BAB IV

IMPLEMENTASI DAN EVALUASI

4.1. Kebutuhan Sistem

Sebelum mengimplementasikan dan menjalankan Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris 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

Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris dijalankan pada perangkat *handphone* (telepon genggam) Android, jadi perangkat keras yang digunakan adalah *handphone* Android. Spesifikasi *handphone* yang dibutuhkan untuk menjalankan aplikasi ini adalah:

1. Layar berwarna 16M *colors*
2. *Internal memory* 20,28MB
3. *Support* GPRS dan Wifi
4. *CPU* 800MHz *processor*

4.1.2. Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang digunakan untuk mengembangkan Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris adalah:

1. Sistem Operasi Froyo 2.2
2. Android SDK r10-linux
3. ADT Plugin *for* Eclipse

4. Eclipse 3.6.0
5. Sqlite3.6.2
6. Terminal/cmd

4.2. Pembuatan Program

Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris dibangun dengan menggunakan Android SDK r10-linux dan *text editor* Eclipse 3.6.0 yang didukung dengan ADT Plugin *for* Eclipse. Penulisan kode program pata *text editor* disimpan dalam file dengan extensi *.java dan *.xml. Android SDK akan meng-*compile file* berektensi *.java dan *.xml tersebut dan membuat *package* untuk aplikasi *mobile* belajar melafalkan bahasa Inggris berekstensi *.apk. Nantinya, *file* *.apk tersebut yang akan digunakan untuk menjalankan aplikasi melalui *handphone* android.

4.3. Implemtasi Sistem

Setelah kebutuhan sistem terpenuhi, langkah selanjutnya adalah mengimplementasikan rancangan sistem ke dalam sebuah Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris. Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris terbagi dalam beberapa modul, yaitu modul untuk mengolah pelafalan, *maintenance* data pengguna, *maintenance* data kategori, *maintenance* data *item*, dan modul untuk mengolah nilai kedalam grafik. Implemtasi aplikasi *mobile* belajar melafalkan bahasa nggris akan dijelaskan sebagai berikut.

4.3.1. Tampilan Pengenalan

Gambar 4.1 Tampilan Pengenalan

Gambar 4.2 Tampilan Daftar Pengguna

Pada saat aplikasi dijalankan yang pertama kali muncul adalah tampilan pengenalan atau sambutan. Dari tampilan ini, pengguna dapat melihat sambutan pertama yaitu berupa tampilan pengenalan yang memberi informasi kepada pengguna bahwa aplikasi ini adalah Aplikasi Belajar Melafalkan Bahasa Inggris. Tampilan pengenalan dapat dilihat pada gambar 4.1. Pada tampilan tersebut, hanya sebatas pengenalan awal terhadap aplikasi yang akan digunakan.

4.3.2. Tampilan Daftar Pengguna

Pada saat tampilan pengenalan aplikasi telah selesai, maka yang pertama kali muncul adalah daftar pengguna, dimana fungsi utama tampilan ini adalah memperlihatkan daftar pengguna yang telah mendaftar. Pengguna diharapkan untuk menambah pengguna baru untuk dapat menggunakan aplikasi ini. Tampilan daftar pengguna dapat dilihat pada gambar 4.2.

```
public void btTambah_Click(View view){
    try{
        //menuju intent PPengguna (Pengaturan Pengguna)
        Intent PPengguna = new Intent(this,PPengguna.class);
        PPengguna.putExtra("asal", "MPengguna");
        startActivity(PPengguna);
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.3 Proses pada Tombol Tambah

```
try{
    int idPeng=(int) id;
    String nmPeng="";
    //ambil nama pengguna
    Cursor c=(Cursor) getListView().getItemAtPosition(position);
    startManagingCursor(c);
    if(c!=null){
        if(c.moveToPosition(position)){
            nmPeng= c.getString(c.getColumnIndex(DatabaseHelper.cNmPeng));
        }
    }

    //setting id dan nm pengguna sbg global variabel
    ((SetApp)getApplication()).setIdPengguna(idPeng);
    ((SetApp)getApplication()).setNamaPengguna(nmPeng);
    //------------------------------------------------

    Intent MMenu=new Intent(this,MMenu.class);
    startActivity(MMenu);
    finish();
}catch(Exception e){
    Alerts.CatchError(this, e.toString());
}
```

Gambar 4.4 Proses Mengaktifkan Pengguna dan Pindah ke Menu Utama

Pada tampilan tersebut, terdapat tombol tambah pengguna dimana berfungsi untuk mengarahkan pengguna dari daftar pengguna (`intent MPengguna`) ke pengaturan pengguna (`intent PPengguna`) agar dapat menambah pengguna baru, dapat dilihat pada gambar 4.3.

Apabila pengguna telah menambah pengguna baru, maka pengguna dapat melanjutkan dengan memilih nama pengguna yang telah terdaftar untuk mengaktifkan pengguna dan melanjutkan ke menu berikutnya, dapat dilihat pada gambar 4.4.

```
if (keyCode == KeyEvent.KEYCODE_BACK) {
    //alert sebelum keluar
    AlertDialog.Builder b = Alerts.Pesan(this, "Keluar", "Apakah anda yakin?");
    b.setCancelable(false);
    b.setPositiveButton("Ya", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            MPengguna.this.finish();
        }
    });
    b.setNegativeButton("Tidak", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            dialog.cancel();
        }
    });
    AlertDialog alert = b.create();
    alert.show();

    return false;
}
```

Gambar 4.5 Proses pada Tombol Back

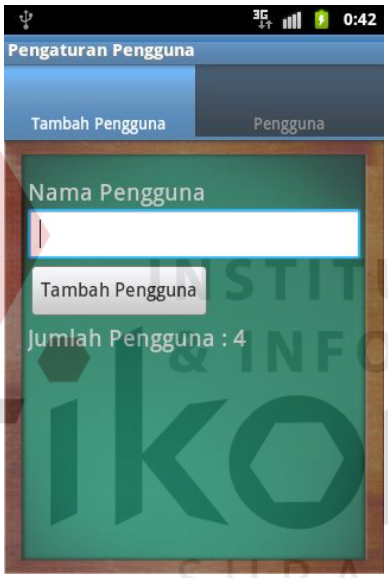
Pada proses tersebut terdapat pengaturan nama pengguna pada global variable (`setIdPengguna()` dan `setNamaPengguna()`), agar dapat diketahui pengguna yang sedang aktif saat ini dan proses pengalihan ke menu utama (`intent MMenu`). Terdapat proses lain, yaitu proses yang berfungsi untuk keluar dari aplikasi pada saat pengguna menekan tombol *back* (kembali) pada *device*. Pada proses ini, apabila pengguna menekan tombol *back* maka akan muncul sebuah pesan untuk meyakinkan pengguna dan setelah dikonfirmasi maka

proses akan diselesaikan (`finish()`). Proses untuk keluar dari aplikasi dengan menggunakan tombol *back* pada *device* dapat dilihat pada gambar 4.5.

4.3.3. Tampilan Pengaturan Pengguna

Pada tampilan pengaturan pengguna terdapat dua buah *tab* yaitu *tab* tambah pengguna yang berfungsi untuk menambah pengguna baru dan *tab* pengguna yang berfungsi untuk melihat daftar pengguna yang telah terdaftar.

A. Tampilan Tab Tambah Pengguna



Gambar 4.6 Tampilan Tab Tambah Pengguna

```
if(namaPeng.length()<3){
    Alerts.Peringatan(this,"Peringatan", "Nama Pengguna Minimal 3 Huruf ");
```

Gambar 4.7 Proses Validasi Masukkan Pengguna

Tampilan *tab* tambah pengguna dapat dilihat pada gambar 4.6. pada tampilan ini, terdapat sebuah *edit text* nama pengguna dimana berfungsi untuk menerima masukkan pengguna, dan *text view* jumlah pengguna untuk mengetahui jumlah pengguna yang terdaftar. pada tampilan ini, terdapat

proses validasi masukkan pengguna dan proses menambahkan pengguna baru yang dilanjutkan dengan pengalihan ke *tab* pengguna.

Proses validasi masukkan pengguna dapat dilihat pada gambar 4.7, dimana berfungsi untuk mengecek isi masukkan pengguna, apakah tidak kurang dari tiga huruf. Proses menambah pengguna dan pengalihan ke *tab* pengguna atau pengalihan ke tampilan daftar pengguna, dapat dilihat pada gambar 4.8. Pada proses tersebut akan memasukkan data pengguna kedalam *database* jika tidak terdapat kesalahan, setelah itu dialihkan ke *tab* pengguna (`intent TGridPengguna`) atau tampilan daftar pengguna (apabila asal *intent* sebelumnya adalah `MPengguna`).

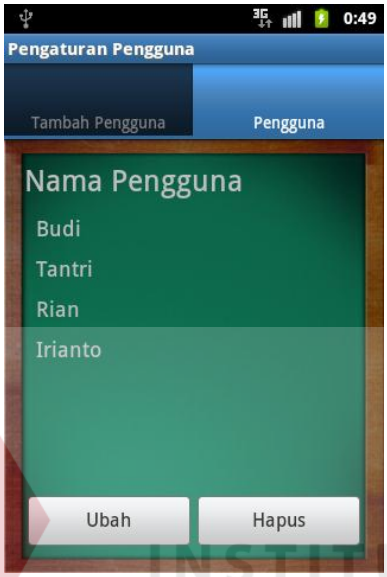
```
}else{
    Pengguna peng=new Pengguna(namaPeng);
    if(dbHelper.AddPengguna(peng)!=0){
        //alert klo diklik bener
        AlertDialog.Builder bldr = Alerts.Pesan(this, "Tambah Pengguna Baru", "Berhasil");
        bldr.setPositiveButton("Ok", new OnClickListener() {
            @Override
            public void onClick(DialogInterface arg0, int arg1) {
                // TODO Auto-generated method stub
                tbNmPengguna.setText("");
                lbJmlPeng.setText("Jumlah Pengguna : "+String.valueOf(dbHelper.getCountPengguna()));

                if(asal!=null){
                    if(asal.equals("MPengguna")){
                        ((TabActivity)getParent()).finish();
                    }else{
                        //Pindah ke tab Pengguna
                        ((TabActivity)getParent()).getTabHost().setCurrentTab(1);
                    }
                }
            }
        });

        AlertDialog diag=bldr.create();
        diag.show();
    }else{
        //alert klo diklik gagal
        Alerts.Peringatan(this, "Tambah Pengguna Baru", "Gagal");
        tbNmPengguna.setText("");
    }
}
```

Gambar 4.8 Proses Menambah Pengguna dan Pengalihan Tab atau Intent

B. Tampilan Tab Daftar Pengguna

Tampilan *tab* pengguna pertama kali akan menampilkan daftar pengguna, dapat dilihat pada gambar 4.9. Tampilan *tab* daftar pengguna berfungsi untuk

menampilkan pengguna yang telah terdaftar. Pada tampilan ini, terdapat *grid view* pengguna yang berfungsi untuk menampilkan data pengguna dan dua buah tombol yaitu ubah yang berfungsi untuk mengalihkan ke tampilan *tab* ubah pengguna dan hapus yang berfungsi untuk menghapus pengguna.



Gambar 4.9 Tampilan Tab Daftar Pengguna

```
private void LoadGrid(){
    dbHelper=new DatabaseHelper(this);
    try{
        Cursor c= dbHelper.getIsiPengguna();
        startManagingCursor(c);
        //binding
        String[] from= {DatabaseHelper.cNmPeng,"_id"};
        int[] to= {R.id.cNmPeng};
        SimpleCursorAdapter sca = new SimpleCursorAdapter(this, R.layout.grid_row_pengguna, c, from, to);
        //menampilkan
        gridPengguna.setAdapter(sca);
    }catch(Exception e){
        Alerts.CatchError(TGridPengguna.this,e.toString());
    }
}
```

Gambar 4.10 Proses Me-Binding dan Menampilkan pada Grid View

Proses menampilkan data pengguna pada *grid view* dapat dilihat pada gambar 4.10, dimana proses tersebut menampilkan dan *binding* data pengguna dari *database* kedalam *grid view*. Proses pengalihan dari *tab* daftar pengguna ke *tab* ubah pengguna tidak menggunakan *intent*, tetapi langsung didalam *class* tersebut, proses tersebut dapat dilihat pada gambar 4.11.

