Jurnal Rekursif, Vol. 6 No. 1 Maret 2018, ISSN 2303-0755
http://ejournal.unib.ac.id/index.php/rekursif/

IMPLEMENTASI ALGORITMA *HORSPOOL* PADA APLIKASI KAMUS BAHASA LINTANG - INDONESIA BERBASIS ANDROID

Virnakimlin Frigustini [1], Aan Erlansari[2], Desi Andreswari[3]

[1,2,3] Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu,
Jln. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[1]virnakimlin.25@gmail.com
[2]aan_erlanshari@unib.ac.id
[3]desi_andreswari@unib.ac.id

Abstrak: Bahasa Lintang adalah salah satu bahasa daerah yang berasal dari Provinsi Sumatera Selatan. Bahasa Lintang merupakan salah satu bahasa daerah kebanggaan Indonesia yang harus dipertahankan. Aplikasi Kamus Lintang-Indonesia berbasis Android menerapkan algoritma pencocokan string *Horspool* dalam pencariannya dan *database* SQLite sebagai penyimpanan data bahasa Lintang. Untuk mendapatkan hasil pencarian, algoritma *Horspool* menggunakan kata pada proses pencarian sebagai *pattern* dan kosakata yang diinputkan kedalam *database* sebagai teks. Hasil penelitian ini adalah sebuah aplikasi kamus Lintang-Indonesia yang dapat dijalankan pada sistem operasi Android secara *offline* dan diharapkan dapat memberikan kemudahan bagi para pengguna aplikasi kamus bahasa Lintang dalam mencari dan mempelajari setiap kosakata bahasa Lintang secara digital yang dapat diakses kapanpun dan dimanapun. Berdasarkan hasil pengujian *black box*, sistem ini dapat melakukan pencarian kata dalam kamus Lintang-Indonesia dengan tingkat kelayakan 4,17 dalam skala *likert* yang termasuk dalam kategori baik.

Kata Kunci: Bahasa Lintang, Pencocokan *String*, *Horspool*, Android.

Abstract: Lintang language is one of the local languages from Province of South Sumatera. This language is considered to be one of exceptional local languages in Indonesia which needs to be preserved. Android-Based Lintang-Indonesian Dictionary Application applies Horspool string matching algorithm in searching process and SQLite database as Lintang language data storage. To gain search results, Horspool algorithm uses word in searching process as pattern dan vocabulary inputted in database as text. The result of this research was an Android-Based Lintang-Indonesian Dictionary Application which can operate in Android operating system in offline mode and it was expected to provide simplicity for the users in finding and learning every vocabulary in Lintang digitally with non-stop access, anytime and anywhere. Based on the result of black box test, this system was capable of searching words in Lintang-Indonesia dictionary with appropriateness level of 4,17 in likert scale included in the good category.

Keywords: Lintang Language, String Machine, Horspool, Android.

I. PENDAHULUAN

A. Latar Belakang Masalah

Bahasa Lintang Empat Lawang berasal dari Bahasa Melayu kuno yang banyak dipengaruhi Bahasa Jawa dan ada juga yang berasal dari Bahasa Arab dan Inggris, sebelum berkuasanya Pangeran-pangeran. Pengucapan kata dalam Bahasa Empat Lawang dulunya banyak memakai lafal A (Karena berasal dari huruf *ulu*). Setelah di Lintang Empat Lawang berkuasa Pangeran-pangeran dan Puyang yang menurunkan keturunan dimasing-masing daerah, timbullah sedikit perbedaan pengucapan pada akhir kata. Ada yang menggunakan lafal ***O*** diakhir suku kata (Pengaruh Bahasa Palembang dan Jawa), dan pengucapan menggunakan lafal ***E*** pada akhir suku kata.

Seiring perkembangan zaman, masyarakat Lintang modern mulai kurang memahami kosakata bahasa Lintang akibat pengaruh budaya dari luar. Masyarakat yang tinggal di perantauan biasanya hanya mengenal sedikit kosakata bahasa Lintang sebagai bahasa ibu karena jarang digunakan untuk berkomunikasi secara lisan. Diperlukan upaya dengan membuat dokumentasi untuk menjaga bahasa Lintang agar tidak hilang sebagai salah satu bahasa daerah di Indonesia. Salah satunya berupa diktat yang disusun oleh bapak Sultan Bustari, karena keprihatinan beliau tentang bahasa Lintang. Kamus ini terbatas keberadaannya karena masih berupa diktat dan belum ada yang dicetak untuk dijual secara umum. Sehingga dibuatlah Aplikasi kamus berbasis Android ini untuk memberikan kemudahan bagi pengguna dalam mencari arti kata bahasa Lintang ke bahasa Indonesia secara praktis kapanpun dan dimanapun melalui *smartphone.*

Algoritma pencocokan *string* sendiri merupakan algoritma yang paling penting dalam pemrosesan teks. Algoritma ini juga merupakan komponen dasar dalam implementasi perangkat-perangkat lunak dalam kebanyakan sistem operasi yang ada saat ini [1]. Salah satu algoritma pencocokan *string* adalah algoritma *Horspool*. Algoritma *Horspool* merupakan turunan dari algoritma *Boyer-Moore* dan mudah dalam implementasinya. Ketika panjang dari *pattern* kecil, sangat tidak efisien untuk menggunakan algoritma *Boyer-Moore.* Algoritma *Horspool* hanya menggunakan perpindahan karakter-buruk yang terjadi pada *Boyer-Moore*. Untuk melakukan dan menghitung nilai pergeseran *bad-character* adalah dengan melihat karakter paling kanan pada *window*. Nilai pergeseran ini dihitung pada tahap praproses untuk semua karakter pada set alfabet sebelumnya. Algoritma ini lebih efisien digunakan ketika ditemukan panjang *pattern* yang kecil [2].

Berdasarkan latar belakang yang telah dijelaskan sebelumnya maka dilaksanakan penelitian yang membahas tentang "Implementasi Algoritma *Horspool* Pada Aplikasi Kamus Bahasa Lintang - Indonesia Berbasis Android".

B. Rumusan Masalah

Adapun rumusan masalah berdasarkan latar belakang di atas adalah sebagai berikut :

1. Bagaimana membuat Aplikasi Kamus Bahasa Lintang - Indonesia berbasis Android?
2. Bagaimana penerapan algoritma *Horspool* pada Kamus Bahasa Lintang - Indonesia?

C. Batasan Masalah

Oleh karena penelitian ini sangatlah luas, untuk itu penulis memberikan batasan masalah sebagai berikut :

1. Aplikasi ini hanya bersifat satu arah dapat melakukan pencarian kosakata dari bahasa Lintang ke bahasa Indonesia.
2. Dialek yang dipakai dalam kamus ini adalah menggunakan lafal ***O*** diakhir suku kata bahasa Lintang.
3. Pencarian yang dilakukan hanya berupa kata dasar tidak memakai imbuhan.
4. Aplikasi dibuat berdasarkan Buku Diktat Kamus Lintang IV Lawang - Indonesia Karya Sultan Bustari, S.Pd.I dengan jumlah 2999 kosakata.
5. Aplikasi hanya sesuai dengan *smartphone* berbasis Android minimal versi 4.0 *Ice Cream Sandwich.*

D. Tujuan Penelitian

Adapun tujuan dari penelitian ini untuk membangun Aplikasi Kamus Bahasa Lintang - Indonesia berbasis Android dan menerapkan algoritma *Horspool* pada Kamus Bahasa Lintang - Indonesia.

E. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah membantu masyarakat dalam mencari arti kata pada kamus bahasa Lintang-Indonesia yang ingin diketahui secara lebih mudah dan lebih praktis.

II. LANDASAN TEORI

A. String Matching

Pencocokan *string* merupakan bagian penting dari sebuah proses pencarian *string* (*string searching*) dalam sebuah dokumen. Hasil dari pencarian sebuah *string* dalam dokumen tergantung dari teknik atau cara pencocokan *string* yang digunakan. Pencocokan string (*string matching)* diartikan sebagai sebuah permasalahan untuk menemukan pola susunan karakter *string* di dalam *string* lain atau bagian dari isi teks [3].

Pencocokan *string* (*string matching*) secara garis besar dapat dibedakan menjadi dua yaitu : *Exact string matching*, *Inexact string matching* atau *Fuzzy string matching* [4].

B. Algoritma Horspool

Algoritma *Horspool* adalah penyederhanaan dari algoritma *Boyer-Moore* yang dibuat oleh R. Nigel Horspool. Masalah dalam pencarian teks ini adalah mencari dalam teks yang besar untuk menemukan *pattern* pertama. Karena teks yang dicari bisa sangat besar (memungkinkan ratusan ribu karakter) maka penting untuk menggunakan teknik yang lebih efisien. Algoritma *Horspool* bekerja dengan metode yang hampir sama dengan algoritma *Boyer-Moore* namun tidak melakukan lompatan berdasarkan karakter pada *pattern* yang ditemukan tidak cocok pada teks [5].

Algoritma *Horspool* mempunyai nilai pergeseran karakter yang paling kanan dari *window*. Pada tahap observasi awal (*preprocessing*), nilai *shift* akan dihitung untuk semua karakter. Pada tahap ini, dibandingkan *pattern* dari kanan ke kiri hingga kecocokan atau ketidakcocokan *pattern* terjadi. Karakter yang paling kanan pada *window* digunakan sebagai indeks dalam melakukan nilai *shift*. Dalam kasus ketidakcocokan (karakter tidak terdapat pada *pattern*) terjadi, *window* digeser oleh panjang dari sebuah *pattern*. Jika tidak, *window* digeser menurut karakter yang paling kanan pada *pattern* [6].

C. Pencarian Dengan Algoritma Horspool

Terdapat dua tahap pada pencocokan *string* menggunakan algoritma *Horspool* [7], yaitu:

1. Tahap pra proses

   Pada tahap ini, dilakukan observasi *pattern* terhadap teks untuk membangun sebuah tabel *bad-match* yang berisi nilai *shift* ketika ketidakcocokan antara *pattern* dan teks terjadi. Secara sistematis, langkah-langkah yang dilakukan algoritma *Horspool* pada tahap praproses adalah :

   a. Algortima *Horspool* melakukan pencocokan karakter ter-kanan pada *pattern*.
   b. Setiap karakter pada *pattern* ditambah ke dalam tabel *bad-match* dan dihitung nilai *shift*-nya.
   c. Karakter yang berada pada ujung *pattern* tidak dihitung dan tidak dijadikan karakter ter-kanan dari karakter yang sama dengannya.
   d. Apabila terdapat dua karakter yang sama dan salah satunya bukan karakter ter-kanan, maka karakter dengan indeks terbesar yang dihitung nilai *shift*-nya.
   e. Algoritma *Horspool* menyimpan panjang dari *pattern* sebagai panjang nilai *shift* secara *default* apabila karakter pada teks tidak ditemukan dalam *pattern*.
   f. Nilai (*value*) *shift* yang akan digunakan dapat dicari dengan perhitungan panjang dari *pattern* dikurang indeks terakhir karakter dikurang 1, untuk masing-masing karakter, $value = m - i - 1$.

2. Tahap pencarian

   Secara sistematis, langkah-langkah yang dilakukan algoritma *Horspool* pada tahap praproses adalah:

   a. Dilakukan perbandingan karakter paling kanan *pattern* terhadap *window*.
   b. Tabel *bad-match* digunakan untuk melewati karakter ketika ketidakcocokan terjadi.
   c. Ketika ada ketidakcocokan, maka karakter paling kanan pada *window* berfungsi sebagai landasan untuk menentukan jarak *shift* yang akan dilakukan.
   d. Setelah melakukan pencocokan (baik hasilnya cocok atau tidak cocok) dilakukan pergeseran ke kanan pada *window*.
   e. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks atau ketika *pattern* cocok dengan teks

D. Penataan Kata Bahasa Lintang

Penataan kata dalam bahasa lintang [8] sebagai berikut:

1. Vokal *A* diakhir kata dari kata Bahasa Nasional diganti dengan *O* atau *E*

   Contoh : *Kemano (e)* berasal dari kata *Kemana*

   Buto (e) berasal dari kata *Buta*

2. Huruf *R* awal kata kata diganti dengan huruf *G*

   Contoh : *Gebus* berasal dari kata *Rebus*

   Gagi berasal dari kata *Ragi*

3. Huruf *R* di akhir kata yang sebelumnya vocal *I* dan *U* diganti huruf *K*

   Contoh : *Ayik* berasal dari kata *Air*

   Tiduk berasal dari kata *Tidur*

4. Huruf *R* di akhir kata yang didahului vocal *A* diganti dengan huruf *GH*

   Contoh : Ulagh berasal dari kata Ular

   Sebagh berasal dari kata *Sebar*

5. Huruf *R* di tengah suku kata atau pada awalan diganti dengan huruf *GH* dan *G*

   Contoh : *Beghas* berasal dari kata *Beras*

   Keghing berasal dari kata *Kering*

   Keterangan : Huruf R diganti dengan Gh adalah dari bahasa Palembang yang tidak bisa mengucapkan huruf R dengan sebagaimana mestinya

6. Huruf *H* pada awal kata dihilangkan.

   Contoh : *Apal* berasal dari kata *Hapal*

   Ulu berasal dari kata *Hulu*

   Idangan berasal dari kata *Hidangan*

7. Pengucapan vocal *I* diucapkan kadang-kadang miring (antara *I dan E)*. Ada kata kata sama seperti Bahasa Indonesia, tapi punya arti tersendiri

   Contoh : *Tetap* artinya *Menyadap air pada batang/kayu untuk obat*

   Undang-undang artinya *Dinding serambi*

9. Ada kata setelah mendapat imbuhan menjadi kata dasar

   Contoh : *Rombongan* berarti *Kelompok*

   Be Rombongan berarti *Berlobang di tengah*

   Impit berarti *Samping*

   Ber impit berarti *Terpaksa, Terdesak*

10. Apabila ada kata yang memakai

    Contoh : *Nardar, Nordor, Ngarkar, Benetar, Celagum, Celabak, Celabuk, Benantum*

11. Apabila kata memakai vocal E atau I berarti kecil,Ringan, lemah, pelan.

    Contoh : *Nirdir, Ngerker, Beneter, Celapek, Kitil-kitil, Gemibil, Kemepek*

12. Untuk memudahkan membaca yang tulisan sama sedangkan membacanya beda, pada bahasa daerah, maka penulis menulis :

    a. Dalam penulisan vocal *E* ditulis *double* EE guna untuk memudahkan membedakan/membaca antara E dan *E.*

       Contoh : B*ent**eel*, N*eed**ee*

       Endeep = rendah

       Eendeep = sopan, lemah lembut

    b. Penulisan huruf *K* akhir kata diganti dengan (‘)

       Contoh : *Balik* ditulis *Bali’*

       Tiduk ditulis *Tidu’*

       Masak ditulis *masa’*

E. UML (Unified Modelling Language)

UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek [9].

