BAB IV

IMPLEMENTASI DAN EVALUASI SISTEM

4.1 Kebutuhan Sistem

Sebelum mengimplementasikan dan menjalankan Aplikasi Kamus Indonesia – Korea dibutuhkan perangkat keras dan perangkat lunak dengan kondisi tertentu agar aplikasi dapat berjalan dengan baik. Adapun kebutuhan perangkat lunak dan perangkat keras adalah sebagai berikut :

4.1.1 Kebutuhan Perangkat Keras

Perangkat keras adalah komponen fisik peralatan yang membentuk sistem komputer, serta peralatan lain yang mendukung komputer dalam menjalankan tugasnya. Sifat umum dari perangkat keras adalah dapat dilihat dan dipegang bentuk fisiknya. Adapun perangkat keras minimal yang dibutuhkan untuk menjalankan aplikasi ini yaitu :

1. Handphone beroperating sistem Android minimal v2.2 (Froyo)
2. Internal memory 160 MB storage, 384MB RAM
3. Layar TFT capacitive touchscreen , 256K colors
4. Resolusi 240 x 320 pixels, 3.14 inches

4.1.2 Kebutuhan Perangkat Lunak

Perangkat lunak merupakan kebalikan dari perangkat keras dimana mempunyai bentuk fisik yang tidak dapat dipegang. Adapun perangkat lunak yang dibutuhkan yaitu:

1. Sistem operasi menggunakan *Microsoft Vista* (32 atau 64 bit).
2. *Database* untuk pengolahan data menggunakan SQLite.

3. Bahasa pemrograman java menggunakan *Java Development Kid* (JDK) 1.6 dan *Java Runtime Environment* (JRE).
4. *Integrated Development Environment* (IDE) Eclipse 3.4 atau 3.5 untuk pengembangan aplikasi.

4.2 Implementasi Sistem

4.2.1 Intro

Gambar 4.1 Halaman Intro Aplikasi

Didalam aplikasi ini terdiri dari menu yang dapat digunakan oleh pengguna dan penjelasan untuk halaman intro ini adalah sebagai halaman pembuka ke menu utama.

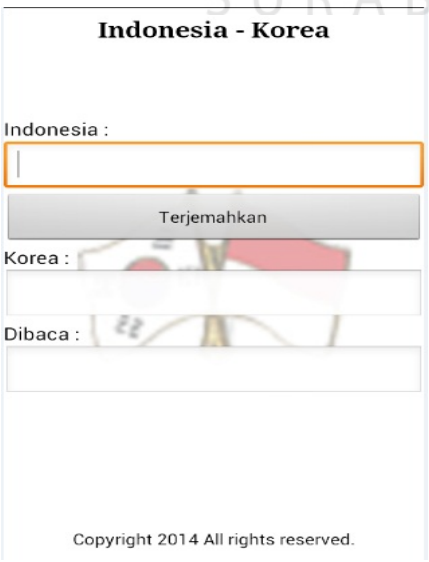
4.2.2 Halaman Utama Website Admin Login

Gambar 4.2 Halaman Menu Utama

Halaman menu utama ini menampilkan berbagai menu pada aplikasi kamus Korea-Indonesia yang dapat digunakan oleh pengguna.

4.2.3 Halaman Menu Indonesia – Korea

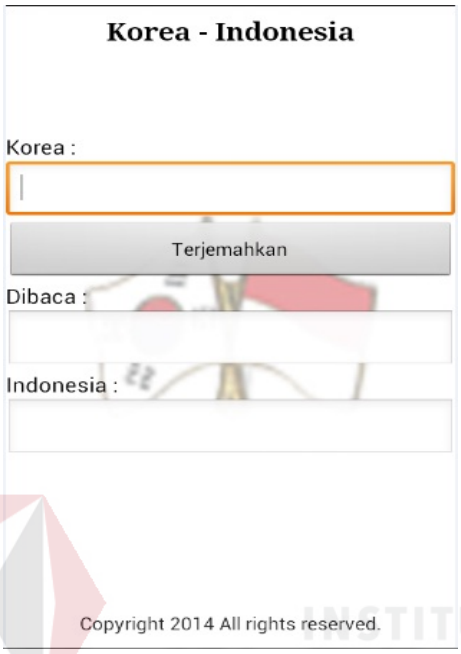
Halaman menu ini digunakan untuk mencari kata dari indonesia ke korea. Disini tersedia kolom indonesia dan tombol terjemahkan.