```
private void contentUbah(){
    /*------------------
     * Memanggil layout add pengguna dan digunakan untuk fungsi ubah
     */
    berada=1;//klo 1 ada di ubah
    setContentView(R.layout.add_pengguna);

    btAdd=(Button)findViewById(R.id.btAdd);
    lbJmlPeng=(TextView)findViewById(R.id.lbJmlPeng);
    tbNmPengguna=(EditText)findViewById(R.id.tbNmPengguna);

    tbNmPengguna.setText(nmPeng.getText().toString());
    btAdd.setText("Ubah Pengguna");
    lbJmlPeng.setVisibility(8);
}
public int getIdPeng() {
    return idPeng;
}
```

Gambar 4.11 Proses Pengalihan Tab

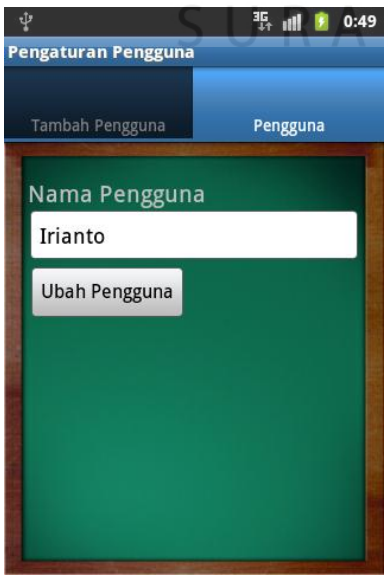
```
public void btUbah_Click(View view){
    try{
        if(nmPeng!=null){
            contentUbah();
        }else{
            Alerts.Peringatan(TGridPengguna.this, "Peringatan",
            "Untuk menggunakan fungsi ini\nSilahkan pilih pengguna terlebih dahulu");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
public void btHapus_Click(View view){
    try{
        if(nmPeng!=null){
            AlertDialog.Builder bldr = Alerts.Pesan(this, "Hapus Pengguna", "Apakah anda yakin menghapus "+nmPeng.getText().toString()+" ?");
            bldr.setPositiveButton("Ya", new OnClickListener() {
                public void onClick(DialogInterface diag, int arg1) {
                    // TODO Auto-generated method stub
                    try{
                            Pengguna peng=new Pengguna(nmPeng.getText().toString());
                            peng.setIdPengguna(idPeng);

                            DatabaseHelper db=new DatabaseHelper(TGridPengguna.this);
                            if(db.DeletePengguna(peng)!=0){
                                LoadGrid();
                            }else{
                                Alerts.Peringatan(TGridPengguna.this, "Peringatan","Fungsi hapus gagal");
                            }
                    }catch(Exception e){
                        Alerts.CatchError(TGridPengguna.this, e.toString());
                    }
                }
            });
            bldr.setNegativeButton("Tidak", null);
            AlertDialog psn = bldr.create();
            psn.show();
        }else{
            Alerts.Peringatan(TGridPengguna.this, "Peringatan", "Untuk menggunakan fungsi ini\nSilahkan pilih pengguna terlebih dahulu");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.12 Proses Menghapus Pengguna

Proses menghapus pengguna dapat dilihat pada gambar 4.12, dimana proses ini melakukan penghapusan data pengguna pada *database* jika tidak terjadi kesalahan.

C. Tampilan Tab Ubah Pengguna

Tampilan *tab* ubah pengguna berfungsi untuk mengubah nama pengguna yang tersimpan pada database dan ditampilkan setelah menekan tombol ubah, dapat dilihat pada gambar 4.13. Tampilan ini menggunakan tampilan *tab* tambah pengguna tetapi terdapat beberapa modifikasi penyesuaian, yaitu *text view* jumlah pengguna dihilangkan dan nama tombol diubah menjadi "ubah pengguna", dimana memiliki fungsi yang sama, tetapi pada awal ditampilkan akan memampilkan nama pengguna sesuai dengan "id_pengguna". Pada tampilan ini, terdapat proses validasi yang sama seperti *tab* tambah pengguna. Proses ubah nama pengguna dapat dilihat pada gambar 4.14, dimana berfungsi untuk mengubah nama pengguna pada *database* jika tidak terjadi kesalahan.



Gambar 4.13 Tampilan Tab Ubah Pengguna

```
}else{
    Pengguna peng=new Pengguna(nm_peng);
    peng.setIdPengguna(idPeng);

    DatabaseHelper db=new DatabaseHelper(TGridPengguna.this);

    if(db.UpdatePengguna(peng)!=0){
        AlertDialog.Builder bldr = Alerts.Pesan(this, "Ubah Pengguna", "Berhasil");
        bldr.setPositiveButton("Ok", new OnClickListener() {
            public void onClick(DialogInterface diag, int arg1) {
                // TODO Auto-generated method stub
                try{
                    contentGrid();
                }catch(Exception e){
                    Alerts.CatchError(TGridPengguna.this, e.toString());
                }
            }
        });
        AlertDialog psn = bldr.create();
        psn.show();
    }else{
        Alerts.Peringatan(this, "Ubah Pengguna", "Gagal");
    }
}
```

Gambar 4.14 Proses Mengubah Nama Pengguna

4.3.4. Tampilan Menu Utama

Gambar 4.15 Tampilan Menu Utama

Tampilan menu utama adalah tampilan awal setelah pengguna mendaftar pengguna baru, dapat dilihat pada gambar 4.15. Pada tampilan ini, terdapat beberapa tombol yaitu tombol belajar yang berfungi mengalihkan pada tampilan

menu belajar, tombol nilai yang berfungsi mengalihkan pada tampilan menu nilai, tombol data yang mengalihkan pada tampilan menu data, tombol cara menggunakan yang berfungsi mengalihkan pada tampilan cara menggunakan, tombol tentang yang berfungsi mengalihkan pada tampilan tentang dan tombol keluar yang berfungsi untuk mengakhiri aplikasi.

4.3.5. Tampilan Belajar

Gambar 4.16 Tampilan Belajar

Tampilan belajar berfungsi untuk menampilkan menu belajar, dapat dilihat pada gambar 4.16. Pada tampilan ini, terdapat dua tombol yaitu tombol latihan yang berfungsi untuk mengalihkan ke tampilan latihan dan tombol ujian yang berfungsi untuk mengalihkan ke menu kategori. Pada tampilan ini, terdapat proses mengalihkan tampilan belajar ke tampilan selanjutnya dan proses pengiriman *String* "asal" kepada `intent MKategori` (menu kategori).

Pada proses yang ditunjukkan gambar 4.17, menu belajar akan dialihkan ke *intent* baru berdasarkan tombol yang dipilih. serta string "asal" akan dikirimkan ke `intent MKategori` menggunakan "`putExtra("String name",`

"String value")", dimana berfungsi agar MKategori dapat memproses sesuai asal *intent*.

```
public void btLatihan_click(View view){
    try{
        Intent TSpeech=new Intent(this,TSpeech.class);
        startActivity(TSpeech);
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
public void btUjian_click(View view){
    try{
        Intent MKategori=new Intent(this,MKategori.class);
        MKategori.putExtra("asal", "ujian");
        startActivity(MKategori);
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.17 Proses Mengalihkan Tampilan Belajar

4.3.6. Tampilan Latihan

Gambar 4.18 Tampilan Latihan

Tampilan latihan berfungsi agar pengguna dapat melatih pelafalan sebelum melakukan ujian bahasa Inggris, dapat dilihat pada gambar 4.18. Pada tampilan ini, terdapat sebuah tombol lafal yang berfungsi untuk

mengaktifkan penerimaan pelafalan pengguna dan *list view* untuk menampilkan hasil identifikasi pelafalan pengguna berupa beberapa teks kemungkinan.

```
// cek apakah activity recognition didukung?
PackageManager pm = getPackageManager();
List<ResolveInfo> activities = pm.queryIntentActivities(new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
if (activities.size() != 0) {
    speakButton.setOnClickListener(this);
} else {
    speakButton.setEnabled(false);
    speakButton.setText("Pengenalan Suara Tidak Aktif");
}
```

Gambar 4.19 Proses Pengecekan Dukungan Speech Input

```
//klik untuk memulai activity voiceRecognition
public void onClick(View v) {
    if (v.getId() == R.id.btn_speak) {
        startVoiceRecognitionActivity();
    }
}

//menjalankan speech voiceRecognition
private void startVoiceRecognitionActivity() {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,"en-US");
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Silahkan melafalkan kata");
    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
}
```

Gambar 4.20 Proses Identifikasi Pelafalan Pengguna

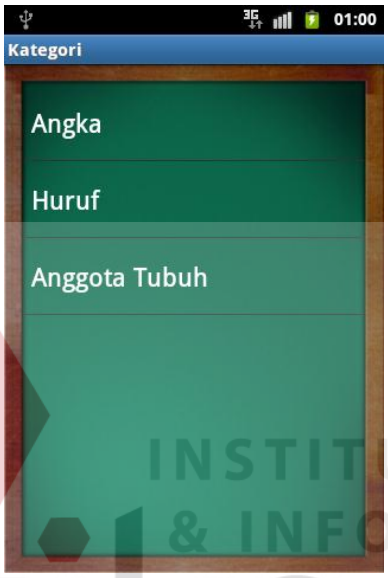
```
//menangani hasil kembali (hasil identifikasi pelafalan)
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == VOICE_RECOGNITION_REQUEST_CODE && resultCode == RESULT_OK) {
        // Fill the list view with the strings the recognizer thought it could have heard

        ArrayList<String> matches = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        mList.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,matches));
    }

    super.onActivityResult(requestCode, resultCode, data);
}
```

Gambar 4.21 Proses Menampilkan Hasil Identifikasi kedalam List View

Pada tampilan ini, terdapat proses pengecekan dukungan *speech input*, proses identifikasi pelafalan pengguna, dan proses menampilkan hasil identifikasi kedalam *list view*. Proses pengecekan dukungan *speech input* dapat dilihat pada gambar 4.19. Pada proses ini, dukungan *speech input* dicek terlebih dahulu, agar dapat melanjutkan pada proses identifikasi pelafalan. Proses identifikasi pelafalan pengguna dapat dilihat pada gambar 4.20.

Pada proses ini, pelafalan pengguna akan diidentifikasi langsung ke *sever google* menggunakan koneksi internet. Proses menampilkan hasil identifikasi kedalam *list view* dapat dilihat pada gambar 4.21. Pada proses ini, hasil identifikasi akan ditampilkan pada *list view*.

4.3.7. Tampilan Daftar Kategori



Gambar 4.22 Tampilan Kategori

```
//mengambil String_name "asal"
asal=getIntent().getStringExtra("asal");
```

```
if(asal.equals("ujian")){
    Intent MItem=new Intent(this,MItem.class);
    startActivity(MItem);
}else{
    Intent GNilai=new Intent(this,GridNilai.class);
    GNilai.putExtra("asal", asal);
    startActivity(GNilai);
}
```

Gambar 4.23 Proses Pengolahan String “asal”

Tampilan Kategori berfungsi untuk menampilkan kategori yang tersedia, dapat dilihat pada gambar 4.22. Pada tampilan ini, hanya terdapat *list view* yang menampilkan kategori standar dan kategori dari *database*. Pada tampilan ini,

terdapat proses pengolahan *string* "asal" yang diterima dari *intent* sebelumnya, dapat dilihat pada gambar 4.23. Pada proses ini, *string* "asal" yang akan digunakan untuk pengecekkan agar dapat melanjutkan ke *intent* berikutnya berdasarkan *value* dari *string* "asal".

4.3.8. Tampilan Item

Gambar 4.24 Tampilan Item

Tampilan *item* berfungsi untuk menampilkan *item* berdasarkan kategori, dapat dilihat pada gambar 4.24. Pada tampilan ini, terdapat *text view* benar yang berfungsi untuk menampilkan nilai benar dalam pelafan, *text view* salah yang berfungsi untuk menampilkan jumlah kesalahan yang dilakukan, *text view* soal yang berfungsi untuk menampilkan soal ke- dari jumlah soal yang ada, *text view* judul/kategori *item* yang berfungsi untuk menampilkan nama kategori, *image view* gambar *item* yang berfungsi untuk menampilkan gambar *item*, *text view* kata dalam bahasa Inggris yang berfungsi untuk menampilkan kata dalam bahasa Inggris, *image view* gambar kata cara ucap yang berfungsi untuk menampilkan kata cara ucap sesuai kamus bahasa Inggris, *text view* kata dalam

bahasa Indonesia yang berfungsi untuk menampilkan kata dalam bahasa Indonesia, dan tombol lafal untuk menerima pelafalan pengguna.