F. Angket

Angket atau Kuesioner merupakan teknik pengumpulan data yang efesien apabila peneliti tahu dengan siapa variabel akan diukur dan tahu apa yang bisa diharapkan dari responden. Angket dapat berupa pertanyaan-pertanyaan tertutup atau terbuka, dapat diberikan kepada responden secara langsung atau dikirim melaui pos atau internet. Untuk

data angket yang diperoleh berupa nilai skor. Untuk menentukan skor pilihan jawaban angket menggunakan skala *Likert* [10].

G. Skala Likert

Skala *Likert* digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Untuk setiap pilihan jawaban diberi skor. Dengan Skala *Likert*, variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrument yang dapat berupa pertanyaan atau pernyataan. Ciri khas dari skala ini adalah bentuk jawaban dari pertanyaan menggunakan skala *Likert* mempunyai gradasi sangat positif sampai sangat negatif [10].

III. METODE PENELITIAN

A. Jenis Penelitian

Adapun jenis penelitian yang digunakan adalah Penelitian Terapan karena penelitian ini mengaplikasikan teori untuk rancangan aplikasi kamus bahasa Lintang - Indonesia berbasis Android.

B. Sarana Pendukung

Sarana pendukung yang akan digunakan dalam aplikasi kamus bahasa Lintang-Indonesia berbasis android adalah:

1. Perangkat Keras (*Hardware*) yang digunakan adalah spesifikasi Intel® CoreTM i3-2350M, 2 GB DDR3 Memory, 14.0" HD LED LCD, 500 GB HDD, Intel® HD Grapics dan *Smartphone* Samsung Galaxy Grand Prime, Android 4.4.4 Kitkat.
2. Perangkat Lunak (*Software*) yang digunakan antara lain :
    1) Sistem Operasi Windows 7
    2) Eclipse Indigo Uno
    3) SQLite Manager

C. Teknik Pengumpulan Data

Pada tahap ini peneliti akan melakukan analisis kebutuhan sistem dengan teknik pengumpulan data menggunakan teknik reduksi data yang bersumber buku diktat Bahasa Lintang. Pengumpulan informasi diawali dengan melakukan observasi dan studi pustaka terhadap kamus bahasa Lintang. Selanjutnya menentukan batasan masalah yang akan diteliti, menentukan perangkat keras, perangkat lunak, *database* dan bahasa pemrograman *Java* serta mengamati beberapa informasi lain yang dapat dijadikan sebagai referensi yang akan digunakan untuk merancang aplikasi.

D. Alir Penelitian

Penelitian dilakukan berdasarkan diagram alir dibawah ini, hal ini juga disesuaikan dengan metode pengembangan sistem yang dipilih yaitu *waterfall*. Adapun diagram alir pada penelitian ini dapat dilihat pada gambar 1.

Gambar 1 Diagram Alir Penelitian

E. Metode Pengujian

Pengujian yang digunakan yaitu *black-box testing*. Ketika perangkat lunak komputer sudah dipertimbangkan maka *black-box testing* dilakukan untuk menguji antarmuka perangkat lunak. *Black-box testing* mengkaji beberapa aspek dari sistem tanpa memperhatikan struktur

logis internal perangkat lunak. Hal yang akan diuji pada percobaan *black-box* antara lain :

1. Pengujian fungsional sistem dari aplikasi kamus bahasa Lintang - Indonesia berbasis Android.
2. Pengujian untuk mengamati *CPU execution time* yang dibutuhkan aplikasi kamus bahasa Lintang - Indonesia berbasis Android.
3. Pengujian untuk mengamati aplikasi kamus bahasa Lintang - Indonesia berbasis Android.
4. Pengujian untuk jenis *file* lain sebagai pengayaan.

F. Metode Uji Kelayakan Sistem

Uji Kelayakan dilakukan untuk mendapatkan penilaian langsung terhadap sistem yang dihasilkan. Target dari pengujian kelayakan sistem ini adalah responden (calon pemakai sistem). Adapun tahapan dari uji kelayakan ini adalah:

1. Angket

   Angket merupakan daftar pertanyaan yang diajukan pada seorang responden untuk mencari jawaban dari permasalahan yang diteliti. Teknik pemilihan responden (sampel) dilakukan dengan metode *random sampling*. *Random sampling* yaitu pengambilan sampel secara acak yang memberikan peluang yang sama bagi setiap anggota populasi untuk dipilih menjadi anggota sampel. Pada Penelitian ini Populasi yang diambil adalah masyarakat Lintang yang berdomisili di Lingkar Timur Kota Bengkulu.

2. Tabulasi Data

   Proses perhitungan data angket menggunakan skala *Likert.* Sebelum melakukan perhitungan dengan menggunakan skala *likert,* maka terlebih dahulu dicari interval kelas.

   Skala *Likert* adalah perhitungan skor pada tiap-tiap interval dari pernyataan yang diberikan ke responden. Hasil dari proses perhitungan disajikan dalam bentuk tabel. Sehingga didapatkan nilai uji kelayakan terhadap sistem.

IV. ANALISA DAN PERANCANGAN

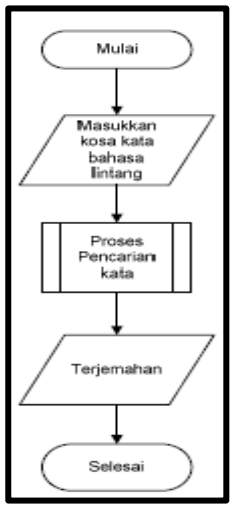
A. Analisis Sistem

1. Analisis Alur Sistem

   Analisis alur sistem merupakan bagian penelitian yang menganalisis sistem yang ada untuk merancang sistem baru atau memperbaharui sistem yang ada. Bagian ini merupakan bagian yang penting, karena hasil dari sistem yang dibuat tergantung dari analisis yang dilakukan.

a. Alur Sistem

   Alur sistem merupakan hasil analisis perancangan dari tahapan kerja sistem yang dibuat. Alur ini dimulai dari pengguna memasukkan *input* data sampai menghasilkan *output* data. *Input* berupa kosakata bahasa Lintang dan *output* berupa terjemahan bahasa Lintang kedalam bahasa Indonesia.



