# PEMBUATAN PERANGKAT LUNAK UNTUK PEMROSESAN PERINTAH PADA KALENDER BERBASIS ANDROID

**ALBER JONATHAN CHRISTIANTO CHENDRA**

Jurusan Informatika Universitas Surabaya

nahtanoj_rebla@hotmail.com

**NJOTO BENARKAH**

benarkah@staff.ubaya.ac.id

**MONICA WIDIASRI**

monica@ubaya.ac.id

***Abstrak***–Perkembangan teknologi memberikan banyak pilihan bagi manusia untuk berinteraksi dengan komputer. Salah satu cara untuk berinteraksidengan komputer adalah pengenalan ucapan. Penelitian ini dibuat untuk menunjukkan ucapan pengguna dapat digunakan untuk memberi masukan kepada aplikasi kalenderberbasis Android. Hasil penelitian ini menunjukkan bahwa pengenalan ucapan dapatmembantu pengguna dalam menjalankan aplikasi kalenderdi Android. Perintah tersebut digunakan untuk menjalankan fungsi-fungsi kalender berdasarkan aturan Context-Free Grammar yang telah dibuat.Hasil uji coba dan evaluasi yang sudah dilakukan menunjukkan bahwa aplikasi pemroses perintah ucapan yang dibuat berhasilmembantu pengguna untuk menjalankan aplikasi kalender pada Android.

***Kata kunci:***speech recognition, Android, calendar, command processing

## PENDAHULUAN

Manusia membuat komputer pada awalnya untukmelakukanperhitungan yang rumitsecaraefektif dan efisien. Perkembangan teknologi pada beberapa dekade terakhir turut mengubah fungsi komputer, yaitu dari suatu alat yang hanya digunakan untuk melakukan perhitungan matematis secara cepat menjadi suatu alat yang digunakan untuk membantupekerjaan manusia di berbagai bidang kehidupan. Perkembangan teknologi tidak hanya mengubah fungsi komputer, tetapi juga mengubah bentuk dan ukuran komputer.Komputer yang pada awalnya berukuran sangat besar kemudiandibuat dalam berbagai ukuran yang lebih kecil, mulai dari komputer *desktop* yang digunakan untuk rumah dan kantor sampai dengan *smartphone*yang berukuran sebesartelapak

tangan seseorang. *Smartphone*merupakan salah satu jenis komputer yang paling sering digunakan saat ini karena ukuran yang kecil dan mampu membantu pekerjaan manusia dengan mudah.Android merupakan salah satu sistem operasi *smartphone* terpopuler di seluruh dunia.

Android memiliki banyak aplikasi yang digunakan untuk membantu pekerjaan manusia, seperti jam, perambah web, dan kalender. Aplikasi kalender standar yang digunakan Android memanfaatkan sentuhan jari seseorang sebagai sarana penerima masukan.Android sendiri memiliki fitur untuk mengenali ucapan seseorang dan mengubah ucapan tersebut ke dalam bentuk teks. Fitur tersebut dapat dimanfaatkan untuk memberikan alternatif bagi pengguna dalam memberikan masukan untuk aplikasi kalender yang digunakan. Sayangnya,aplikasi Android yang sudah menggunakan pengenalan ucapan untuk menerima masukan masih kurang memperhatikan mengenai fitur kalender pada Android. Salah satu masalah yang sering dihadapi adalah tidak adanya standar API untuk kalender yang digunakan oleh *smartphone* Android sebelum versi Ice Cream Sandwich, sehingga pembuat aplikasi sulit mengakses kalender melalui aplikasi lain.

Penelitian ini dilakukan dengan tujuan membuat aplikasi Android yang dapat menerima masukan berupa ucapan. Aplikasi tersebut kemudian memproses masukan menggunakan aturan sesuai Context-Free Grammar yang dibuat sendiri untuk menjalankan fungsi-fungsi kalender pada Android. Hasil yang diharapkan dari penelitian ini adalah aplikasi Android yang dapat menerima ucapan pengguna sebagai perintah untuk menjalankan kalender.