```
//load item berdasarkan kategori
if(kat<0){
    ItemStandar itStdr = new ItemStandar();
    item= itStdr.getItem(kat);
}else{
    ItemDatabase itDb = new ItemDatabase(this, kat);
    item = itDb.getItem();
}
//set jumlah soal
jmlSoal=item[0].length;
soalKe++;

lbSoal.setText(soalKe+"/"+jmlSoal);

//volume media
initVolume=am.getStreamVolume(AudioManager.STREAM_MUSIC);
//item standar mulai
lbJudul.setText(judul);
//cek isi item baru ditampilkan
if (jmlSoal!=0){
    item(item,idx);
}else{
    AlertDialog.Builder b = Alerts.Pesan(this, "Peringatan", "Tidak Memiliki Item");
    b.setPositiveButton("Keluar", new OnClickListener() {
        @Override
        public void onClick(DialogInterface arg0, int arg1) {
            // TODO Auto-generated method stub
            MItem.this.finish();
        }
    });

    AlertDialog keluar=b.create();
    keluar.show();
}
```

```
//item yang ditampilkan
void item(Object[][] str,int i){
    if(kat<0){
        imgItem.setImageResource((Integer) str[0][i]);
        imgKUcap.setImageResource((Integer) str[2][i]);
    }else{
        imgItem.setImageBitmap((Bitmap) str[0][i]);
        imgKUcap.setImageBitmap((Bitmap) str[2][i]);
    }
    lbKIng.setText((CharSequence) str[1][i]);
    lbKInd.setText((CharSequence) str[3][i]);
}
```

Gambar 4.25 Proses Menampilkan Isi Item

Pada tampilan ini, terdapat beberapa proses yaitu proses menampilkan isi *item*, proses menampilkan isi *item* selanjutnya, proses simpan nilai, proses dengar, dan proses lafal. Proses menampilkan isi *item* dapat dilihat

pada gambar 4.25. Pada proses ini, isi *item* diambil dari `ItemStandar()` atau `ItemDatabase()` dan akan ditampilkan. Proses menampilkan isi *item* selanjutnya dapat dilihat pada gambar 4.26. Pada proses ini, isi *item* selanjutnya ditampilkan bila pelafalan dianggap benar. Proses simpan nilai dapat dilihat pada gambar 4.27. Pada proses ini, nilai akan disimpan apabila *item* terakhir telah benar pelafalannya.

```
//item selanjutnya
void itemNext(final Object[][] str){
    if(idx<str[0].length-1){
        idx++;
        item(str,idx);
    }else if(idx==str[0].length-1){
        //kirim nilai ke database dan tampilkan grafik
        simpanNilai();
    }
}
```

Gambar 4.26 Proses Menampilkan Isi Item Selanjutnya

```
void simpanNilai(){
    try{
        int hasilUji=0;
        hasilUji=(100/jmlSoal)*score;

        String format="dd-MMM-yyyy";
        SimpleDateFormat sdf = new SimpleDateFormat(format);
        String tanggal= sdf.format(new Date());

        Nilai nilai=new Nilai(hasilUji, tanggal,kat,idPeng);
        dbHelper.AddNilai(nilai);

        AlertDialog.Builder bldr = Alerts.Pesan(this, "Hasil Ujian Anda", Integer.toString(hasilUji));
        bldr.setPositiveButton("Grafik", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                try{
                    Intent GNilai=new Intent(MItem.this,GridNilai.class);
                    GNilai.putExtra("asal", "pribadi");
                    startActivity(GNilai);
                    MItem.this.finish();
                }catch(Exception e){
                    Alerts.CatchError(MItem.this, e.toString());
                }
            }
        });
```

Gambar 4.27 Proses Simpan Nilai

```
/*menentukan bahasa yang digunakan untuk melafalkan
teks kata bahasa Inggris*/
public void onInit(int status){
    Locale loc = new Locale("US","","");
    if(tts.isLanguageAvailable(loc) >= TextToSpeech.LANG_AVAILABLE){
      tts.setLanguage(loc);
    }
}
```

```
//mendengarkan
void itemDengar(String[][]str){
    if(idx>=0 || idx<str[0].length-1){
        tts.speak(str[4][idx], TextToSpeech.QUEUE_FLUSH, null);
    }
}
```

Gambar 4.28 Proses Dengar

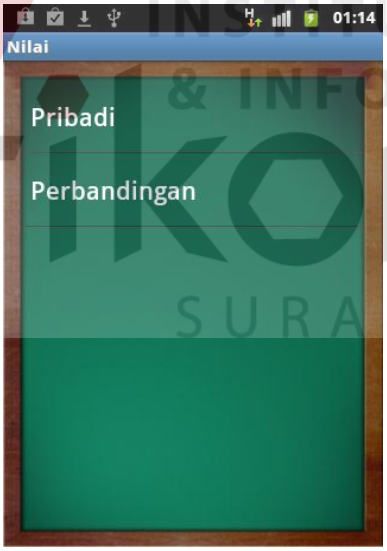
```
public void cekLafal(ArrayList<?> matches){
    int jmlSpeech= matches.size();
    String ktSpeech="";
    String kemungkinan = "\n";
    boolean cekSpeech=false;

    for(int i=0;i<jmlSpeech;i++){
        ktSpeech=matches.get(i).toString();

        kemungkinan+="\n> "+ktSpeech;

        if(((String) item[4][idx]).equalsIgnoreCase(ktSpeech)){
            cekSpeech=true;
            break;
        }
    }

    if(cekSpeech){
        //item selanjutnya
        score++;
        salahKe="";
        kesempatan=2;


        lbSalah.setText(salahKe);
        lbScore.setText("Benar: "+score);
        if(soalKe<jmlSoal){
            soalKe++;
            lbSoal.setText(soalKe+"/"+jmlSoal);
        }
        itemNext(item);
```

Gambar 4.29 Proses Lafal

Proses dengar dapat dilihat pada gambar 4.28. Pada proses ini, fungsi dengar menggunakan *package Text_to_Speech* (tts) untuk melafalkan teks kata dalam bahasa Inggris yang tersimpan didalam *database*. Proses lafal dapat dilihat pada gambar 4.29. Pada proses ini, fungsi lafal menggunakan *package speech input* untuk mengidentifikasi suara pengguna kedalam teks (sebagian proses yang terjadi sama dengan proses pada tampilan latihan).

4.3.9. Tampilan Nilai

Tampilan Nilai berfungsi untuk menampilkan menu nilai, dimana terdapat dua pilihan yaitu nilai pribadi dan nilai perbangingan, dapat dilihat pada gambar 4.30. Pada tampilan ini, kedua pilihan berfungsi sama yaitu mengalihkan ke tampilan daftar kategori.

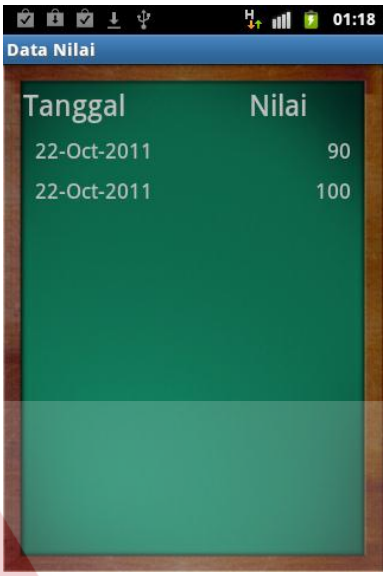


Gambar 4.30 Tampilan Nilai

4.3.10. Tampilan Daftar Nilai Pribadi

Tampilan daftar nilai pribadi berfungsi untuk menampilkan nilai kesalahan pribadi yang disertai tanggal pelaksanaan ujian sesuai kategori tertentu, dapat dilihat pada gambar 4.31. Pada tampilan ini, terdapat *grid view* yang

berfungsi untuk menampilkan total nilai. Pada tampilan ini, terdapat proses menampilkan nilai dari *database* kedalam *grid view*, dapat dilihat pada gambar 4.32.



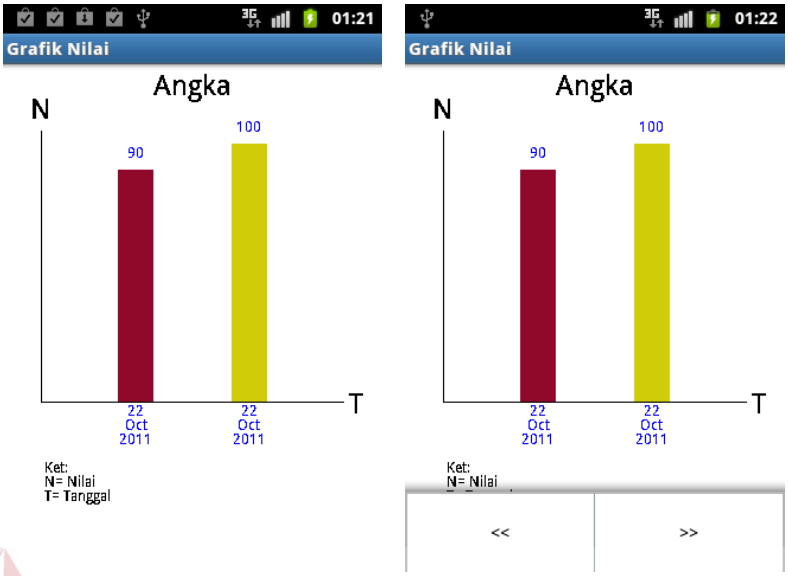
Gambar 4.31 Tampilan Daftar Nilai Pribadi

```
if(asal.equals("pribadi")){
    c= dbHelper.getNilaiBerdasar(idKat, idPeng);
    startManagingCursor(c);

    String[] from= {DatabaseHelper.cNilai,DatabaseHelper.cTgl};
    int[] to= {R.id.cNilai,R.id.cTgl};
    SimpleCursorAdapter sca = new SimpleCursorAdapter(this, R.layout.grid_row_nilai, c, from, to);

    gridNilai.setAdapter(sca);
```

```
//Nilai pribadi pengguna
public Cursor getNilaiBerdasar(int idKat,int idPeng){
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor cur=db.rawQuery("Select "+
                           cIdNilai+" AS _id, "+
                           cTgl+", "+
                           cNilai+
                       " from "+tblNilai+
                       " where "+cIdKatNilai+"="+idKat+" and "+
                                 cIdPengNilai+"="+idPeng,null);
    return cur;
}
```

Gambar 4.32 Proses Menampilkan Nilai dari Database

4.3.11. Tampilan Grafik Nilai Pribadi



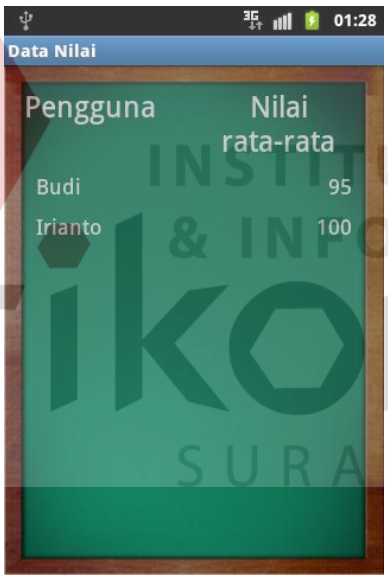
Gambar 4.33 Tampilan Grafik Nilai Pribadi