Gambar 4.3 Halaman Menu Pencarian Indonesia – Korea

4.2.4 Halaman Menu Korea – Indonesia

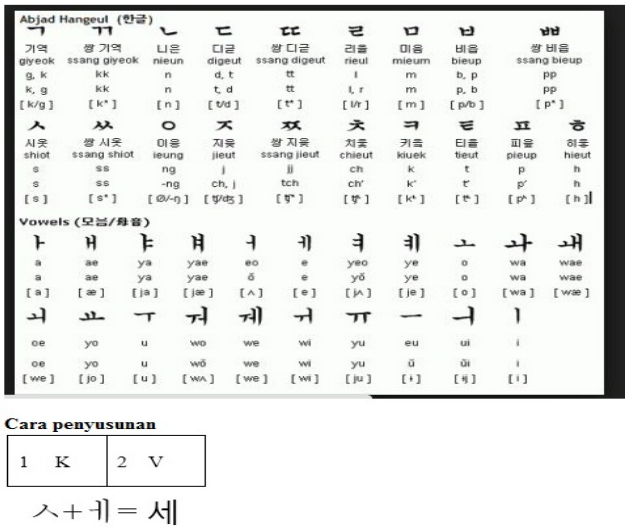
Halaman ini digunakan untuk mencari kata dari bahasa Korea ke bahasa Indonesia.



Gambar 4.4 Halaman Menu Korea Indonesia

4.2.5 Halaman Huruf *Hangul*

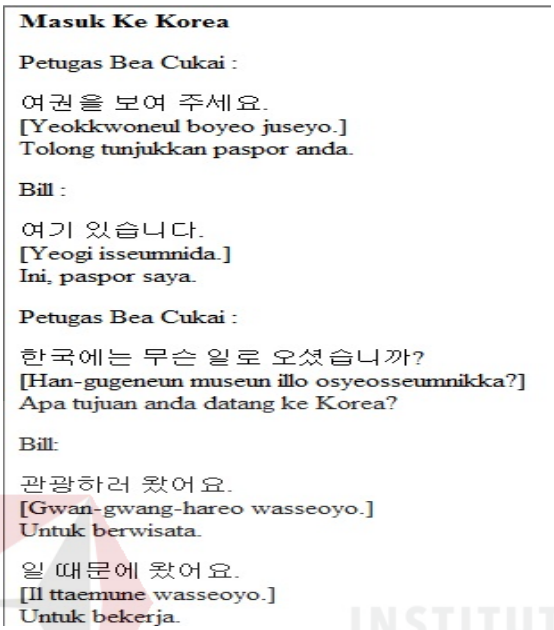
Halaman huruf *hangul* ini terdapat tabel macam – macam huruf *hangul* serta cara penyusunannya .



Gambar 4.5 Halaman Huruf *Hangul*

4.2.6 Halaman Contoh Percakapan

Halaman contoh percakapan ini hanya menampilkan beberapa contoh percakapan.



Gambar 4.6 Halaman Menu Contoh Percakapan

4.3 Hasil Pengujian dan Evaluasi Aplikasi

Uji coba dan evaluasi bertujuan untuk memastikan bahwa aplikasi telah dibuat dengan benar sesuai dengan kebutuhan atau tujuan yang diharapkan. Kekurangan atau kelemahan aplikasi pada tahap ini akan dievaluasi sebelum diimplementasikan secara nyata. Pengujian terhadap aplikasi menggunakan metode *Whitebox Testing* dan *Blackbox Testing*.

4.3.1 White box Testing

Tujuan dari pengujian ini yaitu untuk mengetahui secara internal cara kerja suatu perangkat lunak. Pada pengujian ini fungsi yang diuji yaitu proses mencari padanan kata menggunakan metode padanan kata menggunakan metode *Binary Search*.

1. Pengurutan Kata di *Array*

Dalam menjalankan metode *Binary Search* data yang digunakan harus terurut. Untuk menghubungkan database fisik dan aplikasi, data yang terurut akan disimpan sementara ke dalam *array*.