## METODE PENELITIAN

Penelitian mengenai pembuatan aplikasi pemroses perintah ucapan pada kalender berbasis Android dirancang dalam beberapa tahap, yaitu pengumpulan data, analisis, desain, implementasi, uji coba, evaluasi, serta kesimpulan dan saran. Pengumpulan data dilakukan dengan denganmengumpulkan informasi mengenai pembuatan aplikasi pada Android, pemrosesan perintah menggunakan Context-Free Grammar, dan

pengenalan ucapan. Informasi diperoleh melalui studi literatur dan artikel-artikel maupun jurnal dari internet.Tahap analisis dilaksanakan dengan melakukan pengujian terhadap3 aplikasi yang menggunakan pengenalan ucapan untuk menerima masukan, yaitu Utter, Skyvi, dan Speaktoit Assistant. Ketiga aplikasi digunakan selama beberapa saat dan kemudian dibandingkan satu dengan yang lain untuk mencari kelebihan dan kekurangan masing-masing aplikasi. Hasil perbandingan tersebut digunakan untuk mengetahui kebutuhan aplikasi yang akan dibuat.

Desain yang dibuat untuk penelitian terbagi menjadi 3, yaitu desain proses, desain *user interface*, dan desain Context-Free Grammar. Desain proses digambarkan dengan *flow chart diagram*. Tahap implementasi didasarkan kepada hasil analisis dan desain yang sudah dilakukan pada 2 tahap sebelumnya. Tahap uji coba dilakukan pada aplikasi menggunakan skenario yang dibuat berdasarkan kebutuhan sistem danhasil desain proses. Evaluasi dilakukan dengan memberikan kuesioner pada 10 responden yang sudah melakukan uji coba terhadap aplikasi. Responden menerima kuesioner dan aplikasi melalui *e-mail*.Responden perlu menjawab beberapa pertanyaan mengenai aplikasi untuk memastikan mereka sudah menguji aplikasi tersebut sebelum mengisi kuesioner. Kesimpulan dan saran diperoleh setelah menyelesaikan tahap uji coba dan evaluasi.

## HASIL DAN PEMBAHASAN

Analisis yang sudah dilakukan terhadap 3 aplikasi sejenis menunjukkan kelebihan dan kekurangan masing-masing aplikasi. Kelebihan Utter dibandingkan aplikasi sejenis adalah memiliki banyak alternatif perintah, sehingga pengguna tidak dibatasi dengan satu kata untuk menjalankan suatu fungsi. Selain itu, Utter memiliki banyak pilihan*engine* untuk mengenali ucapan pengguna, sehingga pengguna dapat memilih sendiri *engine*yang dianggap memiliki akurasi pengenalan ucapan terbaik. Kekurangan Utter dibandingkan aplikasi sejenis adalah tampilan tombol mikrofon tidak berada pada posisi yang mudah dilihat oleh pengguna, sehingga cukup sulit digunakan oleh pengguna awam. Hasil pengenalan

ucapan dan tanggapan dari aplikasi tidak ditampilkan oleh Utter. Hal ini memang menghemat ruang pada aplikasi, tetapi berpotensi membingungkan pengguna karena terdapat kemungkinan pengguna kurang memperhatikan apa yang disampaikan aplikasi, sehingga pengguna tidak mengetahui tanggapan aplikasi.