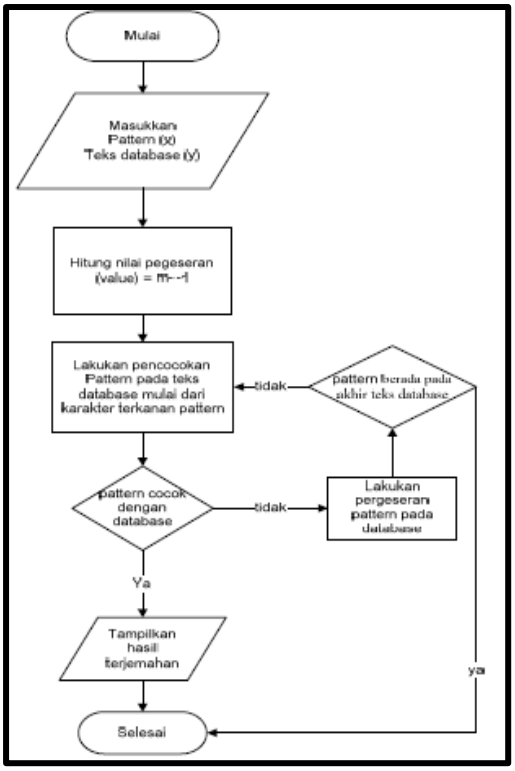
Gambar 2 Diagram Alir Sistem

Berdasarkan diagram alir pada gambar 2, setelah user memasukkan

kata bahasa lintang kemudian sistem akan memproses pencarian kata. kemudian menampilkan hasil terjemahan.

b. Alur Proses Pencarian

Berdasarkan diagram alir pada gambar 2, terdapat proses yang lebih spesifik, yaitu proses pencarian kata. Dalam proses ini, menggambarkan proses spesifik yang dimulai dari *input* berupa kata/*pattern* hingga menghasilkan *output* berupa terjemahan kedalam bahasa Indonesia. Diagram alir dari proses pencarian kata dapat dilihat pada gambar 3.



Gambar 3 Diagram Alir Proses Pencarian Kata

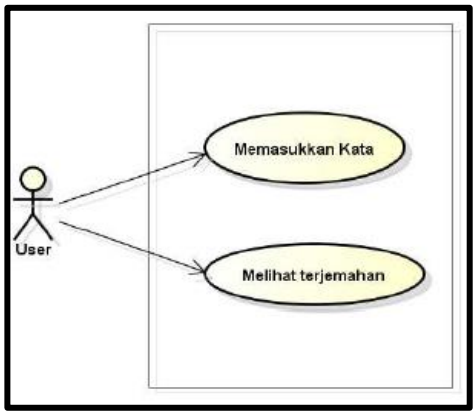
Berdasarkan diagram alir pada gambar 3 terjadi proses pencarian kata setelah *pattern* dimasukkan kemudian dihitung *value* (nilai pergeseran setiap karakter) dengan rumus *value* = $m - i - 1$. Setelah itu lakukan pencocokan *pattern* dengan teks *database* dimulai dari karakter terkanan *pattern* sebagai patokan pergeseran. Apabila *pattern* cocok dengan *database* maka tampilkan hasil terjemahan. Jika tidak cocok maka lakukan pergeseran kekanan sesuai *value*, proses dilakukan berulang sampai terjadi kecocokan. Jika terjadi ketidakcocokan sampai akhir teks maka proses selesai.

B. Perancangan Model UML (Unified Modelling Language)

Perancangan model *UML* ditujukan untuk memberikan gambaran secara umum tentang aplikasi yang akan dibangun. Membuat model dari sebuah sistem sangatlah penting agar kita dapat memahami sistem semacam itu secara menyeluruh. Pada aplikasi ini perancangan aplikasi dilakukan dengan memodelkan permasalahan dalam bentuk diagram-diagram *UML*. Diagram *UML* ini dibuat dengan menggunakan *Astah Communnity*.

1. *Use Case Diagram*

Use case diagram menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar. Adapaun *use case* pada aplikasi ini terlihat pada gambar 4.



Gambar 4 *Use Case Diagram*

Tabel 1 Daftar Aktor

Term	***Description***
User	Setiap individu yang merupakan penguna dari aplikasi Kamus bahasa Lintang

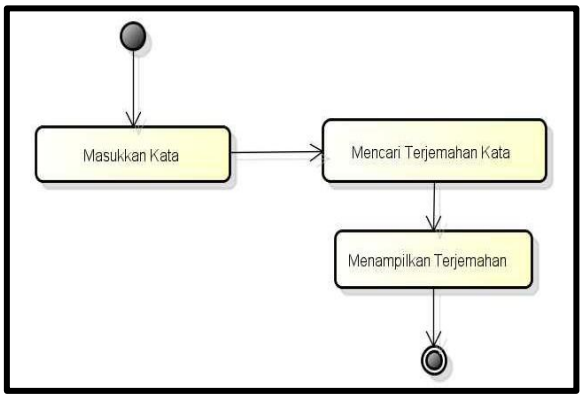
Adapun keterangan dari *use case diagram* yang terdapat pada Gambar 4 dapat dilihat pada tabel 2:

Tabel 3 Daftar *Use Case*

Use Case Name	***Use Case Description***
Memasukkan Kata	*Use Case* ini mendeskripsikan *input* kata yang akan dicari terjemahannya ke dalam bahasa Indonesia
Melihat terjemahan	*Use Case* ini mendeskripsikan proses pencarian hasil terjemahan yang telah di-*input*-kan

2. *Activity Diagram*

Diagram Aktivitas menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan akhir dari aktivitas. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Berikut adalah *activity diagram* dari aplikasi kamus yang dapat dilihat pada gambar 5:

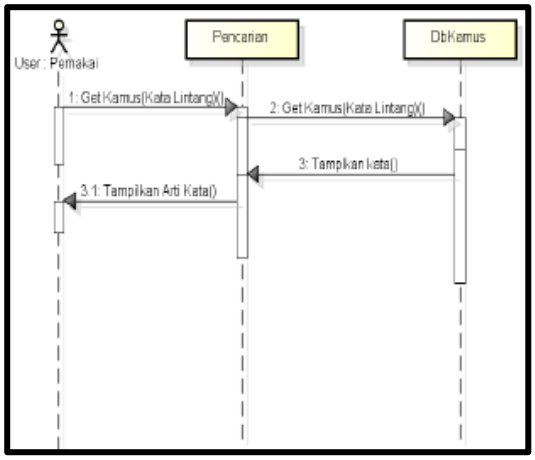


Gambar 5 *Activity Diagram*

Pada gambar 5 dapat dijelaskan bahwa pengguna saat masuk ke aplikasi akan langsung masuk ke halaman Pencarian untuk melakukan pencarian kata dengan memasukkan kata dalam bahasa Lintang yang ingin dicari terjemahannya setelah itu sistem otomatis akan menampilkan terjemahan. Apabila telah selesai melakukan operasi-operasi tersebut maka pengguna dapat keluar dari aplikasi kamus bahasa Lintang ini.