```
Grafik = new Grafik(this);
setContentView(Grafik);
//set ukuran Layar
Grafik.setMaxTL(getWindowManager().getDefaultDisplay().getHeight());
Grafik.setMaxLL(getWindowManager().getDefaultDisplay().getWidth());
Grafik.setJudul(((SetApp)getApplication()).getNmKategori());

asal=getIntent().getStringExtra("asal");
idKat = ((SetApp)getApplication()).getIdKategori();
idPeng = ((SetApp)getApplication()).getIdPengguna();

sOffset=getIntent().getIntExtra("offset", 0);
for(int i=0;i<nX.length;i++){
    btsPersegi=btsKiri+3;
    nGMax=(btsBawah-btsAtas);
    kiri=kanan+btsPersegi;
    atas=btsAtas+(nGMax-(nX[i]*(nGMax/nMax)));
    kanan=kiri+15;
    bawah=btsBawah;

    //gambar persegi
    paint.setAntiAlias(false);
    if(i%2==0){
        paint.setColor(Color.GREEN);
    }else{
        paint.setColor(Color.RED);
    }
    //persegi=kiri atas kanan bawah
    canvas.drawRect(kiri,atas, kanan, bawah, paint);

    //gambar teks n
    paint.setColor(Color.BLUE);
    paint.setTextSize(fontSize);//ukuran font berpengaruh
    canvas.drawText(Integer.toString(nX[i]), kiri+(15/2), (atas-10), paint);

    //gambar teks tgl
    String tglS[]= tglX[i].split("-");
    for(int j=0;j<tglS.length;j++){
        //mengikuti ukuran font
        canvas.drawText(tglS[j], kiri+(15/2), bawah+(fontSize*(j+1)), paint);
    }
}
```

Gambar 4.34 Proses Mengolah Total Nilai Kedalam Bentuk Grafik

Tampilan Grafik Nilai Pribadi berfungsi untuk menampilkan total nilai pribadi kedalam grafik, dapat dilihat pada gambar 4.33. Pada tampilan ini, menggunakan *view* Grafik yang berfungsi untuk menggambar bentuk persegi serta komponen dari grafik. Pada tampilan ini, terdapat proses mengolah total nilai dari *database* agar dapat ditampilkan kedalam bentuk grafik, dapat dilihat pada gambar 4.34. Pada proses ini, *view* Grafik berasal dari "*class* Grafik" yang telah dibuat secara manual dan terdapat `onDraw(Canvas canvas)` agar dapat mengolah nilai kesalahan kedalam bentuk grafik.

4.3.12. Tampilan Daftar Nilai Perbandingan



Gambar 4.35 Tampilan Daftar Nilai Perbandingan

Tampilan daftar nilai perbandingan berfungsi untuk menampilkan seluruh total nilai rata-rata dari pengguna sesuai kategori tertentu, dapat dilihat pada gambar 4.35. Pada tampilan ini, sama halnya dengan tampilan daftar nilai pribadi, terdapat *grid view* yang berfungsi untuk menampilkan total nilai rata-rata dan nama pengguna. Pada tampilan ini, terdapat proses menampilkan perbandingan nilai antar pengguna, dapat dilihat pada gambar 4.36. Pada proses ini, nilai tiap

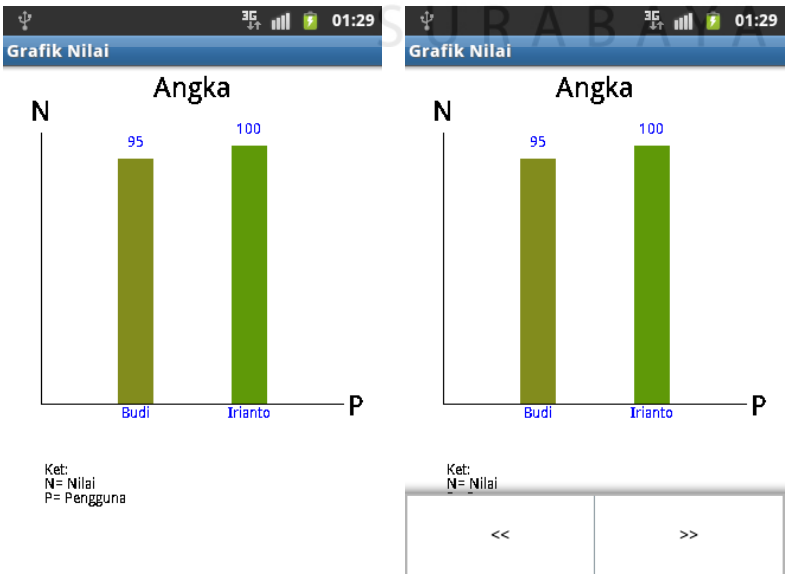
pengguna akan dibuat menjadi nilai rata-rata dan akan ditampilkan berdasarkan kategori tertentu.

```
}else if(asal.equals("perbandingan")){
    lbTgl.setText("Pengguna");
    lbKesalahan.setText("Kesalahan rata-rata");
    c= dbHelper.getNilaiRata(idKat);
    startManagingCursor(c);

    String[] from= {DatabaseHelper.cNmPeng,"nilai_rata"};
    int[] to= {R.id.cTgl,R.id.cNilai};
    SimpleCursorAdapter sca = new SimpleCursorAdapter(this, R.layout.grid_row_nilai, c, from, to);

    gridNilai.setAdapter(sca);
}
```

```
//Nilai rata rata pengguna
public Cursor getNilaiRata(int idKat){
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor cur=db.rawQuery("select "+
                    "n."+cIdNilai+" AS _id, "+
                    "p."+cNmPeng+", "+
                    "sum(n."+cNilai+")/count(n."+cNilai+") AS \"nilai_rata\""+
                " from "+tblNilai+" n, "+tblPeng+" p"+
                " where n."+cIdPengNilai+"=p."+cIdPeng+
                  " and n."+cIdKatNilai+"="+idKat+
                " group by n."+cIdPengNilai,null);

    return cur;
}
```

Gambar 4.36 Proses Menampilkan Total Nilai Rata-Rata

4.3.13. Tampilan Grafik Nilai Perbandingan



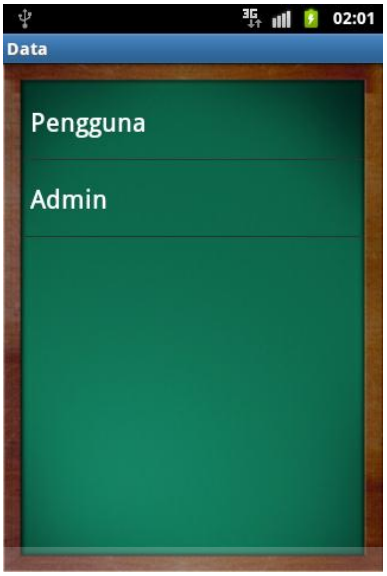
Gambar 4.37 Tampilan Grafik Nilai Perbandingan

Tampilan daftar grafik nilai perbandingan berfungsi untuk menampilkan total nilai rata-rata antar pengguna kedalam bentuk grafik, dapat dilihat pada gambar 4.37. Pada tampilan ini, terdapat *view* Grafik yang berfungsi untuk mengolah total nilai rata-rata kedalam bentuk grafik. Pada tampilan ini, proses yang terjadi sama seperti pengolahan total nilai pripadi kedalam bentuk grafik yaitu pengolahan total_nilai_rata-rata kedalam bentuk grafik, dapat dilihat pada gambar 4.38. Pada proses ini, total nilai rata-rata akan diolah agar dapat ditampilkan dalam bentuk grafik, serta terdapat menu option untuk menangani perpindahan data pribadi/nilai_rata per lima data.

```
}else if(asal.equals("perbandingan")){
    Cursor c = dbHelper.getNilaiRata(idKat, offset,limit);
    startManagingCursor(c);
    if(c!=null){
        if(c.moveToFirst()){
            do{
                n=Integer.toString(c.getInt(c.getColumnIndex("nilai_rata")));
                k=c.getString(c.getColumnIndex(DatabaseHelper.cNmPeng));

                arrNilai.add(new String[] {n,k});
            }while(c.moveToNext());
        }
    }
}
public boolean onOptionsItemSelected(MenuItem item) {
    Intent GrafikNilai =new Intent(this, GrafikNilai.class);
    GrafikNilai.putExtra("asal", asal);
    int btsMin=sOffset-slimit;
    int btsMax= sOffset+slimit;
    switch (item.getItemId()) {
    case 1:
        if(btsMin<0){
            Alerts.Peringatan(this, "Data Awal", "Tidak ada data Lagi");
        }else{
            GrafikNilai.putExtra("offset", btsMin);
            startActivity(GrafikNilai);
            finish();
        }
        break;
    case 2:
        if(btsMax<btsDataMax){
            GrafikNilai.putExtra("offset", btsMax);
            startActivity(GrafikNilai);
            finish();
        }else{
            Alerts.Peringatan(this, "Data Terakhir", "Tidak ada data Lagi");
        }
        break;
    }

    super.onOptionsItemSelected(item);
    return false;
}
```

Gambar 4.38 Proses Mengolah Kesalahan Rata-Rata Kedalam Bentuk Grafik

4.3.14. Tampilan Menu Data



Gambar 4.39 Tampilan Data

Tampilan menu data berfungsi untuk menampilkan pilihan untuk pengaturan data pengguna dan data admin, dapat dilihat pada gambar 4.39. Pada tampilan ini, terdapat *list view* yang berfungsi untuk menampilkan pilihan pengaturan data pengguna dan data admin. Proses yang terjadi pada tampilan ini, yaitu proses pengalihan ke *intent* selanjutnya.

Gambar 4.40 Tampilan Tab Tambah Kategori

4.3.15. Tampilan Pengaturan Kategori

Tampilan pengaturan kategori terdapat dua buat *tab* yaitu *tab* tambah kategori yang berfungsi untuk menambah kategori baru dan *tab* kategori yang berfungsi untuk melihat daftar kategori.

A. Tampilan Tab Tambah Kategori

```
if(namaKat.length()<3){
    Alerts.Peringatan(this,"Peringatan", " Nama Kategori Minimal 3 Huruf ");
}else{

    Kategori kat=new Kategori(namaKat);

    if(dbHelper.AddKategori(kat)!=0){
        //alert klo diklik bener
        AlertDialog.Builder bldr= Alerts.Pesan(this, "Tambah Kategori Baru", "Berhasil");
        bldr.setPositiveButton("Ok",new OnClickListener() {
            public void onClick(DialogInterface arg0, int arg1) {
                // TODO Auto-generated method stub
                tbNmKat.setText("");
                lbJmlKat.setText("Jumlah Kategori : "+String.valueOf(dbHelper.getCountKategori()));

                //Pindah ke tab Kategori
                ((TabActivity)getParent()).getTabHost().setCurrentTab(1);
            }
        });

        AlertDialog diag=bldr.create();
        diag.show();
    }else{
        //alert klo diklik gagal
        Alerts.Peringatan(this, "Tambah Kategori Baru", "Gagal");
        tbNmKat.setText("");
    }
}
```

Gambar 4.41 Proses Tambah Kategori

Tampilan *tab* tambah kategori seperti halnya *tab* tambah pengguna, yang berfungsi untuk menambah kategori baru, dapat dilihat pada gambar 4.40. Pada tampilan ini, proses penyimpanan kategori sama halnya pada *tab* tambah pengguna, dimana setelah menambah kategori baru, maka dialihkan ke *tab* kategori, dapat dilihat pada gambar 4.41.

B. Tampilan Tab Daftar Kategori

Tampilan *tab* daftar kategori pertama kali akan menampilkan kategori yang terdapat didalam *database* kedalam *grid view*, dapat dilihat pada gambar 4.42. Pada tampilan ini, terdapat *grid view* kategori yang berfungsi untuk menampilkan data kategori dan dua tombol yaitu tombol ubah dan hapus.

Gambar 4.42 Tampilan Tab Daftar Kategori

```
void LoadGrid(){
    dbHelper=new DatabaseHelper(this);
    try{
        Cursor c= dbHelper.getIsiKategori();
        startManagingCursor(c);

        String[] from= {DatabaseHelper.cNmKat,"_id"};
        int[] to= {R.id.cNmKat};
        SimpleCursorAdapter sca = new SimpleCursorAdapter(this, R.layout.grid_row_kategori, c, from, to);

        gridKategori.setAdapter(sca);
    }catch(Exception e){
        Alerts.CatchError(TGridKategori.this,e.toString());
    }
}
```

Gambar 4.43 Proses Me-Binding dan Menampilkan pada Grid View

Proses menampilkan data kategori pada *grid view* dapat dilihat pada gambar 4.43, dimana proses tersebut menampilkan dan me-binding data kategori dari database kedalam *grid view*. Proses yang terjadi pada tombol ubah adalah mengalihkan ke tampilan ubah kategori, dapat dilihat pada gambar 4.44. Proses yang terjadi pada tombol hapus adalah menghapus kategori dari *database*, dapat dilihat pada gambar 4.45.