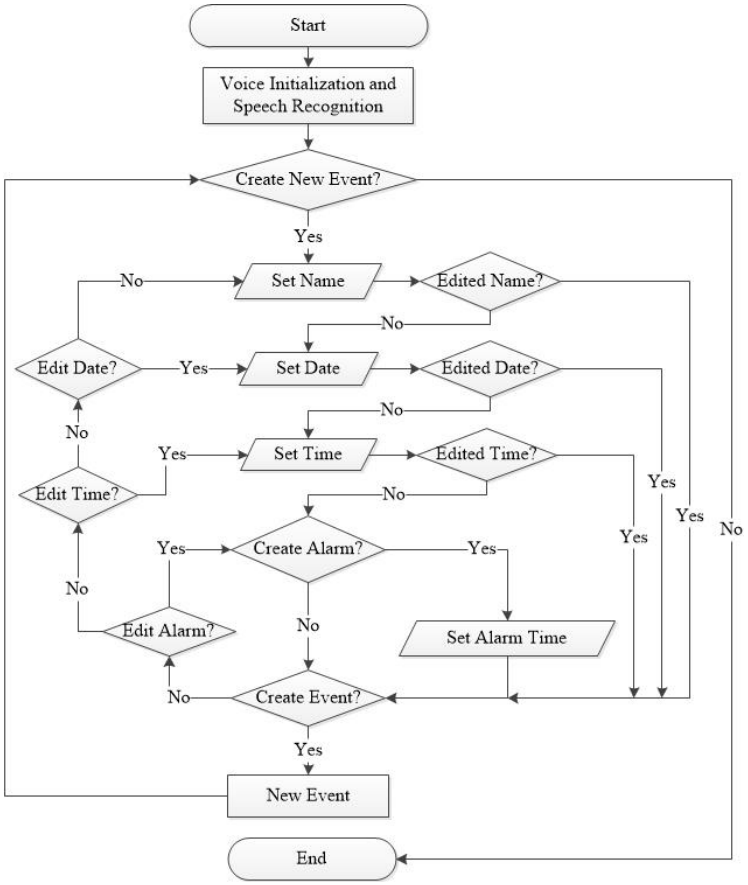
Skyvi memiliki keunggulan dibandingkan aplikasi sejenis dalam hal*user interface*. Peletakan tombol mikrofon yang berukuran besar pada bagian bawah aplikasi membuat pengguna mudah memahami cara penggunaan Skyvi. Tetapi, *engine* yang digunakan Skyvi untuk mengenali ucapan tidak seakurat *engine* milik aplikasi sejenis, sehingga kesalahan terjemahan sering terjadi. Fitur yang ditawarkan Skyvi juga lebih sedikit dibandingkan dengan aplikasi sejenis.

Aplikasi ketiga, yaitu Speaktoit Assistant, memiliki kelebihan dan kekurangannya sendiri. Speaktoit Assistant menggunakan tampilan avatar asisten pribadi dan tombol mikrofon berwarna putih yang kontras dengan latar belakang hitam yang digunakan Assistant. Hasil pengenalan ucapan dan tanggapan dari Assistant juga ditampilkan pada layar dan dibacakan. Kekurangan Assistant adalah aturan perintah yang digunakan mirip antara satu perintah dengan perintah yang lain, sehingga sering terjadi kesalahpahaman dalam pemrosesan perintah.

Kebutuhan sistem untuk aplikasi pemroses perintah ucapan pada kalender dirumuskan berdasarkan hasil analisis yang sudah dilakukan. Hasil rumusan kebutuhan sistem disimpulkan menjadi 9poin, yaitu pengenalan ucapan dalam bentuk perintah, pembuatan jadwal baru, perubahan jadwal, pengecekan jadwal, penghapusan jadwal, pembuatan alarm untuk jadwal, penghapusan alarm untuk jadwal, penambahan perintah, dan penghapusan perintah.

Terdapat 5 proses yang dibuatuntuk aplikasi, yaitu proses pembuatan jadwal baru, proses perubahan jadwal, proses penghapusan jadwal, proses pengecekan jadwal, dan proses pengecekan kalender.Seluruh proses diawali dengan inisialiasi suara untuk aplikasi dan penerimaan masukan berupa ucapan pengguna.Langkah selanjutnya pada proses pembuatan jadwal baru

