

**IMPLEMENTASI ALGORITMA CAVERPHONE 2.0 UNTUK
PENCARIAN KATA BERDASARKAN KEMIRIPAN
PENGUCAPAN PADA APLIKASI KAMUS INGGRIS-INDONESIA**

SKRIPSI

Konsentrasi Komputasi Cerdas dan Visualisasi

Untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer



Disusun Oleh :

ANDREAS TOMMY CHRISTIAWAN

NIM. 0810680022

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA/ ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

**IMPLEMENTASI ALGORITMA CAVERPHONE 2.0 UNTUK
PENCARIAN KATA BERDASARKAN KEMIRIPAN
PENGUCAPAN PADA APLIKASI KAMUS INGGRIS-INDONESIA**

SKRIPSI

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer



Disusun oleh :

ANDREAS TOMMY CHRISTIAWAN

NIM. 0810680022

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Arief Andy Soebroto, ST., MKom.

NIP. 19720425 199903 1 002

Dosen Pembimbing II

Drs. Achmad Ridok, M.Kom

NIP. 19680825 199403 1 002

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA *CAVERPHONE 2.0* UNTUK
PENCARIAN KATA BERDASARKAN KEMIRIPAN PENGUCAPAN
PADA APLIKASI KAMUS-INGGRIS INDONESIA

SKRIPSI
KONSENTRASI KOMPUTASI CERDAS DAN VISUALISASI

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh :

ANDREAS TOMMY CHRISTIAWAN

NIM. 0810680022

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 25 Maret 2013

Penguji I

Candra Dewi, S.Kom., M.Sc.
NIP. 19771114 200312 2 001

Penguji II

Fitra A. Bachtiar, ST., M.Eng.
NIK. 840628 16 1 1 0427

Penguji III

Himawat Arvadita, ST., M.Sc.
NIP. 19801018 200801 1 003

Mengetahui,

Ketua Program Studi Teknik Informatika



Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah dipublikasikan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Maret 2013

Mahasiswa,



Andreas Tommy Christiawan
0810680022

PENGANTAR

Puji syukur penulis panjatkan kehadiran Tuhan Yang Maha Esa karena hanya dengan rahmat dan karunia-Nya, penulis telah menyelesaikan skripsi dengan judul “Implementasi Algoritma *Coverphone 2.0* untuk Pencarian Kata berdasarkan Kemiripan Pengucapan pada Aplikasi Kamus Inggris-Indonesia” dengan baik. Skripsi ini diajukan untuk memenuhi persyaratan kelulusan dalam menyelesaikan pendidikan dan memperoleh gelar Sarjana Komputer konsentrasi Komputasi Cerdas dan Visualisasi di Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Dalam menyelesaikan skripsi ini, penulis banyak mendapat bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Bapak Arief Andy Soebroto, ST., M.Kom. selaku Dosen Pembimbing I dan Bapak Drs. Achmad Ridok, M.Kom. selaku Dosen Pembimbing II yang telah memberikan banyak arahan, masukan, dan bimbingannya kepada penulis selama proses penyelesaian skripsi ini.
2. Bapak Wiji Setiawan P. dan Ibu Christiana Christien Berlina selaku orang tua penulis beserta seluruh keluarga yang telah memberikan dukungan dan bantuan baik lahir maupun batin, dan doa yang senantiasa teriring demi terselesaikannya skripsi ini.
3. Bapak Drs. Marji, MT. dan Bapak Issa Arwani, S.Kom, M.Sc. selaku Ketua dan Sekretaris Program Studi Teknik Informatika.
4. Ibu Candra Dewi, S.Kom., M.Sc. selaku dosen penguji I, Bapak Fitra A. Bachtiar, ST., M.Eng. selaku dosen penguji II, serta Bapak Himawat Aryadita, ST., M.Sc. selaku dosen penguji III yang telah memberikan kritik dan saran untuk skripsi ini.
5. Bapak Himawat Aryadita, ST.,M.Sc. selaku dosen penasehat akademik yang senantiasa memberikan nasehat kepada penulis selama menempuh masa studi.
6. Seluruh Dosen Jurusan Teknik Informatika dan Civitas Akademika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya yang telah

banyak memberi bantuan dan dukungan selama penulis menempuh masa studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

7. Davi Nugraha P.P, Febri Abdullah, Ari Hernawan, M. Aminul Akbar, Nanda Yustina, Dewi Rokhmah yang telah memberikan banyak bantuan dalam proses pengerjaan skripsi ini.
8. Sahabat-sahabat tercinta Ayok Luthfi Hidayat, Iqra Ahmadya, Arif Fahmi, Rahma Indah, Gita Ayu Anjayani, Novita Rudiarsih, Ninda Arfi, Niken Hendrakusma, Cantika Perviana, Fauziah Mayasari, Rafika Dewi Mutik, Kusumaning Hati Pambayun yang senantiasa memberikan semangat dan doa demi terselesaikannya skripsi ini.
9. Teman-teman angkatan 2008 Teknik Informatika tercinta yang telah memberikan bantuan dan pengalaman selama menjadi mahasiswa di Universitas Brawijaya.
10. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Maka, saran dan kritik yang membangun dari semua pihak sangat diharapkan demi penyempurnaan selanjutnya. Semoga skripsi ini dapat bermanfaat dan berguna bagi semua pihak, baik penulis maupun pembaca.

Malang, 1 Februari 2013

Penulis

ABSTRAK

Andreas Tommy Christiawan. 2012. Implementasi Algoritma *Caverphone 2.0* untuk Pencarian Kata berdasarkan Kemiripan Pengucapan pada Aplikasi Kamus Inggris-Indonesia. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Arief Andy Soebroto, ST., M.Kom. dan Drs. Achmad Ridok, M.Kom.

Bahasa Inggris merupakan bahasa yang berlaku secara universal. Di era modern sekarang ini, setiap orang dituntut untuk dapat menguasai bahasa Inggris. Ada beberapa hal yang sangat penting bagi setiap orang untuk mengasah kemampuan berbahasa Inggris. Salah satunya adalah *pronunciation* atau pengucapan. *Pronunciation* dianggap sulit oleh sebagian orang karena ada beberapa bunyi yang tidak ada dalam bahasa Indonesia. Ditambah lagi ada banyak kosakata dalam bahasa Inggris yang memiliki kemiripan pengucapan. Untuk mencari kata bahasa Inggris yang memiliki kemiripan pengucapan dapat digunakan beberapa algoritma salah satunya adalah algoritma *Caverphone 2.0*. Algoritma *Caverphone 2.0* mengubah setiap kata menjadi kode fonetis sehingga dapat diketahui kata bahasa Inggris apa saja yang memiliki kemiripan pengucapan. Implementasi algoritma *Caverphone 2.0* dalam kamus Inggris-Indonesia dapat membantu seseorang untuk mengasah kemampuan *pronunciation* bahasa Inggris mereka. Analisis kebutuhan dilakukan dengan menganalisis *Use Case Diagram*. Implementasi perancangan menggunakan bahasa pemrograman Java.

Pengujian fungsionalitas terhadap 6 tindakan dalam *use case diagram* dengan metode *black-box testing* menunjukkan bahwa Aplikasi Kamus Inggris-Indonesia dengan algoritma *Caverphone 2.0* ini telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

Hasil pengujian ini menunjukkan bahwa kualitas hasil keluaran tertinggi Aplikasi Kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0* adalah sebesar 100%. Sedangkan kualitas hasil keluaran terendah pada Aplikasi Kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0* adalah 16,44%. Hasil pengujian untuk setiap jenis klasifikasi konsonan dapat dilihat pada bab pengujian.

Kata kunci : kamus inggris-indonesia, *phonetic string matching*, algoritma *caverphone 2.0*

ABSTRACT

Andreas Tommy Christiawan. 2012. *Algorithm Caverphone 2.0 Implementation for Search word similarity based on the pronunciation of English-Indonesian Dictionary Application . Program of Informatic Technology and Computer Science, Brawijaya University.*

Advisors : Arief Andy Soebroto, ST., M.Kom. and Drs. Achmad Ridok, M.Kom.

English is an universal language. In today's modern era, every person required to know English. There are some things that are very important for everyone to whet English language skills. One is the pronunciation. Pronunciation is considered difficult by most people because there are some sounds that do not exist in Indonesia language. Beside that, there are a lot of words in English that have similar pronunciation. Multiple algorithms can be used to search for English words that have similar pronunciation. One of the algorithm is Caverphone 2.0. Caverphone 2.0 algorithm change every single word into phonetic code. So, we can know any English words that have similar pronunciation. The implementation of Caverphone 2.0 algorithm to English-Indonesia dictionary can help people to sharpen their English pronunciation skills. The analysis is done by analyzing of Use Case Diagram. The implementation of the design using Java programming language.

The functionality testing of the 6 actions in the use case diagram with black-box testing methods, dictionary English-Indonesia with Caverphone 2.0 algorithm application to meet the needs that have been described in the requirement analysis phase.

The test results showed that the highest quality of the output is 100%. While the lowest quality of the output is 16,44%. The test results for each type of classification of consonants can be seen in the test section.

Key words : english-indonesia dictionary, phonetic string matching, caverphone 2.0 algorithm

DAFTAR ISI

PENGANTAR	i
ABSTRAK	iii
ABSTRACT	iv
DAFTAR ISI	v
DAFTAR TABEL	viii
DAFTAR GAMBAR	ix
DAFTAR LAMPIRAN	xi
I. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	2
1.3 Rumusan Masalah	2
1.4 Batasan Masalah	2
1.5 Tujuan	3
1.6 Manfaat	3
1.7 Sistematika Penulisan	4
II. TINJAUAN PUSTAKA	5
2.1 Kajian Pustaka	5
2.2 Pengenalan Sistem Temu Kembali Informasi	6
2.2.1 Tujuan Sistem Temu Kembali Informasi	6
2.2.2 <i>Tokenization</i>	6
2.2.3 Pembetulan Ejaan	7
2.3 Fonetik Bahasa Inggris	7
2.3.1 Klasifikasi Konsonan	7
2.4 <i>Phonetic String Matching</i>	9
2.5 Kamus	10
2.6 Bahasa Pemrograman Java	11
2.7 <i>Unified Modelling Language</i>	13
2.7.1 Diagram pada UML	13
2.8 Algoritma <i>Levenshtein Distance</i>	15

2.9 Algoritma <i>Caverphone 2.0</i>	16
III. METODOLOGI PENELITIAN.....	20
3.1 Studi Literatur	21
3.2 Analisis Kebutuhan.....	21
3.2.1 Analisis Masukan Sistem.....	21
3.2.2 Analisis Keluaran Sistem.....	21
3.3 Pengumpulan Data	22
3.4 Perancangan	22
3.4.1 Desain Perancangan Aplikasi Kamus Inggris-Indonesia menggunakan Algoritma <i>Caverphone 2.0</i>	22
3.4.2 Perancangan Algoritma <i>Levenshtein Distance</i>	23
3.4.3 Perancangan Algoritma <i>Caverphone 2.0</i>	23
3.4.4 Perancangan Antarmuka Aplikasi	23
3.5 Implementasi	23
3.6 Pengujian Perangkat Lunak.....	24
3.7 Pengambilan Kesimpulan dan Saran.....	25
IV. PERANCANGAN	26
4.1 Perancangan Aplikasi.....	26
4.1.1 Diagram Blok.....	26
4.1.2 Analisis Kebutuhan	28
4.1.2.1 Identifikasi Aktor.....	28
4.1.2.2 Daftar Kebutuhan.....	28
4.1.3 <i>Use Case Diagram</i>	29
4.1.4 <i>Sequence Diagram</i>	31
4.1.4.1 <i>Sequence Diagram</i> Kata Mirip Pengucapan	31
4.1.4.2 <i>Sequence Diagram</i> Koreksi Ejaan Salah	32
4.1.4.3 <i>Sequence Diagram</i> Arti Kata dan Pengucapan	33
4.1.4 <i>Class Diagram</i>	34
4.2 Perancangan Tabel	35
4.3 Perancangan Algoritma.....	37
4.3.1 Perancangan Algoritma <i>Caverphone 2.0</i>	37

4.3.2 Perancangan Algoritma <i>Levenshtein Distance</i>	40
4.4 Perancangan Antarmuka	42
4.5 Contoh Penghitungan Manual Algoritma <i>Caverphone 2.0</i>	44
4.6 Contoh Pencocokan dengan Metode <i>Phonetic String Matching</i>	47
V. IMPLEMENTASI.....	50
5.1 Spesifikasi Sistem	50
5.1.1 Spesifikasi Perangkat Keras.....	51
5.1.2 Spesifikasi Perangkat Lunak.....	51
5.2 Implementasi Algoritma.....	51
5.3.1 Implementasi Algoritma <i>Caverphone 2.0</i>	52
5.3.2 Implementasi Algoritma <i>Levenshtein Distance</i>	53
5.4 Implementasi Antarmuka.....	56
VI. PENGUJIAN DAN ANALISIS.....	62
6.1 Pengujian.....	62
6.1.1 Pengujian Validasi	62
6.1.1.1 Hasil Pengujian Validasi.....	65
6.1.2 Pengujian Akurasi.....	68
6.2 Analisis.....	71
VII. PENUTUP	73
7.1 Kesimpulan	73
7.2 Saran.....	73
DAFTAR PUSTAKA	DP-1
LAMPIRAN	L-1



DAFTAR TABEL

Tabel 2.1	Pembagian Konsonan Bahasa Inggris.....	9
Tabel 2.2	Tahap Algoritma <i>Caverphone 2.0</i>	17
Tabel 4.1	Identifikasi Aktor.....	28
Tabel 4.2	Daftar Kebutuhan Fungsional <i>User</i>	28
Tabel 4.3	<i>Rule</i> Algoritma <i>Caverphone 2.0</i>	38
Tabel 4.4	Contoh Penghitungan Manual Algoritma <i>Caverphone 2.0</i>	44
Tabel 5.1	Spesifikasi Perangkat Keras Komputer	51
Tabel 5.2	Spesifikasi Perangkat Lunak Komputer	51
Tabel 6.1	<i>Test Case</i> untuk Pengujian Validasi <i>Input</i> Kata	62
Tabel 6.2	<i>Test Case</i> untuk Pengujian Validasi Arti Kata	63
Tabel 6.3	<i>Test Case</i> untuk Pengujian Validasi Pengucapan Kata	63
Tabel 6.4	<i>Test Case</i> untuk Pengujian Validasi Kata Mirip Pengucapan	64
Tabel 6.5	<i>Test Case</i> untuk Pengujian Validasi Cek Kata Tidak Tersedia	64
Tabel 6.6	<i>Test Case</i> untuk Pengujian Validasi Bantuan Cara Pengucapan	65
Tabel 6.7	Hasil Pengujian Validasi Aplikasi	65
Tabel 6.50	Hasil Pengujian Akurasi Algoritma.....	70

DAFTAR GAMBAR

Gambar 2.1	Proses Pembentukan dan Eksekusi Program di Dalam Java	12
Gambar 2.2	Skema Pengkompilasian hingga Pengeksekusian Kode Java.....	12
Gambar 2.3	Unsur-Unsur Pembentuk UML.....	13
Gambar 2.4	Algoritma <i>Levenshtein Distance</i>	16
Gambar 3.1	Diagram Alir Metode Penelitian.....	20
Gambar 4.1	Pohon Perancangan.....	26
Gambar 4.2	Diagram Blok Perancangan Aplikasi.....	27
Gambar 4.3	Diagram <i>Use Case</i> Aplikasi.....	29
Gambar 4.4	<i>Sequence Diagram</i> Kata Mirip Pengucapan.....	31
Gambar 4.5	<i>Sequence Diagram</i> Koreksi Ejaan Salah	32
Gambar 4.6	<i>Sequence Diagram</i> Arti Kata dan Pengucapan.....	33
Gambar 4.7	<i>Class Diagram</i> Aplikasi.....	34
Gambar 4.8	Struktur Tabel Aplikasi.....	36
Gambar 4.9	<i>Pseudocode</i> Algoritma <i>Caverphone 2.0</i>	37
Gambar 4.10	Diagram Alir Algoritma <i>Caverphone 2.0</i>	39
Gambar 4.11	<i>Pseudocode</i> Algoritma <i>Levenshtein Distance</i>	40
Gambar 4.12	Diagram Algoritma <i>Levenshtein Distance</i>	41
Gambar 4.13	Perancangan Antarmuka Aplikasi Halaman <i>Home</i>	42
Gambar 4.14	Perancangan Antarmuka Aplikasi Halaman Bantuan.....	43
Gambar 4.15	Diagram Blok Pencocokan Metode <i>Phonetic String Matching</i>	48
Gambar 5.1	Pohon Implementasi	50
Gambar 5.2	Tampilan <i>Sourcecode</i> Proses Pendeklarasian Variabel <i>Class</i> Fonetik	61
Gambar 5.3	Tampilan <i>Sourcecode</i> Proses Penghilangan Simbol-Simbol.....	53
Gambar 5.4	Tampilan <i>Sourcecode</i> Proses <i>Levenshtein Distance</i>	54
Gambar 5.5	Tampilan <i>Sourcecode</i> Proses Pendeklarasian Variabel <i>Main</i> <i>Program</i>	57
Gambar 5.6	Tampilan <i>Sourcecode</i> Konstruktork <i>Main Program</i>	57
Gambar 5.7	Tampilan <i>Sourcecode Method</i> Isidata.....	58

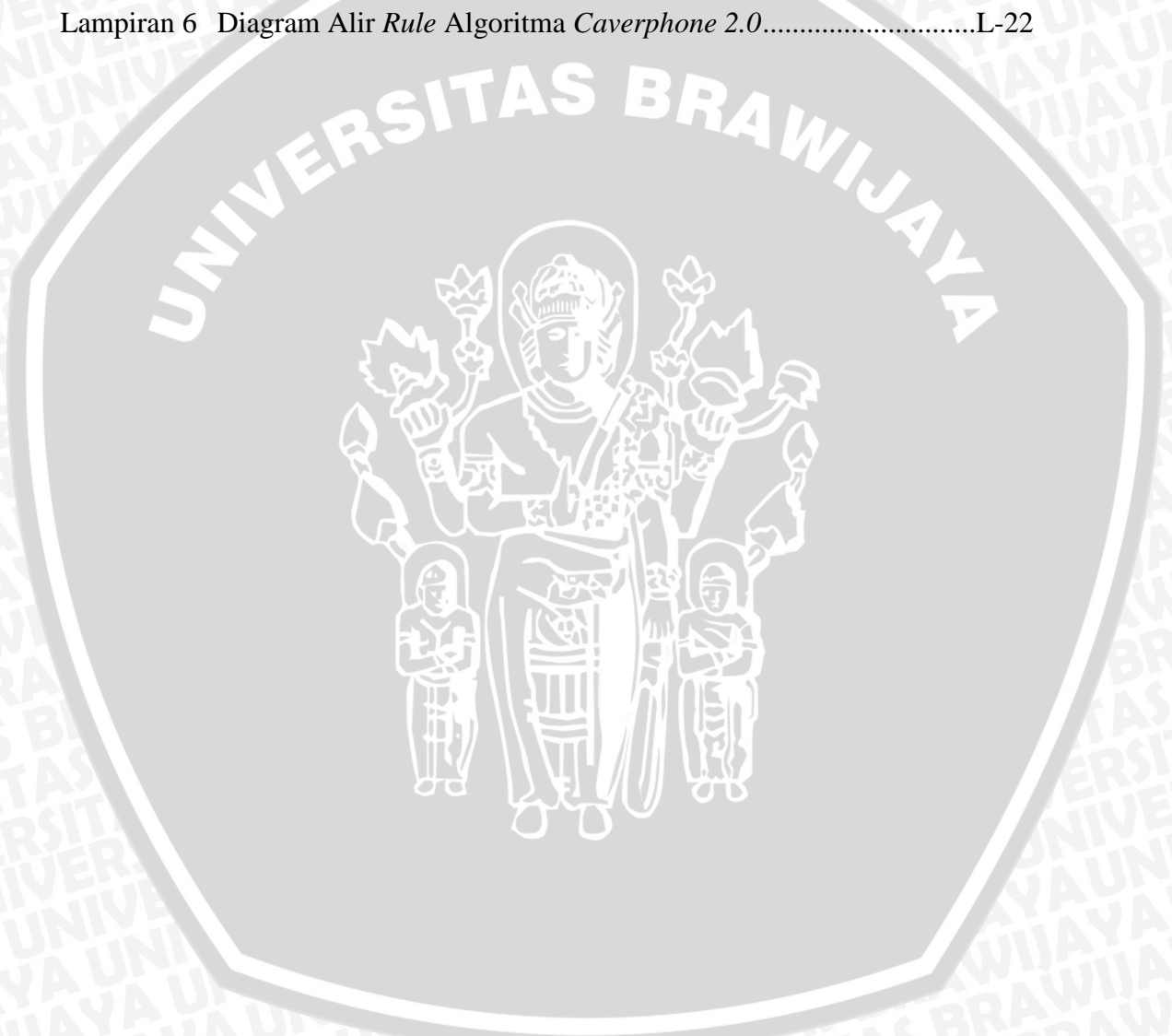
Gambar 5.8 Tampilan Aplikasi Kamus Inggris-Indonesia menggunakan Algoritma Caverphone 2.061

Gambar 6.1 Diagram Alir Pengujian Akurasi69



DAFTAR LAMPIRAN

Lampiran 1	Daftar Kata yang Digunakan pada Pengujian Akurasi	L-1
Lampiran 2	Hasil Pengujian Akurasi pada Awal Kata.....	L-4
Lampiran 3	Hasil Pengujian Akurasi pada Tengah Kata	L-9
Lampiran 4	Hasil Pengujian Akurasi pada Akhir Kata.....	L-14
Lampiran 5	<i>Sourcecode</i> Tahap Perubahan Kata menjadi Kode Fonetis	L-19
Lampiran 6	Diagram Alir <i>Rule</i> Algoritma <i>Caverphone 2.0</i>	L-22



BAB I PENDAHULUAN

1.1 Latar Belakang Permasalahan

Bahasa Inggris merupakan salah satu bahasa resmi di dunia yang digunakan untuk berkomunikasi. Dalam perkembangan teknologi saat ini, bahasa Inggris merupakan suatu hal yang wajib kita kuasai khususnya di dalam dunia bisnis dan pergaulan internasional. Untuk menguasai bahasa Inggris diperlukan pembelajaran secara terus menerus atau dapat menggunakan sarana lain yakni dengan bantuan alat penerjemah yang dalam hal ini adalah kamus [HAM-12].