```
public void btUbah_Click(View view){
    try{
        if(nmKat!=null){
            contentUbah();
        }else{
            Alerts.Peringatan(TGridKategori.this, "Peringatan", "Untuk menggunakan fungsi ini\nSilahkan pilih kategori terlebih dahulu");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
private void contentUbah(){
    /*-----------------
     * Memanggil layout add kategori dan digunakan untuk fungsi ubah
     */
    berada=1;//klo 1 ada di ubah
    setContentView(R.layout.add_kategori);

    btAdd=(Button)findViewById(R.id.btAdd);
    lbJmlKat=(TextView)findViewById(R.id.lbJmlKat);
    tbNmKategori=(EditText)findViewById(R.id.tbNmKat);

    tbNmKategori.setText(nmKat.getText().toString());
    btAdd.setText("Ubah Kategori");
    lbJmlKat.setVisibility(8);
}
public int getIdKat() {
    return idKat;
}
```

Gambar 4.44 Proses Pengalihan Tab

```
public void btHapus_Click(View view){
    try{
        if(nmKat!=null){
            AlertDialog.Builder bldr = Alerts.Pesan(this, "Hapus Kategori", "Apakah anda yakin menghapus "+nmKat.getText().toString()+" ?");
            bldr.setPositiveButton("Ya", new OnClickListener() {
                public void onClick(DialogInterface diag, int argl) {
                    // TODO Auto-generated method stub
                    try{
                            Kategori kat=new Kategori(nmKat.getText().toString());
                            kat.setIdKategori(idKat);

                            DatabaseHelper db=new DatabaseHelper(TGridKategori.this);
                            if(db.DeleteKategori(kat)!=0){
                                LoadGrid();
                            }else{
                                Alerts.Peringatan(TGridKategori.this, "Peringatan","Fungsi hapus gagal");
                            }
                    }catch(Exception e){
                        Alerts.CatchError(TGridKategori.this, e.toString());
                    }
                }
            });
            bldr.setNegativeButton("Tidak", null);
            AlertDialog psn = bldr.create();
            psn.show();
        }else{
            Alerts.Peringatan(TGridKategori.this, "Peringatan", "Untuk menggunakan fungsi ini\nSilahkan pilih kategori terlebih dahulu");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.45 Proses Menghapus Kategori

C. Tampilan Tab Ubah Kategori

Gambar 4.46 Tampilan Tab Ubah Kategori

Tampilan *tab* ubah kategori berfungsi untuk mengubah nama kategori yang telah tersimpan pada *database* dan ditampilkan setelah menekan tombol ubah, dapat dilihat pada gambar 4.46. Tampilan ini menggunakan tampilan *tab* tambah kategori tetapi terdapat beberapa modifikasi penyesuaian, yaitu *text view* jumlah kategori dihilangkan dan nama tombol diubah menjadi "ubah kategori", dimana memiliki fungsi yang sama, tetapi pada awal ditampilkan akan menampilkan nama kategori sesuai dengan "id_kategori". Pada tampilan ini, terdapat proses validasi yang sama seperti *tab* tambah kategori. Proses ubah nama kategori dapat dilihat pada gambar 4.47, dimana berfungsi untuk mengubah nama kategori pada *database* jika tidak terjadi kesalahan.

4.3.16. Tampilan Pengaturan Item

Tampilan Pengaturan *item* terdapat dua buah *tab* yaitu *tab* tambah *item* yang berfungsi untuk menambah *item* dan *tab item* yang berfungsi untuk melihat daftar *item* yang telah terdaftar.

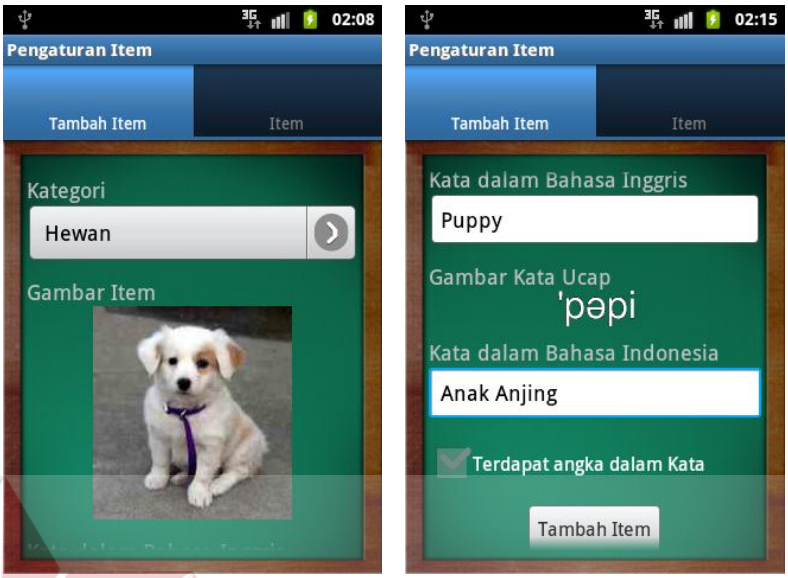
A. Tampilan Tab Tambah Item

```
public void btAddKategori_Click(View view){
    try{
        Kategori kat=new Kategori(tbNmKategori.getText().toString());
        kat.setIdKategori(idKat);

        DatabaseHelper db=new DatabaseHelper(TGridKategori.this);

        if(db.UpdateKategori(kat)!=0){
            AlertDialog.Builder bldr = Alerts.Pesan(this, "Ubah Kategori", "Berhasil");
            bldr.setPositiveButton("Ok", new OnClickListener() {
                public void onClick(DialogInterface diag, int argl) {
                    // TODO Auto-generated method stub
                    try{
                        contentGrid();
                    }catch(Exception e){
                        Alerts.CatchError(TGridKategori.this, e.toString());
                    }
                }
            });
            AlertDialog psn = bldr.create();
            psn.show();
        }else{
            Alerts.Peringatan(this, "Ubah Kategori", "Gagal");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.47 Proses Ubah Kategori

Tampilan *tab* tambah *item* berfungsi untuk menambah *item* baru kedalam *database*, dapat dilihat pada gambar 4.48. Pada tampilan ini, terdapat *spinner* kategori yang berfungsi untuk menampilkan kategori yang telah ditambahkan kedalam *database*, *image view* gambar *item* yang berfungsi untuk menampilkan gambar *item* yang dipilih pada *gallery*, *edit text* kata dalam bahasa Inggris yang berfungsi untuk menerima masukkan berupa kata dalam bahasa Inggris, *image view* gambar kata ucap yang berfungsi untuk menampilkan gambar kata ucap yang dipilih pada *gallery*, *check box* terdapat angka dalam kata yang berfungsi untuk menampilkan *text view* dan *edit text* teks angka Inggris, *edit text* teks angka Inggris yang berfungsi untuk menerima masukkan teks dan angka

dalam bahasa Inggris dan tombol tambah *item* yang berfungsi untuk menambah *item*.



Gambar 4.48 Tampilan Tab Tambah Item

```
dbHelper=new DatabaseHelper(this);

Cursor c = dbHelper.getIsiKategori();
startManagingCursor(c);

String[] from={DatabaseHelper.cNmKat,"_id"};
int[] to={R.id.lbNmKategori};

SimpleCursorAdapter sca=new SimpleCursorAdapter(this,
        R.layout.isi_spinner_kategori,
        c,from,to);

cbKategori.setAdapter(sca);

cbKategori.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View selectedView,
            int position, long id) {
        // TODO Auto-generated method stub
        idKat=(int) id;
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        // TODO Auto-generated method stub

    }
});
```

Gambar 4.49 Proses Menampilkan Kategori kedalam Spinner

```
case PICK_IMAGE: // request code
    if(resultCode == Activity.RESULT_OK) { // jika result ok
        Uri imageUri = data.getData(); // ambil image URI
        if(setGbr==0){
            gbrItem.setImageURI(imageUri); // menggunakannya
            pathGbrItem=getRealPathFromURI(imageUri);
        }else{
            gbrKtUcap.setImageURI(imageUri);
            pathGbrKtUcap=getRealPathFromURI(imageUri);
        }
    }
```

Gambar 4.50 Proses Menampilkan Gambar pada Image View

```
if(ktBing.length()!=0 && ktBind.length()!=0 && teksAngka.length()!=0 && pathGbrItem!=null && pathGbrKtUcap!=null){
    Item itm= new Item(bmGbrItem,ktBing,bmGbrKtUcap,ktBind,teksAngka,idKatItem);
}else{
    //peringatan isian
    String peringatan="";
    if(ktBing.length()==0){
        peringatan+="> Kata Bahasa Inggris Tidak Boleh Kosong\n";
    }
    if(ktBind.length()==0){
        peringatan+="> Kata Bahasa Indonesia Tidak Boleh Kosong\n";
    }
    if(teksAngka.length()==0){
        peringatan+="> Teks Angka Inggris Tidak Boleh Kosong\n";
    }
    if(pathGbrItem==null){
        peringatan+="> Harus Memilih Gambar Item\n";
    }
    if(pathGbrKtUcap==null){
        peringatan+="> Harus Memilih Gambar Kata Ucap\n";
    }
    Alerts.Peringatan(this, "Peringatan", peringatan);
}
```

Gambar 4.51 Proses Validasi Masukkan

Pada tampilan ini, terdapat proses menampilkan kategori kedalam *spinner*, proses menampilkan gambar pada *image view* setelah dipilih, proses validasi masukkan, dan proses tambah *item*. Proses menampilkan kategori kedalam *spinner* dapat dilihat pada gambar 4.49. Pada proses ini, data kategori akan di*binding* dan ditampilkan pada *spinner.* Proses menampilkan gambar pada *image view* dapat dilihat pada gambar 4.50. Pada proses ini, *image view*

akan menampilkan hasil pilih dari *intent* selanjutnya (menerima hasil balik). Proses validasi masukkan dapat dilihat pada gambar 4.51. Pada proses ini, tiap masukkan dari admin akan divalidasi terlebih dahulu. Proses tambah *item* dapat dilihat pada gambar 4.52. Pada proses ini, hasil masukkan admin setelah divalidasi akan ditambahkan kedalam *database*.

