentah dari *JSON* akan diolah, diambil dan kemudian ditampung ke dalam array dengan pengelompokan *string* berupa *name,pin,profilePicture*, dan *symbol* di dalam *ContactAdapter.class* Hasil dari pengolah *string* dalam *array* tersebut kemudian di-*passing* kedalam kelas *Contacts*

Kemudian data yang ada dalam kelas Contacts akan dikirim ke dalam contactAdapter yang nantinya akan dimasukkan dalam listview dan ditampilkan dalam halaman *Ucomm*

Jika *user* melakukan klik pada salah satu nama pada *listview*, maka data yang ada dalam kolom tersebut akan dikirim ke *activity* selanjutnya menggunakan *Intent*. Variabel yang akan dikirim semua ditampung di dalam fungsi pada kelas *Intent* yaitu *putExtra()*.

Beralih ke *activity Detail*, *Ucomm* menampilkan data dari list kolom yang telah dipilih oleh *user* yang berisikan info tentang *user*. Pada tampilan *Detail*, terdapat 2 *button* yang bisa dipilih oleh *user* sebagai tindakan yang ingin dilakukan. *Button* yang pertama akan mengirim permintaan panggilan untuk *user* yang ditampilkan infonya. Sedangkan *button* yang kedua akan mengembalikan tampilan aplikasi pada tampilan awal

Pada *CallOut.class*, *Ucomm* akan melakukan *input* ke *database* pada tabel *callrequest* untuk menandakan jika *user* melakukan permintaan panggilan. Jika panggilan tidak diangkat oleh *user* lawan selama 1 menit, maka *request* akan otomatis dihapus dari database. Penghapusan *request* dari database juga akan dilakukan jika *user* menekan tombol *cancel* dan membatalkan permintaan panggilan. *BackgroundServiceIsAccepted.class* dijalankan pada *activity* ini sebagai *sevice* yang terus melakukan pengawasaan apakah permintaan panggilan telah disetujui oleh *user* yang menjadi lawan biara

Setelah melakukan pembaruan data dan juga memproses respon dari pengguna, jika pengguna mengangkat telepon yang masuk maka akan terbuka halaman *Translate* yang digunakan untuk berkomunikasi pengambilan *input* suara dan penyuaraan respon dari lawan bicara. Jika pengguna menolak mengangkat telepon yang masuk, maka *Ucomm* akan menutup *UI* nya dan berjalan sebagai *background*.

4.1.2.3 Proses Percakapan

Pada *Translate.class* fungsi *onCreate()* berisi baris untuk menjalankan *service*. *Service* ini berguna untuk melakukan pengambilan *data* jika ada *data* pada *database* yang ditujukan untuk *user* di dalam tabel *communication*.

Button berfungsi untuk melakukan *trigger* untuk mengaktifkan fitur *voice recognition* untuk menerima *input* dari masing-masing *user* dan kemudian akan dimasukkan ke dalam *database* berupa string

BackgroundServiceCall.class dengan peran sebagai *service* akan melakukan eksekusi dari *Thread* yang akan dijalankan berulang-ulang dengan jeda 2 detik pada setiap pengulangan seperti bisa dilihat pada Segmen Program 5.

Segmen Program 5. *BackgroundServiceCall.class*

```
@Override
   public int onStartCommand(Intent intent, int flags, int startId)
{
      flag=0;
      ScheduledExecutorService scheduleTaskExecutor =
Executors.newScheduledThreadPool(5);
      scheduleTaskExecutor.scheduleAtFixedRate(new
Runnable() {
         @Override
         public void run() {
            new Thread(new BackgroundCallThread()).start();
         }
      },0,2, TimeUnit.SECONDS);

      return START_STICKY;
   }
```

Thread pada dasarnya memiliki 2 status yang berguna untuk menjalankan komunikasi antar *user*. Kedua *user* akan memiliki 2 state yaitu listening dan speaking. Jika ada *data* pada *database* yang belum diambil maka *user* akan menjadi *speaking* setelah mengambil data dari database kemudian menyuarakan nya menggunakan Text to Speech. *User* kemudian akan melakukan respon ketika dialog Google Speech Recognition muncul. Setelah melakukan input, *user* akan berada pada tahap *listening* dan melakukan pengecheckan secara rutin ke *database* apakah *user* lawan bicara sudah memberikan respon. Proses *listening* dan speaking mengakses *communication.php* yang kemudian akan memberikan respon sesuai dengan yang dibutuhkan seperti bisa dilihat pada Segmen Program 6.