Berikut adalah *source code* dari proses pengurutan di *array*.

```
ix=0;
        datakata = new DataKata(Intro.this);
               dbkata = datakata.getWritableDatabase();
               kataCursor = dbkata.rawQuery("SELECT id_kata, indonesia, korea,
lafal, indeks_i_k, indeks_k_i FROM kata order by indonesia", null);
               if (kataCursor.moveToFirst()) {
                      for (; !kataCursor.isAfterLast();
kataCursor.moveToNext()) {
                             ix++;

      datakata.updateIK(dbkata,kataCursor.getString(0).toString(),Integer.to
String(ix));

                      }

               }
        ix=0;
        datakata = new DataKata(Intro.this);
               dbkata = datakata.getWritableDatabase();
               kataCursor = dbkata.rawQuery("SELECT id_kata, indonesia, korea,
lafal, indeks_i_k, indeks_k_i FROM kata order by korea", null);
               if (kataCursor.moveToFirst()) {
                      for (; !kataCursor.isAfterLast();
kataCursor.moveToNext()) {
                             ix++;

       datakata.updateKI(dbkata,kataCursor.getString(0).toString(),Integer.to
String(ix));
```

Gambar 4.7 *Source Code* Mengurutkan Kata di *Array*

Dalam mengurutkan data terdapat 2(dua) syntak, yang pertama adalah syntak untuk kata Indonesia dan syntak yang kedua adalah untuk kata Korea.

- Untuk kata Indonesia ("SELECT id_kata, indonesia, korea, lafal, indeks_i_k, indeks_k_i FROM kata order by indonesia", null);

  Tabel berisikan id_kata, indonesia, korea, lafal diurutkan berdasarkan (order by indonesia) lalu disimpan di indeks i_k.

- Untuk kata Korea ("SELECT id_kata, indonesia, korea, lafal, indeks_i_k, indeks_k_i FROM kata order by korea", null);

  Tabel berisikan id_kata, indonesia, korea, lafal diurutkan berdasarkan (order by indonesia) lalu disimpan di indeks k_i.

Data pertama dibandingkan dengan data yang terdapat pada indek selanjutnya, apabila lebih mesar maka data ditukar, begitu juga seterusnya.

Setelah data tertukar semua, maka selanjutnya dilakukan pengecekan nilai data dan selanjutnya disimpan sementara di *array*.

2. Proses Pengambilan Data di *Array*

Setelah data tersimpan di *array* kemudian dijalankan proses selanjutnya yaitu pengambilan data pada *array* untuk metode *Binary Search*. Berikut *source code* pengambilan data dari *array*.


```
ix=0;
        datakata = new DataKata(IndonesiaKorea.this);
                db = datakata.getWritableDatabase();
                kataCursor = db.rawQuery("SELECT id_kata, indonesia, korea,
lafal, indeks_i_k, indeks_k_i FROM kata order by indeks_i_k", null);
                if (kataCursor.moveToFirst()) {
                        for (; !kataCursor.isAfterLast();
kataCursor.moveToNext()) {
                                ix++;

        //datakata.updateIK(db,kataCursor.getString(0).toString(),Integer.toSt
ring(ix));
                                 String id = kataCursor.getString(0).toString();
                          String i = kataCursor.getString(1).toString();
                          String k = kataCursor.getString(2).toString();
                          String l = kataCursor.getString(3).toString();
                          String ik = kataCursor.getString(4).toString();
                          String ki = kataCursor.getString(5).toString();

                         ListKata.add(new Kata(id,i,k,l,ik,ki));
                        }

                }
```

```
ix=0;
        datakata = new DataKata(KoreaIndonesia.this);
               db = datakata.getWritableDatabase();
               kataCursor = db.rawQuery("SELECT id_kata, indonesia, korea,
lafal, indeks_i_k, indeks_k_i FROM kata order by indeks_k_i", null);
               if (kataCursor.moveToFirst()) {
                      for (; !kataCursor.isAfterLast();
kataCursor.moveToNext()) {
                              ix++;

       //datakata.updateIK(db,kataCursor.getString(0).toString(),Integer.toSt
ring(ix));
                               String id = kataCursor.getString(0).toString();
                        String i = kataCursor.getString(1).toString();
                        String k = kataCursor.getString(2).toString();
                        String l = kataCursor.getString(3).toString();
                        String ik = kataCursor.getString(4).toString();
                        String ki = kataCursor.getString(5).toString();

                       ListKata.add(new Kata(id,i,k,l,ik,ki));
                      }
```