Pronunciation atau pengucapan adalah hal terbesar yang dapat dilihat orang dari bahasa Inggris. *Pronunciation* harus dipelajari sekalipun kita menganggap sudah bisa berkomunikasi bahasa Inggris. Oleh karena itu, *pronunciation* dianggap sulit bagi sebagian orang karena ada beberapa bunyi dalam bahasa Inggris yang tidak terdapat dalam bahasa Indonesia [PAC-10].

Terdapat lebih dari 40 bunyi bahasa Inggris dari 26 alfabet. Sistem pengejaan bahasa Inggris selalu gagal untuk merepresentasikan sebuah keambiguan bunyi kata-kata. Alfabet fonetik dapat digunakan untuk menangani masalah keambiguan bunyi karena setiap simbol merepresentasikan satu bunyi dan setiap bunyi direpresentasikan hanya oleh satu simbol [NOO-10].

Kemiripan bunyi atau pengucapan dalam bahasa Inggris dapat dikenali dengan menggunakan metode pencocokan *string*. Metode pencocokan berdasarkan kemiripan pengucapan disebut dengan metode *phonetic string matching*. Salah satu algoritma untuk pencocokan *string* berdasarkan kemiripan pengucapan adalah algoritma *Caverphone*. Algoritma *Caverphone* mempunyai dua buah versi yaitu *Caverphone* dan *Caverphone 2.0*. Algoritma *Caverphone 2.0* yang merupakan salah satu algoritma untuk proses *phonetic string matching* dapat digunakan untuk pencocokan kata bahasa Inggris secara umum [HOO-04].

Penggabungan metode *phonetic string matching* menggunakan algoritma *Caverphone 2.0* dan penggunaan alfabet fonetik untuk mengenali pengucapan bahasa Inggris pada aplikasi kamus Inggris-Indonesia dapat dijadikan salah satu

solusi untuk mempelajari pengucapan kata bahasa Inggris. Sehubungan dengan hal tersebut penulis mengambil judul “Implementasi Algoritma *Caverphone 2.0* untuk Pencarian Kata Berdasarkan Kemiripan Pengucapan pada Aplikasi Kamus Inggris-Indonesia”. Harapan yang ingin diwujudkan penulis adalah agar perangkat lunak ini nantinya dapat membantu melatih *pronunciation* atau pengucapan pengguna aplikasi kamus sehingga kemampuan berbahasa Inggris pengguna pun dapat meningkat.

1.2 .Identifikasi Masalah

Identifikasi masalah pada pembahasan ini terkait dengan hal – hal seputar perancangan dan pengembangan, teori yang mendukung, dan penjelasan tentang algoritma *Caverphone 2.0*. Selain itu juga akan dibahas tentang implementasi algoritma *Caverphone 2.0* dalam pembuatan Aplikasi Kamus Inggris-Indonesia.

1.3 Rumusan Masalah

Berdasarkan permasalahan yang diangkat pada bagian latar belakang, maka rumusan masalah dikhususkan pada:

1. Bagaimana perancangan Aplikasi Kamus Inggris-Indonesia menggunakan Algoritma *Caverphone 2.0*?
2. Bagaimana implementasi algoritma *Caverphone 2.0* pada Aplikasi Kamus Inggris-Indonesia?
3. Bagaimana hasil pengujian algoritma *Caverphone 2.0* pada Aplikasi Kamus Inggris-Indonesia?

1.4 Batasan Masalah

Agar diperoleh hasil pembahasan yang sesuai dengan apa yang diharapkan, maka perlu diberikan pembatasan masalah pada pengembangan perangkat lunak ini yaitu:

1. Daftar kata yang digunakan untuk pembuatan aplikasi berjumlah 23.623 kata bahasa Inggris. Sumber daftar kata bahasa Inggris didapatkan dari internet dengan alamat <http://jatiblack.com/membuat-kamus-inggris-indonesia-dengan-php>.

2. Pengujian akurasi yang dilakukan sebatas pencocokan hasil keluaran kata sesuai dengan jenis klasifikasi konsonan bahasa Inggris.

1.5 Tujuan

Berdasarkan permasalahan yang diangkat pada bagian rumusan masalah, maka tujuan dari pengembangan perangkat lunak ini dikhususkan pada:

1. Merancang Aplikasi Kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0*.
2. Mengimplementasikan algoritma *Caverphone 2.0* pada Aplikasi Kamus Inggris-Indonesia.
3. Mendapatkan hasil pengujian dari algoritma *Caverphone 2.0* yang telah diimplementasikan pada Aplikasi Kamus Inggris-Indonesia.

1.6 Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari implementasi algoritma *Caverphone 2.0* pada Aplikasi Kamus Inggris-Indonesia ini adalah sebagai berikut:

1. Bagi penulis:
 - a. Mengaplikasikan ilmu yang didapat selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya.
 - b. Mendapatkan pemahaman tentang perancangan dan pengembangan aplikasi kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0*.
2. Bagi pengguna
 - a. Membantu pengguna untuk mempelajari pengucapan kata dalam bahasa Inggris dan membandingkan dengan kata lain yang memiliki kemiripan pengucapan.
 - b. Meningkatkan kemampuan *pronunciation* pengguna dengan cara melatih pengucapan setiap kata bahasa Inggris dan kata lain yang memiliki kemiripan pengucapan.

1.7 Sistematika Penulisan

Penyusunan tugas akhir ini menggunakan kerangka pembahasan yang tersusun sebagai berikut:

- a. **Bab I Pendahuluan:** bab ini merupakan dasar dari penyusunan tugas akhir ini yang terdiri dari latar belakang, identifikasi masalah, rumusan masalah, batasan masalah, tujuan, manfaat, sistematika penulisan dan metodologi.
- b. **Bab II Landasan Teori:** bab ini berisi referensi atau sumber-sumber yang berhubungan dengan permasalahan dalam tugas akhir yang meliputi : Sistem Temu Kembali Informasi (*Information Retrieval*), fonetik bahasa inggris, kamus, metode *phonetic string matching*, Java, *MySQL*, *Unified Modeling Language*, algoritma *Levenshtein Distance*, dan algoritma *Caverphone 2.0*.
- c. **Bab III Metodologi penelitian:** bab ini menjelaskan bagaimana perancangan dan pembuatan aplikasi kamus inggris-indonesia menggunakan algoritma *Caverphone 2.0* dilakukan mulai dari pengumpulan data, pembuatan diagram alir, dan pembuatan UML (Unified Modeling Language).
- d. **Bab IV Perancangan Sistem:** bab ini menjelaskan hasil dari proses perancangan dan pembuatan aplikasi kamus inggris-indonesia menggunakan algoritma *Caverphone 2.0* dalam mengatasi permasalahan yang diuraikan pada rumusan masalah.
- e. **Bab V Pengujian Sistem:** bab ini menjelaskan hasil pengujian aplikasi meliputi pengujian validasi dan pengujian akurasi.
- f. **Bab VI Kesimpulan dan Saran:** bab ini menjelaskan kesimpulan yang dapat diambil dari proses perancangan dan pembuatan aplikasi kamus inggris-indonesia menggunakan algoritma *Caverphone 2.0* disertai saran yang mendukung untuk penelitian berikutnya.

BAB II

TINJAUAN PUSTAKA

Pada bab ini berisi pembahasan tentang kajian pustaka dan dasar teori yang berhubungan dengan pengembangan perangkat lunak yang dilakukan. Kajian pustaka berupa pembahasan dari penelitian sebelumnya dan dasar teori berupa penjelasan tentang Sistem Temu Kembali Informasi (*Information Retrieval*), dasar teori mengenai *phonetic string matching*, kamus, Java, *MySQL*, *Unified Modeling Language*, Algoritma *Levenshtein Distance*, dan algoritma *Caverphone 2.0*.

2.1 Kajian Pustaka

Dewi Primasari dari Institut Pertanian Bogor pernah melakukan penelitian tentang kinerja beberapa algoritma *phonetic string matching*. Objek penelitian yang diteliti adalah algoritma *soundex*, algoritma *phonix4*, *phonix8*, dan *phonixE*. Kesimpulan dari penelitian tersebut adalah bahwa algoritma *phonix4* dan *phonixE* memiliki kinerja yang lebih baik jika dibandingkan dengan algoritma *soundex* dan *phonix8*. Selain itu juga disimpulkan bahwa algoritma *soundex* dan *phonix* dapat bekerja dengan baik pada dokumen berbahasa indonesia. [PRI-97].

Mokhamad Syaroni dan Rinaldi Munir pernah melakukan penelitian untuk analisis algoritma *phonetic string matching*. Objek penelitian yang diteliti adalah algoritma *soundex*, algoritma *metaphone*, dan *caverphone*. Kesimpulan dari penelitian tersebut adalah bahwa *phonetic string matching* sangat bermanfaat untuk pencarian kata dengan kata masukan yang salah. Selain itu, disimpulkan bahwa kemampuan terbaik ditunjukkan oleh algoritma *caverphone* [SYA-05].

Kesimpulan yang dapat diambil adalah pencocokan *string* dengan metode *phonetic string matching* sangat bermanfaat untuk pencarian kata yang salah. Selain itu, kemampuan terbaik dari algoritma *phonetic string matching* ditunjukkan oleh algoritma *caverphone*. Berdasarkan hal tersebut, maka penulis menggunakan algoritma *caverphone 2.0* sebagai objek tugas akhir yang berjudul

“Implementasi Algoritma *Caverphone 2.0* untuk Pencarian Kata berdasarkan Kemiripan Pengucapan pada Aplikasi Kamus Inggris-Indonesia”.

2.2 Pengenalan Sistem Temu Kembali Informasi (*Information Retrieval*)

Arti dari Sistem Temu Kembali Informasi (*Information Retrieval*) dapat menjadi sangat luas. Namun, dalam lingkup pembelajaran Sistem temu Kembali Informasi (*Information Retrieval*) dapat didefinisikan sebagai berikut [MAN-09]:

“*Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)*”.

Pernyataan di atas dapat diartikan sebagai berikut.

“Sistem temu Kembali Informasi adalah menemukan material pada lingkungan yang tidak terstruktur (biasanya teks) yang memuaskan kebutuhan informasi dari koleksi yang besar (biasanya disimpan dalam komputer)”.

1.2.1 Tujuan Sistem Temu Kembali Informasi (*Information Retrieval*)

Untuk beberapa tujuan, ada sedikitnya 3 tujuan yang ingin dicapai dalam penggunaan Sistem Temu Kembali Informasi (*Information Retrieval*) yaitu [MAN-09]:

1. Untuk memproses koleksi dokumen yang besar secara cepat.
2. Untuk melakukan operasi pencocokan yang lebih fleksibel.
3. Untuk mendapatkan perankingan hasil temu kembali.

1.2.2 *Tokenization*

Tokenization adalah sebuah proses melakukan pemotongan sekumpulan karakter yang berurutan menjadi bagian-bagian yang disebut *tokens*. Mungkin juga dalam waktu yang sama membuang karakter tertentu seperti tanda baca [MAN-09].

Contoh:

Input : Friends, Romans, Countrymen, lend me you ears; (*sentence/ kalimat*)

Output :

Friends	Romans	Countrymen	lend	me	your	ears
---------	--------	------------	------	----	------	------

Sumber: [MAN-09]

1.2.3 Pembetulan Ejaan

Kesalahan yang sering terjadi dalam pencarian data pada Sistem Temu Kembali Informasi (*Information Retrieval*) adalah kesalahan penulisan ejaan kata. Kesalahan semacam ini dapat mengurangi akurasi pencarian hasil *query* yang dimasukkan. Untuk meminimalisir terjadinya hal tersebut, maka Sistem Temu Kembali Informasi (*Information Retrieval*) menyertakan pembetulan ejaan kata di dalamnya.

Ada dua prinsip dasar yang paling mendasari algoritma pengkoreksi ejaan yaitu [MAN-09]:

1. Dari berbagai alternatif pembetulan ejaan untuk *query* yang salah eja, pilih yang paling “dekat”.
2. Ketika ada dua pembetulan ejaan *query* yang sama, maka pilih yang paling umum.

2.3 Fonetik Bahasa Inggris

Menurut Bertil Malmberg sebagaimana dikutip oleh [SYA-05], fonetik (*phonetics*) adalah ilmu yang menyelidiki bunyi bahasa tanpa melihat fungsi bunyi itu sebagai pembeda makna dalam suatu bahasa (*language*).

2.3.1 Klasifikasi Konsonan

Menurut Jones Daniel sebagaimana dikutip oleh [SYA-05], konsonan didasarkan pada alat bicara yang menghasilkannya dapat dibedakan menjadi tujuh kelompok yaitu sebagai berikut:

1. *Labial* atau bunyi bibir yang dapat dibedakan lagi menjadi dua golongan yaitu sebagai berikut:
 - a. *Bilabial*, bunyi diartikulasi oleh dua bibir. Contoh: bunyi p, b, m, w.
 - b. *Labio-dental*, bunyi diartikulasi oleh bibir bawah dan gigi atas. Contoh: bunyi f, v.
2. *Dental*, bunyi diartikulasi oleh ujung lidah dengan gigi atas. Contoh: bunyi th (dalam kata *thin*).
3. *Alveolar*, bunyi diartikulasi oleh ujung lidah dengan punggung gigi (*teeth ridge*). Contoh: bunyi d, t, n, l, r, s, z.

4. *Palato-alveolar*, bunyi yang memiliki artikulasi *alveolar* diikuti dengan naiknya lidah sampai pada langit-langit mulut secara simultan. Contoh: bunyi c, j, sh (dalam kata *show*).
5. *Palatal*, bunyi diartikulasi oleh bagian depan lidah dengan langit-langit keras (*hard palate*). Contoh: bunyi y.
6. *Velar*, bunyi diartikulasi oleh bagian belakang lidah dengan langit-langit lunak (*soft palate*). Contoh: bunyi k, g.
7. *Glottal*, bunyi diartikulasi oleh *glottis*. Contoh: bunyi h.

Sedangkan berdasarkan cara artikulasi (dihambat), menurut Jones Daniel sebagaimana dikutip [SYA-05] konsonan dibedakan menjadi delapan golongan yaitu:

1. Konsonan hambat letup (*plosive*), merupakan konsonan yang terjadi dengan hambatan penuh arus udara kemudian hambatan itu dilepaskan secara tiba-tiba. Contoh: b, d, g, k, p, t.
2. Konsonan nasal atau sengau (*nasal*), merupakan konsonan yang dibentuk dengan menghambat rapat (menutup) jalan udara dari paru-paru melalui rongga mulut, beserta dengan itu langit-langit lunak beserta anak tekaknya diturunkan sehingga udara keluar melalui rongga hidung. Contoh: m, n.
3. Konsonan paduan (*affricate*), merupakan konsonan hambat jenis khusus dimana proses terjadinya dengan menghambat penuh arus udara dari paru-paru, kemudian hambatan itu dilepaskan secara pelan-pelan. Tempat artikulasinya pada *palato-alveolar*. Contoh: c, j.
4. Konsonan sampingan (*lateral*), merupakan konsonan yang dibentuk dengan menutup arus udara di tengah rongga mulut sehingga udara keluar melalui kedua samping atau sebuah samping saja. Tempat artikulasinya pada *alveolar*. Contoh: l.
5. Konsonan geseran (*fricative*), merupakan konsonan yang dibentuk dengan menyempitkan jalannya arus udara yang dihembuskan dari paru-paru, sehingga jalannya udara terhalang dan keluar dengan bergeser. Contoh: f, v, r, s, z, th (dalam kata *thin*), sh (dalam kata *show*), h.

6. Konsonan getar (*rolled*), merupakan konsonan yang dibentuk dengan menghambat jalannya arus udara yang dihembuskan dari paru-paru secara berulang-ulang dan cepat dan terjadi banyak sentuhan (*tap*) yang terjadi antara ujung lidah dengan langit-langit atau gusi belakang. Contoh: *rolled* r (sangat jarang ditemukan).
7. Konsonan sentuhan kuat (*flapped*), merupakan konsonan dengan proses yang hampir sama dengan konsonan *rolled* tetapi hanya terjadi satu sentuhan (*tap*) antara ujung lidah dengan langit-langit atau gusi belakang. Contoh: *flapped* r (sangat jarang ditemukan).
8. Semi vokal (*semi-vowels*), bunyi yang secara praktis termasuk konsonan tetapi karena pada waktu diartikulasikan belum membentuk konsonan murni, maka bunyi-bunyi itu disebut semi-vokal. Contoh: w, y.

Berikut ini adalah tabel pembagian konsonan bahasa Inggris berdasarkan alat bicara yang menghasilkan dan cara artikulasinya.

Tabel 2.1 Pembagian Konsonan Bahasa Inggris

Klasifikasi Konsonan	Labial		Dental	Alveolar	Palato-Alveolar	Palatal	Velar	Glottal
	Bilabial	Labio-dental						
<i>Plosive</i>	b, p			t, d			k, g	
<i>Affricate</i>					c, j			
<i>Nasal</i>	w			n				
<i>Lateral</i>				l				
<i>Rolled</i>								
<i>Flapped</i>								
<i>Fricative</i>		f, v	Th	s, z, r	sh			h
<i>Semi-vowel</i>	m					y		

Sumber: [SYA-05]

2.4 Phonetic String Matching

Pembetulan bunyi (*Phonetic Correction*) adalah kesalahan pengejaan yang muncul karena kesalahan *user* dalam memasukkan *query* yang memiliki bunyi seperti istilah target [MAN-09]. Ada beberapa algoritma yang cukup sering digunakan untuk proses *phonetic string matching* di antaranya adalah *soundex*, *metaphone*, dan *caverphone* [SYA-05]. Selain itu, ada juga algoritma lain yaitu *phonix* yang terbagi lagi menjadi *phonix4*, *phonix8*, dan *phonixE* [PRI-97].