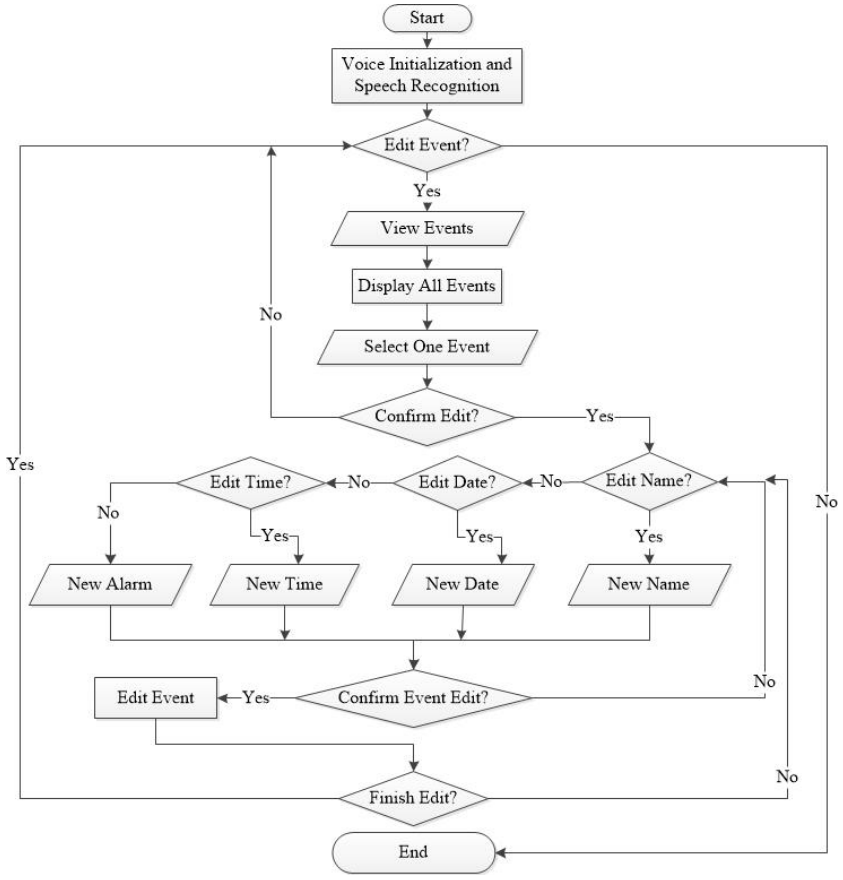
adalah mengecek kata yang diucapkan. Jika urutan perintah yang diucapkan sesuai dengan perintah “new event”, maka aplikasi melanjutkan proses dengan menerima nama jadwal yang ingin dibuat, disusul dengan menerima tanggal dan waktu pelaksanaan jadwal. Tanggal dan waktu pelaksanaan jadwal diterima dalam 2 tahap berbeda. Aplikasi kemudian memberikan opsi pembuatan alarm bagi jadwal. Pilihan tidak melanjutkan proses ke konfirmasi pembuatan jadwal, sedangkan pilihan ya melanjutkan proses ke penentuan waktu alarm dan konfirmasi pembuatan jadwal. Aplikasi dapat melanjutkan proses ke tahap perubahan bagian jadwal jika pembuatan jadwal tidak mendapat konfirmasi. Aplikasi melanjutkan proses ke penyimpanan jadwal ke dalam basis data jika konfirmasi pembuatan jadwal diperoleh. Proses pembuatan jadwal baru pun berakhir.Gambar 1 menunjukkan *flow chart* dari proses pembuatan jadwal baru.




Gambar 1. *Flow chart* proses pembuatan jadwal baru

Proses perubahan jadwal melanjutkan inisialisasi suara dan penerimaan masukan dengan menerima nomor jadwal yang ingin diubah.

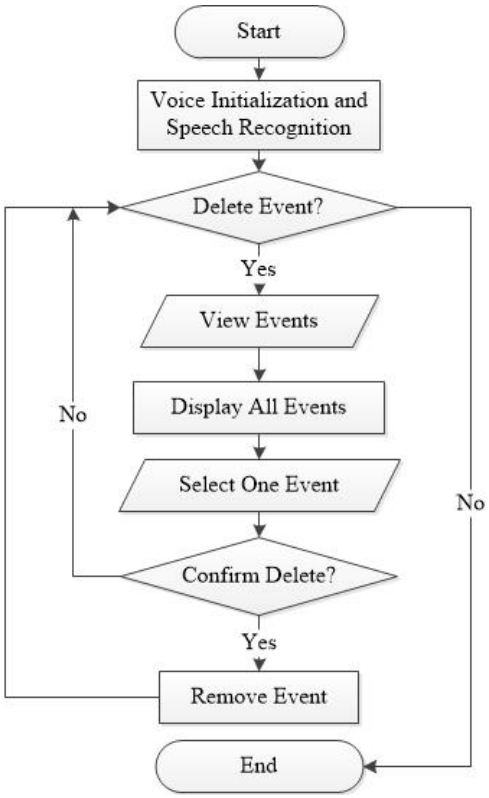
Proses kemudian berlanjut ke konfirmasi nomor jadwal yang ingin diubah. Aplikasi mengembalikan proses ke penerimaan nomor jadwal jika konfirmasi tidak diberikan, sedangkan jika konfirmasi diberikan, aplikasi melanjutkan proses ke penerimaan bagian jadwal yang ingin diubah. Bagian jadwal yang dapat diubah adalah detil nama jadwal, tanggal pelaksanaan jadwal, waktu pelaksanaan, dan alarm untuk jadwal.Proses perubahan jadwal kemudian dilanjutkan setelah pemilihan bagian jadwal yang ingin diubah dengan menetapkan nilai baru untuk bagian tersebut. Aplikasi kemudian meminta konfirmasi perubahan jadwal. Jika konfirmasi tidak diterima, maka proses kembali ke tahap pemilihan bagian jadwal yang ingin diubah. Jika konfirmasi diterima, maka jadwal diubah dan proses perubahan jadwal pun selesai.*Flow chart* proses perubahan jadwal dapat dilihat pada Gambar 2.