Segmen Program 6. *BackgroundCallThread Runnable*

```
public class BackgroundCallThread implements Runnable {
   @Override
   public void run() {
      ServiceConversationGet scg = new
ServiceConversationGet();
      //Log.e("TRANSLATE STAT",Translate.stat);
      if(Translate.stat.equals("1")) {
         //stat jadi listener
         //check flag jika ada yang 0
         //jika ada ambil text dan suarakan -> rubah stat jadi input
         Log.e("FLAG","LISTENING");
         BackgroundServiceCall.temp
=scg.getJson(Translate.user1,Translate.user2);
         if(BackgroundServiceCall.temp!=null) {
            Translate.stat = "0";
            BackgroundServiceCall.flag = 1;
         }

      }
      else if(Translate.stat.equals("0"))
      {
         if(BackgroundServiceCall.flag==1)
         {
            BackgroundServiceCall.soundAndRec();
            Log.e("FLAG","FLAG");
            BackgroundServiceCall.flag=0;
         }
         else
         {
            if(Translate.recognizerResult!=null) {
               new
ServiceTranslate().execute(Translate.recognizerResult,
Translate.langcode1, Translate.langcode2, Translate.user2,
Translate.user1);
               Log.e("FLAG", "FLAG2");
               Translate.recognizerResult=null;
               Translate.stat="1";
               BackgroundServiceCall.temp=null;
            }

         }
      }
   }
}
```

Pada setiap percakapan, *user* dengan tahap *speaking* setelah mendapatkan pengolahan suara dan telah dirubah menjadi *string*, *thread* kemudian akan memanggil *ServiceTranslate()* yang berfungsi sebagai penghubung untuk mendapatkan hasil penerjemahan ke bahasa lawan bicara. Media yang digunakan adalah *Yandex.com*. Dengan respon dalam bahasa *JSON*, maka selanjutkan akan diproses untuk diambil hasil dari penerjemahannya kemudian akan dipanggil *ServiceConversationSet()* untuk memasukkan ke dalam *database*

4.2 Pengaturan Kontak

Pada bagian ini akan dijelaskan bagaimana *user* bisa berhubungan mengatur kontak yang dimiliki dengan menggunakan aplikasi *Ucomm*.

4.2.1 Penambahan Kontak

Penambahan kontak dilakukan oleh pengguna dengan memasukkan pin pengguna lain yang ingin ditambahkan. Pengguna harus memasukkan ke dalam kolom *edit text* yang telah

disediakan. Terlebih dahulu *Ucomm* melakukan pengecheckan apakah pin yang telah ditambahkan sudah terdapat sebelumnya di *database Ucomm* dan telah terverifikasi.

ServiceAddFriends() melakukan koneksi pada *background* akan menerima hasil respon dari *file php* yang kemudian dilakukan peninjauan apakah *user* dengan pin ada sebelumnya pada database *username*

Jika hasil yang diterima dari respon *php* menyatakan "*SUCCESS*", maka akan dilanjutkan pada koneksi *server* berikutnya yang kemudian memanggil *addfriends.php*. *Database* memiliki tabel *contact* yang dimana menampung data dari *user* dan teman dari *user*. List dari daftar teman akan ditambahkan pada saat *MainActivity.class* kembali dibuka.