Secara umum, langkah yang ditempuh algoritma *phonetic string matching* dalam pencocokan kata adalah sebagai berikut [SYA-05]:

1. Menerima *input* berupa dua *string* yang akan dicocokkan.
2. Mengubah dua *string* masukan menjadi dua buah kode fonetis (*phonetic code*).
3. Membandingkan dua buah kode fonetis yang dihasilkan, jika kode fonetis sama maka dua *string* dianggap cocok (mirip cara pengucapan), jika kode fonetis berbeda maka dua *string* dianggap tidak cocok.

2.5 Kamus

Kamus adalah buku yang berisi daftar kosakata suatu bahasa yang disusun secara alfabetis dengan disertai penjelasan makna dan keterangan lain yang diperlukan serta dilengkapi dengan contoh pemakaian dalam kalimat. Kamus pada awalnya dicetak dan berbentuk buku yang seiring perkembangannya mulai digantikan dengan kamus elektronik [HAM-12].

Kamus dapat dibedakan menurut jenisnya masing-masing. Klasifikasi kamus menurut bahasa sasaran dapat dibedakan sebagai berikut [HAM-12]:

1. Kamus Ekabahasa : Kamus yang bahasa sumbernya sama dengan bahasa sasaran.
Contoh : Kamus Besar Bahasa Indonesia
2. Kamus Dwibahasa : Kamus yang bahasa sumbernya tidak sama dengan bahasa sasarannya.
Contoh : Kamus Inggris Indonesia
3. Kamus Aneka Bahasa : Kamus yang kata-kata bahasa sumber dijelaskan dengan padanannya dalam tiga bahasa atau lebih.

Klasifikasi kamus berdasarkan ukurannya dapat dibedakan seperti di bawah ini [HAM-12]:

1. Kamus Besar : Kamus yang memuat semua kosakata, gabungan kata, idiom, ungkapan, peribahasa, akronim, singkatan, dan semua bentuk gramatika bahasa tersebut.

2. Kamus Terbatas : Kamus dengan jumlah kata yang dimasukkan sebagai lema dibatasi, begitu pula dengan makna dan keterangan-keterangan lain.

Klasifikasi kamus berdasarkan isinya dapat dibedakan seperti di bawah ini [HAM-12]:

1. Kamus Umum : Kamus yang memuat seluruh kata secara umum. Kata-kata yang agak khas atau spesifik tidak dimuat dalam kamus tersebut.
2. Kamus Khusus : Kamus yang memuat disiplin ilmu pada bidang ilmu tertentu.

2.6 Bahasa Pemrograman Java

Java adalah bahasa pemrograman serbaguna. Java mendukung sumber daya internet yang saat ini populer yaitu *World Wide Web* atau yang sering disebut web saja. Java juga mendukung aplikasi klien/ server, baik dalam jaringan lokal (LAN) maupun jaringan berskala luas (WAN) [KAD-09].

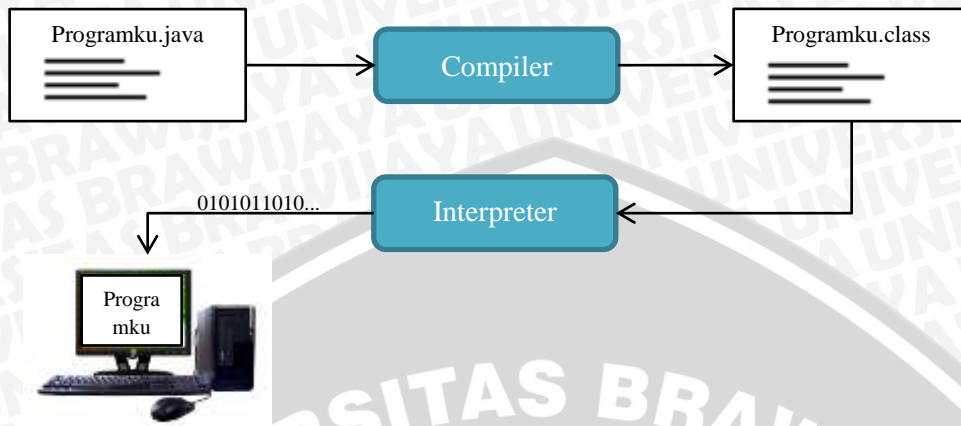
Java merupakan bahasa pemrograman yang saat ini sedang “naik daun” dan banyak digunakan oleh para *programmer* dan *software developer* untuk mengembangkan berbagai tipe aplikasi, mulai dari aplikasi *console*, aplikasi *desktop*, *applet* (aplikasi yang berjalan di lingkungan *web browser*), sampai ke aplikasi yang berskala *enterprise*.

Bahasa pemrograman Java dikategorikan menjadi 3 edisi yaitu sebagai berikut [RAH-09]:

- J2SE (*Java 2 Platform – Standard Edition*) yang digunakan untuk pembuatan aplikasi-aplikasi *desktop* dan *applet*.
- J2EE (*Java 2 Platform – Enterprise Edition*) yang digunakan untuk pembuatan aplikasi *multitier* berskala *enterprise*.
- J2ME (*Java 2 Platform – Micro Edition*) yang digunakan untuk pembuatan aplikasi yang dijalankan di lingkungan perangkat mikro seperti *handphone* dan *PDA*.

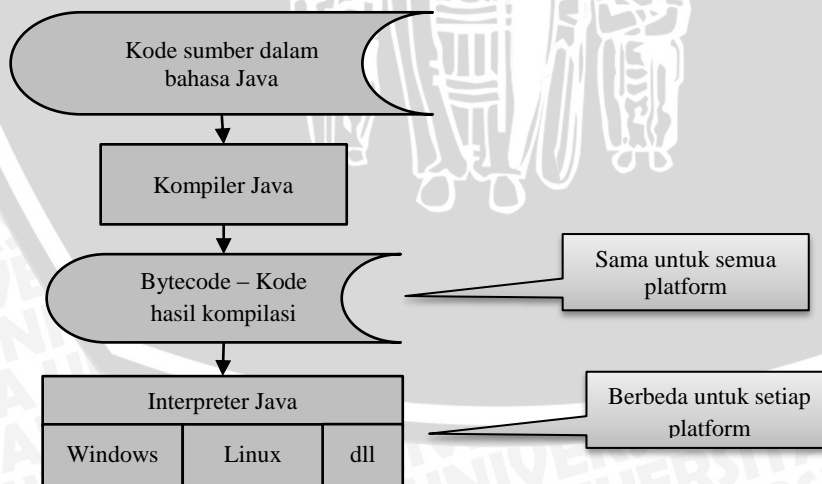
Bahasa pemrograman Java menerapkan dua proses utama yaitu berperan sebagai *compiler* dan *interpreter*. Kode program yang ditulis dengan bahasa Java

akan dikompilasi oleh *compiler* menjadi suatu kode objek yang disebut *Bytecode*. Selanjutnya, *Bytecode* akan dieksekusi baris demi baris oleh *interpreter*.



Gambar 2.1 Proses pembentukan dan eksekusi program di dalam Java
Sumber: [RAH-09]

Bytecode dapat dianggap sebagai sekumpulan perintah dalam bahasa mesin untuk sebuah JVM (*Java Virtual Machine*). Setiap *interpreter* Java, baik yang berupa *development tool* maupun sebuah *web browser*, merupakan implementasi dari JVM. Program yang dibuat dengan Java tidak mungkin dapat dijalankan di dalam komputer maupun alat lain yang tidak memiliki JVM. Dengan adanya konsep *Bytecode* ini, maka bahwa sekali kita menulis program Java dan melakukan kompilasi terhadapnya, maka *bytecode*-nya dapat dijalankan dalam *platform* manapun selama *platform* tersebut memiliki JVM [RAH-09].

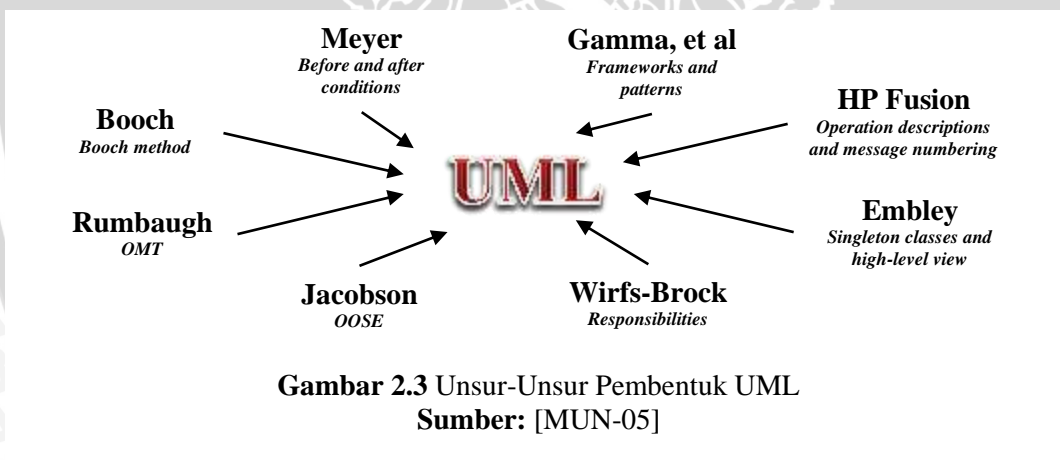


Gambar 2.2 Skema pengkompilasian hingga pengeksekusian kode Java
Sumber: [KAD-09]

2.7 Unified Modeling Language

UML (*Unified Modeling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain [MUN-05].

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Dengan UML, metode Booch, OMT, dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya [MUN-05].



Gambar 2.3 Unsur-Unsur Pembentuk UML
Sumber: [MUN-05]

2.7.1 Diagram pada UML

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Jenis diagram pada pemodelan UML antara lain sebagai berikut [ASH-12]:

1. Use Case Diagram

Use Case adalah abstraksi dari interaksi antara sistem dan aktor. *Use Case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem

dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

2. *Class Diagram*

Class adalah deskripsi kelompok objek –objek dengan properti, perilaku, dan relasi yang sama. Sehingga dengan adanya *Class* diagram dapat memberikan pandangan global atas sebuah sistem.

3. *Component Diagram*

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer dan tidak berada di benak analis. *Component* merupakan implementasi *software* dari sebuah atau lebih *class*.

4. *Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya.

5. *State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu objek dari suatu *class* dan keadaan yang menyebabkan *state* berubah. *State Class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

6. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Kegunaannya untuk menunjukkan rangkaian pesan dan interaksi antar objek.

7. *Collaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagram*. *Collaboration Diagram* menggambarkan objek dan hubungannya (mengacu ke konteks). Jika penekanan pada urutan digunakan *sequence diagram*, tapi jika penekanan pada konteks digunakan *collaboration diagram*.

8. Activity Diagram

Menggambarkan rangkaian aliran dari aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya.

2.8 Algoritma Levenshtein Distance

Kata kunci atau yang biasa disebut dengan *query* pada pencarian informasi digunakan sebagai kriteria pencarian yang tepat dan sesuai dengan kebutuhan. Ejaan kata kunci yang benar menjadi penting untuk meningkatkan hasil pencarian informasi. Kesalahan ejaan dapat disebabkan oleh beberapa hal di antaranya [SUT-09]:

1. Ketidaktahuan penulisan.

Kesalahan ini biasanya konsisten dan kemungkinan berhubungan dengan bunyi kata dan penulisan yang seharusnya.

2. Kesalahan dalam pengetikan.

Kesalahan ini biasanya lebih tidak konsisten tapi mungkin berhubungan erat dengan posisi tombol papan ketik dan pergerakan jari.

3. Kesalahan transmisi dan penyimpanan.

Berhubungan dengan pengkodean pada jalur mekanisme transmisi data.

Algoritma *Levenshtein* atau sering disebut dengan *Levenshtein distance* atau *edit distance* merupakan algoritma pencarian jumlah perbedaan *string* yang ditemukan oleh Vladimir Levenshtein, seorang ilmuwan Rusia, pada tahun 1965 [MAD-09]. *Levenshtein distance* antara dua *string* ditentukan berdasarkan jumlah minimum perubahan/ pengeditan yang diperlukan untuk melakukan transformasi dari satu bentuk *string* ke bentuk *string* yang lain. Operasi yang dilakukan dan diperbolehkan digunakan dalam menentukan *levenshtein distance* ini ada 3 macam operasi sebagai berikut [RIZ-10].

1. Insertion

Insertion adalah operasi melakukan penyisipan sebuah karakter ke dalam sebuah *string* tertentu.

2. Deletion

Deletion adalah operasi melakukan penghilangan atau penghapusan sebuah karakter tertentu dari sebuah *string*.

3. Substitution

Substitution adalah operasi menukarkan sebuah karakter pada *string* tertentu dengan karakter lain.

```
int LevenshteinDistance(char s[1..m], char t[1..n])
{
// d is a table with m+1 rows and n+1 columns
declare int d[0..m, 0..n]

for i from 0 to m
d[i, 0] := i // deletion
for j from 0 to n
d[0, j] := j // insertion
for j from 1 to n
{
for i from 1 to m
{
if s[i] = t[j] then
d[i, j] := d[i-1, j-1]
else
d[i, j] := minimum
(
d[i-1, j] + 1, // deletion
d[i, j-1] + 1, // insertion
d[i-1, j-1] + 1 //substitution
)
}
}
return d[m,n]
}
```

Gambar 2.4 Algoritma *Levenshtein Distance*

Sumber: [RIZ-10]

2.9 Algoritma *Caverphone 2.0*

Algoritma *Caverphone* untuk *phonetic string matching* dikembangkan pertama kali oleh David Hood dalam proyek Caversham, Universitas Otago, New Zealand pada tahun 2002. Pada saat itu, algoritma ini hanya dikhususkan untuk pencocokan string nama dalam bahasa inggris saja.

Pada tahun 2004, David Hood kembali mengembangkan algoritma ini menjadi algoritma *Caverphone 2.0*. Algoritma *Caverphone 2.0* tidak hanya

digunakan untuk proses pencocokan nama saja, tetapi juga pencocokan kata dalam bahasa inggris [SYA-05].

Aturan dari algoritma *Caverphone 2.0* untuk mengenali kemiripan bunyi pada kata bahasa inggris dapat dijelaskan sebagai berikut [HOO-04]:

1. Mulai dengan sebuah kata.
2. Ubah menjadi huruf kecil.
3. Menghilangkan semua yang bukan merupakan alfabet standar (a-z).
4. Menghilangkan huruf e pada karakter terakhir.
5. Aturan lain dapat dilihat pada tabel 2.2.

Tabel 2.2 Tahap Algoritma *Caverphone 2.0*

Awal	Akhir	Keterangan
cough	cou2f	
rough	rou2f	
tough	tou2f	
enough	enou2f	
trough	trou2f	
gn	2n	
mb	m2	
cq	2q	
ci	si	
ce	se	
cy	sy	
tch	2ch	
c	k	
q	k	
x	k	
v	f	
dg	2g	
tio	sio	
tia	sia	
d	t	
ph	fh	
b	p	
sh	s2	
z	s	
huruf vokal awal	A	
huruf vokal lain	3	
j	y	
y3 awal kata	Y3	
y awal kata	A	
y	3	
3gh3	3kh3	

Awal	Akhir	Keterangan
gh	22	
g	k	
kumpulan huruf s	S	
kumpulan huruf t	T	
kumpulan huruf p	P	
kumpulan huruf k	K	
kumpulan huruf f	F	
kumpulan huruf m	M	
kumpulan huruf n	N	
w3	W3	
wh3	Wh3	
w pada karakter paling akhir	3	Jika kata berakhiran huruf w
w	2	
h karakter awal	A	
h selain karakter awal	2	
r3	R3	
r pada karakter paling akhir	3	Jika kata berakhiran huruf r
r	2	
l3	L3	
l pada karakter paling akhir	3	Jika kata berakhiran huruf l
l	2	
menghilangkan semua karakter 2		
3 pada karakter terakhir	A	Jika kata berakhiran dengan 3
menghilangkan semua karakter 3		
menambahkan 10 karakter 1 di akhir kata		
mengambil 10 karakter pertama sebagai kode fonetis		

Sumber: [HOO-04]

Tabel 2.2 menunjukkan bagaimana perubahan setiap huruf sebelum dan sesudah diproses oleh algoritma *caverphone 2.0*. Setiap kata bahasa Inggris dipisahkan menjadi setiap karakter yang nantinya akan diubah satu per satu sehingga menghasilkan kode fonetis.

Contoh:

- Jika ada kata *cough*, maka diubah menjadi *cou2f*. Selanjutnya, *c* diubah menjadi *k* dan *ou* diubah menjadi *33*. Sehingga setelah diproses menjadi *k332f*. Maka, kode fonetis akan menjadi **KF11111111**.
- Jika ada kata *bad*, maka huruf *b* diubah menjadi *p*, huruf *a* diubah menjadi *3*, dan huruf *d* diubah menjadi *t*. Sehingga setelah diproses menjadi *p3t*. Maka, kode fonetis akan menjadi **PT11111111**.

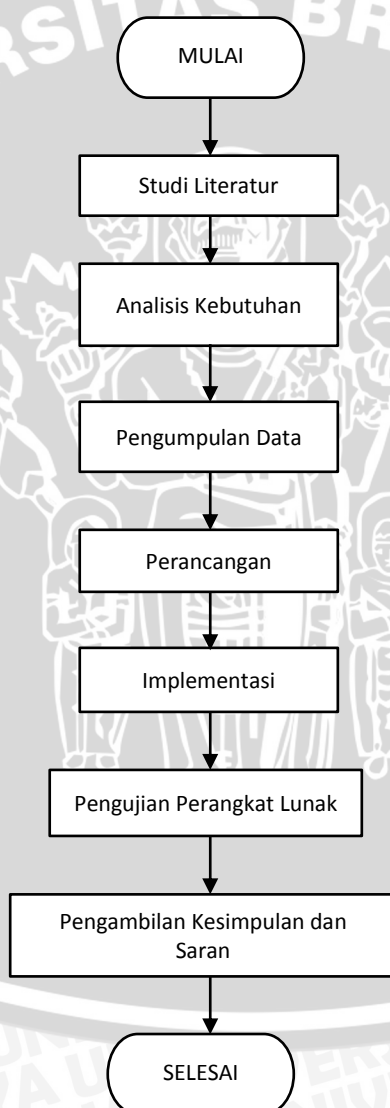
- Jika ada kata pigeon, maka huruf p tetap, huruf i,e,o diubah menjadi 3, huruf g diubah menjadi k, huruf n tetap. Sehingga setelah diproses menjadi p3k33n. Maka, kode fonetis akan menjadi **PKN1111111**.



BAB III

METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam pengerjaan tugas akhir, yaitu studi literatur, analisis kebutuhan, pengumpulan data, perancangan, implementasi, pengujian dari aplikasi perangkat lunak yang akan dibuat, pengambilan kesimpulan dan saran sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya. Berikut ini merupakan diagram alir runtutan pengerjaan penelitian ini:



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Studi literatur mempelajari mengenai penjelasan dasar teori yang digunakan untuk menunjang penulisan tugas akhir. Studi literatur dalam penulisan ini disusun sebagai berikut:

1. Dasar teori Sistem Temu Kembali Informasi (*Information Retrieval*).
2. Dasar Teori Fonetik bahasa Inggris.
3. Dasar Teori Metode *Phonetic String Matching*.
4. Dasar Teori Kamus.
5. Dasar Teori Algoritma *Caverphone 2.0*.
6. Dasar Teori Algoritma *Levenshtein Distance*.
7. Bahasa pemrograman Java.
8. *Unified Modeling Language*.

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Metode analisis yang digunakan adalah menggunakan bahasa pemodelan UML (*Unified Modeling Language*). Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem. Diagram *use case* digunakan untuk mengetahui aktor, bagaimana skenario aktor, bagaimana penggunaan aplikasi dan mengidentifikasi semua kebutuhan (*requirements*) fungsionalitas aplikasi.

3.2.1 Analisis Masukan Sistem

Masukan yang dibutuhkan aplikasi kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0* ini terdiri dari kata bahasa Inggris yang diinputkan *user* sebagai kata kunci *query* untuk proses *phonetic string matching* menggunakan algoritma *caverphone 2.0*.