Gambar 2. *Flow chart* proses perubahan jadwal

Proses penghapusan jadwal mirip seperti proses perubahan jadwal. Setelah melakukan inisialisasi suara dan menerima masukan ucapan, proses

dilanjutkan dengan menerima nomor jadwal yang ingin dihapus dan meminta konfirmasi penghapusan jadwal. Proses kembali ke penerimaan nomor jadwal jika konfirmasi tidak diberikan kepada aplikasi dan proses dilanjutkan ke penghapusan jadwal dari basis data jika konfirmasi diberikan kepada aplikasi. Gambar 3 merupakan *flow chart* dari proses penghapusan jadwal.




Gambar 3. *Flow chart* penghapusan jadwal

Proses pengecekan jadwal dan proses pengecekan kalender menggunakan alur kerja yang hampir sama. Setelah inisialisasi suara dan menerima masukan berupa ucapan pengguna, proses dilanjutkan dengan menampilkan jadwal bagi proses pengecekan jadwal dan menampilkan kalender bagi proses pengecekan kalender.

Context-Free Grammar (CFG)dibuat untuk menetapkan aturan-aturan yang digunakan untuk mengatur masukan yang dapat diterima aplikasi. CFG terdiri dari terminal (**Σ**), non-terminal (**N**), *start symbol* (S), dan *production rule* (**P**). Pada pembuatan aplikasi yang dilakukan, CFG menggunakan *semantic rule*untuk membedakan satu aturan dengan aturan

yang lain. Penggunaan *semantic rule* diperlukan karena non-terminal yang digunakan oleh aplikasi tidak saling berhubungan, sehingga membutuhkan aturan untuk menentukan kondisi yang perlu terpenuhi sebelum suatu *production rule* dapat dijalankan. CFG dan *semantic rule*-nya adalah sebagai berikut:

CFG + *semantic rule*:

**Σ: {commands, events, event, numbers, date, month, answer, year, part, hour, minute, alarm, alHour, alMinute}**

**N: {S, New, NumEvent, Name, EventDate, EventTime, Confirm, EditPart, ConAlarm, AlarmTime}**

**S: S**

**P: {**

| | |
|---|---|
| **S → New** | {S.value = New.value<br>state.value = “EventsDel” \|\| “EventsUp” \|\| “Event” } |
| **New → commands events** | {if state.value = “Start”<br>New.value = commands.value + events.value} |
| **S→ NumEvent** | {S.value = NumEvent.value<br>if state.value = “EventsDel“<br>state.value = “Delete”<br>if state.value = “EventsUp”<br>state.value = “Edit” } |
| **NumEvent → numbers** | {if state.value = “EventsDel” \|\| “EventsUp”<br>NumEvent.value = numbers.value} |
| **S → Name** | {S.value = Name.value |

| | |
|---|---|
| | if edit.value = false state.value = “Date” |
| | if edit.value = true state.value = “Answer”} |
| **Name → event** | {if state.value = “Event” |
| | Name.value = event.value} |
| **S→ EventDate** | {S.value = EventDate.value |
| | if state.value = “Date” state.value = “Time” |
| | if edit.value = true state.value = “Answer”} |
| **EventDate → date month** | {if state.value = “Date” EventDate.value = |
| | date.value + months.value + |
| | Calendar.getYear() } |
| **EventDate → month date** | {if state.value = “Date” EventDate.value = |
| | date.value + months.value + |
| | Calendar.getYear() } |
| **EventDate →** | |
| **date month year** | {if state.value = “Date” EventDate.value = |
| | date.value + months.value + year.value} |
| **EventDate →** | |
| **month date year** | {if state.value = “Date" EventDate.value = |
| | date.value + months.value + year.value} |
| **S → EventTime** | {S.value = EventTime.value |
| | ifstate.value = “Time” state.value = “Alarm” |
| | if edit.value = true state.value = “Answer”} |
| **EventTime → hour minute** | {if state.value = “Time” EventTime.value = |
| | hour.value+ minute.value} |