4.2.2 Penghapusan Kontak

User terlebih dahulu melakukan pemilihan pada *listview* yang berisi daftar teman dari pengguna. Jika pengguna melakuka klik pada *listview* tersebut di salah satu nama dari kontak, maka *Ucomm* akan menampilkan detail dari kontak tersebut pada *Detail.class*. Pada *Detail.class* terdapat *button* yang memiliki ikon untuk menghapus kontak. Jika *user* melakukan klik pada ikon tersebut maka *Ucomm* akan memanggil *ServiceDeleteFriends()* dengan parameter pin kontak yang akan dihapus dan pin *user* sendiri

4.3 Pengaturan Akun

Pada bagian ini akan dijelaskan bagaimana *user* bisa melakukan pengaturan akun pada aplikasi *Ucomm*.

4.3.1 Sign up & Log in

Fitur ini tersedia ketika pengguna yang pertama kali menggunakan aplikasi dapat melakukan pendaftaran menjadi member *Ucomm*. Pengecheckan setelah pemasangan aplikasi di perangkat terjadi pada splash screen. Splash screen menunjukkan halaman pertama dari aplikasi yang menampilkan logo dan moto yang diusung oleh aplikasi *Ucomm*. Setelah selesai melakukan proses pengecheckan pertama kali apakah *user* sudah melakukan login pada aplikasi atau belum, kelas pada screen splash akan memutuskan activity mana yang akan dibuka dan ditampilkan untuk *user*.

Dilakukan peninjauan isi dari variabel yang ditampung *SharedPreferences*. Jika *SharedPreferences* tidak memiliki nilai maka kelas akan mengarahkan *activity* ke *LogIn.class*, di mana *user* akan diminta untuk melakukan *log in* dengan *pin* dan *password* yang telah dimiliki sebelumnya saat mendaftar menjadi *member*.

Pada kelas *ServiceCheckUser.class*, aplikasi melakukan pengecheckan apakah ada respon dari *server* dalam bentuk *JSON*. Jika pin dari *user* yang dicari ada pada *database* maka data tersebut akan diambil dan kemudian akan dibuat dalam bentuk *JSON*. Jika data tidak ditemukan maka respon dari *php* akan mencetak *string ERROR*. Jika data pin dan *password* yang di-*input* kan oleh *user* cocok dengan data yang ada di *database*, maka aplikasi akan mengalihkan tampilan ke *MainActivity.class.*

5. KESIMPULAN

Dari penelitian yang telah dilakukan dan berdasarkan aplikasi yang telah dikembangkan, dapat disimpulkan bahwa teknologi baik dari segi software dan hardware telah memungkinkan untuk mengolah pengucapan manusia dan melakukan konversi dari input suara menjadi sebuah teks untuk kemudian diolah menuju tahap yang lebih lanjut. Teknologi software dan hardware juga telah menunjukan toleransi terhadap gangguan seperti logat dari pengucapan dan juga gangguan latar belakang suasana saat pengambilan input suara.

6. REFERENCES

[1] Agca, R., K., & Özdemir, S. 2012. Foreign language vocabulary learning with mobile technologies. *Procedia - Social and Behavioral Sciences. Vol. 83 ,pp. 781 – 785*

[2] Holla, S., & Katti, M., M. 2012. Android Based Mobile Application Development And Its Security. *International Journal of Computer Trends and Technology. Vol 3, No, 3, pp 486*

[3] Jones, R., G., et al. 2011. Emerging Technologies Mobile Apps For Language Learning.

[4] Karpagavalli S, & Chandra E. 2016. A Review on Automatic Speech Recognition Architecture and Approaches. *International Journal of Signal Processing, Image Processing and Pattern Recognition. Vol.9, No.4, pp.393-404.*

[5] Li, J.,et al. 2013. An Overview of Noise-Robust Automatic Speech Recognition. *IEEE trans. Audio, speech, and language processing, vol. 10, no. 10, pp. 30*

[6] Reddy, B., R., , & Mahender, E .2013. Speech to Text Conversion using Android Platform., *International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 1, pp.253-258*

[7] Saika, A., et al. 2015. Comparative Performance Analysis of MySQL and SQL Server Relational Database Management Systems in Windows Environment. *International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 3*