3.2.2 Analisis Keluaran Sistem

Keluaran aplikasi berupa arti kata bahasa Inggris dalam bahasa Indonesia dan cara pengucapannya. Selain itu, juga akan muncul pula daftar kata bahasa

Inggris lain yang memiliki kemiripan pengucapan/ bunyi dengan kata bahasa Inggris hasil masukan dari *user*.

3.3 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data untuk daftar kata yang digunakan sebagai data kamus. Daftar kata yang digunakan diperoleh dari <http://jatiblack.com/membuat-kamus-inggris-indonesia-dengan-php> yang terdiri dari 23.623 kata bahasa Inggris. Data yang diperoleh ini terdiri dari daftar kata bahasa Inggris beserta arti kata dalam bahasa Indonesia.

3.4 Perancangan

Perancangan perangkat lunak digunakan untuk memenuhi kebutuhan fungsional aplikasi. Tahap Perancangan aplikasi meliputi desain perancangan aplikasi secara umum, perancangan algoritma *Caverphone 2.0*, perancangan algoritma *levenshtein distance*, dan perancangan antarmuka aplikasi.

Desain perancangan aplikasi secara umum akan menggambarkan bagaimana aplikasi nantinya akan dibuat beserta proses yang terjadi di dalamnya. Perancangan algoritma *Caverphone 2.0* akan menggambarkan bagaimana *pseudocode* dan tahapan algoritma *Caverphone 2.0* untuk mendapatkan kode fonetis dari setiap kata bahasa Inggris. Perancangan algoritma *Levenshtein Distance* akan menggambarkan bagaimana *pseudocode* dan tahapan algoritma *Levenshtein Distance* dalam mendapatkan jarak terpendek antar kata sehingga dapat melakukan pembetulan terhadap kata yang salah eja. Perancangan antarmuka aplikasi akan menggambarkan desain tampilan aplikasi yang akan dibuat.

3.4.1 Perancangan Aplikasi Kamus Inggris-Indonesia menggunakan Algoritma *Caverphone 2.0*

Desain perancangan aplikasi kamus Inggris-Indonesia secara umum akan menggambarkan jalannya proses yang terjadi pada aplikasi. Desain perancangan aplikasi secara umum akan digambarkan dalam bentuk diagram blok dan diagram

alir. Diagram blok membantu menggambarkan jalannya aplikasi secara umum, sedangkan diagram alir akan menjelaskan rancangan aplikasi secara lebih detail.

3.4.2 Perancangan Algoritma *Levenshtein Distance*

Proses algoritma *Levenshtein Distance* adalah dengan mencari jarak terpendek dari dua buah kata yang dibandingkan. Semakin pendek jarak antar dua kata, maka akan semakin besar peluang kemiripan kata tersebut. Kata yang memiliki jarak paling pendek akan menjadi kandidat kuat untuk menggantikan kata yang mengalami kesalahan ejaan. Perancangan algoritma *Levenshtein Distance* akan digambarkan dalam bentuk *pseudocode* dan diagram alir untuk memperjelas setiap tahap.

3.4.3 Algoritma *Caverphone 2.0*

Proses algoritma *Caverphone 2.0* adalah dengan mencari kode fonetis dari setiap kata bahasa Inggris. Jika dua buah kata memiliki kode fonetis yang sama, maka dua kata tersebut akan dikatakan memiliki kemiripan pengucapan. Sebaliknya, jika dua buah kata memiliki kode fonetis yang berbeda, maka dua kata tersebut akan dikatakan memiliki perbedaan pengucapan. Perancangan algoritma *Caverphone 2.0* akan digambarkan dalam bentuk *pseudocode* dan diagram alir.

3.4.4 Perancangan Antarmuka

Desain perancangan antarmuka aplikasi ditujukan untuk menggambarkan bagaimana tampilan aplikasi yang akan berinteraksi dengan pengguna/ *user*. Perancangan antarmuka akan menjelaskan bagian-bagian tampilan aplikasi beserta fungsi-fungsi menu dan kegunaan tombol. Dalam perancangan antarmuka aplikasi digunakan gambar secara manual untuk menunjukkan setiap bagian aplikasi.

3.5 Implementasi

Implementasi aplikasi dilakukan dengan mengacu kepada perancangan aplikasi. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa

pemrograman JAVA dan DBMS MySQL. Implementasi algoritma *Caverphone 2.0* pada aplikasi kamus Inggris-Indonesia meliputi:

- Implementasi algoritma *caverphone 2.0* untuk mendapatkan kata yang memiliki kemiripan pengucapan.
- Implementasi algoritma *levenshtein distance* untuk mendapatkan kata yang dapat menggantikan kata yang salah eja.
- Implementasi antarmuka aplikasi.

3.6 Pengujian Perangkat Lunak

Pengujian perangkat lunak pada penelitian ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya. Pengujian perangkat lunak meliputi:

1. Pengujian validasi

Pengujian validasi dilakukan dengan menguji fungsionalitas aplikasi. Pengujian dilakukan untuk mengetahui apakah sistem yang dibangun telah sesuai dengan daftar kebutuhan. Pengujian validasi dilakukan dengan cara menjalankan fungsi aplikasi sesuai daftar kebutuhan. Kemudian, dapat ditentukan apakah semua kebutuhan sudah dipenuhi dan semua fungsi dapat berjalan dengan baik.

2. Pengujian akurasi

Pengujian akurasi dilakukan dengan menguji kesesuaian hasil yang dikeluarkan oleh algoritma *caverphone 2.0* dengan klasifikasi konsonan bahasa Inggris. Pengujian dilakukan untuk mengetahui tingkat kecocokan yang dihasilkan oleh algoritma *caverphone 2.0*. Proses pengujian akurasi dapat dijelaskan sebagai berikut:

- Pengujian akurasi dilakukan untuk awal, tengah, dan akhir kata. Masing-masing bagian dilakukan pengujian untuk semua jenis klasifikasi konsonan bahasa Inggris yang berjumlah 14 jenis.
- Pengujian akurasi dilakukan sebanyak 10 kali untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris.

- Hasil pengujian akurasi masing-masing jenis klasifikasi konsonan adalah presentase jumlah keluaran kata aplikasi dibandingkan dengan jumlah kata yang memiliki kemiripan pengucapan.

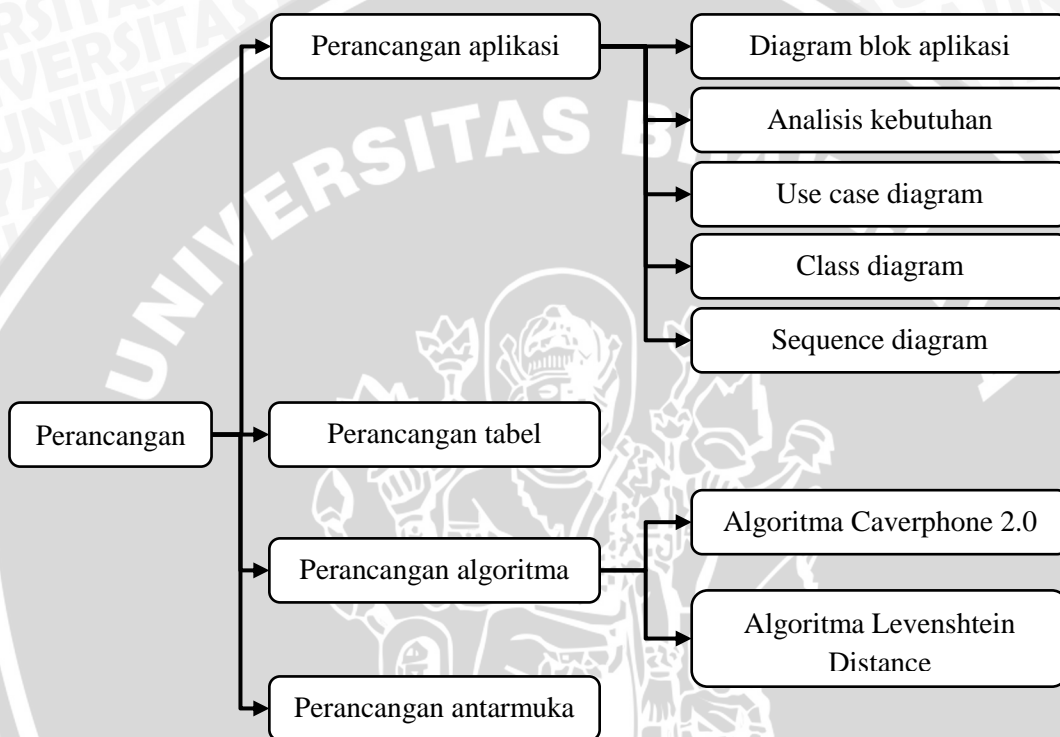
3.7 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.



BAB IV PERANCANGAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam tahap perancangan. Perancangan yang dilakukan meliputi perancangan aplikasi, perancangan basis data, perancangan algoritma, dan perancangan antarmuka atau tampilan aplikasi. Pohon perancangan aplikasi dapat dilihat pada gambar 4.1.



Gambar 4.1 Pohon Perancangan
Sumber: [Perancangan]

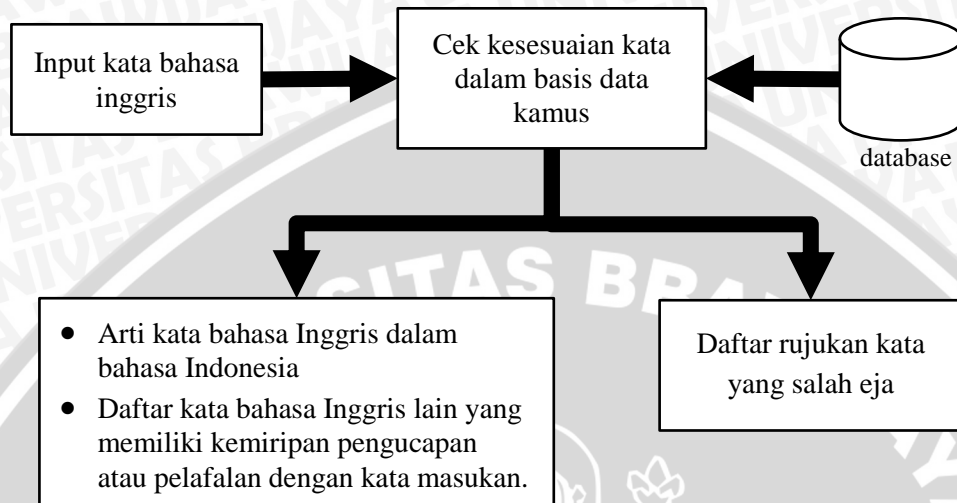
4.1 Perancangan Aplikasi

Perancangan aplikasi merupakan tahap awal dari perancangan perangkat lunak. Perancangan aplikasi dilakukan untuk mengetahui aplikasi yang akan dibuat secara umum. Perancangan aplikasi meliputi diagram blok aplikasi, analisis kebutuhan, diagram *use case*, diagram *class*, dan diagram *sequence*.

4.1.1 Diagram Blok Aplikasi

Diagram blok aplikasi menggambarkan setiap blok atau bagian dari aplikasi kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0*.

Program dirancang agar dapat digunakan untuk membantu dalam proses pembelajaran pengucapan atau pelafalan kata bahasa Inggris dan mengetahui arti dari setiap kata bahasa Inggris dalam bahasa Indonesia. Gambaran umum konsep aplikasi yang akan dibangun ditunjukkan pada Gambar 4.2.



Gambar 4.2 Diagram Blok Perancangan Aplikasi

Sumber: [Perancangan]

Penjelasan diagram blok perancangan aplikasi pada gambar 4.2 dapat dijelaskan sebagai berikut:

1. Pengguna/ *user* menginputkan kata bahasa Inggris yang ingin diketahui arti kata dalam bahasa Indonesia beserta cara pengucapannya.
2. Aplikasi akan mengecek ketersediaan kata bahasa Inggris hasil inputan *user* dengan daftar kata dalam basis data kamus.
3. Jika kata bahasa Inggris hasil inputan *user* tersedia dalam basis data, maka aplikasi akan menampilkan arti kata bahasa Inggris dalam bahasa Indonesia beserta cara pengucapannya.
4. Selain itu, aplikasi juga akan menampilkan daftar kata bahasa Inggris lain yang memiliki kemiripan pengucapan dengan kata bahasa Inggris yang diinputkan *user*.
5. Jika kata bahasa Inggris hasil inputan *user* tidak tersedia dalam basis data, maka aplikasi akan menampilkan daftar rujukan kata/ *suggested word* yang tersedia dalam basis data.

4.1.2 Analisis Kebutuhan

Pengembangan sebuah perangkat lunak bertujuan untuk menghasilkan perangkat lunak yang dapat memenuhi kebutuhan *user*. Setiap pengembangan sebuah sistem perangkat lunak memerlukan adanya dokumentasi terhadap kebutuhan-kebutuhan *user* agar tujuan tersebut tercapai.

Proses analisis kebutuhan dilakukan dengan acuan pengumpulan dan penetapan kebutuhan-kebutuhan dari aplikasi yang akan dibangun. Analisis kebutuhan dibagi menjadi dua tahap yaitu tahap identifikasi aktor dan tahap penjabaran daftar kebutuhan.

4.1.2.1 Identifikasi Aktor

Tahap identifikasi aktor dilakukan untuk mengidentifikasi aktor yang berinteraksi dengan aplikasi. Identifikasi aktor akan dijelaskan pada tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi Aktor
Pengguna/ <i>User</i>	Pengguna atau <i>User</i> merupakan aktor yang menggunakan fasilitas aplikasi kamus Inggris-Indonesia untuk mencari arti kata bahasa Inggris dalam bahasa Indonesia. Selain itu pengguna atau <i>User</i> juga dapat mencari padanan kata bahasa Inggris yang memiliki kemiripan pengucapan.

Sumber: [Perancangan]

4.1.2.2 Daftar Kebutuhan

Pada tahap daftar kebutuhan akan dijelaskan kebutuhan fungsional yang diperlukan oleh pengguna atau *user* dalam menggunakan aplikasi kamus Inggris-Indonesia. Spesifikasi kebutuhan fungsional pengguna atau *user* ditunjukkan pada tabel 4.2.

Tabel 4.2 Daftar Kebutuhan Fungsional *User*

No	Kebutuhan
01	Aplikasi kamus Inggris-Indonesia harus dapat memberikan fasilitas untuk memasukkan kata bahasa Inggris yang ingin dicari arti kata dalam bahasa Indonesia dan padanan kata

No	Kebutuhan
	bahasa Inggris yang memiliki kemiripan pengucapan.
02	Aplikasi kamus Inggris-Indonesia harus dapat digunakan untuk menampilkan arti kata bahasa Inggris yang dimasukkan pengguna atau <i>user</i> dalam bahasa Indonesia.
03	Aplikasi kamus Inggris-Indonesia harus dapat digunakan untuk menampilkan cara pengucapan bahasa Inggris dalam bahasa Indonesia
04	Aplikasi kamus Inggris-Indonesia harus dapat digunakan untuk menampilkan kata bahasa Inggris yang memiliki kemiripan pengucapan dengan kata bahasa Inggris hasil masukan pengguna atau <i>user</i> .
05	Aplikasi kamus Inggris-Indonesia harus dapat digunakan untuk mengoreksi kesalahan penulisan kata bahasa Inggris yang salah eja yang dimasukkan oleh pengguna atau <i>user</i> .
06	Aplikasi kamus Inggris-Indonesia harus menyediakan fasilitas bantuan atau <i>help</i> untuk menampilkan cara pembacaan setiap pengucapan kata bahasa Inggris dalam bahasa Indonesia.

Sumber: [Perancangan]

4.1.3 Use Case Diagram

Diagram *use case* adalah salah satu diagram untuk memodelkan aspek perilaku aplikasi. Diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas aplikasi yang diinisialisasi oleh aktor. Pemodelan diagram *use case* untuk aplikasi kamus Inggris-Indonesia dapat dilihat pada gambar 4.3.



Gambar 4.3 Diagram Use Case Aplikasi

Sumber: [Perancangan]

Diagram use case pada gambar 4.3 di atas terdiri dari 6 use case yaitu sebagai berikut:

1. *Use case* memasukkan kata bahasa Inggris.
 - Proses yang terjadi dalam *use case* ini adalah *user* memberikan masukan kata bahasa Inggris yang ingin diketahui arti, cara pengucapan, dan kata bahasa Inggris lain yang mirip pengucapan.
 - *User* memberikan masukan kata bahasa Inggris dengan cara mengetik pada textfield.
2. *Use case* mendapatkan arti kata bahasa Inggris.
 - Proses yang terjadi dalam *use case* ini adalah *user* memberikan masukan kata bahasa Inggris.
 - *User* menekan tombol proses.
 - *User* mendapatkan arti kata bahasa Inggris dalam bahasa Indonesia.
3. *Use case* mendapatkan cara pengucapan kata bahasa Inggris.
 - Proses yang terjadi dalam *use case* ini adalah *user* memberikan masukan kata bahasa Inggris.
 - *User* menekan tombol proses.
 - *User* mendapatkan cara pengucapan kata bahasa Inggris berupa alfabet fonetik internasional.
4. *Use case* mendapatkan kata bahasa Inggris yang memiliki kemiripan pengucapan.
 - Proses yang terjadi dalam *use case* ini adalah *user* memberikan masukan kata bahasa Inggris.
 - *User* menekan tombol proses.
 - *User* mendapatkan kata bahasa Inggris lain yang memiliki kemiripan pengucapan dengan kata bahasa Inggris hasil masukan *user*.
5. *Use case* pengkoreksian kata yang salah eja.
 - Proses yang terjadi dalam *use case* ini adalah *user* memberikan masukan kata bahasa Inggris.
 - *User* menekan tombol proses.

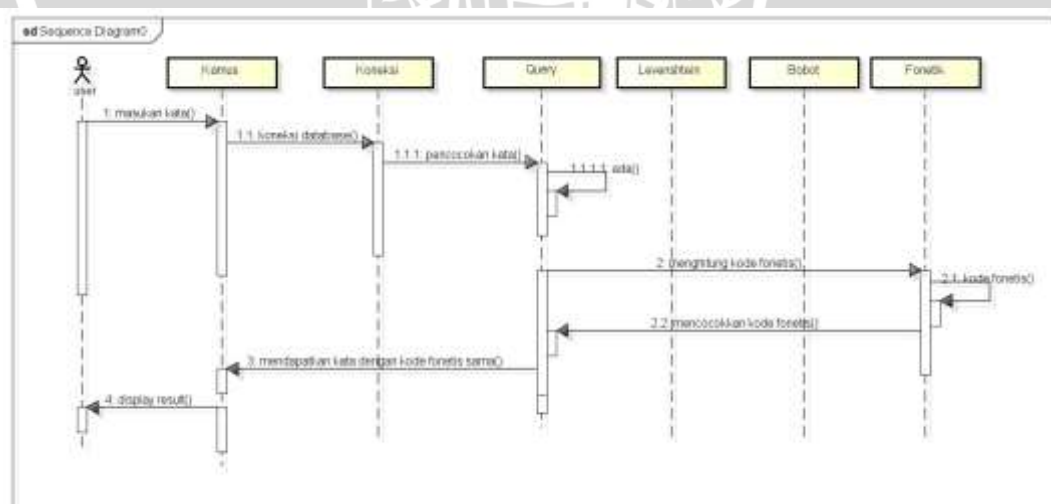
- Jika aplikasi tidak menemukan kecocokan antara kata hasil masukan *user* dengan daftar kata bahasa Inggris dalam *database*, maka *user* mendapatkan koreksi kata yang salah eja berupa *suggested word*.
6. *Use case* melihat bantuan cara pembacaan alfabet fonetik.
- Proses yang terjadi dalam *use case* ini adalah *user* menekan menu bantuan.
 - Aplikasi menampilkan halaman bantuan pembacaan alfabet fonetik pada *user*.

4.1.4 Sequence Diagram

Sequence diagram adalah diagram yang menunjukkan alur atau urutan jalannya aplikasi. Terdapat 3 *sequence diagram* pada aplikasi kamus Inggris-Indonesia yaitu *sequence diagram kata mirip pengucapan*, *sequence diagram koreksi ejaan salah*, dan *sequence diagram arti kata dan pengucapan*. Pemodelan masing-masing *sequence diagram* dapat dijelaskan sebagai berikut.