```
Item itm= new Item(bmGbrItem,ktBing,bmGbrKtUcap,ktBind,teksAngka,idKatItem);

if(dbHelper.AddItem(itm)!=0){
    //alert klo diklik bener
    AlertDialog.Builder bldr= Alerts.Pesan(this, "Tambah Item Baru", "Berhasil");
    bldr.setPositiveButton("Ok",new OnClickListener() {
        public void onClick(DialogInterface arg0, int arg1) {
            //Pindah ke tab Item
            ((TabActivity)getParent()).getTabHost().setCurrentTab(1);
        }
    });

    AlertDialog diag=bldr.create();
    diag.show();
}else{
    //alert klo diklik gagal
    Alerts.Peringatan(this, "Tambah Item Baru", "Gagal");
}
```

Gambar 4.52 Proses Tambah Item

B. Tampilan Pencarian Gambar

Tampilan pencarian gambar berfungsi untuk menampilkan gambar-gambar yang terdapat pada *sdcard*/kartu memori, yang akan dipilih sebagai gambar *item* atau gambar kata ucap, dapat dilihat pada gambar 4.53. Pada tampilan ini, android telah menyediakan fungsi untuk mengakses aplikasi *gallery*.

Pada tampilan ini, terdapat beberapa proses yaitu proses menampilkan gambar dengan menggunakan fungsi `Intent.ACTION_PICK` yang dapat mengakses aplikasi *gallery* yang telah tersedia pada sistem operasi Android dapat dilihat pada gambar 4.54. Pada proses ini, akan menampilkan aplikasi *gallery* dan

gambar yang telah dipilih akan ditampilkan pada *image view* dengan menggunakan hasil kembali berupa alamat content/`Uri`.

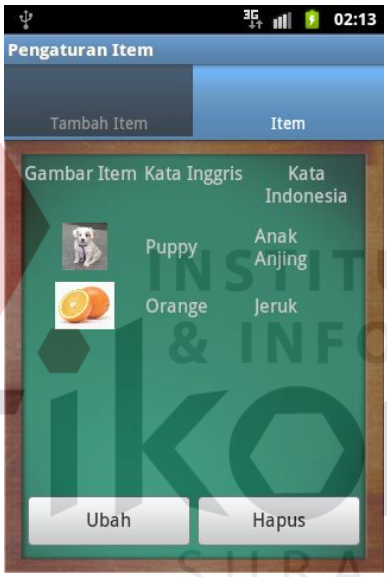
Gambar 4.53 Tampilan Pencarian Gambar

```
public void gbrItem_Click(View view){
    setGbr=0;
    startActivityForResult(new Intent(Intent.ACTION_PICK,
            android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI),
            PICK_IMAGE);
}

public void gbrKtUcap_Click(View view){
    setGbr=1;
    startActivityForResult(new Intent(Intent.ACTION_PICK,
            android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI),
            PICK_IMAGE);
}
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch(requestCode) {
        case PICK_IMAGE: // request code
            if(resultCode == Activity.RESULT_OK) { // jika result ok
                Uri imageUri = data.getData(); // ambil image URI
                if(setGbr==0){
                    gbrItem.setImageURI(imageUri); // menggunakannya
                    pathGbrItem=getRealPathFromURI(imageUri);
                }else{
                    gbrKtUcap.setImageURI(imageUri);
                    pathGbrKtUcap=getRealPathFromURI(imageUri);
                }
            }
    }
}
```

Gambar 4.54 Proses Menampilkan Gallery dan Menggunakan Uri pada Image View

C. Tampilan Tab Daftar Item

Tampilan *tab* daftar *item* pertama kali akan menampilkan daftar *item* yang telah ditambahkan kedalam *database*, dapat dilihat pada gambar 4.55. Pada tampilan ini, terdapat *grid view* yang berfungsi untuk menampilkan gambar *item*, kata dalam bahasa Inggris dan kata dalam bahasa Indonesia untuk mewakili *item* yang telah tersimpan didalam *database*, tombol ubah yang berfungsi mengalihkan dari *tab* daftar *item* ke *tab* ubah *item*, tombol hapus yang berfungsi untuk menghapus *item*.



Gambar 4.55 Tampilan Tab Daftar Item

Pada tampilan ini, terdapat beberapa proses yaitu proses menampilkan gambar *item*, kata dalam bahasa Inggris dan Indonesia pada *grid view* dapat dilihat pada gambar 4.56. Pada proses ini, *grid view* menampilkan gambar *item*, kata dalam bahasa Inggris dan Indonesia yang berasal dari *database*. Proses pengalihan *tab* dapat dilihat pada gambar 4.57. Pada proses ini, *tab* daftar *item* akan dialihkan ke *tab* ubah *item*, tetapi langsung didalam *class* tersebut tanpa menggunakan *intent*. Proses menghapus *item* dapat dilihat pada gambar 4.58,

dimana proses ini melakukan penghapusan data *item* pada *database* jika tidak terjadi kesalahan.

```
private void LoadGrid() {
    dbHelper=new DatabaseHelper(this);
    try{
        Cursor c=dbHelper.getIsiItem();
        startManagingCursor(c);

        String[] from={DatabaseHelper.cGbrItem,DatabaseHelper.cKtIng,DatabaseHelper.cKtInd,"_id"};
        int[] to={R.id.cGbrItem,R.id.cKtIng,R.id.cKtInd};
        SimpleCursorAdapter sca = new SimpleCursorAdapter(this, R.layout.grid_row_item, c, from, to);
        /*-------------------------------------------------------
         * viewBinder membantu untuk dapat menggunakan isi cursor
         *pada gambar/widget yang membutuhkan perlakuan khusus
         */
        sca.setViewBinder(new ViewBinderItem());

        gridItem.setAdapter(sca);
    }catch(Exception e){
        Alerts.CatchError(this,e.toString());
    }
}
```

Gambar 4.56 Proses Menampilkan pada Grid View

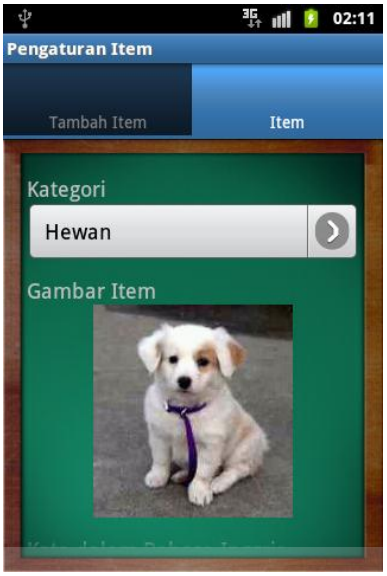
```
public void btUbah_Click(View view){
    try{
        if(ktIngG!=null){
            contentUbah();
        }else{
            Alerts.Peringatan(this, "Peringatan", "Untuk menggunakan fungsi ini\nSilahkan pilih Item terlebih dahulu");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.57 Proses Mengalihkan Tab

```
//fungsi tombol hapus
public void btHapus_Click(View view){
    try{
        if(ktIngG!=null){
            AlertDialog.Builder bldr = Alerts.Pesan(this, "Hapus Item", "Apakah anda yakin menghapus "+ktIngG.getText().toString()+" ?");
            bldr.setPositiveButton("Ya", new OnClickListener() {
                public void onClick(DialogInterface diag, int argl) {
                    // TODO Auto-generated method stub
                    try{
                            Item itm=new Item();
                            itm.setIdItem(idItem);

                            DatabaseHelper db=new DatabaseHelper(TGridItem.this);

                            if(db.DeleteItem(itm)!=0){
                                LoadGrid();
                            }else{
                                Alerts.Peringatan(TGridItem.this, "Peringatan","Fungsi hapus gagal");
                            }
                    }catch(Exception e){
                        Alerts.CatchError(TGridItem.this, e.toString());
                    }
                }
            });
            bldr.setNegativeButton("Tidak", null);
            AlertDialog psn = bldr.create();
            psn.show();
        }else{
            Alerts.Peringatan(TGridItem.this, "Peringatan", "Untuk menggunakan fungsi ini\nSilahkan pilih item terlebih dahulu");
        }
    }catch(Exception e){
        Alerts.CatchError(this, e.toString());
    }
}
```

Gambar 4.58 Proses Menghapus Item

D. Tampilan Tab Ubah Item



Gambar 4.59 Tampilan Tab Ubah Item

```
if(ktBing.length()!=0 && ktBind.length()!=0 && teksAngka.length()!=0){
    boolean gItem=false;
    boolean gKUcap=false;
    if(pathGbrItem!=null){
        gItem=true;
    }
    if(pathGbrKtUcap!=null){
        gKUcap=true;
    }
    Item itm= new Item(bmGbrItem,ktBing,bmGbrKtUcap,ktBind,teksAngka,idKatItem);
    itm.setIdItem(idItem);

    DatabaseHelper db=new DatabaseHelper(this);

    if(db.UpdateItem(itm,gItem,gKUcap)!=0){
        AlertDialog.Builder bldr = Alerts.Pesan(this, "Ubah Item", "Berhasil");
        bldr.setPositiveButton("Ok", new OnClickListener() {
            public void onClick(DialogInterface diag, int argl) {
                // TODO Auto-generated method stub
                try{
                    contentGrid();
                }catch(Exception e){
                    Alerts.CatchError(TGridItem.this, e.toString());
                }
            }
        });
        AlertDialog psn = bldr.create();
        psn.show();
    }else{
        Alerts.Peringatan(this, "Ubah Item", "Gagal");
    }
```

Gambar 4.60 Proses Ubah Item

Tampilan *tab* ubah *item* berfungsi untuk mengubah isi dari *item* yang ada didalam *database* dan ditampilkan setelah menekan tombol ubah, dapat dilihat pada gambar 4.59. Tampilan ini menggunakan tampilan *tab* tambah *item* tetapi

terdapat beberapa modifikasi penyesuaian, yaitu nama tombol diubah menjadi "ubah *item*", maka memiliki fungsi yang sama, tetapi pada awal ditampilkan akan menampilkan isi *item* sesuai dengan "id_item". Pada tampilan ini, terdapat proses validasi yang sama seperti *tab* tambah *item*. Proses ubah *item* dapat dilihat pada gambar 4.60, dimana berfungsi untuk mengubah isi *item* pada *database* jika tidak terjadi kesalahan.

4.4. Evaluasi Sistem

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 akan dilakukan dengan menggunakan *emulator* Android (kecuali pada uji coba penggunaan *speech input* dimana harus terkoneksi dengan *server* google dan menggunakan *device* yang telah terdaftar, pengujian dilakukan pada *handphone* Android 2.2 Froyo).

4.4.1. Uji Coba Fungsi Aplikasi

Pengujian ini, dilakukan untuk mengetahui apakah fungsi-fungsi yang ada pada aplikasi berjalan dengan baik atau tidak. Adapun fungsi-fungsi yang akan diujikan adalah:

A. Fungsi Menampilkan Isi Item Standar dan Database

Uji coba ini, bertujuan untuk mengetahui apakah fungsi menampilkan isi *item* standar dan *database* dapat dilakukan melalui aplikasi. Pada uji coba, pertama dilakukan pemilihan kategori standar yang berisi *item* standar dan kategori yang tersimpan pada *database* yang berisi *item* yang tersimpan pada

database. Aplikasi kemudian akan mengambil data dan akan menampilkannya. Proses pengambilan dan menampilkan data *item* standar dan *database* berdasarkan kategori yang dipilih, dan apabila kategori tersebut tidak memiliki *item* maka tidak dapat menampilkan isi *item* serta akan memberi peringatan. Data kategori yang digunakan agar dapat menampilkan *item* berdasar kategori dapat dilihat pada tabel 4.1. Hasil uji coba dapat dilihat pada tabel 4.2.

Tabel 4.1 Kategori

Nama Kategori	Keterangan
Angka	kategori standar
Huruf	kategori standar
Bagian Tubuh	kategori standar
Hewan	kategori *database*
Buah	kategori *database*

Tabel 4.2 Hasil Uji Coba Menampilkan Isi Item Standar dan Database

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
1	Menampilkan isi *item* standar (data ada)	Memilih kategori angka	Tampil isi *item standar*	Sesuai
2	Menampilkan isi *item* standar (data tidak ada)	Memilih kategori huruf	Muncul pesan "Tidak Memiliki *item"*	Sesuai
3	Menampilkan isi *item* yang tersimpan pada *database* (data ada)	Memilih kategori hewan	Tampil isi *item* yang tersimpan pada *database*	Sesuai
4	Menampilkan isi *item* yang tersimpan pada *database* (data tidak ada)	Memilih kategori buah	Muncul pesan "Tidak Memiliki *item"*	Sesuai

B. Fungsi Dengar

Uji coba ini, bertujuan untuk mengetahui apakah fungsi dengar atau meminta pelafalan sistem dapat dilakukan melalui aplikasi. Pada uji coba, pertama dilakukan permintaan pelafalan sistem dengan menekan tombol dengar.

Aplikasi akan mengecek ketersediaan bahasa yang telah ditentukan, apabila mendukung maka sistem akan melafalkan teks yang telah tersedia. Hasil uji coba dapat dilihat pada tabel 4.3.