3. Sequence Diagram

Sequence diagram adalah suatu penyajian perilaku yang tersusun sebagai rangkaian langkah-langkah percontohan dari waktu ke waktu. *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output*. Berikut adalah *Sequence diagram* aplikasi kamus bahasa Lintang yang dapat dilihat pada gambar Gambar 6.

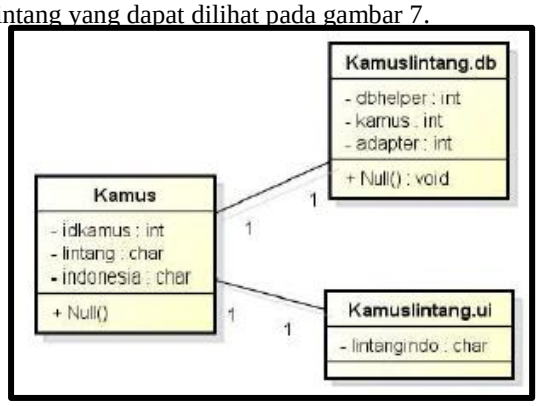


Gambar 6. *Sequence Diagram*

Diagram pada gambar 6 menunjukkan serangkaian kegiatan yang dilakukan oleh pengguna atau terhadap sistem. Saat *user* aplikasi maka akan langsung tampil halaman Lintang-Indonesia, melakukan pencarian kata bahasa Lintang dengan cara mengetikkan kata dalam bahasa Lintang. Kemudian sistem akan mengecek satu-satu kata dari ribuan kata di *database* untuk ditampilkan hasilnya secara otomatis akan tampil kata yang dicari beserta artinya.

4. Class Diagram

Class diagram adalah diagram yang menunjukan kelas-kelas yang ada dari sebuah sistem dan hubungannya secara logika. *Class diagram* menggambarkan struktur statis dari sebuah sistem. Karena itu *class diagram* merupakan tulang punggung atau kekuatan dasar dari hampir setiap metode berorientasi objek termasuk *UML*. Pada perancangan hubungan atau relasi kelas meliputi : satu ke satu (1..1), dan satu Berikut adalah diagram kelas yang digunakan untuk memvisualisasikan struktur kelas-kelas yang terdapat dalam aplikasi kamus bahasa Lintang yang dapat dilihat pada gambar 7.



Gambar 7 *Class Diagram*

Gambar 7 menunjukkan relasi antar kelas pada sistem. Beberapa kelas tersebut antara lain: *Kamus, Kamuslintang.db*¸dan *Kamuslintang.ui*. Kelas *Kamus* memiliki relasi satu ke satu (1...1) dengan *Kamuslintang.db*. *Kamus* memiliki relasi satu ke satu (1...1) dengan *Kamuslintang.ui.*

2. Perancangan *Database*

Tujuan dari perancangan *database* atau basis data adalah sebagai media penyimpanan data kamus kata yang akan dipanggil oleh sistem ketika mencari kata pada aplikasi kamus. Perancangan *database* pada sistem ini menggunakan *SQLite*. *Database* kamus terdiri dari satu tabel, yaitu tabel tb_data yang dapat dilihat pada tabel 4:

Tabel 4 Struktur Tabel tb_data

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	IdKamus	INTEGER	1	null	1
1	lintang	TEXT	0	null	0
2	indonesia	TEXT	0	null	0

Pada tabel 4 dapat dijelaskan bahwa struktur tabel tb_data terdiri dari *Column ID* yang mempunyai *type Integer* dan dijadikan kode kunci, *field* IdKamus dengan *type Integer*, Lintang dengan *type text* dan *field* Indonesia dengan *type text.*

V. HASIL DAN PEMBAHASAN

A. Hasil Implementasi Algoritma Horspool

Penerapan algoritma *Horspool* dalam sistem yang dibuat adalah pada proses pencarian kata dari bahasa Lintang ke bahasa Indonesia. Adapun langkah pertama yang dilakukan adalah melakukan pembuatan *database* dalam *SQLITE*, data yang dimasukkan adalah kata dari diktat kamus Lintang IV Lawang yang disusun oleh Sultan Bustari. Selanjutnya pencarian *pattern* pada aplikasi akan diproses dalam *database* sesuai dengan pencarian algoritma *Horspool*. Untuk *pattern* yang sesuai dengan *database* maka akan dimunculkan hasil berupa kata yang mengandung *pattern* yang di *input*kan. Jika *pattern* tidak ada dalam *database* maka kata tidak ditemukan.

Langkah pertama yang dilakukan adalah tahapan *preprocessing* yaitu membuat tabel *bad-match.* Tabel ini diciptakan merujuk kepada *pattern* yang akan dicari oleh karena itu jika *pattern* berubah maka tabel juga akan berubah.

Untuk menggambarkan rincian algoritma, akan diberikan contoh kasus *pattern* P =

"DOGHO". Hasil *preprocessing* untuk *pattern* DOGHO terlihat pada tabel 5.

Pattern: DOGHO D O G H O
0 1 2 3 4

Tabel 5 Tabel *bad-match*

Karakter	*Index*	*Value*
D	0	4
O	1	3
G	2	2
H	3	1
*	-	5

value = 5 – 0 – 1 = 4
value = 5 – 1 – 1 = 3
value = 5 – 2 – 1 = 2
value = 5 – 3 – 1 = 1
** : karakter yang tidak dikenali*

1) Contoh untuk teks Buku_Dogho:

Inisialisasi awal untuk teks T = "BUKU_DOGHO" terlihat pada Tabel 6 berikut.

Tabel 6 Inisialisasi awal *bad-match* teks *Buku_Dogho*

m	1	2	3	4	5	6	7	8	9	10
T	B	U	K	U	_	D	O	G	H	O
P	D	O	G	H	O					
i	0	1	2	3	4					

Pembuatan *bad-match* terlihat pada Tabel 7 berikut.

Tabel 7 Pembuatan *bad-match*

P	D	O	G	H	*
i	0	1	2	3	-
v	4	3	2	1	5

Seperti yang terlihat pada Tabel 7, inisialisasi awal *bad-match* dilakukan. Setiap teks dan *pattern* masing-masing diberi nilai *m* dan *i*, dimana *m* sebagai panjang *pattern* dan *i* sebagai indeks. Tabel 7 menunjukkan nilai pergeseran *bad-match* dengan menghitung nilai *v* seperti yang telah dilakukan pada Tabel 5. Pada tahap awal pencarian, dilakukan perbandingan karakter paling kanan *pattern* terhadap *window*. Apabila terjadi ketidakcocokan, akan dilakukan pergeseran ke kanan untuk melewati karakter yang tidak cocok dimana nilai

pergeserannya terdapat pada tabel *badmatch.* Karakter paling kanan teks pada *window* berfungsi sebagai landasan untuk menentukan jarak geser yang akan dilakukan. Hal ini terlihat pada Tabel 8 berikut.