| | |
|---|---|
| **S → Confirm** | {S.value = Confirm.value |
| | if state.value=“Answer” || “Delete” |
| | state.value=“Start” |
| | if state.value = “Edit” state.value = “Part” } |
| **Confirm→ answer** | {if state.value = “Answer” |
| | Confirm.value = answer.value} |
| **S → EditPart** | {S.value = EditPart.value |
| | if EditPart.value = “name” |
| | state.value = “Event” |
| | if EditPart.value = “date” state.value = “Date” |
| | if EditPart.value = “time” |
| | state.value = “Time” |
| | if EditPart.value = “alarm” |
| | state.value = “Alarm”} |
| **EditPart → part** | {if state.value = “Part” |
| | EditPart.value = part.value} |
| **S → ConAlarm** | {S.value = ConAlarm.value |
| | ifConAlarm.value = “yes” |
| | **s**tate.value = “alarmTime” |
| | ifConAlarm.value = “no” |
| | **s**tate.value = “Answer”} |
| **ConAlarm → alarm** | {if state.value = “Alarm” |
| | ConAlarm.value = alarm.value} |

**S → AlarmTime** {S.value = AlarmTime.value

**s**tate.value = “Answer”}

**AlarmTime →**

**alHour alMinute** {if state.value = “alarmTime”

AlarmTime.value =

alHour.value + alMinute.value}

**}**

Desain proses menunjukkan bahwa inisialisasi suara dan pengenalan ucapan selalu digunakan dalam semua proses. Implementasi inisialisasi suara dalam program dilakukan dengan memanfaatkan kelas TextToSpeech milik Android dan implementasi pengenalan ucapan dilakukan menggunakan *speech recognition engine* milik pihak ketiga yang terdapat dalam *smartphone*. *Engine* yang umum digunakan adalah Google Voice. Hasil implementasi untuk inisialisasi suara ditunjukkan pada Listing 1 dan hasil implementasi untuk pengenalan ucapan ditunjukkan pada Listing 2.

```
private TextToSpeech tts;
tts = new TextToSpeech(this, this);
String text = "", record = "";

@Override
publicvoid onInit(int code) {
if (code == TextToSpeech.SUCCESS) {
tts.setLanguage(Locale.getDefault());
      if (tts != null) {
      text = "What do you want to do?";
      if (text != null) {
            if (!tts.isSpeaking()) {
            tts.speak
                  (text, TextToSpeech.QUEUE_FLUSH, null);
            record += text + "\n";
      ((TextView)findViewById(R.id.txtRecord)).
                  setText(record);
            }
      }
      }
   } else {
      tts = null;
      Toast.makeText(this, "Failed to initialize TTS
      engine.", Toast.LENGTH_SHORT).show();
}
}
```

Listing 1. Implementasi Inisialisasi Suara

```
protectedstaticfinalintREQUEST_OK = 1;
String command = "", record = "", category = "", type =
"", text = "";
List<String>spokenWords = new LinkedList<String>();
private TextToSpeech tts;
@Override
publicvoid onClick(View arg0) {
Intent intentSpeech = new Intent
(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intentSpeech.putExtra
(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "en-US");
try {
        startActivityForResult(intentSpeech, REQUEST_OK);
    } catch (Exception e) {
      Toast.makeText(this, "Error initializing speech to
text engine.", Toast.LENGTH_LONG).show();
    }
}
@Override
protectedvoid onActivityResult(int requestCode, int
resultCode, Intent data) {
   super.onActivityResult(requestCode, resultCode, data);
if (requestCode==REQUEST_OK&& resultCode==RESULT_OK)
{
      ArrayList<String> thingsUserSaid = data.
      getStringArrayListExtra(RecognizerIntent.EXTRA_RESUL
      TS);
      record += thingsUserSaid.get(0) + "\n";
for (int i=0; i < thingsUserSaid.get(0).length();
           i++) {
if (thingsUserSaid.get(0).charAt(i) != ' ') {
      command += String.valueOf
            (thingsUserSaid.get(0).charAt(i));
         } elseif (thingsUserSaid.get(0).charAt(i)
                  == ' ') {
      spokenWords.add(command);
      command = "";
         }
      }
if (command != "") {
spokenWords.add(command);
command = "";
      }
   }
}
```