Tabel 4.3 Hasil Uji Coba Fungsi Dengar / Meminta Pelafalan Sistem

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
5	Meminta pelafalan sistem	Menekan gambar item	Sistem melafalkan teks	Sesuai

C. Fungsi Lafal

Uji coba ini, bertujuan untuk mengetahui apakah fungsi lafal atau meminta pencocokan pelafalan pengguna serta penilaian dapat dilakukan melalui aplikasi. Pada uji coba, pertama dilakukan pengecekkan dukungan *voice recognition*, setelah itu sesuai pengaturan *speech input* diminta pelafalan pengguna. Aplikasi akan menampilkan tampilan siap menerima pelafalan pengguna dan setelah pengguna melafalkan, maka sistem akan memproses suara yang dihasilkan dengan mengirim pada *server* google untuk diidentifikasi/dicocokan. Hasil pencocokan akan dikembalikan berupa beberapa kemungkinan teks, dimana hasil tersebut akan dicocokkan lagi dengan kata bahasa Inggris. Aplikasi akan menambahkan nilai apabila kata bahasa Inggris dengan hasil identifikasi pelafalan cocok setelah itu melanjutkan ke *item* selanjutnya. Pengguna diberi tiga kali kesempatan untuk mencocokkan pelafalan, dan jika kesempatan habis maka akan melanjutkan ke *item* selanjutnya. Aplikasi akan menyimpan nilai total kedalam *database* bila semua *item* telah selesai dilaksanakan. Hasil uji coba dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil Uji Coba Fungsi Lafal dan Penilaian

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
6	Mengecek dukungan *voice recognition* (mendukung)	Masuk *form* ujian *item* / *form* latihan	Tombol "Lafal" aktif	Sesuai
7	Mengecek dukungan *voice recognition* (tidak mendukung)	Masuk *form* ujian *item* / *form* latihan	Tombol "Lafal" tidak aktif	Sesuai
8	Mencocokkan pelafalan pada *form* latihan (suara dikenali)	Menekan tombol "Lafal", lalu pengguna melafalkan kata Inggris	Menampilkan daftar kata hasil identifikasi pada *list view*	Sesuai
9	Mencocokkan pelafalan pada *form* latihan (suara tidak dikenali)	Menekan tombol "Lafal", lalu pengguna melafalkan kata Inggris	Muncul pesan "Pelafanan Tidak dikenali"	Sesuai
10	Mencocokkan pelafalan pada *form* ujian *item* (cocok)	Menekan tombol "Lafal", lalu pengguna melafalkan kata Inggris	Menambah nilai benar, lalu lanjut *item* berikutnya.	Sesuai
11	Mencocokkan pelafalan pada *form* ujian *item* (tidak cocok)	Menekan tombol "Lafal", lalu pengguna melafalkan kata Inggris	Muncul pesan "Salah", lalu menambah tanda salah	Sesuai
12	Lanjut *item* berikut bila melakukan kesalahan	Pengguna melafalkan kata Inggris dengan salah sebanyak tiga kali	Muncul pesan "Salah", lalu lanjut *item* berikutnya	Sesuai
13	Menyimpan total nilai akhir (tersimpan)	Melafalkan *item* terakhir dengan benar	Muncul pesan "Nilai dan menampilkan tombol *"Grid* Nilai*"* dan "Kategori"	Sesuai
14	Menyimpan total nilai akhir (tidak tersimpan)	Melafalkan *item* terakhir dengan benar	Muncul pesan "Simpan Nilai Gagal"	Sesuai

D. Fungsi Pengolahan Nilai kedalam Grafik

Uji coba ini, bertujuan untuk mengetahui apakah fungsi pengolahan nilai kedalam grafik dapat dilakukan melalui aplikasi. Pada uji coba, pertama dilakukan

pengambilan nilai maximal (grafik nilai pribadi) atau nilai rata-rata maksimal (grafik nilai perbandingan), setelah itu mengambil nilai per lima data. Proses selanjutnya adalah mengolah kedalam grafik, sesuai nilai-nilai yang telah didapat (grafik akan menyesuaikan dengan nilai) dan proses pindah halaman. Hasil uji coba dapat dilihat pada tabel 4.5.

Tabel 4.5 Hasil Uji Coba Fungsi Pengolahan Nilai kedalam Grafik

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
15	Menampilkan nilai dalam bentuk grafik	Menekan menu *option* "Grafik Nilai"	Menampilkan *form* grafik	Sesuai
16	Pindah *page* berikutnya	Menekan menu *option* ">>"	Menampilkan grafik nilai	Sesuai
17	Pindah *page* berikutnya (data terakhir)	Menekan menu *option* ">>"	Muncul pesan "Data Terakhir, tidak ada data lagi"	Sesuai
18	Pindah *page* sebelumnya	Menekan menu *option* "<<"	Menampilkan grafik nilai	Sesuai
19	Pindah *page* sebelumnya (data awal)	Menekan menu *option* "<<"	Muncul pesan "Data Awal, tidak ada data lagi"	Sesuai

E. Fungsi Tampilkan Daftar Data

Uji coba ini, bertujuan untuk mengetahui apakah fungsi tampilkan daftar pengguna, kategori dan *item* dapat dilakukan melalui aplikasi. Pada uji coba, pertama dilakukan pada saat awal penggunaan aplikasi menampilkan daftar pengguna dan pemilihan *tab* daftar data pada saat berada pada pengaturan data. Proses menampilkan daftar data, diambil dari *database* yang ditampilkan pada *list view* dan *grid view*. Hasil uji coba dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil Uji Coba Tampilan Daftar Data

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
20	Menampilkan daftar pengguna pada *listview* (berhasil)	Masuk *form* daftar pengguna	Data pengguna tampil pada *listview*	Sesuai
21	Menampilkan daftar pengguna pada *listview* (gagal)	Masuk *form* daftar pengguna	Muncul peringatan error	Sesuai
22	Menampilkan daftar pengguna pada *gridview* (berhasil)	Masuk *form* pengaturan pengguna pada tab pengguna	Data pengguna tampil pada *gridview*	Sesuai
23	Menampilkan daftar pengguna pada *gridview* (gagal)	Masuk *form* pengaturan pengguna pada tab pengguna	Muncul peringatan error	Sesuai
24	Menampilkan daftar kategori pada *listview* (berhasil)	Masuk *form* daftar kategori	Data kategori tampil pada *listview*	Sesuai
25	Menampilkan daftar kategori pada *listview* (gagal)	Masuk *form* daftar kategori	Muncul peringatan error	Sesuai
26	Menampilkan daftar kategori pada *gridview* (berhasil)	Masuk *form* pengaturan kategori pada tab kategori	Data kategori tampil pada *gridview*	Sesuai
27	Menampilkan daftar kategori pada *gridview* (gagal)	Masuk *form* pengaturan kategori pada tab kategori	Muncul peringatan error	Sesuai
28	Menampilkan daftar *item*, berupa gambar, kata bahasa Inggris, kata bahasa Indonesia pada *gridview* (berhasil)	Masuk *form* pengaturan *item* pada tab *item*	Data *item* tampil pada *gridview*	Sesuai
29	Menampilkan daftar *item*, berupa gambar, kata bahasa Inggris, kata bahasa Indonesia pada *gridview* (gagal)	Masuk *form* pengaturan *item* pada tab *item*	Muncul peringatan error	Sesuai
30	Menggunakan tombol ubah pada *gridview*(berhasil)	Memilih salah satu data, lalu menekan tombol “Ubah”	Mengalihkan pada *tab* ubah	Sesuai
31	Menggunakan tombol ubah pada *gridview* (gagal)	Memilih salah satu data, lalu menekan tombol “Ubah”	Muncul peringatan untuk memilih data terlebih dulu	Sesuai

Test Case ID	Tujuan	*Input*	*Output* yang diharapkan	Status
32	Menggunakan tombol hapus pada *gridview* (berhasil)	Memilih salah satu data, lalu menekan tombol “Hapus”	Menghapus data	Sesuai
33	Menggunakan tombol hapus pada *gridview* (gagal)	Memilih salah satu data, lalu menekan tombol “Hapus”	Muncul peringatan untuk memilih data terlebih dulu	Sesuai

F. Fungsi Tambah Data

Uji coba ini, bertujuan untuk mengetahui apakah fungsi tambah data dapat dilakukan melalui aplikasi. Pada uji coba, pertama dilakukan memasukkan isian data, setelah itu divalidasi, jika validasi benar maka akan tersimpan kedalam *database*. Aplikasi akan memberi peringatan bila terjadi kegagalan dalam menambah data baru. Hasil uji coba dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Uji Coba Tambah Data

Test Case ID	Tujuan	*Input*	*Output* yang diharapkan	Status
34	Menambah data pengguna (berhasil)	Memasukkan nama pengguna =”Irianto”, lalu tekan tombol “Tambah Pengguna”	Muncul pesan “Berhasil Menambah Pengguna”	Sesuai
35	Menambah data pengguna (gagal)	Memasukkan nama pengguna =”Ir” , lalu tekan tombol “Tambah Pengguna”	Muncul peringatan “nama pengguna tidak boleh kurang dari tiga huruf”	Sesuai
36	Menambah data kategori (berhasil)	Memasukkan nama kategori =”Hewan” , lalu tekan tombol “Tambah Kategori”	Muncul pesan “Berhasil Menambahkan Kategori”	Sesuai
37	Menambah data kategori (gagal)	Memasukkan nama kategori =”Hw” , lalu tekan tombol “Tambah Kategori”	Muncul peringatan “nama kategori tidak boleh kurang dari tiga huruf”	Sesuai
38	Menampilkan gambar pada *image view*	Memilih gambar pada *gallery*	Gambar tampil pada *image view*	Sesuai

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
39	Menambah data *item* (berhasil)	Memilih nama kategori=”Hewan”, lalu memasukkan kata Inggris=”Fish”, kata indonesia =”Ikan”, gambar *item*=”gambar ikan”, gambar kata ucap=”gambar kata ucap ikan”, lalu tekan tombol “Tambah *Item*”	Muncul pesan “Berhasil Menambah *Item”*	Sesuai
40	Menambah data *item* (gagal)	Memilih nama kategori=”Hewan”, lalu memasukkan kata Inggris=”Fish”, kata indonesia =”Ikan” , lalu tekan tombol “Tambah *Item*”	Muncul peringatan “Gambar *item*, Gambar kata ucap harus dipilih”	Sesuai

G. Fungsi Ubah Data

Uji coba ini, bertujuan untuk mengetahui apakah fungsi ubah data telah berjalan dengan benar atau tidak. Hasil uji coba fungsi ubah data dapat dilihat pada tabel 4.8.

Tabel 4.8 Hasil Uji Coba Ubah Data

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
41	Mengubah data pengguna (berhasil)	Memasukkan nama pengguna =”Pratama”, lalu tekan tombol “Ubah Pengguna”	Muncul pesan “Berhasil Mengubah Pengguna”	Sesuai
42	Mengubah data pengguna (gagal)	Memasukkan nama pengguna =”Pr”, lalu tekan tombol “Ubah Pengguna”	Muncul peringatan “Nama pengguna tidak boleh kurang dari tiga huruf”	Sesuai

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
43	Mengubah data kategori (berhasil)	Memasukkan nama kategori =”Fauna”, lalu tekan tombol “Ubah Kategori”	Muncul pesan “Berhasil Mengubah Kategori”	Sesuai
44	Mengubah data kategori (gagal)	Memasukkan nama kategori =”Fa”, lalu tekan tombol “Ubah Kategori”	Muncul peringatan “Nama kategori tidak boleh kurang dari tiga huruf”	Sesuai
45	Mengubah data *item* (berhasil)	Memasukkan kata Inggris =”Dolphin”, kata indonesia = ”Lumba-lumba” , lalu tekan tombol “Ubah *Item*”	Muncul pesan “Berhasil Mengubah *Item*	Sesuai
46	Mengubah data *item* (gagal)	Memasukkan kata Inggris =””, kata indonesia = ””, lalu tekan tombol “Ubah *Item*”	Muncul peringatan “Kata Inggris tidak boleh kosong, Kata Indonesia tidak boleh kosong”	Sesuai

H. Fungsi Hapus Data

Uji coba ini, bertujuan untuk mengetahui apakah fungsi hapus data telah berjalan dengan benar atau tidak. Hasil uji coba hapus data dapat dilihat pada tabel 4.9.