Tabel 8 Iterasi algoritma *Horspool* pertama untuk teks Buku_Dogho

m	1	2	3	4	5	6	7	8	9	10
T	B	U	K	U	_	D	O	G	H	O
P	D	O	G	H	O					
i	0	1	2	3	4					

Terdapat ketidakcocokan seperti yang terlihat pada Tabel 8. Karakter "_" adalah karakter paling kanan teks pada *window*. Pada tabel *bad-match*, nilai geser karakter tak dikenali "_" adalah 5. Maka, dilakukan pergeseran ke kanan pada *window* sebanyak 5 kali. Hal ini terlihat pada Tabel 9.

Tabel 9 Iterasi algoritma *Horspool* kedua untuk teks Buku_Dogho

m	1	2	3	4	5	6	7	8	9	10
T	B	U	K	U	_	D	O	G	H	O
P						D	O	G	H	O
i						0	1	2	3	4

Pada Tabel 9, *window* telah berada pada akhir teks dan semua *pattern* cocok dengan teks. Seluruh pencocokan karakter menggunakan algoritma *Horspool* telah selesai dan berhenti pada iterasi kedua.

2) Contoh untuk teks Dogho:

Inisialisasi awal untuk teks T = "DOGHO" terlihat pada Tabel 10 berikut.

Tabel 10 Inisialisasi awal *bad-match teks Dogho*

m	1	2	3	4	5
T	D	O	G	H	O
P	D	O	G	H	O
i	0	1	2	3	4

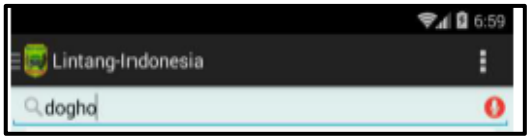
Pembuatan *bad-match* dapat dilihat pada Tabel 7. proses yang dilakukan sama seperti pada contoh teks "Buku_Dogho". Iterasi pertama untuk teks "Dogho" dapat dilihat pada tabel 11 dimana pattern dan teks langsung terjadi

kecocokan tanpa melakukan pergeseran.

Tabel 11 Iterasi algoritma *Horspool* pertama untuk teks Dogho

m	1	2	3	4	5
T	D	O	G	H	O
P	D	O	G	H	O
i	0	1	2	3	4

Tampilan pencarian pada aplikasi dengan mengetikkan kata "dogho" terlihat pada gambar 10.



Gambar 10. Pencarian kata "dogho"

Terjadi proses pencarian kedalam *database* dengan algoritma *horspool* sehingga didapat hasil seperti terlihat pada gambar 11.

Gambar 11. Hasil Pencarian Kata "Dogho"

Hasil yang didapat saat pencarian kata "dogho" pada aplikasi yaitu terdapat dua kata "Buku_Dogho" dan "Dogho".

B. Hasil Implementasi Aplikasi

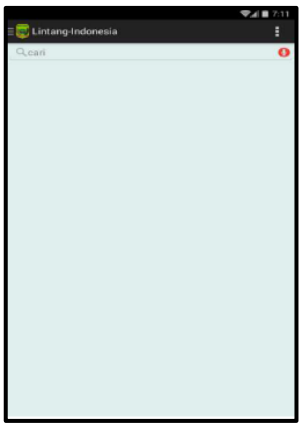
Berikut adalah tampilan ketika aplikasi dijalankan pada emulator Android yang disebut *Android Virtual Devices* (AVD).

1. Tampilan Halaman Pencarian Kata Lintang - Indonesia

   Ketika aplikasi dijalankan, tampilan yang pertama kali muncul dapat dilihat pada gambar 12.

Gambar 12. Halaman Menu *Splash*

Tampilan *splash* adalah tampilan pertama kali ketika aplikasi di jalankan. Tampilan *splash* hanya muncul beberapa detik saja, setelah itu langsung masuk ke *form* pencarian tanpa menekan tombol apapun. Tampilan pencarian kata Lintang-Indonesia dapat dilihat pada gambar 13.

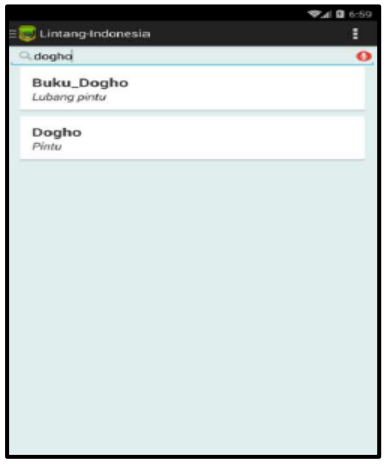


Gambar 13 Tampilan Pencarian Kata Lintang – Indonesia

Setelah halaman *splash*, maka yang muncul di layar emulator adalah halaman Pencarian Kata Lintang - Indonesia.

2. Tampilan Halaman Saat Menginputkan Kata yang Akan Dicari

   Berikut adalah tampilan saat menginputkan kata yang akan dicari, dapat dilihat pada gambar 14.



Gambar 14 Tampilan Menu Pencarian Lintang-Indonesia

Halaman pencarian adalah halaman yang menampilkan semua hasil dari satu kata yang di-*input*-kan dan hasil yang muncul adalah semua kata-kata yang terkandung dari satu kata yang diinputkan oleh *user*. Seperti contoh yang terdapat pada gambar 14, *user* menginputkan kata "dogho", sistem akan otomatis mencari dan menampilkan semua yang mengandung kata "dogho" beserta artinya akan ditampilkan pada *list view*. Pada proses ini sistem mengecek satu persatu kata dalam *database* yang ada.

C. Pengujian Black Box Testing

Pengujian aplikasi ini menggunakan metode *black box testing*. Pengujian *black box* dilakukan untuk menguji apakah sistem yang dikembangkan sesuai dengan apa yang tertuang dalam spesifikasi fungsional sistem. *Black box* juga digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang. Kebenaran perangkat lunak yang diuji hanya dilihat berdasarkan keluaran yang dihasilkan dari data atau kondisi masukan yang diberikan untuk fungsi yang ada tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut. Proses pengujian *black box* secara lengkap.

Dari keluaran yang dihasilkan, kemampuan program dalam memenuhi kebutuhan pemakai dapat diukur sekaligus sehingga dapat diiketahui kesalahankesalahannya. Beberapa jenis kesalahan yang dapat diidentifikasi adalah: fungsi tidak benar atau hilang, kesalahan antar muka atau *interface*, kesalahan pada struktur data dan kesalahan performasi.

Tabel 12 berikut menyajikan hasil *black box testing* yang dilakukan penulis terhadap aplikasi yang dibangun.