4.1.4.1 Sequence Diagram Kata Mirip Pengucapan

Sequence diagram kata mirip pengucapan menampilkan urutan jalannya aplikasi kamus Inggris-Indonesia ketika berhubungan dengan algoritma *caverphone 2.0*. Pada diagram ini dijelaskan proses yang terjadi selama pengubahan kata menjadi kode fonetis hingga dapat menampilkan kata yang memiliki kemiripan pengucapan. Pemodelan *sequence diagram kata mirip pengucapan* dapat dilihat pada gambar 4.4.



Gambar 4.4 *Sequence Diagram Kata Mirip Pengucapan*

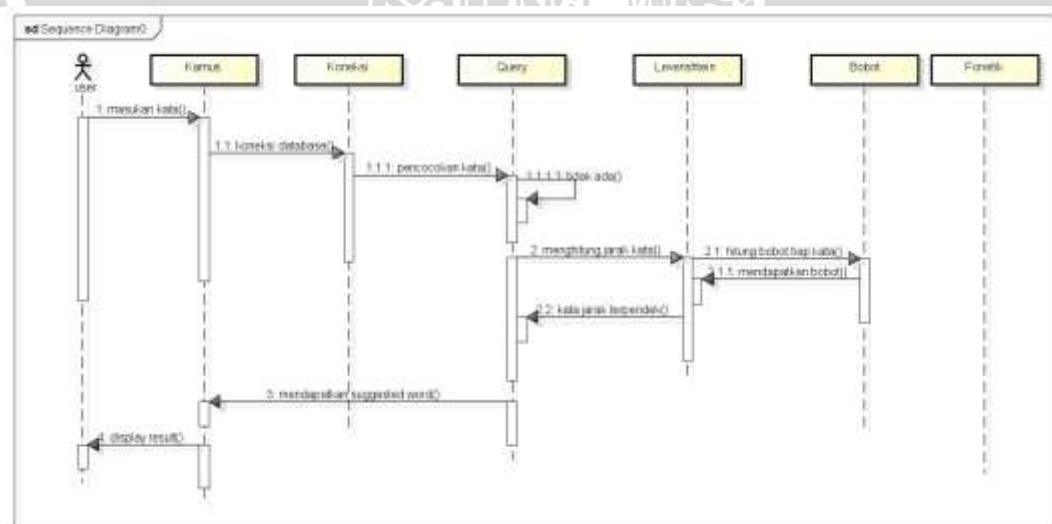
Sumber: [Perancangan]

Diagram *sequence* pada gambar 4.4 di atas dapat dijelaskan seperti di bawah ini:

1. User memasukkan kata bahasa Inggris yang ingin diketahui artinya dalam bahasa Indonesia pada halaman utama.
2. Aplikasi akan membuka koneksi dengan database.
3. Aplikasi akan mencocokkan kata yang dimasukkan oleh *user* apakah tersedia dalam *database* atau tidak.
4. Kata hasil masukan *user* yang tersedia dalam *database* akan dihitung kode fonetisnya dengan menggunakan algoritma *caverphone 2.0*.
5. Aplikasi akan mencocokkan kode fonetis kata hasil masukan *user* dengan kode fonetis kata dalam *database*.
6. Aplikasi akan menampilkan kata bahasa Inggris lain yang memiliki kemiripan pengucapan. Kata yang memiliki kemiripan pengucapan adalah kata bahasa Inggris yang memiliki kesamaan kode fonetis.

4.1.4.2 Sequence Diagram Koreksi Ejaan Salah

Sequence diagram koreksi ejaan salah menampilkan urutan jalannya aplikasi kamus Inggris-Indonesia ketika berhubungan dengan algoritma *levenshtein distance*. Pemodelan *sequence diagram koreksi ejaan salah* dapat dilihat pada gambar 4.5.



Gambar 4.5 *Sequence Diagram Koreksi Ejaan Salah*

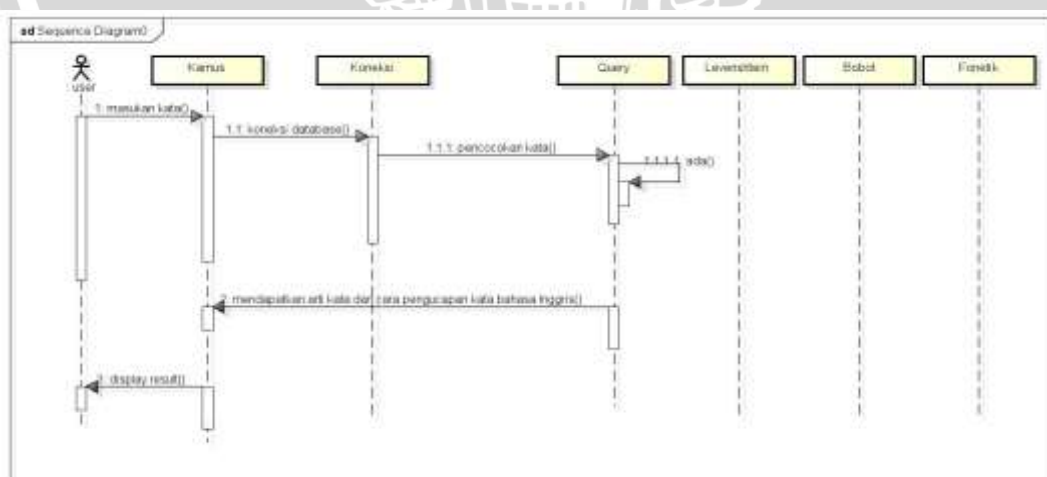
Sumber: [Perancangan]

Diagram *sequence* pada gambar 4.5 di atas dapat dijelaskan seperti di bawah ini:

1. User memasukkan kata bahasa Inggris yang ingin diketahui artinya dalam bahasa Indonesia pada halaman utama.
2. Aplikasi akan membuka koneksi dengan *database*.
3. Aplikasi akan mencocokkan kata yang dimasukkan oleh *user* apakah tersedia dalam *database* atau tidak.
4. Kata hasil masukan *user* yang tidak tersedia dalam *database* akan dihitung jarak antar kata dengan kata dalam *database* menggunakan algoritma *levenshtein distance*.
5. Aplikasi akan mendapatkan bobot jarak setiap kata.
6. Aplikasi akan mendapatkan kata dengan jarak terpendek dengan kata dalam *database*.
7. Aplikasi akan menampilkan *suggested word* sesuai kata yang terdapat dalam *database* dan memiliki jarak yang paling pendek.

4.1.4.3 Sequence Diagram Arti Kata dan Pengucapan

Sequence diagram arti kata dan pengucapan menampilkan urutan jalannya aplikasi kamus Inggris-Indonesia ketika menampilkan arti kata bahasa Inggris dalam bahasa Indonesia beserta cara pengucapannya. Pemodelan *sequence diagram arti kata dan pengucapan* dapat dilihat pada gambar 4.6.



Gambar 4.6 Sequence Diagram Arti Kata dan Pengucapan

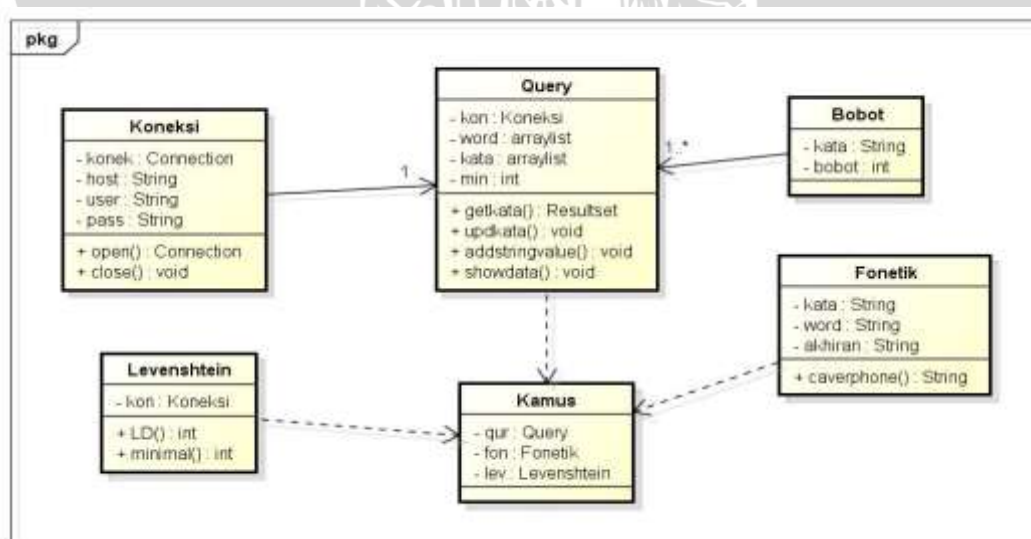
Sumber: [Perancangan]

Diagram *sequence* pada gambar 4.5 di atas dapat dijelaskan seperti di bawah ini:

1. User memasukkan kata bahasa Inggris yang ingin diketahui artinya dalam bahasa Indonesia pada halaman utama.
2. Aplikasi akan membuka koneksi dengan *database*.
3. Aplikasi akan mencocokkan kata yang dimasukkan oleh *user* apakah tersedia dalam *database* atau tidak.
4. Ketika ditemukan dalam *database*, maka kemudian aplikasi akan mengembalikan nilai berupa arti kata dan cara pengucapan kata bahasa Inggris hasil masukan pengguna.
5. Aplikasi akan menampilkan arti kata dan cara pengucapan kata bahasa Inggris hasil masukan pengguna.

4.1.5 Class Diagram

Class diagram adalah diagram kelas yang akan diimplementasikan pada aplikasi kamus Inggris-Indonesia. Berdasarkan objek-objek yang terdapat pada *sequence diagram*, maka pada kelas diagram ini terdapat 6 buah kelas yaitu *main class* yang diberi nama Kamus, kelas Koneksi, kelas Query, kelas Fonetik, kelas Levenshtein dan kelas Bobot. Pemodelan *class diagram* untuk aplikasi kamus Inggris-Indonesia dapat dilihat pada gambar 4.4.



Gambar 4.7 Diagram Class Aplikasi

Sumber: [Perancangan]

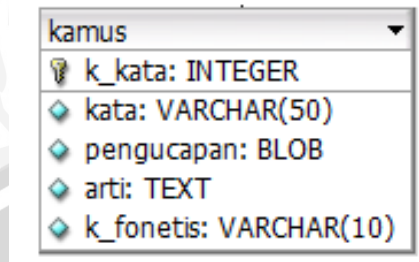
Diagram *class* pada gambar 4.4 di atas dapat dijelaskan seperti di bawah ini:

1. Kelas *Koneksi* merupakan kelas yang digunakan untuk menghubungkan koneksi dengan *database* MySQL. Kelas ini terdiri dari 2 *method* yaitu *open()* dan *close()*. *Method open()* digunakan untuk membuka koneksi dengan *database* MySQL, sedangkan *method close()* digunakan untuk memutuskan koneksi dengan *database* MySQL.
2. Kelas *Query* merupakan kelas yang digunakan untuk menampung semua operasi *query* yang berhubungan dengan *database* MySQL. Kelas ini terdiri dari 4 *method* yaitu *getkata()*, *updkata()*, *addstringvalue()*, dan *showdata()*. *Method getkata()* digunakan untuk mendapatkan *resultset* dari *database* kamus, *method updkata()* digunakan untuk melakukan proses *update* kode fonetik dalam *database*, *method addstringvalue* digunakan untuk memasukkan kata ke dalam *arraylist*, *method showdata* digunakan untuk menampilkan data kata.
3. Kelas *Fonetik* merupakan kelas yang digunakan untuk mengerjakan proses dari algoritma *caverphone 2.0*. Kelas ini memiliki *method* yang bernama *caverphone()* yang akan mengembalikan nilai kembalian berupa kode fonetik dari setiap kata.
4. Kelas *Levenshtein* merupakan kelas yang digunakan untuk mengerjakan proses dari algoritma *levenshtein distance*. Kelas ini memiliki *method* yang bernama *LD()* dan *minimal()*. *Method LD()* akan mengembalikan nilai berupa jarak. *Method minimal()* akan mengembalikan nilai berupa jarak minimal.
5. Kelas *Bobot* merupakan kelas yang digunakan untuk menyimpan variabel bertipe data *String* dan *Integer* yang akan digunakan untuk membangun *arraylist*.
6. Kelas *kamus* merupakan kelas utama yang akan melakukan pemanggilan ke kelas-kelas yang lainnya.

4.2 Perancangan Tabel

Database atau basis data berfungsi sebagai tempat menyimpan data. Tahap perancangan basis data digunakan untuk merancang basis data yang akan dibuat

agar masukan dan keluaran aplikasi sesuai dengan apa yang diharapkan. Basis data berisi kumpulan tabel beserta relasi antar tabel. Dalam pembuatan aplikasi ini, digunakan sebuah tabel yang dapat dilihat pada gambar 4.7.



kamus
k_kata: INTEGER
kata: VARCHAR(50)
pengucapan: BLOB
arti: TEXT
k_fonetis: VARCHAR(10)

Gambar 4.8 Struktur Tabel pada Aplikasi
Sumber: [Perancangan]

Penjelasan dari tabel untuk pembuatan aplikasi pada gambar 4.7 adalah sebagai berikut:

1. Perancangan basis data kamus Inggris-Indonesia terdiri dari tabel yang bernama kamus.
2. Tabel kamus terdiri dari 5 kolom yaitu k_kata, kata, pengucapan, arti, dan k_fonetis.
3. Kolom k_kata bertipe *integer* untuk menyimpan no_urut setiap kata bahasa Inggris.
4. Kolom kata bertipe *varchar(50)* untuk menyimpan setiap kata bahasa Inggris dan maksimal panjang kata yang dapat dimasukkan adalah 50 karakter.
5. Kolom pengucapan bertipe *blob* untuk menyimpan cara pengucapan atau pelafalan setiap kata bahas Inggris.
6. Kolom arti bertipe *text* untuk menyimpan arti kata setiap kata bahasa Inggris. Tipe *text* dipilih dikarenakan arti kata biasanya memiliki jumlah karakter yang sangat panjang.
7. Kolom k_fonetis bertipe *varchar(10)* untuk menyimpan kode fonetis dari setiap kata bahasa Inggris. Panjang kode fonetis maksimal yang dapat dimasukkan adalah 10 karakter. Hal ini sudah sesuai dengan hasil keluaran algoritma *caverphone 2.0* yang berupa 10 karakter kode fonetis.

4.3 Perancangan Algoritma

Perancangan algoritma dalam pembuatan aplikasi kamus Inggris-Indonesia ini akan dibahas seputar perancangan algoritma *caverphone 2.0* dan algoritma *levenshtein distance*.

4.3.1 Perancangan Algoritma *Caverphone 2.0*

Proses yang terjadi dalam algoritma *Caverphone 2.0* adalah dengan mencari kode fonetis dari dua *String* yang dibandingkan. Kemudian ditentukan apakah kedua *String* memiliki kode fonetis yang sama atau tidak. Jika kedua *String* memiliki kode fonetis yang sama, maka kedua *String* dianggap memiliki kemiripan bunyi. Sebaliknya, jika kedua *String* memiliki kode fonetis yang berbeda, maka kedua *String* dianggap tidak memiliki kemiripan bunyi.

Pseudocode yang menampilkan proses algoritma *Caverphone 2.0* dapat dilihat pada gambar 4.8.

<p><u>Nama Algoritma</u> : <i>Caverphone 2.0</i></p> <p><u>Deklarasi</u> String data (kata kunci)</p> <p><u>Deskripsi</u> Input : data Proses : 1. Menerima masukan kata bahasa Inggris. 2. <i>Mereplace</i> setiap karakter pada String data sesuai rule algoritma <i>Caverphone 2.0</i>.</p> <p>Output : Kode fonetis setiap kata</p>
--

Gambar 4.9 *Pseudocode* Algoritma *Caverphone 2.0*

Sumber: [Perancangan]

Pada proses ketiga pada gambar 4.8 dapat dilihat bahwa setiap karakter pada *String* data dilakukan proses *replace* sesuai *rule* atau aturan yang terdapat dalam algoritma *Caverphone 2.0*. Rule algoritma *Caverphone 2.0* dapat dilihat pada tabel 4.3.

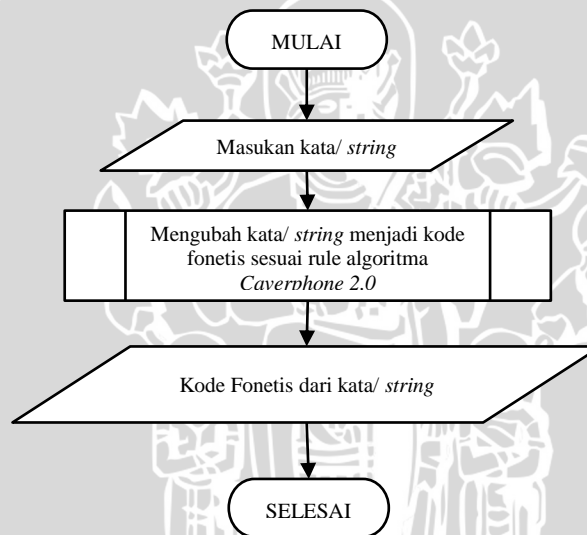
Tabel 4.3 Rule Algoritma Caverphone 2.0

Awal	Akhir	Keterangan
cough	cou2f	
rough	rou2f	
tough	tou2f	
enough	enou2f	
trough	trou2f	
gn	2n	
mb	m2	
cq	2q	
ci	si	
ce	se	
cy	sy	
tch	2ch	
c	k	
q	k	
x	k	
v	f	
dg	2g	
tio	sio	
tia	sia	
d	t	
ph	fh	
b	p	
sh	s2	
z	s	
huruf vokal awal	A	
huruf vokal lain	3	
j	y	
y3 awal kata	Y3	
y awal kata	A	
y	3	
3gh3	3kh3	
gh	22	
g	k	
kumpulan huruf s	S	
kumpulan huruf t	T	
kumpulan huruf p	P	
kumpulan huruf k	K	
kumpulan huruf f	F	
kumpulan huruf m	M	
kumpulan huruf n	N	
w3	W3	
wh3	Wh3	
w pada karakter paling akhir	3	Jika kata berakhiran huruf w
w	2	
h karakter awal	A	
h selain karakter awal	2	

r3	R3	
r pada karakter paling akhir	3	Jika kata berakhiran huruf r
r	2	
l3	L3	
l pada karakter paling akhir	3	Jika kata berakhiran huruf l
l	2	
menghilangkan semua karakter 2		
3 pada karakter terakhir	A	Jika kata berakhiran dengan 3
menghilangkan semua karakter 3		
menambahkan 10 karakter 1 di akhir kata		
mengambil 10 karakter pertama sebagai kode fonetis		

Sumber: [HOO-04]

Sedangkan untuk diagram alir dari algoritma *Caverphone 2.0* dapat dilihat pada gambar 4.9.



Gambar 4.10 Diagram Alir Algoritma *Caverphone 2.0*
 Sumber: [Perancangan]

Berikut ini adalah penjelasan tahap-tahap proses yang dilakukan algoritma *Caverphone 2.0*:

1. Mengambil masukan kata/ *String*.
2. Mengubah kata/ *String* menjadi kode fonetis sesuai dengan aturan yang telah ditentukan algoritma *Caverphone 2.0*. (aturan algoritma *Caverphone 2.0* dapat dilihat pada gambar 4.3.1.2.
3. Mendapatkan kode fonetis dari kata/ *String*.

Diagram alir untuk rule algoritma *caverphone* 2.0 dapat dilihat pada **lampiran 6**. Setelah kata/ *String* diubah menjadi kode fonetis, kemudian kode fonetis inilah yang nantinya akan dibandingkan dengan kode fonetis kata lain untuk menyimpulkan apakah kata memiliki kemiripan pengucapan atau tidak.