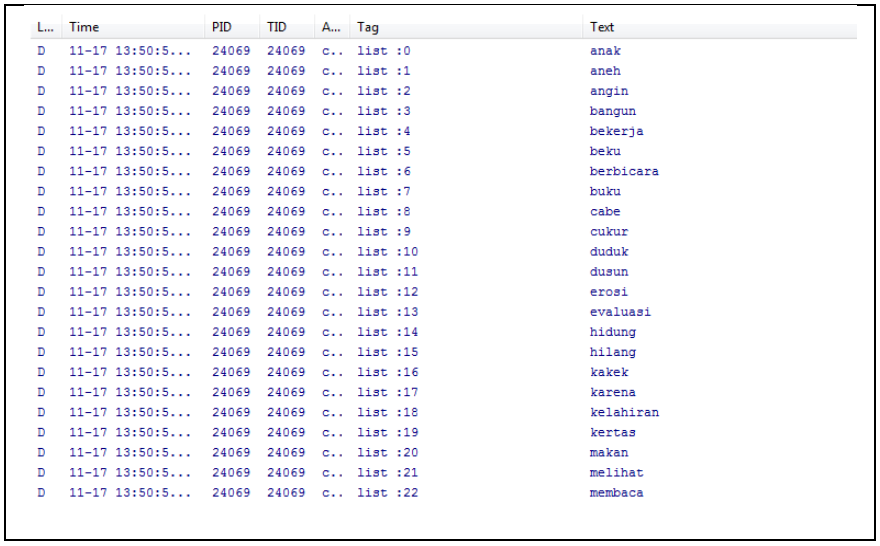
Gambar 4.8 *Source Code* Mengambil Data di *Array*

Cara penyusunan datanya adalah :

- Pada menu Indonesia-Korea alphabet pertama Indonesia adalah A dan indeks pertama adalah 0 maka kata pertama pada huruf A dan tiap alphabet pada kata dilooping sampai ketemu yang paling awal dimasukkan ke dalam indeks 0 dan alphabet selanjutnya dimasukkan ke indeks 1 sampai indeks akhir.
- Pada menu Korea-Indonesia alphabet pertama Korea adalah ㄱ dan dimasukkan ke indeks 0 dan alphabet selanjutnya dimasukkan ke indeks 1 sampe indeks akhir.

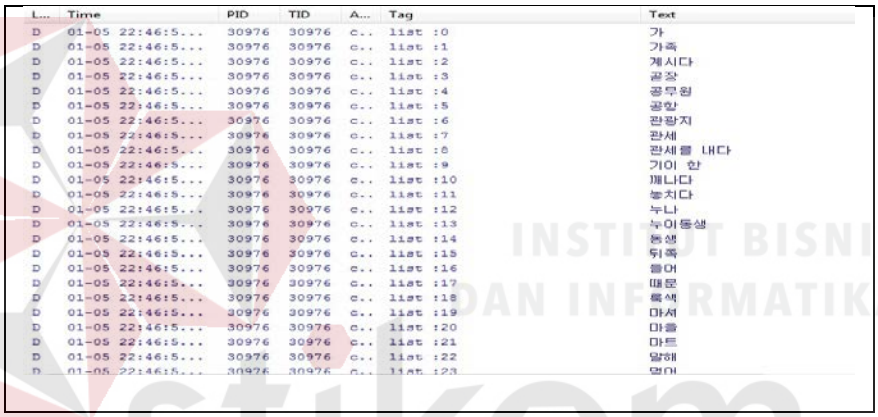
Setelah data diurutkan, kemudian data akan tampil pada program dengan menggunakan syntak “ `for (int x=0; x<ix; x++) { Log.d("list :"+x,ListKata.get(x).getIndonesia());` ”.

Gambar berikut adalah list data yang sudah terurut.