Listing 2. Implementasi Pengenalan Ucapan Pengguna

Uji coba dilakukan terhadap aplikasi yang sudah dibuat untuk mengetahui apakah seluruh proses yang ada dalam aplikasi sudah berjalan tanpa kesalahan. Uji coba dilakukan untuk proses pembuatan jadwal baru, proses perubahan jadwal, dan proses penghapusan jadwal. Seluruh proses sudah berjalan sesuai dengan desain proses yang sudah dibuat. Gambar 4 menunjukkan hasil uji coba pada proses pembuatan jadwal baru.


Gambar 4. Hasil uji coba proses pembuatan jadwalbaru

Tahap evaluasi dijalankan dengan membagikan kuesioner pada 10 responden yang sudah menggunakan aplikasi yang dibuat. Hasil kuesionernya adalah sebagai berikut.

Tabel 1. Hasil kuesioner mengenai aplikasi pada 10 responden

| No. | Penilaian | Sangat Setuju | Setuju | Kurang Setuju | Tidak Setuju |
|---|---|---|---|---|---|
| 1 | *User interface* dari aplikasi pemroses perintah untuk kalender mudah dimengerti. | 50% (5) | 50% (5) | | |
| 2 | Aplikasi pemroses perintah untuk kalender dapat membuat, mengubah, dan menghapus jadwal dengan mudah. | 30% (3) | 60% (6) | 10% (1) | |
| 3 | Aplikasi pemroses perintah untuk kalender dapat membuat dan menghapus alarm dengan mudah. | 60%(6) | 40% (4) | | |
| 4 | Aplikasi pemroses perintah untuk kalender dapat menambahkan jadwal ke Google Calendar dengan mudah. | 40% (4) | 60% (6) | | |
| 5 | Aplikasi pemroses perintah untuk kalender dapat menjalankan aplikasi kalender bawaan pengguna dengan mudah. | 30% (3) | 70% (7) | | |

## KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa aplikasi yang memproses ucapan pengguna untuk menjalankan fungsi-fungsi kalenderberbasis Android dapat membantu pengguna mengoperasikankalender pada Android. Pemrosesan perintah dilakukan dengan menggunakan Context-Free Grammar yang dirancang secara khusus untuk mengolahjadwal pada kalender. Interaksi antara Google Calendar dan aplikasi yang dibuat juga berhasil dilakukan.

Meskipun telah mencapai tujuan awal yang diinginkan, ada beberapa saran yang dapat diberikan untuk pembuatan aplikasi sejenis pada masa mendatang. Aplikasi diharapkan dapat digunakan untuk menerima perintah dan memberikan tanggapan dalam bahasa Indonesia. Aplikasi yang dibuat juga diharapkan dapat memberikan pilihan kepada pengguna mengenai bahasa yang ingin digunakan. Selain itu, aplikasi diharapkan dapat memiliki fitur-fitur kalender yang lebih lengkap, seperti membuat jadwal berulang, jadwal yang waktunya sepanjang hari, dan *reminder*. Aplikasi juga diharapkan dapat mengenali bahasa yang digunakan sehari-hari.

## DAFTAR PUSTAKA

Android, 2014. Accessing Google APIs. https://developer.android.com/google/auth/api-client.html . Diakses pada 15 Desember 2014.

Annuzzi, J., Jr., Darcey, L. & Conder, S., 2013. *Introduction to Android Application Development*. Upper Saddle River, NJ: Addison-Wesley.

Beigi, H., 2011. *Fundamentals of Speaker Recognition*. New York: Springer.

Cinar, O., 2012. *Android Apps with Eclipse*. New York: Apress.

Gargenta, M. & Nakamura, M., 2014. *Learning Android*. Sebastopol: O'Reilly.

Holzner, S., 2004. *Eclipse*. Sebastopol: O'Reilly.

Hopcroft, J.E., Motwani, R. & Ullman, J.D., 2001. *Introduction to Automata Theory, Languages, and Computation*. Boston: Addison-Wesley.

Jurafsky, D.S. & Martin, J.H., 2000. *Speech and Language Processing*. Englewood Cliffs, NJ: Prentice Hall.