4.3.2 Perancangan Algoritma *Levenshtein Distance*

Proses yang terjadi dalam algoritma *Levenshtein Distance* adalah dengan mencari jarak antar kata yang dibandingkan. Jarak antar kata tersebut didapatkan dari 3 proses utama yaitu ada tidaknya penghapusan, ada tidaknya penyisipan, dan ada tidaknya penukaran *string* yang berdekatan. Kemudian dari jarak tersebut akan didapatkan jarak terpendek dari dua buah kata yang dibandingkan. *Pseudocode* algoritma *Levenshtein Distance* dapat dilihat pada gambar 4.10.

Nama Algoritma : *Levenshtein Distance*

Deklarasi

String data (kata kunci query), String word (kata dalam kamus), int m=length data, int n=length word, int d[][] , int i (perulangan data), int j (perulangan word), int cost

Deskripsi

Input : data

Proses :

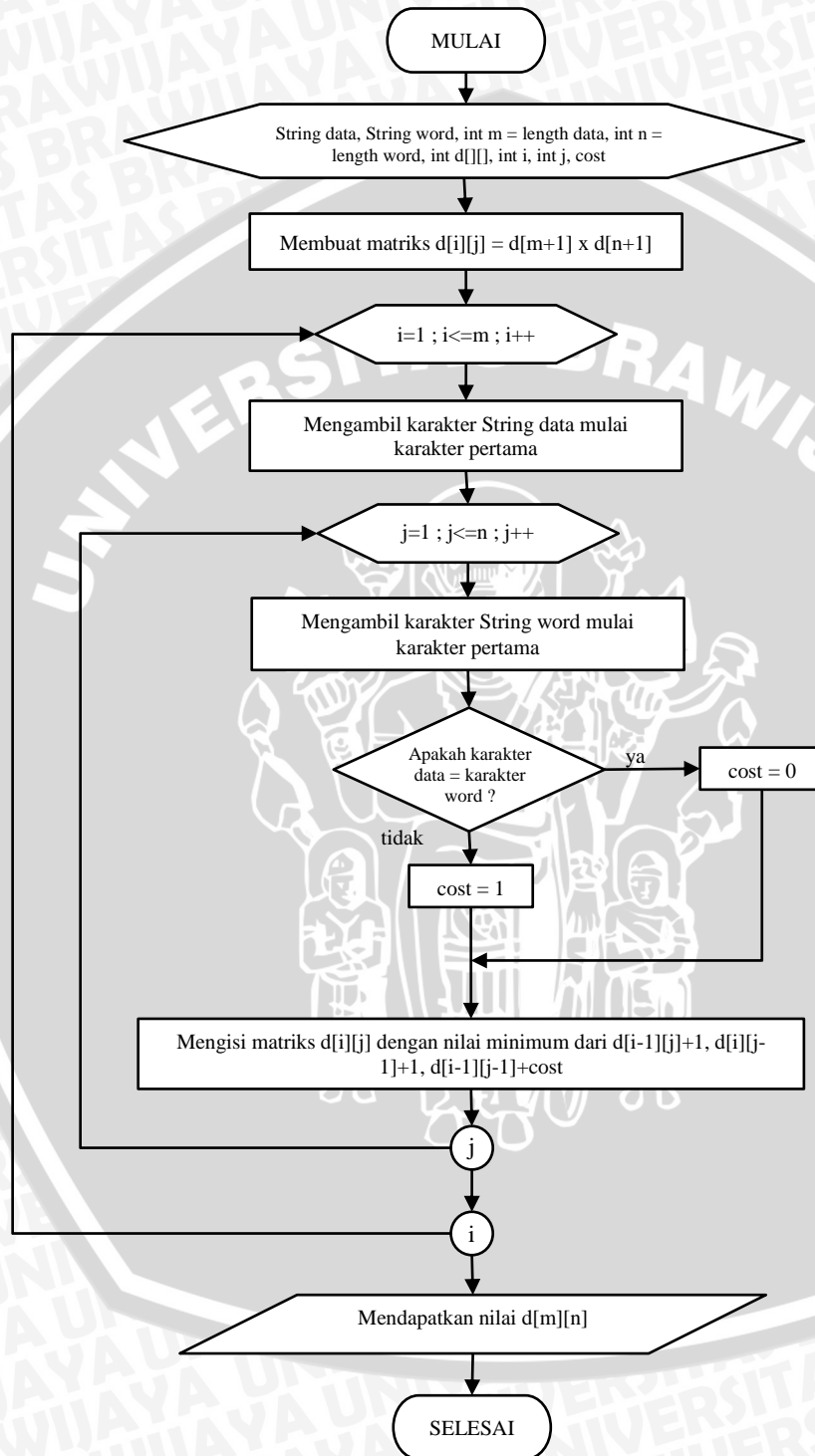
1. Membuat matriks [m+1][n+1]
2. Mengambil karakter dari String data mulai dari index ke-0 sampai n-1 dan memasukkannya ke dalam matriks [m+1]
3. Mengambil karakter dari String word mulai dari index ke-0 sampai m-1 dan memasukkannya ke dalam matriks [n+1]
4. Mencari nilai cost, jika karakter dari String data=karakter dari String word, maka nilai cost=0
5. Jika karakter dari String data != karakter dari String word, maka nilai cost=1
6. Mengisi matriks d[i][j] dengan nilai minimum antara d[i-1][j]+1, d[i][j-1]+1, dan d[i-1][j-1]+cost

Output : nilai *Levenshtein Distance*

Gambar 4.11 *Pseudocode* Algoritma *Levenshtein Distance*

Sumber: [Perancangan]

Diagram alir algoritma *Levenshtein Distance* akan menjelaskan urutan proses yang terjadi dalam algoritma. Diagram alir dari algoritma *Levenshtein Distance* dapat dilihat pada gambar 4.11.



Gambar 4.12 Diagram Alir Algoritma *Levenshtein Distance*

Sumber: [Perancangan]

Berikut ini adalah penjelasan tahap-tahap proses yang dilakukan algoritma *Levenshtein Distance*:

1. Membuat matriks $d[i][j]$ untuk menampung jarak tiap karakter pada setiap kata.
2. Mengambil karakter string 1 mulai karakter pertama.
3. Mengambil karakter string 2 mulai karakter pertama.
4. Mencocokkan karakter pada string 2 dengan karakter pada string 1.
5. Jika karakter sama, maka cost bernilai 0.
6. Jika karakter tidak sama, maka cost bernilai 1.
7. Mengisi matrik $d[i][j]$ dengan nilai minimum dari $d[i-1][j]+1$, $d[i][j-1]+1$, $d[i-1][j-1]+cost$.
8. Mengulangi proses hingga seluruh karakter pada string 2 selesai dihitung.
9. Mengulangi proses hingga seluruh karakter pada string 1 selesai dihitung.

4.4 Perancangan Antarmuka

Perancangan antarmuka sangat diperlukan untuk mempermudah *user* atau pengguna dalam menggunakan aplikasi kamus Inggris-Indonesia. Perancangan antarmuka aplikasi kamus Inggris-Indonesia dapat dilihat pada gambar 4.12 sebagai berikut.

The screenshot shows the home page of the 'KAMUFON' application. At the top, there is a title 'KAMUFON' and a subtitle 'Kamus Inggris-Indonesia dengan Fonetik'. Below the title, there are two buttons: 'HOME' and 'BANTUAN'. The main area contains a text input field labeled 'Masukkan kata bahasa Inggris'. Below the input field are two buttons: 'GENERATE' and 'PROSES'. Underneath these buttons is a box labeled 'Suggested Word'. Below that is another text input field. Below the input field are two more text input fields. At the bottom, there is a box labeled 'Kata yang mirip pengucapan'.

Gambar 4.13 Perancangan Antarmuka Aplikasi Halaman Home

Sumber: [Perancangan]

Keterangan gambar 4.12:

1. *TabPane* untuk masuk ke dalam halaman utama.
2. *TabPane* untuk masuk ke dalam halaman bantuan.
3. *Textfield* untuk masukan kata bahasa Inggris yang ingin dicari arti kata dalam bahasa Indonesia beserta cara pengucapannya.
4. Tombol generate untuk mendapatkan kode fonetis semua kata dalam *database* kamus.
5. Tombol proses untuk memproses kata bahasa Inggris hasil inputan oleh *user*.
6. *Jtable* untuk menampung *suggested word* dari kata bahasa Inggris yang tidak tersedia dalam *database* kamus.
7. *Textarea* untuk menampilkan arti kata dan cara pengucapan kata bahasa Inggris dalam bahasa Indonesia.
8. *Textarea* untuk menampilkan arti kata dan cara pengucapan kata bahasa Inggris yang memiliki kemiripan pengucapan agar dapat dibandingkan dengan kata masukan.
9. *Jtable* untuk menampung kata bahasa Inggris yang memiliki kemiripan pengucapan dengan kata bahasa Inggris masukan.



Gambar 4.14 Perancangan Antarmuka Aplikasi Halaman Bantuan

Sumber: [Perancangan]

Keterangan gambar 4.13:

1. *TabPane* untuk masuk ke dalam halaman utama.
2. *TabPane* untuk masuk ke dalam halaman bantuan.
3. *Textarea* untuk bantuan pembacaan simbol-simbol alfabet fonetik.

4.5 Contoh Penghitungan Manual Algoritma Caverphone 2.0

Pada bab ini akan dibahas mengenai contoh penghitungan manual algoritma *Caverphone 2.0* secara runtut dari awal hingga akhir. Contoh penghitungan manual menunjukkan proses perubahan kata bahasa Inggris menjadi kode fonetis. Pada tabel 4.4 dapat dilihat bahwa masukan berupa kata bahasa Inggris dan keluaran berupa 10 karakter kode fonetis.

Tabel 4.4 Contoh Penghitungan Manual Algoritma *Caverphone 2.0*

Rule Algoritma Caverphone 2.0		Contoh Kata M asukan			
Awal	Akhir	bad	every	autograph	factual
cough	cou2f	↓	↓	↓	↓
rough	rou2f				
tough	tou2f				
enough	enou2f				
trough	trou2f				
gn	2n				
mb	m2				
cq	2q				
ci	si				
ce	se				
cy	sy				
tch	2ch				
c	k				
q	k				
x	k				
v	f	↓	↓	↓	↓
dg	2g				
tio	sio				
tia	sia				
d	t				
ph	fh				
b	p				
sh	s2				
z	s				
huruf vokal awal	A				
huruf vokal lain	3				
j	y				
y3 awal kata	Y3				
y awal kata	A				
y	3				
3gh3	3kh3				
gh	22				
g	k				
kumpulan huruf s	S				
kumpulan huruf t	T				
kumpulan huruf p	P				
kumpulan huruf k	K				
kumpulan huruf f	F				
		bat	efery	autografh	faktual
		pat	Afery	Autografh	f3kt33l
		p3t	Af3ry	A3t3gr3fh	
			Af3r3		
				A3t3kr3fh	
		p3T		A3T3kr3fh	f3kT33l
		P3T		A3T3Kr3fh	f3KT33l
			Af3r3	A3T3Kr3fh	F3KT33l



kumpulan huruf m	M				
kumpulan huruf n	N				
w3	W3				
wh3	Wh3				
w pada karakter paling akhir	3				
W	2				
h karakter awal	A				
h selain karakter awal	2				
r3	R3				
r pada karakter paling akhir	3				
r	2				
l3	L3				
l pada karakter paling akhir	3				
l	2				
menghilangkan semua karakter 2					
3 pada karakter terakhir	A				
menghilangkan semua karakter 3					
menambahkan 10 karakter 1 di akhir kata					
mengambil 10 karakter pertama sebagai kode fonetis					

Sumber: [Perancangan]

Penjelasan secara runtut tabel 4.4 yang menggambarkan proses perubahan kata bahasa Inggris menjadi kode fonetis dapat dijelaskan sebagai berikut:

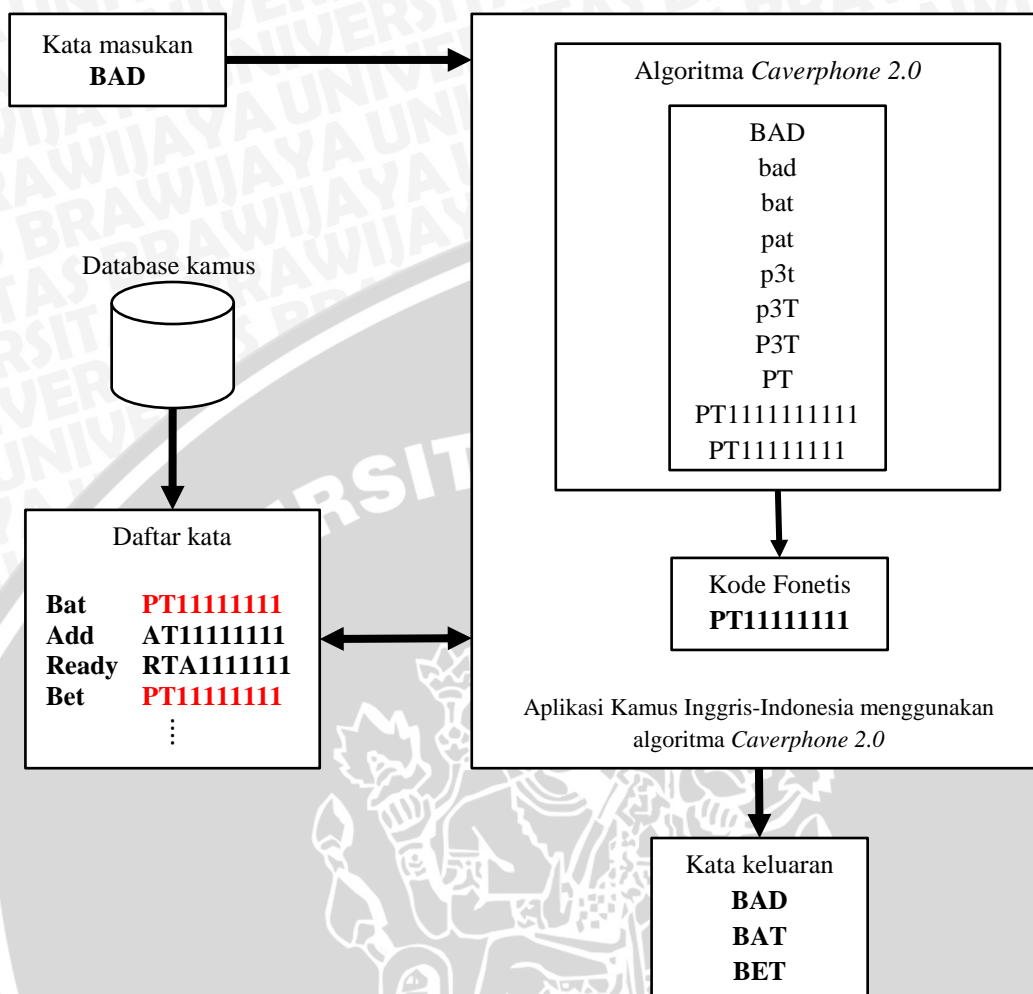
1. Kata **bad**. Dalam proses menjadi kode fonetis, kata **bad** melalui tahapan sebagai berikut:
 - Karakter d diubah menjadi t. Maka, hasilnya akan menjadi **bat**.
 - Karakter b diubah menjadi p. Maka, hasilnya akan menjadi **pat**.
 - Karakter huruf vokal a diubah menjadi 3. Maka, hasilnya akan menjadi **p3t**.
 - Kumpulan karakter t diubah menjadi T. Maka, hasilnya akan menjadi **p3T**.
 - Kumpulan karakter p diubah menjadi P. Maka, hasilnya akan menjadi **P3T**.
 - Menghilangkan karakter 3. Maka, hasilnya akan menjadi **PT**.
 - Kode fonetis kata **bad** ditambah karakter 1 hingga mencapai 10 karakter. Maka, kode fonetis kata **bad** menjadi **PT11111111**.
2. Kata **every**. Dalam proses menjadi kode fonetis, kata **every** melalui tahapan sebagai berikut:

- Karakter v diubah menjadi f. Maka, hasilnya akan menjadi **efery**.
 - Karakter huruf vokal e pada awal kata diubah menjadi A. Maka, hasilnya akan menjadi **Afery**.
 - Karakter huruf vokal e diubah menjadi 3. Maka, hasilnya akan menjadi **Af3ry**.
 - Karakter y diubah menjadi 3. Maka, hasilnya akan menjadi **Af3r3**.
 - Kumpulan karakter f diubah menjadi F. Maka, hasilnya akan menjadi **AF3r3**.
 - Karakter r3 diubah menjadi R3. Maka, hasilnya akan menjadi **AF3R3**.
 - Karakter 3 pada karakter terakhir diubah menjadi A. Maka, hasilnya akan menjadi **AF3RA**.
 - Menghilangkan karakter 3. Maka, hasilnya akan menjadi **AFRA**.
 - Kode fonetis kata **every** ditambah karakter 1 hingga mencapai 10 karakter. Maka, kode fonetis kata **every** menjadi **AFRA111111**.
3. Kata **autograph**. Dalam proses menjadi kode fonetis, kata **every** melalui tahapan sebagai berikut:
- Karakter ph diubah menjadi fh. Maka, hasilnya akan menjadi **autograph**.
 - Karakter huruf vokal a pada awal kata diubah menjadi A. Maka, hasilnya akan menjadi **Autograph**.
 - Karakter huruf vokal lain diubah menjadi 3. Maka, hasilnya akan menjadi **A3t3gr3fh**.
 - Karakter g diubah menjadi k. Maka, hasilnya akan menjadi **A3t3kr3fh**.
 - Kumpulan karakter t diubah menjadi T. Maka, hasilnya akan menjadi **A3T3kr3fh**.
 - Kumpulan karakter k diubah menjadi K. Maka, hasilnya akan menjadi **A3T3Kr3fh**.
 - Kumpulan karakter f diubah menjadi F. Maka, hasilnya akan menjadi **A3T3Kr3Fh**.
 - Karakter h diubah menjadi 2. Maka, hasilnya akan menjadi **A3T3Kr3F2**.
 - Karakter r3 diubah menjadi R3. Maka, hasilnya akan menjadi **A3T3KR3F2**.
 - Menghilangkan semua karakter 2. Maka, hasilnya akan menjadi **A3T3KR3F**.
 - Menghilangkan semua karakter 3. Maka, hasilnya akan menjadi **ATKRF**.

- Kode fonetis kata **autograph** ditambah karakter 1 hingga mencapai 10 karakter. Maka, kode fonetis kata **autograph** menjadi **ATKRF11111**.
- 4. Kata **factual**. Dalam proses menjadi kode fonetis, kata **factual** melalui tahapan sebagai berikut:
 - Karakter c diubah menjadi k. Maka, hasilnya akan menjadi **faktual**.
 - Karakter huruf vokal diubah menjadi 3. Maka, hasilnya akan menjadi **f3kt33l**.
 - Kumpulan karakter t diubah menjadi T. Maka, hasilnya akan menjadi **f3kT33l**.
 - Kumpulan karakter k diubah menjadi K. Maka, hasilnya akan menjadi **f3KT33l**.
 - Kumpulan karakter f diubah menjadi F. Maka, hasilnya akan menjadi **F3KT33l**.
 - Karakter l pada karakter paling akhir diubah menjadi 3. Maka, hasilnya akan menjadi **F3KT333**.
 - Karakter 3 pada karakter paling akhir diubah menjadi A. Maka, hasilnya akan menjadi **F3KT33A**.
 - Menghilangkan semua karakter 3. Maka, hasilnya akan menjadi **FKTA**.
 - Kode fonetis kata **factual** ditambah karakter 1 hingga mencapai 10 karakter. Maka, kode fonetis kata **factual** menjadi **FKTA111111**.

4.6 Contoh Pencocokan dengan Metode *Phonetic String Matching*

Pada bab ini akan dibahas mengenai contoh pencocokan kata bahasa Inggris dengan menggunakan metode *Phonetic String Matching*. Selain itu juga dibahas proses yang terjadi dalam aplikasi dari data masukan hingga keluaran data.