L...	Time	PID	TID	A...	Tag	Text
D	11-17 13:50:5...	24069	24069	c..	list :0	anak
D	11-17 13:50:5...	24069	24069	c..	list :1	aneh
D	11-17 13:50:5...	24069	24069	c..	list :2	angin
D	11-17 13:50:5...	24069	24069	c..	list :3	bangun
D	11-17 13:50:5...	24069	24069	c..	list :4	bekerja
D	11-17 13:50:5...	24069	24069	c..	list :5	beku
D	11-17 13:50:5...	24069	24069	c..	list :6	berbicara
D	11-17 13:50:5...	24069	24069	c..	list :7	buku
D	11-17 13:50:5...	24069	24069	c..	list :8	cabe
D	11-17 13:50:5...	24069	24069	c..	list :9	cukur
D	11-17 13:50:5...	24069	24069	c..	list :10	duduk
D	11-17 13:50:5...	24069	24069	c..	list :11	dusun
D	11-17 13:50:5...	24069	24069	c..	list :12	erosi
D	11-17 13:50:5...	24069	24069	c..	list :13	evaluasi
D	11-17 13:50:5...	24069	24069	c..	list :14	hidung
D	11-17 13:50:5...	24069	24069	c..	list :15	hilang
D	11-17 13:50:5...	24069	24069	c..	list :16	kakek
D	11-17 13:50:5...	24069	24069	c..	list :17	karena
D	11-17 13:50:5...	24069	24069	c..	list :18	kelahiran
D	11-17 13:50:5...	24069	24069	c..	list :19	kertas
D	11-17 13:50:5...	24069	24069	c..	list :20	makan
D	11-17 13:50:5...	24069	24069	c..	list :21	melihat
D	11-17 13:50:5...	24069	24069	c..	list :22	membaca

Gambar 4.9 List Kata Indonesia Teurut



L...	Time	PID	TID	A...	Tag	Text
D	01-05 22:46:5...	30976	30976	c..	list :0	가
D	01-05 22:46:5...	30976	30976	c..	list :1	가족
D	01-05 22:46:5...	30976	30976	c..	list :2	계시다
D	01-05 22:46:5...	30976	30976	c..	list :3	공장
D	01-05 22:46:5...	30976	30976	c..	list :4	공무원
D	01-05 22:46:5...	30976	30976	c..	list :5	공항
D	01-05 22:46:5...	30976	30976	c..	list :6	관광지
D	01-05 22:46:5...	30976	30976	c..	list :7	관세
D	01-05 22:46:5...	30976	30976	c..	list :8	관세를 내다
D	01-05 22:46:5...	30976	30976	c..	list :9	기이 한
D	01-05 22:46:5...	30976	30976	c..	list :10	깨나다
D	01-05 22:46:5...	30976	30976	c..	list :11	놓치다
D	01-05 22:46:5...	30976	30976	c..	list :12	누나
D	01-05 22:46:5...	30976	30976	c..	list :13	누이동생
D	01-05 22:46:5...	30976	30976	c..	list :14	동생
D	01-05 22:46:5...	30976	30976	c..	list :15	뒤쪽
D	01-05 22:46:5...	30976	30976	c..	list :16	들어
D	01-05 22:46:5...	30976	30976	c..	list :17	때문
D	01-05 22:46:5...	30976	30976	c..	list :18	특색
D	01-05 22:46:5...	30976	30976	c..	list :19	마셔
D	01-05 22:46:5...	30976	30976	c..	list :20	마을
D	01-05 22:46:5...	30976	30976	c..	list :21	마트
D	01-05 22:46:5...	30976	30976	c..	list :22	말해
D	01-05 22:46:5...	30976	30976	c..	list :23	멀어

Gambar 4.10 List Kata Korea Terurut

3. Proses pencarian dengan metode *Binary Search.*

Langkah dalam pencarian biner adalah :

1. Mula-mula diambil posisi awal = 1 dan posisi akhir = n

   Berdasarkan data pada kamus Korea – Indonesia terdapat 100 kata. Maka posisi awal adalah indeks ke 0 dan posisi akhir adalah indeks ke 99.