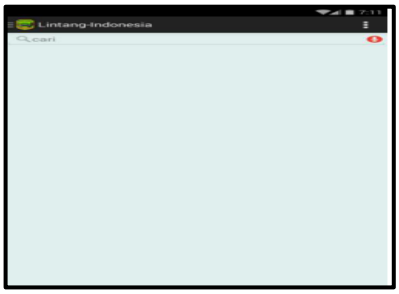
Tabel 12 Hasil *Black Box Testing*

Aktifitas Pengujian	Realisasi yang diharapkan	Hasil
Tampilan Pencarian Kata Lintang-Indonesia	Tampilan untuk memulai kamus lintang - indonesia, Mencari kata berdasarkan Algoritma *Horspool*	Sukses (Gambar 15)
Tampilan Hasil Pencarian Kata Lintang – Indonesia	Tampilan yang menampilkan bahasa lintang dan artinya	Sukses (Gambar 16)

Untuk mulai menguji dengan *black box testing* penulis membagi sistem dalam beberapa *case* atau kasus dan kemudian dianalisis sebagai berikut :

1) Kasus Tampilan halaman pencarian:

Berikut *black box testing* kasus Tampilan halaman pencarian, yang dapat dilihat pada gambar 15.

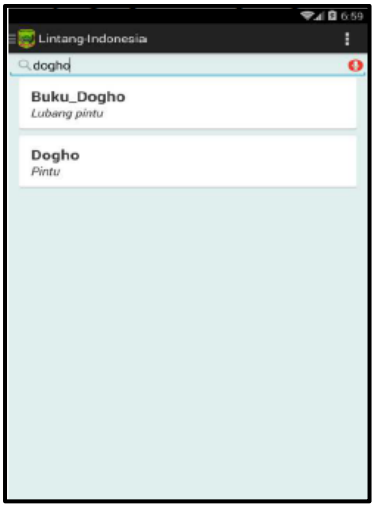


Gambar 15 Halaman Pencarian

Dalam kasus ini saat *user* menjalankan aplikasi akan muncul halaman pencarian kata Lintang – Indonesia setelah tampilan splash. Pada kasus ini aplikasi bekerja sesuai fungsi dan sesuai ancangan begitu juga tidak ada kesalahan *interface*.

2) Kasus Pencarian dengan horspool:

Berikut *black box testing* pencarian kata yang dapat dilihat pada gambar 16.



Gambar 16 Tampilan kasus pencarian kata

Dari pengujian yang dilakukan terhadap kasus pencarian kata, ternyata aplikasi berhasil mencari kata berdasarkan Algoritma *Horspool*, seperti yang terlihat pada contoh gambar 16 di atas, setelah kata diketikkan maka akan otomatis keluar seluruh kata yang mengandung kata tersebut beserta terjemahannya. Sistem mencari atau menyeleksi satu persatu kata di *database* untuk menemukan kata yang dicari.

D. Pengujian Kelayakan

Pengujian kelayakan ini didasarkan pada penilaian *user* berupa angket yang diberikan kepada 30 orang responden yang berdomisili di Lingkar Timur Kota Bengkulu. Penilaian dilakukan dengan pengisian angket setelah penulis mempresentasikan program kepada *user*.

Dalam hal ini *user* dipilih secara acak tanpa memperhatikan strata yang ada dalam populasi dengan rentang usia 18 - 50 tahun. Skala pengukuran untuk menguji kelayakan sistem yang digunakan penulis adalah Skala *Likert*. tahapan dari uji kelayakan ini adalah :

1. Angket

Angket yang dibuat berisikan pertanyaan-pertanyaan berkaitan Pertanyaan berdasarkan beberapa variabel yaitu: tampilan, kemudahan pengguna dan kinerja dari sistem. Variabel yang ada dijadikan sebagai titik tolak untuk menyusun *item-item* instrument yang dapat berupa pertanyaan. Penyusunan bentuk jawaban dari pertanyaan menggunakan skala *Likert*. Untuk Angket penelitian ini diberikan gradasi jawaban : SB = (Sangat Baik) ; B = (Baik) ; CB = (Cukup Baik) ; TB = (Tidak Baik) ; STB = (Sangat Tidak Baik). Dengan bobot penilaian untuk setiap jawaban tersebut adalah SB = 5 ; B = 4 ; CB = 3

; TB = 2; STB = 1

2. Tabulasi Data

Angket yang dibuat kemudian dibagikan kepada responden. Teknik pemilihan responden (sampel) dilakukan dengan metode *simple random sampling* yaitu pemilihan sampel dengan cara *random* atau acak dan didapatkan 30 sampel acak. Sebelum melakukan perhitungan dengan menggunakan skala *Likert*, terlebih dahulu dilakukan pencarian intervalnya dengan persamaan berikut:

$$i = \frac{m - n}{k}$$

$$i = \frac{5 - 1}{5}$$

$$i = \frac{4}{5} = 0{,}8$$

Dengan i = 0,8 dan ketetapan skala terendah adalah 1,00, maka kategori penilaian yang dihasilkan adalah terlihat pada tabel 13:

Tabel 13 Kategori Penilaian

Interval	Kategori
4,24 - 5,04	Sangat baik
3,43 - 4.23	Baik
2,62 - 3,42	Cukup baik
1,81 - 2,61	Tidak baik
1,00 - 1,80	Sangat Tidak Baik

Untuk menguji kelayakan sistem, maka digunakan angket yang telah diberikan kepada 30 orang responden. Dari pengumpulan data menggunakan angket tersebut, maka dilakukan analisis dan perhitungan untuk uji kelayakan sistem. Hasil data yang didapat telah dipersingkat menjadi lebih jelas untuk setiap aspeknya. Berikut ini adalah hasil penilaian dari pengujian terhadap pengguna untuk masing-masing variabel tampilan, kemudahan pengguna, dan kinerja sistem :

a. Variabel Tampilan

Untuk penilaian variabel tampilan didapatkan hasil seperti pada tabel 14.

No	Tampilan (V1)	M	Frekuensi Jawaban				
			SB	B	CB	TB	STB
1	Kesesuaian warna pada aplikasi kamus	3,97	2	25	3	0	0
2	Kesesuaian tulisan pada aplikasi kamus	4	2	26	2	0	0
3	Kesesuaian tampilan pencarian kata	4,13	5	24	1	0	0
	Jumlah Frekuensi Jawaban		9	75	6	0	0
	Persentase rata-rata		10 %	83,33%	6,67 %	0 %	0%
	Total rata-rata kategori	4,03					
	Kategori	"BAIK"					

Tabel 14 Hasil Penilaian Variabel Tampilan

Dari tabel 14 terlihat bahwa penilaian terhadap variabel 1 memiliki nilai rata-rata 4,03. Berdasarkan kategori penilaian pada tabel 13 nilai rata-rata 4,03 berada dalam interval 3,43-4.23. Hal tersebut menyimpulkan bahwa penilaian pada variabel 1 termasuk kategori "Baik".

b. Variabel Kemudahan Pengguna

Untuk penilaian variabel kemudahan pengguna didapatkan hasil seperti pada tabel 15.

Tabel 15 Hasil Penilaian Variabel Kemudahan Pengguna

No	Kemudahan Pengguna (V2)	M	Frekuensi Jawaban				
			SB	B	CB	TB	STB
1	Kemudahan meng*install* aplikasi	4,27	8	22	0	0	0
2	Kemudahan mengoperasikan aplikasi kamus	4,3	9	21	0	0	0
3	Kemudahan memahami informasi yang diberikan	4.13	5	24	1	0	0
	Jumlah Frekuensi Jawaban		22	67	1	0	0
	Persentase rata-rata		24,45%	74,44 %	1,11 %	0%	0%
	Total rata-rata kategori	4,23					
	Kategori	"BAIK"					

Dari Tabel 15 terlihat bahwa penilaian terhadap variabel 2 memiliki nilai rata-rata 4,23.