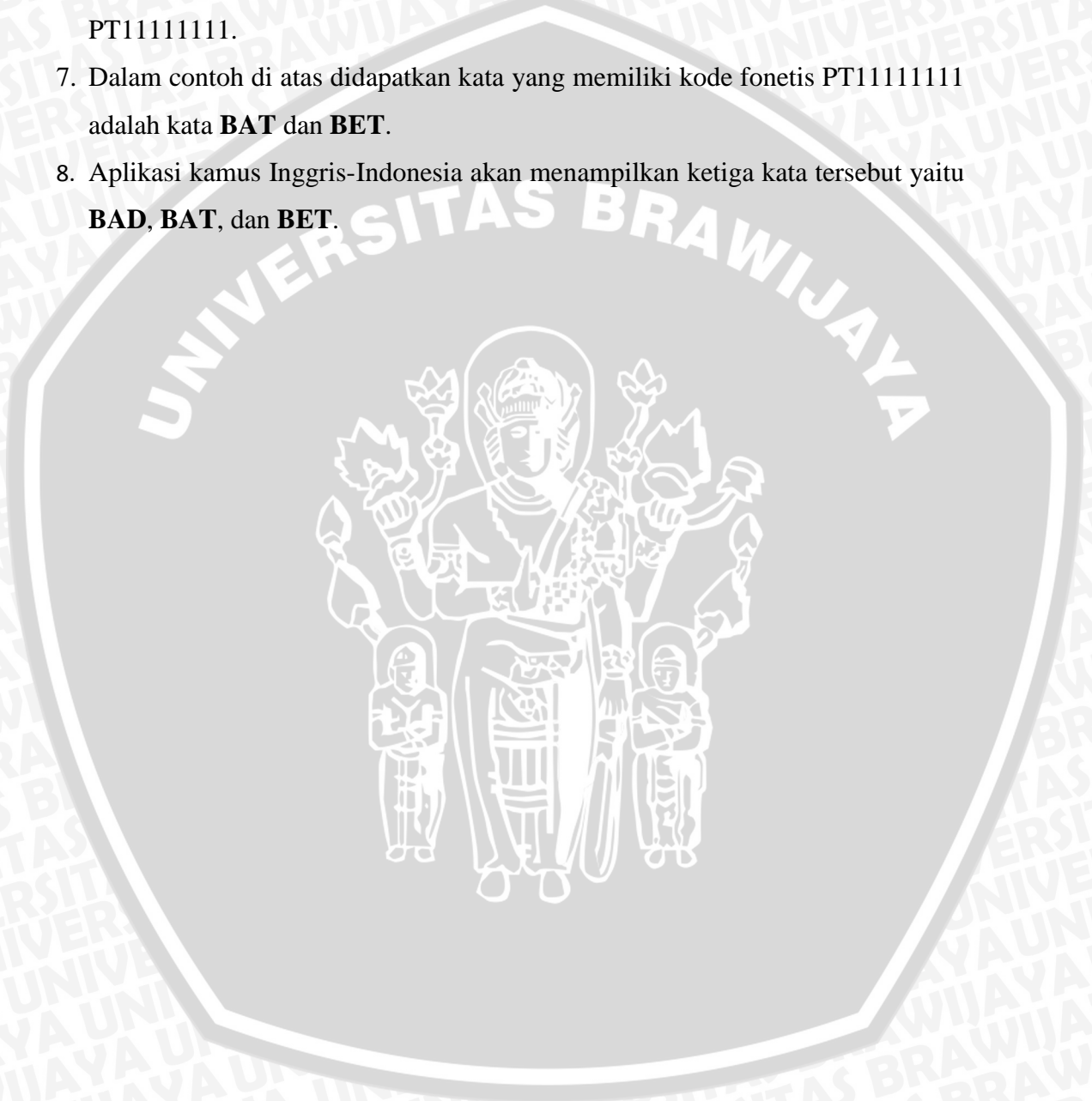


Gambar 4.15 Diagram Blok Pencocokan Kata menggunakan Metode *Phonetic String Matching*

Berikut ini adalah penjelasan diagram blok pencocokan kata menggunakan metode *Phonetic String Matching* pada gambar 4.15:

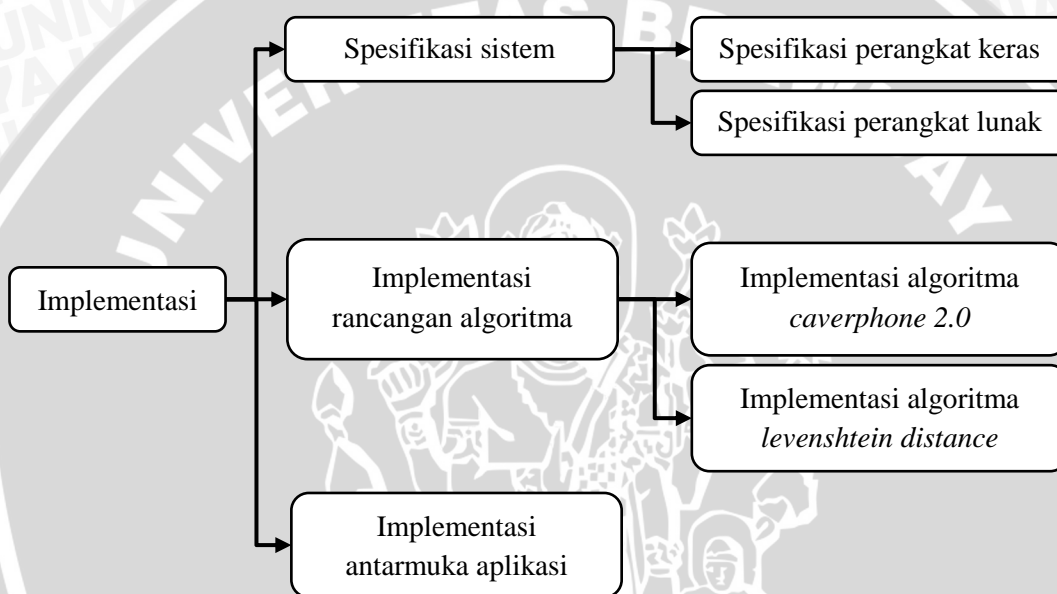
1. Mengambil *input*/ masukan berupa kata. Dalam contoh di atas, kata yang diambil adalah kata **BAD**.
2. Kata **BAD** kemudian diolah pada aplikasi kamus Inggris-Indonesia yang mengimplementasikan algoritma *Caverphone 2.0* di dalamnya.
3. Algoritma *Caverphone 2.0* kemudian akan mengubah kata **BAD** menjadi kode fonetis sesuai aturan yang diterapkan oleh algoritma *Caverphone 2.0*.

4. Dalam contoh di atas terlihat bahwa kata **BAD** akan memiliki kode fonetis PT11111111.
5. Kemudian kode fonetis dari kata **BAD** akan dicocokkan dengan kode fonetis dari semua kata yang ada di dalam *database* kamus.
6. Dalam *database* kamus, kemudian dicari kode fonetis yang sama dengan PT11111111.
7. Dalam contoh di atas didapatkan kata yang memiliki kode fonetis PT11111111 adalah kata **BAT** dan **BET**.
8. Aplikasi kamus Inggris-Indonesia akan menampilkan ketiga kata tersebut yaitu **BAD**, **BAT**, dan **BET**.



BAB V IMPLEMENTASI

Bab ini membahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, implementasi rancangan algoritma, dan implementasi antarmuka aplikasi. Pohon implementasi aplikasi dapat ditunjukkan pada gambar 5.1.



Gambar 5.1 Pohon Implementasi
Sumber: [Implementasi]

1.1 Spesifikasi Sistem

Hasil analisis kebutuhan dan perancangan perangkat lunak yang telah dijelaskan pada Bab 4 menjadi acuan untuk melakukan implementasi menjadi sebuah program aplikasi kamus Inggris-Indonesia yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Nama Komponen	Spesifikasi
Prosesor	Intel ® Pentium ® CPU P6100 @ 2.00 GHz (2CPUs), ~2.0 GHz
Memori (RAM)	4096 MB RAM DDR3
Hard Drive	300 GB hard disk drive
Motherboard	Dell 0KRP5X A03

Sumber: [Implementasi]

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Komputer

Nama Komponen	Spesifikasi	Kegunaan
Sistem Operasi	Windows 7 Ultimate 32-bit	Lingkungan pengembangan aplikasi
Bahasa Pemrograman	Java JDK 1.6.0_11	Platform yang digunakan untuk mengembangkan aplikasi
Tools Pemrograman	Netbeans IDE 6.9.1	Graphic User Interface untuk pengembangan aplikasi berbasis Java
Database	MySQL 5.5.8	Penyimpanan data

Sumber: [Implementasi]

1.2 Implementasi Rancangan Algoritma

Perancangan algoritma dari bab sebelumnya menghasilkan implementasi algoritma antara lain implementasi algoritma *Caverphone 2.0*, dan implementasi algoritma *Levenshtein Distance*.

Pada penulisan laporan skripsi ini hanya dicantumkan implementasi algoritma untuk beberapa proses operasi saja, selebihnya akan disertakan di lampiran.

1.2.1 Implementasi Algoritma *Caverphone 2.0*

Proses pembentukan kode fonetis dari setiap kata bahasa Inggris dilakukan dengan mengubah karakter demi karakter kata bahasa Inggris sesuai rule algoritma *Caverphone 2.0*. Kode fonetis yang didapatkan dari setiap kata bahasa Inggris kemudian akan dimasukkan ke dalam kolom *k_fonetis* pada *database* kamus. Implementasi rancangan algoritma *Caverphone 2.0* mengacu kepada perancangan algoritma *caverphone 2.0* pada sub bab 4.3.1 halaman 40-42. Implementasi rancangan algoritma *caverphone 2.0* dibagi menjadi 3 bagian yaitu:

- *Sourcecode* pendeklarasian variabel.
- *Sourcecode* penghilangan simbol-simbol.
- *Sourcecode* proses pengubahan kata menjadi kode fonetis.

Sourcecode pendeklarasian variabel pada proses algoritma *caverphone 2.0* dapat dilihat pada gambar 5.2.

```
1 public class Fonetik {
2
3     private String kata;
4     private String word;
5     private String akhiran;
6
7 }
```

Gambar 5.2 Tampilan *Sourcecode* Proses Pendeklarasian Variabel *Class* Fonetik
Sumber: [Implementasi]

Penjelasan *sourcecode* proses pendeklarasian variabel pada gambar 5.2 dapat dijelaskan sebagai berikut:

1. Baris 1 adalah *class* Fonetik yang digunakan untuk menampung proses algoritma *caverphone 2.0*.
2. Baris 3 adalah pendeklarasian variabel kata bertipe *String*. Variabel ini digunakan untuk menyimpan kata bahasa Inggris hasil masukan dari pengguna.
3. Baris 4 adalah pendeklarasian variabel word bertipe *String*. Variabel ini digunakan untuk menyimpan kata hasil perubahan tiap tahap menjadi kode fonetis.

4. Baris 5 adalah pendeklarasian variabel akhiran bertipe *String*. Variabel ini digunakan untuk menyimpan akhiran dari kata bahasa Inggris.

Sourcecode penghilangan simbol-simbol pada proses algoritma *caverphone 2.0* dapat dilihat pada gambar 5.3.

```

1 public String caverphone(String kata){
2     this.kata = kata;
3
4     word = this.kata.replaceAll(" ", "");
5     word = word.replaceAll("-", "");
6     word = word.replaceAll("_", "");
7     word = word.replaceAll("!", "");
8     word = word.replaceAll("@", "");
9     word = word.replaceAll("#", "");
10    word = word.replaceAll("$", "");
11    word = word.replaceAll("%", "");
12    word = word.replaceAll(",", "");
13    word = word.replaceAll("\\.", "");
14    word = word.replaceAll("'", "");
15    word = word.replaceAll("\"", "");
16
17    word = word.toLowerCase();
18 }

```

Gambar 5.3 Tampilan *Sourcecode* Proses Penghilangan Simbol-Simbol
Sumber: [Implementasi]

Penjelasan *sourcecode* proses pendeklarasian variabel pada gambar 5.2 dapat dijelaskan sebagai berikut:

1. Baris 1 adalah *method* *caverphone*.
2. Baris 2 adalah pengisian nilai variabel lokal *kata* dengan masukan *kata* dari parameter *method*.
3. Baris 4 adalah proses menghilangkan tanda spasi untuk setiap *kata* masukan.
4. Baris 5-15 adalah proses untuk menghilangkan semua tanda baca dan simbol untuk setiap *kata* masukan.
5. Baris 17 adalah proses untuk mengubah *kata* masukan menjadi *lowercase* atau huruf kecil.

Untuk *sourcecode* tahap-tahap perubahan *kata* menjadi kode fonetis dapat dilihat pada **lampiran 5**.

5.2.2 Implementasi Algoritma *Levenshtein Distance*

Proses *Levenshtein Distance* dilakukan dengan membuat matriks dari dua *kata* yang dibandingkan (*kata* yang salah dengan *kata* pada kamus). Dari setiap

kata yang salah dicari jaraknya dengan seluruh kata yang ada di *database* dan didapatkan nilai *Levenshtein Distance*. Mengacu kepada perancangan algoritma *levenshtein distance* pada sub bab 4.3.2 halaman 43-46, berikut ini adalah implementasi algoritma dari proses *Levenshtein Distance*.

```
1 public class Levenshtein {
2
3     private Koneksi kon;
4
5     public int LD (String data, String word) {
6
7         int d[][];
8         int n;
9         int m;
10        int i;
11        int j;
12        char data_i;
13        char word_j;
14        int cost;
15
16
17        n = data.length();
18        m = word.length();
19
20        if (n == 0) {
21            return m;
22        }
23
24        if (m == 0) {
25            return n;
26        }
27
28        d = new int[n+1][m+1];
29
30
31        for (i = 0; i <= n; i++) {
32            d[i][0] = i;
33        }
34
35
36        for (j = 0; j <= m; j++) {
37            d[0][j] = j;
38        }
39
40
41        for (i = 1; i <= n; i++) {
42
43            data_i = data.charAt(i-1);
44
45
46            for (j = 1; j <= m; j++) {
47
48                word_j = word.charAt(j-1);
49
50                if (data_i == word_j) {
51                    cost = 0;
52                }
53
54                else {
```

```
55         cost = 1;
56     }
57
58     d[i][j] = minimal (d[i-1][j]+1, d[i][j-1]+1, d[i-1][j-1] + cost);
59 }
60 }
61 return d[n][m];
62 }
63 }
```

Gambar 5.4 Tampilan *Sourcecode* Proses *Levenshtein Distance*

Sumber: [Implementasi]

Penjelasan *sourcecode* algoritma *levenshtein distance* pada gambar 5.4 dapat dijelaskan sebagai berikut:

1. Baris 1 adalah *class* *Levenshtein* untuk menampung proses algoritma *levenshtein distance*.
2. Baris 5 adalah *method* *levenshtein distance* yang diberi nama *LD* dengan parameter string *data* dan string *word*.
3. Baris 7 adalah proses deklarasi variabel matriks *d* dua dimensi untuk menampung perhitungan jarak antar karakter pada setiap kata.
4. Baris 8 adalah proses deklarasi variabel *n* bertipe integer untuk menampung banyaknya karakter pada string *data*.
5. Baris 9 adalah proses deklarasi variabel *m* bertipe integer untuk menampung banyaknya karakter pada string *word*.
6. Baris 10 adalah proses deklarasi variabel *i* bertipe integer untuk proses perulangan pada string *data*.
7. Baris 11 adalah proses deklarasi variabel *j* bertipe integer untuk proses perulangan pada string *word*.
8. Baris 12 adalah proses deklarasi variabel *data_i* bertipe *char* untuk menyimpan karakter yang akan dibandingkan dari string *data*.
9. Baris 13 adalah proses deklarasi variabel *word_j* bertipe *char* untuk menyimpan karakter yang akan dibandingkan dari string *word*.
10. Baris 14 adalah proses deklarasi variabel *cost* yang bertipe integer untuk menyimpan nilai *cost* dari hasil perbandingan tiap karakter antara string *data* dan string *word*.

11. Baris 17 adalah proses untuk mengisi variabel n dengan panjang atau jumlah karakter dari string data.
12. Baris 18 adalah proses untuk mengisi variabel m dengan panjang atau jumlah karakter dari string word.
13. Baris 20-26 adalah proses untuk mendapatkan panjang karakter dari tiap string. Jika string data bernilai 0, maka akan mengembalikan panjang karakter string word. Jika string word bernilai 0, maka akan mengembalikan panjang karakter string data.
14. Baris 28-38 adalah proses untuk mengisi indeks matriks d dengan panjang karakter string data dan panjang karakter string word yang ditambah 1.
15. Baris 41-48 adalah proses untuk perulangan karakter pada string data dan string word.
16. Baris 50-56 adalah proses untuk mendapatkan nilai cost. Jika karakter yang dibandingkan antara string data dan string word sama, maka cost akan bernilai 0. Sedangkan, jika karakter yang dibandingkan antara string data dan string word tidak sama, maka cost akan bernilai 1.
17. Baris 58 adalah proses untuk mengisi matriks d dengan nilai minimum antara $d[i-1][j]+1$, $d[i][j-1]+1$, dan $d[i-1][j-1] + \text{cost}$.
18. Baris 61 adalah proses untuk mengembalikan nilai matriks d yang sudah terisi jarak minimal masing-masing karakter antara string data dan string word.

1.3 Implementasi Antarmuka Aplikasi

Program aplikasi kamus Inggris-Indonesia menggunakan metode *phonetic string matching* dengan algoritma *caverphone 2.0* memiliki kegunaan untuk pencarian kata bahasa Inggris yang memiliki kemiripan pengucapan serta mendapatkan arti kata dan cara pengucapan setiap kata bahasa Inggris.

Implementasi rancangan algoritma *main program* dibagi menjadi 3 bagian yaitu:

- *Sourcecode* pendeklarasian variabel.
- *Sourcecode* konstruktor *main program*.

- *Sourcecode method* isidata() yang dibagi menjadi *sourcecode* untuk proses *levenshtein distance* dan *sourcecode* untuk proses *caverphone 2.0.*

Sourcecode pendeklarasian variabel pada *main program* dapat dilihat pada gambar 5.5.

```

1 public class Kamus extends javax.swing.JFrame {
2
3     private Query qur;
4     private Fonetik fon;
5     private Levenshtein lev;
6     private DefaultTableModel model = new DefaultTableModel();
7     private DefaultTableModel model2 = new DefaultTableModel();
8 }

```

Gambar 5.5 Tampilan *Sourcecode* Pendeklarasian Variabel Main Program
Sumber: [Implementasi]

Penjelasan *sourcecode* proses pendeklarasian variabel pada main program gambar 5.5 dapat dijelaskan sebagai berikut:

1. Baris 1 adalah *class* Kamus yang digunakan sebagai main program.
2. Baris 3 adalah pendeklarasian variabel qur bertipe *class* Query. Ini diperlukan agar *class* Kamus dapat membaca kode dari *class* Query.
3. Baris 4 adalah pendeklarasian variabel fon bertipe *class* Fonetik. Ini diperlukan agar *class* Kamus dapat membaca kode dari *class* Fonetik.
4. Baris 5 adalah pendeklarasian variabel lev bertipe *class* Levenshtein. Ini diperlukan agar *class* Kamus dapat membaca kode dari *class* Levenshtein.
5. Baris 6 adalah pendeklarasian *tablemodel* dengan variabel model yang diinisialisasi dengan *new DefaultTableModel()*. Digunakan untuk pemodelan tabel daftar kata yang memiliki kemiripan pengucapan dengan kata masukan pada antarmuka aplikasi.
6. Baris 7 adalah pendeklarasian *tablemodel* dengan variabel model2 yang diinisialisasi dengan *new DefaultTableModel()*. Digunakan untuk pemodelan tabel *suggested word* pada antarmuka aplikasi.