2. Mencari posisi tengah dengan rumus = (posisi awal + posisi akhir) / 2

```
kataindonesia = txtIndonesia.getText().toString();

atas=1;
bawah=ix;


tengah=(atas+bawah)/2;
ketemu=false;
```

```
        binarySearch();
```

Gambar 4.12 *Source Code* Mencari Posisi Tengah

Cara menghitung secara manual adalah Tengah = (0+100) / 2 = 50

Maka posisi tengah ditemukan pada indeks 49 dengan kata " Karena ". Dikarenakan posisis awal adalah indeks 0.

```
D   01-06 11:26:3...   13815  13815  c..  list :45                    kakek
D   01-06 11:26:3...   13815  13815  c..  list :46                    kakek
D   01-06 11:26:3...   13815  13815  c..  list :47                    kanan
D   01-06 11:26:3...   13815  13815  c..  list :48                    kantor
D   01-06 11:26:3...   13815  13815  c..  list :49                    karena
D   01-06 11:26:3...   13815  13815  c..  list :50                    kelahiran
D   01-06 11:26:3...   13815  13815  c..  list :51                    keluarga
```

Gambar 4.13 Hasil Pencarian Posisi Tengah

3. Pada proses pencarian ini diberikan contoh untuk mencari kata "Tidur" pada kata Indonesia yang terdapat pada indeks 96.

```
D   01-06 11:26:3...   13815  13815  c..  list :93                    terminal bis
D   01-06 11:26:3...   13815  13815  c..  list :94                    terus
D   01-06 11:26:3...   13815  13815  c..  list :95                    tidak
D   01-06 11:26:3...   13815  13815  c..  list :96                    tidur
D   01-06 11:26:3...   13815  13815  c..  list :97                    toilet
D   01-06 11:26:3...   13815  13815  c..  list :98                    traffic light
D   01-06 11:26:3...   13815  13815  c..  list :99                    wisata pusat kota
```

Gambar 4.14 Hasil Pencarian Kata Kunci

Untuk membandingkan data yang dicari dengan data tengah dan menampilkan pencarian yang ketemu menggunakan *source code* pada gambar 4.15 berikut .

```
        binarySearch();


                String result = "";
                String result2 = "";

                if(ketemu==false){
                        result = "Terjemahan Not Found";
                        result2 = "Terjemahan Not Found";
                }else{
                        result = ListKata.get(tengah-1).getKorea();
                        result2 = ListKata.get(tengah-1).getLafal();
                }
                txtKorea.setText(result);
                txtLafal.setText(result2);
                         }
```

Gambar 4.15 *Source Code* Membandingkan Data Tengah

Berdasarkan urutan data pada database kata yang dicari pada posisi indeks no 96 sedangkan posisi tengah adalah indeks no 49. Jika elemen tengah tidak sama dengan kata yang dicari, maka proses selanjutnya :

a. Jika elemen tengah lebih besar dari data yang dicari, maka pencarian dilakukan pada setengah *array* pertama. Caranya dengan menggunakan perintah kiri sama dengan tengah ditambah satu.
b. Jika elemen tengah lebih kecil dari data yang dicari, maka pencarian dilakukan pada setengah *array* berikutnya. Caranya dengan menggunakan perintah kanan sama dengan tengah dikurangi satu.
c. Tengah sama dengan kiri ditambah (kanan – kiri) dibagi dua.

Gambar berikut ini adalah *source code* dari proses pencarian *Binary Search*.

```
            public void binarySearch()

                    while ( atas < bawah && ketemu==false) {
                                Log.d("atas,tengah,bawah,ketemu",
    atas+","+tengah+","+bawah+","+ketemu);
                                Log.d("kata 1 vs kata 2", kataindonesia+" vs
    "+ListKata.get(tengah-1).getIndonesia());
                                 if
        (kataindonesia.compareToIgnoreCase(ListKata.get(tengah-
        1).getIndonesia()) == 0) {
                                        ketemu=true;


                                 }
else {
                                if
(kataindonesia.compareToIgnoreCase(ListKata.get(tengah-
1).getIndonesia()) > 0) {
                                        atas = tengah;
                                        //binarySearch();
                                 } else if
(kataindonesia.compareToIgnoreCase(ListKata.get(tengah-
1).getIndonesia()) < 0) {
                                        bawah = tengah;
                                        //binarySearch();
                                 }
                                tengah = ( atas + bawah) / 2;
                        }
```

```
                if(ketemu==false && bawah-atas==1){
                        if
(kataindonesia.compareToIgnoreCase(ListKata.get(bawah-
1).getIndonesia()) == 0) {
                                ketemu=true;
                                tengah=bawah;
                        } else if
(kataindonesia.compareToIgnoreCase(ListKata.get(atas-
1).getIndonesia()) == 0) {
                                ketemu=true;
                                tengah=atas;
                        } else{
                                break;
                        }
```