Tabel 4.9 Hasil Uji Coba Hapus Data

***Test Case* ID**	**Tujuan**	***Input***	***Output* yang diharapkan**	**Status**
47	Menghapus data pengguna (berhasil)	Memilih data pengguna, lalu menekan tombol hapus, dan mengkonfirmasi pesan hapus	Pada *gridview* tidak menampilkan data terhapus, data pengguna terhapus dan nilai ikut terhapus	Sesuai
48	Menghapus data pengguna (gagal)	Memilih data pengguna, lalu menekan tombol hapus, dan mengkonfirmasi pesan hapus	Muncul peringatan “Gagal menghapus data”, pada *gridview* masih menampilkan data terhapus	Sesuai

Test Case ID	Tujuan	*Input*	*Output* yang diharapkan	Status
49	Menghapus data kategori (berhasil)	Memilih data kategori, lalu menekan tombol hapus, dan mengkonfirmasi pesan hapus	Pada *gridview* tidak menampilkan data terhapus, data kategori terhapus dan nilai ikut terhapus	Sesuai
50	Menghapus data kategori (gagal)	Memilih data kategori, lalu menekan tombol hapus, dan mengkonfirmasi pesan hapus	Muncul peringatan "Gagal menghapus data", pada *gridview* masih menampilkan data terhapus	Sesuai
51	Menghapus data *item* (berhasil)	Memilih data *item*, lalu menekan tombol hapus, dan mengkonfirmasi pesan hapus	Pada *gridview* tidak menampilkan data terhapus, data item terhapus	Sesuai
52	Menghapus data *item* (gagal)	Memilih data *item*, lalu menekan tombol hapus, dan mengkonfirmasi pesan hapus	Muncul peringatan "Gagal menghapus data", pada *gridview* masih menampilkan data terhapus	Sesuai

4.4.2. Uji Coba Aplikasi Kepada Pengguna

Pada proses uji coba ini, pengguna diharuskan mencoba Aplikasi Belajar Melafalkan Bahasa Inggris hingga menghasilkan nilai akhir. Uji coba ini dilakukan dengan cara melakukan *survey* langsung kepada 30 orang pengguna termasuk seorang ahli dengan cara mengisi kuesioner pada lampiran Hal-2, dimana karakteristik pengguna dapat dilihat pada tabel 4.10.

Tabel 4.10 Karakteristik Pengguna

No.	Nama Pengguna	Umur (th)	Pekerjaan	Total Nilai Akhir
1	I Gede Bayu Artha Yasa	11	5 SD	100
2	Fitria Rahmawati	13	2 SMP	80
3	Jefta	13	1 SMP	80
4	Riqi	14	2 SMP	90
5	Ni Nyoman Sri Wira W.	14	3 SMP	80
6	Yermias A.	20	Mahasiswa S1 SI	90

No.	Nama Pengguna	Umur (th)	Pekerjaan	Total Nilai Akhir
7	Badar Yasifun Ali	20	Mahasiswa S1 SI	90
8	Rio Andreanto	21	Mahasiswa S1 SI	100
9	Arief F. Wicaksono	21	Mahasiswa S1 SI	70
10	Adrian Chandra	21	Mahasiswa S1 SI	70
11	Ferry	21	Mahasiswa S1 SI	60
12	Nuri Estatika Herga Putri	22	Mahasiswa S1 SI	80
13	Nur Arif F.	22	Mahasiswa S1 SI	80
14	Elvin Al-Mutaqin	22	Mahasiswa S1 SI	80
15	Erma	22	Mahasiswa S1 SI	70
16	Hazar Tyan Pratiwi	22	Mahasiswa S1 SI	60
17	Rian	22	-	70
18	Faizal A.	23	Mahasiswa S1 SI	90
19	Arko A.	23	Mahasiswa S1 SI	90
20	Dion Praisa	23	Mahasiswa S1 SI	90
21	Nita Puspita	23	Mahasiswa S1 SI	90
22	Yendi R.	23	Mahasiswa S1 SI	90
23	Dyah Eka Purnamasari	23	Mahasiswa S1 SI	90
24	Ivana	23	Mahasiswa S1 SI	90
25	Eka Bayu E.	23	Mahasiswa S1 SI	80
26	Ignatius Hadi P.	23	Mahasiswa S1 SI	100
27	Meli	23	Guru Les Bahasa Inggris	100
28	Angriani Angkie	24	Mahasiswa S1 SI	100
29	Bu Rianto	45	Ibu Rumah Tangga	70
30	Haryati	50	Ibu Rumah Tangga	60

Hasil yang ditunjukkan pada tabel 4.10 merupakan total nilai akhir dari penggunaan aplikasi yang bertujuan mengetahui kemampuan pelafalan bahasa Inggris pengguna.

Tabel 4.11 Detail Uji Coba Item Kategori Angka

Angka	Benar Melafalkan tetapi dianggap Salah	Salah	Tidak dikenali	Dikenali sebagai
1	-	3	-	run, why, ron,lion, what
2	-	11	-	do, dove, doe, toe, tow
3	-	-	-	great, free
4	-	13	-	phone, father, form, full
5	-	-	-	fight, fire
6	-	1	-	cheats, sixth, pics
7	-	14	-	saturn, event, heaven
8	-	-	-	aids, age, ache, eggs
9	-	1	-	mine,nines, ninth, mind
10	-	8	-	zen, then, send, ben
total	0	51	0	

Adapun detail uji coba pelafalan oleh pengguna yang akan dicocokkan pada sistem dapat dilihat pada tabel 4.11. Pada tabel ini, kategori yang dicoba adalah kategori angka yang memiliki *item* angka 1-10 dan dilakukan oleh 30 orang pengguna termasuk seorang ahli. Terdapat tiga kolom dimana pelafalan pengguna benar tetapi disalahkan, pelafalan salah, dan pelafalan tidak dikenali.

Hasil kesalahan yang dinilai oleh aplikasi adalah kesalahan pelafalan yang dilakukan sebanyak tiga kali. Dari tabel 4.11 didapat dari 30 orang pengguna telah dianggap melakukan kesalahan pelafalan sebanyak 51, dan banyak pelafalan yang salah terjadi pada item angka dua, empat dan tujuh.

Rekapitulasi kuesioner yang telah diisi oleh 30 orang pengguna termasuk seorang ahli dapat dilihat pada tabel 4.12. Pada tabel tersebut, menjelaskan tentang hasil perhitungan penyataan pengguna terhadap Aplikasi *Mobile* Belajar Melafalkan Bahasa Inggris.

Tabel 4.12 Rekapitulasi Kuesioner

Pertanyaan No.		**Penilaian**					**Σ**	**$\bar{x}$**	**Nilai Akhir**
		1	**2**	**3**	**4**	**5**			
Tampilan									
B	1.	0	0	6	22	2	116	3,8	4
	2.	0	0	1	21	8	127	4,2	
	3.	0	0	10	20	0	110	3,6	
	4.	0	0	11	18	1	110	3,6	
	5.	0	0	2	17	11	129	4,3	
	6.	0	0	3	11	16	133	4,4	
	7.	0	0	3	10	17	134	4,4	
Navigasi									
C	1.	0	1	7	18	4	115	3,8	3,7
	2.	0	1	9	16	4	113	3,7	
	3.	0	1	11	15	3	110	3,6	
	4.	0	0	7	18	5	118	3,9	
	5.	0	1	8	15	6	116	3,8	

Pertanyaan No.		Penilaian					Σ	$\bar{x}$	Nilai Akhir
		1	2	3	4	5			
Materi Pembelajaran									
D	1.	0	0	2	15	13	131	4,3	3,9
	2.	0	0	5	16	9	124	4,1	
	3.	0	4	9	13	4	107	3,5	
	4.	0	0	4	20	6	122	4	
	5.	0	3	9	15	3	108	3,6	
Penilaian									
E	1.	0	0	8	12	10	122	4	3,9
	2.	0	1	7	12	10	121	4	
	3.	0	1	8	15	6	116	3,8	
Manfaat									
F	1.	0	0	9	18	3	114	3,8	4
	2.	0	0	13	15	2	109	3,6	
	3.	0	0	4	13	13	129	4,3	
	4.	0	0	7	14	9	122	4	
	5.	0	0	7	12	11	124	4,1	
	6.	0	0	4	15	11	127	4,2	
Minat dan Motivasi									
G	1.	0	0	6	18	6	120	4	3,9
	2.	0	0	5	21	4	119	3,9	
	3.	0	0	9	18	3	114	3,8	
Lain-lain									
H	1.	0	0	3	14	13	130	4,3	4,3
Ahli Bahasa Inggris (diisi oleh seorang ahli)									
I	1.	0	0	0	1	0	4	4	4
	2.	0	0	0	1	0	4	4	
	3.	0	0	0	1	0	4	4	
	4.	0	0	0	1	0	4	4	

Hasil yang didapat dari tabel 4.12 dapat digunakan untuk menentukan analisis hasil uji coba aplikasi, dimana bertujuan untuk mengetahui apakah aplikasi belajar melafalkan bahasa Inggris dapat dinyatakan layak digunakan dan sesuai dengan yang dibutuhkan serta menarik minat dan motivasi pengguna.

4.4.3. Analisis Hasil Uji Coba Aplikasi Mobile Belajar Melafalkan Bahasa Inggris

Pada bagian ini akan ditampilkan analisis hasil uji coba yang telah dilakukan pada aplikasi dengan hasil sebagai berikut:

1. Analisis hasil uji coba fungsi aplikasi

   Analisis hasil uji coba dari keseluruhan uji coba yang dilakukan akan menentukan kelayakan fungsi-fungsi aplikasi sesuai dengan rancangan yang telah ditetapkan. Fungsi-fungsi aplikasi dinilai layak apabila keseluruhan hasil uji coba ini sesuai dengan *output* yang diharapkan. Pada uji coba yang telah dilakukan pada fungsi-fungsi aplikasi seperti tampak pada *test case* 1-52, dapat disimpulkan bahwa fungsi-fungsi tersebut telah berjalan dengan baik dan tidak terdapat *error*. Fungsi menampilkan isi *item* standar dan *database*, fungsi dengar, fungsi lafal, fungsi pengolahan nilai kedalam grafik, fungsi tampilkan daftar data, fungsi tambah data, fungsi ubah data dan fungsi hapus data dapat berjalan seperti yang diharapkan.

2. Analisis hasil uji coba aplikasi kepada pengguna dilakukan untuk mengetahui respon pengguna, apakah aplikasi ini bermanfaat dan mudah dimengerti oleh pengguna, serta hasil identifikasi aplikasi terhadap pelafalan pengguna dari hasil *survey* (tabel 4.11). Berdasarkan pada tabel 4.12 respon 30 orang pengguna (100%) menilai bahwa aplikasi belajar melafalkan bahasa Inggris:

   a. Tampilan mendapat nilai akhir 4 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 80%. Hasil tersebut menyatakan tampilan yang terdapat pada aplikasi bernilai baik.

   b. Navigasi mendapat nilai akhir 3,7 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 74%. Hasil tersebut menyatakan navigasi yang terdapat pada aplikasi bernilai baik.

c. Materi pembelajaran mendapat nilai akhir 3,9 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 78%. Hasil tersebut menyatakan materi pembelajaran yang terdapat pada aplikasi bernilai baik.
d. Penilaian mendapat nilai akhir 3,9 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 78%. Hasil tersebut menyatakan penilaian yang terdapat pada aplikasi bernilai baik.
e. Manfaat mendapat nilai akhir 4 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 80%. Hasil tersebut menyatakan manfaat yang terdapat pada aplikasi bernilai baik.
f. Minat dan motivasi mendapat nilai akhir 3,9 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 78%. Hasil tersebut menyatakan minat dan motivasi pengguna bernilai baik.
g. Lain-lain mendapat nilai akhir 4,3 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 86%. Hasil tersebut menyatakan bahasa yang digunakan pada tiap halaman jelas/dapat dipahami pengguna bernilai baik.
h. Ahli bahasa Inggris mendapat nilai akhir 4 dari kisaran 1-5, dimana jika diubah kedalam prosentase bernilai 80%. Hasil tersebut menyatakan seorang ahli memberi nilai baik terhadap aplikasi.