Berdasarkan kategori penilaian pada Tabel 13 nilai rata-rata 4,23 berada dalam interval 3,43 - 4.23. Hal tersebut menyimpulkan bahwa penilaian pada variabel 2 termasuk kategori "Baik".

c. Variabel Kinerja Sistem

Untuk penilaian variabel kinerja sistem didapatkan hasil seperti terlihat pada Tabel 16.

Tabel 16 Hasil Penilaian Variabel Kinerja Sistem

No	Kinerja Sistem (V3)	M	Frekuensi Jawaban				
			SB	B	CB	TB	STB
1	Tujuan Aplikasi	4,07	3	26	1	0	0
2	Kesesuaian hasil informasi dengan kebutuhan pengguna	4.2	6	24	0	0	0
3	Kecepatan waktu akses aplikasi	4.5	15	15	0	0	0
	Jumlah Frekuensi Jawaban		24	65	1	0	0
	Persentase rata-rata		26,67%	72,22 %	1,11 %	0 %	0 %
	Total rata-rata kategori	4,26					
	Kategori	"SANGAT BAIK"					

Dari tabel 16 terlihat bahwa penilaian terhadap variabel 3 memiliki nilai rata-rata 4,26. Berdasarkan kategori penilaian pada tabel Tabel 13 nilai rata-rata 4,26 berada dalam interval 4,24 - 5,04. Hal tersebut menyimpulkan bahwa penilaian pada variabel 3 termasuk kategori "Sangat Baik".

d. Penilaian rata-rata Tingkat

Kelayakan Implementasi Algoritma *Horspool* Pada Aplikasi Kamus Bahasa Lintang - Indonesia Berbasis Android.

Dari tabel analisa data yang ada, didapatkan bahwa penilaian terhadap Aplikasi Kamus Bahasa Lintang - Indonesia dengan nilai rata-rata (M) yang diperoleh dari perhitungan jumlah rata-rata aspek penilaian per banyaknya aspek penilaian adalah 4,17 dan apabila dikonversi dalam tabel penilaian berada pada interval 3,43 - 4.23 yang tergolong dalam kategori Baik. Maka dapat disimpulkan bahwa Aplikasi Kamus Bahasa Lintang-Indonesia Berbasis Android telah layak untuk diimplementasikan sebagai alat bantu bagi masyarakat lintang untuk menerjemahkan bahasa Lintang ke Indonesia.

VI. PENUTUP

A. *Kesimpulan*

Berdasarkan pembahasan yang telah diuraikan, kesimpulan dari penelitian ini adalah sebagai berikut :

1. Aplikasi kamus bahasa Lintang-Indonesia berbasis Android dengan menerapkan algoritma *Horspool* dalam proses pencarian berhasil di implementasikan. Aplikasi tersebut dapat berjalan dengan baik tanpa ada *error*.
2. Berdasarkan analisis terhadap penilaian angket yang dilakukan oleh *user*, didapatkan bahwa penilaian terhadap Aplikasi Kamus Bahasa Lintang-Indonesia Berbasis Android mendapatkan nilai rata-rata (M) 4,17 yang artinya aplikasi ini tergolong dalam kategori Baik.

B. *Saran*

Berdasarkan pembahasan yang telah diuraikan, saran dari penelitian ini adalah sebagai berikut :

1. Aplikasi ini merupakan kamus 2 bahasa, kedepannya dapat dikembangkan lebih luas menjadi kamus 3 bahasa, kamus 4 bahasa bahkan kamus 5 bahasa. Seperti Kamus Bahasa Lintang-Indonesia-Inggris atau Kamus Bahasa Indonesia-Lintang-Rejang-Serawai.
2. Pengembangan aplikasi kamus Lintang-Indonesia ini dapat dilakukan dengan mengkombinasikan algoritma *string matching* lainnya dalam proses pencarian kata seperti algoritma *Knuth Morris Pratt*, *Rabin Karp*, atau *Brute Force* agar kinerjanya menjadi lebih baik.

   Menambahkan fasilitas pencarian suara (*voice search)* dan dapat juga menampilkan hasil penerjemahan kata dengan suara.

REFERENSI

[1] Charras, C., & Lecroq, T. (1997). *Handbook of Exact String Matching Algorithms.* United Kingdom: Oxford University Press.

[2] Sheik, S., Anggarwal, S., Poddar, A., Balakhrisnan, N., & Sekar, K. (2004). *A Fast Pattern Matching Algoritm*. *Journal of Chemical Information and Computer Science vol. 44 No. 4* , 1251-1256.

[3] National Institute of Standards and Technology. (1998). *Dictionary of Algorithms and Data Structures*. Dipetik Februari 10, 2017, dari Dictionary of Algorithms and Data Structures: https://xlinux.nist.gov/dads/

[4] Binstock, A., & Rex, J. (1995). *Practical Algorithms for Programmers.* Addison-Wesley Professional.

[5] Horspool, R. (1980). *Practical Fast Searching in String*. *Journal Software Practice and Experience Vol. 10* , 501-506.

[6] Beeza-Yates, R., & Regnier, M. (1992). *Average Running Time of the Boyer-Moore-Horspool Algorithm*. *Journal Theoretical Computer Science Vol. 92 No.1* , 19-31.

[7] Singh, R., & Verma, H. (2011). *A Fast String Matching Algorithm*. *International Journal of Computer Technology and Application Vol.2 No.6* , 1877-1883.

[8] Bustari, S. (2010). *Diktat Kamus Lintang IV Lawang-Indonesia.* Empat Lawang.
[9] A.S, R., & Shalahuddin, M. (2011). *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek).*Bandung: Modula.
[10] Sugiyono. (2011). *Metode Penelitian Kuantitatif Kualitatif dan R&D.* Bandung: Alfabeta.
[11] Fitri, M. (2013). *Rancang Bangun Aplikasi Kamus Bahasa Indonesia-Minang, Minang-Indonesia Berbasis Android.* Bengkulu: Universitas Bengkulu.
[12] Martinus, J. (2013). *Rancang Bangun Aplikasi Dwibahasa (Indonesia-Rejang) Berbasis Android.* Bengkulu: Universitas Bengkulu.
[13] Alfajri, M. Z. (2015). *Aplikasi Kamus Istilah Biologi Berbasis Android.* Yogyakarta: STMIK AMIKOM.
[14] Purwanto, A. (2013). *Pembuatan Aplikasi Kamus Bahasa Indonesia - Bahasa Jawa-Aksara Jawa Berbasis Sistem Android.*Yogyakarta: STMIK AMIKOM.
[15] Sembiring, J. P. (2013). *Perancangan Aplikasi Kamus Bahasa Indonesia – Karo Online Berbasis Web Dengan Metode Sequential Search*. *Pelita Informatika Budi Darma Volume. IV No.2*