Gambar 4.16 *Source Code* Metode *Binary Search*

Berdasarkan proses pencarian selanjutnya akan ditemukan hasilnya yang di tampilkan pada gambar 4.17.

```
.   13815  13815  c..  atas,tengah,bawah,ketemu      1,50,100,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs karena
.   13815  13815  c..  atas,tengah,bawah,ketemu      50,75,100,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs paspor
.   13815  13815  c..  atas,tengah,bawah,ketemu      75,87,100,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs rumah
.   13815  13815  c..  atas,tengah,bawah,ketemu      87,93,100,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs teman
.   13815  13815  c..  atas,tengah,bawah,ketemu      93,96,100,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs tidak
.   13815  13815  c..  atas,tengah,bawah,ketemu      96,98,100,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs toilet
.   13815  13815  c..  atas,tengah,bawah,ketemu      96,97,98,false
.   13815  13815  c..  kata 1 vs kata 2             tidur vs tidur
```

Gambar 4.17 Hasil Pencarian Metode *Binary Search*

4.3.2 Black Box Testing

Pada tahap uji coba ini dilakukan untuk mengetahui fungsi – fungsi pada aplikasi dapat berjalan dengan baik atau tidak berdasarkan tabel *test case*. Desain uji coba fungsi aplikasi sebagai berikut :

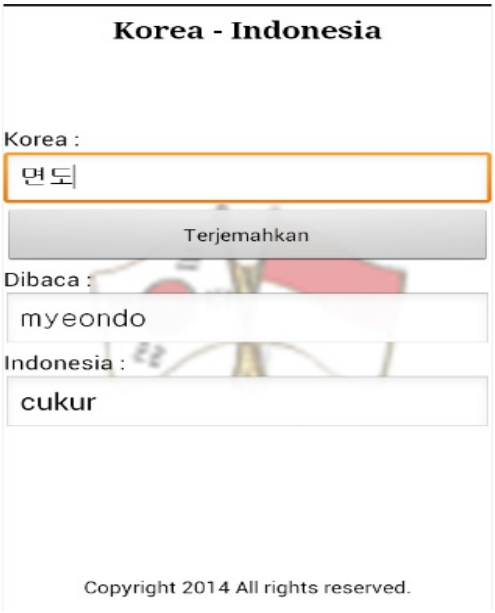
1. Hasil Uji Coba Pencarian Korea - Indonesia

Pengujian dilakukan untuk mengetahui apakah aplikasi kamus Korea-Indonesia dengan menggunakan metode *Binary Search* berbasis android ini telah berjalan sesuai dengan tujuan. Hasil pengujian pada fungsi pencarian kamus Korea - Indonesia dapat dilihat pada tabel 4.1.

Tabel 4.1 Uji Coba Pencarian Korea – Indonesia

Test Case	**Tujuan**	**Input**	**Output yang diharapkan**
1	Mencari Terjemahan kata dari bahasa Korea ke bahasa Indonesia	Memasukkan kata dengan huruf *Hangul*	Aplikasi berhasil menampilkan kata yang dicari beserta latin dan terjemahan ke dalam bahasa Indonesia
2	Menampilkan pesan jika pencarian tidak ketemu	Memasukkan kata yang salah	Form menampilkan kata " Terjemahan *Not Found* "
3	Menampilkan pesan jika yang diinputkan bukan huruf *Hangul*	Memasukkan kata dengan alfabet	form menampilkan kata "Terjemahan *Not Found*"

Pada tabel 4.1 merupakan hasil uji coba pencarian Korea - Indonesia, dimana hasil keluaran sistem sesuai dengan output yang diharapkan. Dibawah ini adalah gambar-gambar pembuktian test case 1, test case 2 dan test case 3.



Gambar 4.18 Pembuktian *Test Case* 1

Gambar 4.19 Pembuktian *Test Case* 2

Gambar 4.20 Pembuktian *Test Case* 3

2. Perancangan Uji Coba Pencarian Indonesia – Korea

Proses pencarian dilakukan dengan cara memasukkan sebuah kata yang ingin dicari menggunakan font biasa kemudian sistem akan melakukan pencarian dan menampilkan hasil pencarian berupa kata yang dicari beserta terjemahan ke bahasa Korea dan latinnya. Desain pengujian pada fungsi pencarian kamus Indonesia - Korea dapat dilihat pada tabel 4.2.

Tabel 4.2 Uji Coba Pencarian Indonesia – Korea

Test Case	**Tujuan**	**Input**	**Output yang diharapkan**
4	Mencari Terjemahan kata dari bahasa Indonesia ke bahasa Korea	Memasukkan kata dengan huruf alfabet	Aplikasi berhasil menampilkan kata yang dicari beserta terjemahan bahasa Korea dan latinnya
5	Menampilkan pesan jika pencarian tidak ketemu	Memasukkan kata yang salah	form menampilkan kata "Terjemahan *Not Found*"
6	Menampilkan pesan jika yang diinputkan bukan huruf alfabet	Memasukkan kata dengan huruf *Hangul*	form menampilkan kata "Terjemahan *Not Found*"

Pada tabel 4.2 merupakan hasil uji coba pencarian Korea - Indonesia, dimana hasil keluaran sistem sesuai dengan output yang diharapkan. Dibawah ini adalah gambar-gambar pembuktian test case 4, test case 5 dan test case 6.

Gambar 4.21 Hasil *Test Case* 4

Gambar 4.22 Hasil *Test Case* 5

Gambar 4.23 Hasil *Test Case* 6

3. Perancangan Uji Coba Huruf *Hangul*

Proses pencarian dilakukan dengan cara memilih menu huruf *hangul*. Desain pengujian pada fungsi huruf *hangul* dapat dilihat pada tabel 4.3.

Tabel 4.3 Uji Coba Huruf *Hangul*

Test Case	Tujuan	Input	Output yang diharapkan
7	Menampilkan tabel macam – macam huruf *hangul* dan cara penyusunan huruf *hangul*	Memilih menu "huruf *hangul*" pada menu utama	Aplikasi berhasil menampilkan tabel yang berisi huruf *hangul* dan tabel cara penyusunan huruf *hangul*

Gambar 4.24 Hasil Test Case 7

4. Perancangan Uji Coba Contoh Percakapan

Proses pencarian dilakukan dengan cara memilih menu contoh percakapan. Desain pengujian pada fungsi contoh percakapan dapat dilihat pada tabel 4.4.

Tabel 4.4 Uji Coba Contoh Percakapan

Test Case	Tujuan	Input	Output yang diharapkan
8	Menampilkan contoh-contoh percakapan	Memilih menu “contoh percakapan” pada menu utama	Aplikasi berhasil menampilkan contoh percakapan dalam bahasa Indonesia dan bahasa Korea beserta latinnya

Gambar 4.25 Menu Contoh Percakapan

Gambar 4.26 Hasil Test Case 8

Berdasarkan uji coba yang dilakukan dapat disimpulkan kemampuan dan kelemahan aplikasi sebagai berikut :

1. Kemampuan Aplikasi

Kemampuan dari aplikasi yang dibangun antara lain :

1. Aplikasi dapat memberikan informasi hasil pencarian arti kata dari bahasa Indonesia dan bahasa Korea beserta huruf *hangul* dan cara bacanya.
2. Aplikasi dapat memberikan informasi berupa tampilan huruf *hangul*, cara penyusunannya, contoh percakapan.

2. Kelemahan Aplikasi

Kelemahan dari aplikasi yang dibangun antara lain :

1. Karena aplikasi bersifat *offline* penambahan kata bisa dilakukan ketika ada notifikasi *update* sehingga disarankan menggunakan *wifi* untuk menghemat kuota *internet*.