Sourcecode konstruktor pada main program/ *class* Kamus dapat dilihat pada gambar 5.6.

```

1 public Kamus()
2 {
3     initComponents();
4     model.addColumn("Kata yang memiliki kemiripan pengucapan");
5     model2.addColumn("Suggested Word");

```



```

6
7         jTable1.setModel(model);
8         jTable2.setModel(model2);
9
10        qur = new Query();
11    }

```

Gambar 5.6 Tampilan *Sourcecode* Konstruktor Main Program
Sumber: [Implementasi]

Penjelasan *sourcecode* konstruktor pada main program gambar 5.6 dapat dijelaskan sebagai berikut:

1. Baris 1 adalah konstruktor Kamus.
2. Baris 3 adalah pemanggilan *method* `initComponents()`.
3. Baris 4 adalah penambahan kolom pada model dengan nama kolom 'kata yang memiliki kemiripan pengucapan' yang akan ditampilkan pada antarmuka aplikasi.
4. Baris 5 adalah penambahan kolom pada model2 dengan nama kolom 'suggested word' yang akan ditampilkan pada antarmuka aplikasi.
5. Baris 7 adalah proses pengisian nilai `jTable1` oleh nilai model. Kode ini untuk menampilkan model pada `jTable1` pada antarmuka aplikasi.
6. Baris 8 adalah proses pengisian nilai `jTable2` oleh nilai model2. Kode ini untuk menampilkan model2 pada `jTable2` pada antarmuka aplikasi.
7. Baris 10 adalah instance *class* `Query` yang bertujuan agar semua kode pada *class* `Query` dapat terbaca oleh *class* main program.

Sourcecode method `isiData()` pada main program/ *class* `Kamus` dapat dilihat pada gambar 5.7.

```

1  public void isiData()
2  {
3      String inputan;
4      String kata;
5      ResultSet hasil ;
6      ResultSet hsl ;
7      String kt=null;
8      String katas;
9      String pengucapan;
10     String arti;
11     String gabung[]=new String[1000];
12     String charSet = null;
13     int jarak;
14
15     inputan = jTextField1.getText();
16     hsl = qur.getkata("select kata from kata where kata = '" +
17     inputan + "'");

```

```
18     try
19     {
20         while (hsl.next())
21         {
22             kt = hsl.getString("kata");
23         }
24     }
25     catch(SQLException ex)
26     {
27
28     Logger.getLogger(Kamus.class.getName()).log(Level.SEVERE, null, ex);
29     }
30
31     if(kt==null)
32     {
33         lev = new Levenshtein();
34         hasil = qur.getkata("select kata from kata");
35         try {
36             while (hasil.next() == true) {
37                 katas = hasil.getString("kata");
38                 jarak = lev.LD(inputan, katas);
39                 qur.addStringValue(katas, jarak);
40             }
41         } catch (SQLException ex) {
42
43     Logger.getLogger(Kamus.class.getName()).log(Level.SEVERE, null, ex);
44         }
45         qur.showdata();
46         jTextArea1.setText("Maaf kata tidak tersedia");
47         jTextArea2.setText(null);
48         System.out.println(""+qur.getmin());
49
50         for(int z=0;z<qur.getkata().size();z++)
51         {
52
53     System.out.println("kata="+qur.getkata().get(z));
54             model2.addRow(new Object[]
55     {qur.getkata().get(z)});
56         }
57     }
58     else
59     {
60         fon = new Fonetik();
61         kata = fon.caverphone(inputan);
62
63         hasil = qur.getkata("select kata from kata where
64     k_fonetis = '"+kata+"' and kata<> '"+inputan+"'");
65         try
66         {
67             int a=0;
68             while(hasil.next()==true)
69             {
70                 model.addRow(new Object[]{
71                     hasil.getString("kata")
72                 });
73                 a=a+1;
74             }
75         }catch (Exception ex)
76         {
77         }
78         hasil = qur.getkata("select kata, pengucapan, arti from
79     kata where kata='"+inputan+"'");
80         try{
```

```

81     hasil.next();
82     katas = hasil.getString("kata");
83     pengucapan = new String(hasil.getBytes("pengucapan"));
84     arti = hasil.getString("arti");
85     gabung[0] = "kata
86     :"+katas+"\npengucapan:"+pengucapan+"\narti:"+arti;
87     jTextArea1.setText(gabung[0]);
88     jTextArea2.setText(null);
89     }catch (Exception ex){
90
91     JOptionPane.showMessageDialog(null,"Error"+ex.getMessage());
92     }
93     }
94     }

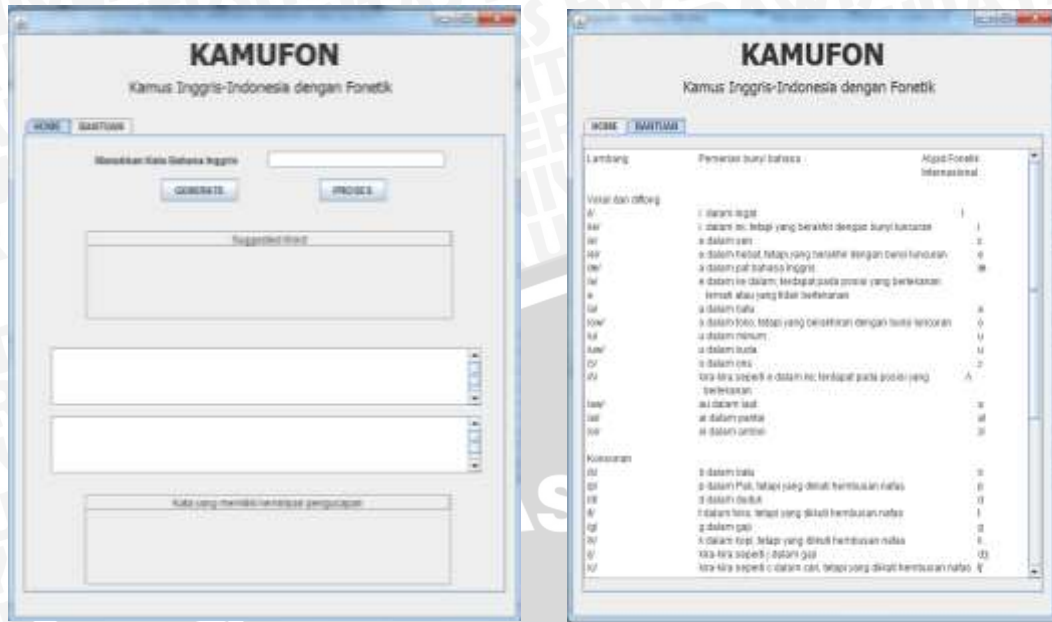
```

Gambar 5.7 Tampilan *Sourcecode Method* isidata
Sumber: [Implementasi]

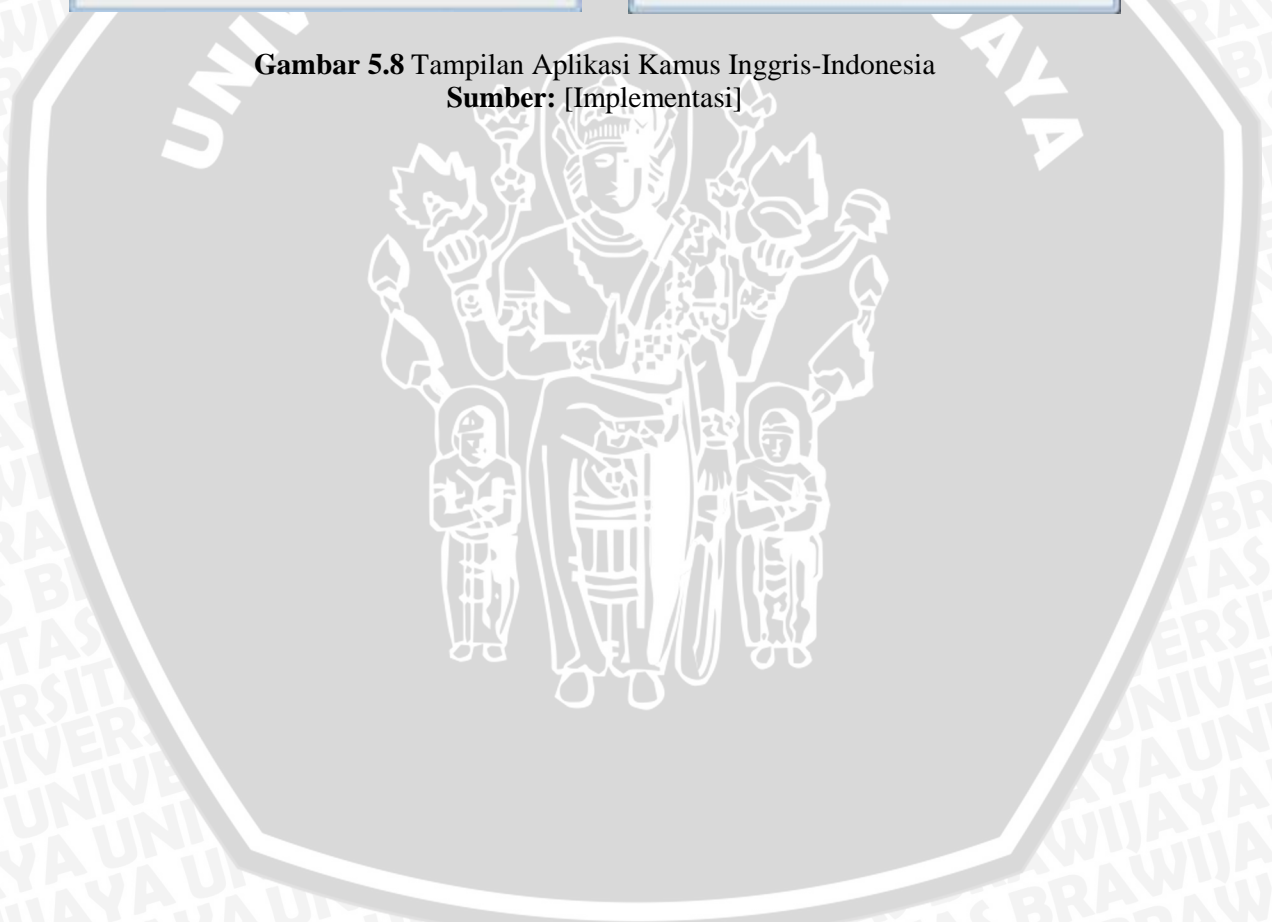
Penjelasan *sourcecode method* isidata() pada main program gambar 5.7 dapat dijelaskan sebagai berikut:

1. Baris 1 adalah *method* isidata().
2. Baris 3-13 adalah pendeklarasian variabel yang digunakan pada *method* isidata().
3. Baris 15 adalah mengisi nilai variabel inputan dengan nilai dari `jTextField1`.
4. Baris 16-29 adalah proses pengecekan apakah kata bahasa Inggris yang dimasukkan pada `jTextField1` ada dalam kamus atau tidak.
5. Baris 31-55 adalah proses pencarian *suggested word* menggunakan algoritma *levenshtein distance* yang dipanggil dari *class* Levenshtein jika kata masukan bahasa Inggris tidak ditemukan dalam kamus.
6. Baris 58-77 adalah proses pencarian kata yang memiliki kemiripan pengucapan menggunakan algoritma *caverphone 2.0* yang dipanggil dari *class* Fonetik jika kata masukan bahasa Inggris ditemukan dalam kamus.
7. Baris 78-94 adalah proses ditampilkannya kata bahasa Inggris, cara pengucapan, beserta arti kata dalam bahasa Indonesia dari kata masukan bahasa Inggris pada `jTextfield1`.

Berikut ini adalah tampilan utama aplikasi kamus Inggris-Indonesia menggunakan algoritma *caverphone 2.0*.



Gambar 5.8 Tampilan Aplikasi Kamus Inggris-Indonesia
 Sumber: [Implementasi]



BAB VI PENGUJIAN DAN ANALISIS

Bab ini akan membahas mengenai pengujian dari aplikasi perangkat lunak yang telah dibangun. Proses pengujian terdiri dari dua tahap yaitu pengujian validasi dan pengujian akurasi. Pengujian validasi dilakukan untuk menilai apakah aplikasi perangkat lunak sudah dapat menjalankan fungsi-fungsi yang diharapkan. Pengujian akurasi digunakan untuk menilai tingkat kecocokan hasil yang dikeluarkan oleh aplikasi.

6.1 Pengujian

Pengujian yang akan dilakukan pada tahap ini meliputi pengujian validasi dan pengujian akurasi.

6.1.1 Pengujian Validasi

Pengujian validasi dilakukan untuk menilai apakah aplikasi perangkat lunak sudah dapat menjalankan fungsi-fungsi yang diharapkan sebagaimana mestinya. Kebutuhan yang telah dirumuskan dalam analisis kebutuhan akan dijadikan acuan untuk melakukan pengujian validasi.

Untuk mengetahui kesesuaian antara kebutuhan dengan kinerja aplikasi, pada setiap kebutuhan dilakukan proses pengujian dengan kasus uji masing-masing.

Tabel 6.1 Test Case untuk Pengujian Validasi *input* Kata

Nama Kasus Uji	<i>Input</i> kata bahasa Inggris
Tujuan Pengujian	Untuk menguji validitas kinerja dari aplikasi dalam memberikan fasilitas <i>input</i> kata bahasa Inggris agar <i>user</i> dapat memasukkan kata bahasa Inggris untuk diketahui artinya, cara pengucapannya, dan kata lain yang memiliki kemiripan bunyi
Prosedur Uji	<ol style="list-style-type: none"> 1. Menjalankan aplikasi kamus 2. <i>User</i> mengetikkan kata bahasa Inggris yang ingin diproses pada <i>textfield</i> yang telah disediakan

Hasil yang Diharapkan	Aplikasi dapat menampilkan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i>
-----------------------	---

Sumber: [Pengujian dan Analisis]

Tabel 6.2 Test Case untuk Pengujian Validasi Arti Kata

Nama Kasus Uji	Arti kata bahasa Inggris
Tujuan Pengujian	Untuk menguji validitas kinerja dari aplikasi dalam memberikan fasilitas untuk menampilkan arti kata bahasa Inggris yang dimasukkan oleh <i>user</i> .
Prosedur Uji	<ol style="list-style-type: none"> 1. Menjalankan aplikasi kamus 2. <i>User</i> mengetikkan kata bahasa Inggris yang ingin diproses pada <i>textfield</i> yang telah disediakan 3. <i>User</i> menekan tombol proses
Hasil yang Diharapkan	Aplikasi dapat menampilkan arti kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i>

Sumber: [Pengujian dan Analisis]

Tabel 6.3 Test Case untuk Pengujian Validasi Pengucapan Kata

Nama Kasus Uji	Pengucapan kata bahasa Inggris
Tujuan Pengujian	Untuk menguji validitas kinerja dari aplikasi dalam memberikan fasilitas untuk menampilkan cara pengucapan kata bahasa Inggris yang dimasukkan oleh <i>user</i> .
Prosedur Uji	<ol style="list-style-type: none"> 1. Menjalankan aplikasi kamus 2. <i>User</i> mengetikkan kata bahasa Inggris yang ingin diproses pada <i>textfield</i> yang telah disediakan 3. <i>User</i> menekan tombol proses
Hasil yang Diharapkan	Aplikasi dapat menampilkan cara pengucapan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i>

Sumber: [Pengujian dan Analisis]

Tabel 6.4 Test Case untuk Pengujian Validasi Kata Mirip Pengucapan

Nama Kasus Uji	Kata bahasa Inggris yang mirip pengucapan
Tujuan Pengujian	Untuk menguji validitas kinerja dari aplikasi dalam memberikan fasilitas untuk menampilkan kata bahasa Inggris lain yang memiliki kemiripan pengucapan dengan kata bahasa Inggris yang dimasukkan oleh <i>user</i> .
Prosedur Uji	<ol style="list-style-type: none"> 1. Menjalankan aplikasi kamus 2. <i>User</i> mengetikkan kata bahasa Inggris yang ingin diproses pada <i>textfield</i> yang telah disediakan 3. <i>User</i> menekan tombol proses
Hasil yang Diharapkan	Aplikasi dapat menampilkan kata bahasa Inggris lain yang memiliki kemiripan pengucapan dengan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i>

Sumber: [Pengujian dan Analisis]

Tabel 6.5 Test Case untuk Pengujian Validasi Cek Kata Tidak Tersedia

Nama Kasus Uji	Cek Kata Tidak Tersedia
Tujuan Pengujian	Untuk menguji validitas kinerja dari aplikasi dalam menyediakan fasilitas cek kata yang tidak tersedia dalam kamus agar <i>user</i> dapat mengetahui apakah kata yang dimasukkan sudah sesuai dengan daftar kata dalam kamus
Prosedur Uji	<ol style="list-style-type: none"> 1. Menjalankan aplikasi kamus 2. <i>User</i> mengetikkan kata bahasa Inggris yang ingin diproses pada <i>textfield</i> yang telah disediakan 3. <i>User</i> menekan tombol proses
Hasil yang Diharapkan	Aplikasi dapat menampilkan daftar rujukan kata bahasa Inggris lain atau <i>suggested word</i> kata lain yang tersedia dalam kamus

Sumber: [Pengujian dan Analisis]

Tabel 6.6 Test Case untuk Pengujian Validasi Bantuan Cara Pengucapan


Nama Kasus Uji	Bantuan Cara Pengucapan
Tujuan Pengujian	Untuk menguji validitas kinerja dari aplikasi dalam menyediakan fasilitas bantuan cara pembacaan alfabet fonetis agar <i>user</i> dapat mengetahui cara pembacaan alfabet fonetis dari setiap kata bahasa Inggris
Prosedur Uji	1. Menjalankan aplikasi kamus 2. <i>User</i> menekan tombol bantuan
Hasil yang Diharapkan	Aplikasi dapat menampilkan cara pembacaan alfabet fonetis sesuai atura IPA (International Phonetic Alphabet)



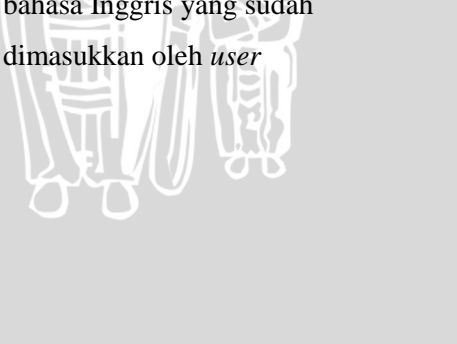
Sumber: [Pengujian dan Analisis]

6.1.1.1 Hasil Pengujian Validasi


Dari kasus uji yang telah dilaksanakan pada sub pokok bahasan 6.1.1, maka didapatkan hasil seperti yang ditunjukkan pada tabel 6.1.1.1.

Tabel 6.7 Hasil Pengujian Validasi Aplikasi

No	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	<i>Input</i> kata bahasa Inggris	Aplikasi dapat menampilkan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i>	Aplikasi dapat menampilkan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i> 	Valid
2	Arti kata bahasa Inggris	Aplikasi dapat menampilkan arti kata bahasa	Aplikasi dapat menampilkan arti kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i>	Valid

		<p>Inggris yang sudah dimasukkan oleh <i>user</i></p>		
3	<p>Pengucapan kata bahasa Inggris</p>	<p>Aplikasi dapat menampilkan cara pengucapan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i></p>	<p>Aplikasi dapat menampilkan cara pengucapan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i></p> 	<p>Valid</p>
4	<p>Kata bahasa Inggris yang mirip pengucapan</p>	<p>Aplikasi dapat menampilkan kata bahasa Inggris lain yang memiliki kemiripan pengucapan dengan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i></p>	<p>Aplikasi dapat menampilkan kata bahasa Inggris lain yang memiliki kemiripan pengucapan dengan kata bahasa Inggris yang sudah dimasukkan oleh <i>user</i></p> 	<p>Valid</p>

				
5	Cek kata tidak tersedia	Aplikasi dapat menampilkan daftar rujukan kata bahasa Inggris lain atau <i>suggested word</i> kata lain yang tersedia dalam kamus	<p>Aplikasi dapat menampilkan daftar rujukan kata bahasa Inggris lain atau <i>suggested word</i> kata lain yang tersedia dalam kamus</p> 	Valid
6	Bantuan cara pengucapan	Aplikasi dapat menampilkan cara pembacaan alfabet fonetis sesuai atura IPA	<p>Aplikasi dapat menampilkan cara pembacaan alfabet fonetis sesuai atura IPA (International Phonetic Alphabet)</p>	Valid

		(International Phonetic Alphabet)	
--	--	-----------------------------------	--

Sumber: [Pengujian dan Analisis]

6.1.2 Pengujian Akurasi

Pengujian akurasi aplikasi kamus Inggris-Indonesia menggunakan algoritma *caverphone 2.0* dilakukan dengan mengacu kepada tabel klasifikasi konsonan bahasa Inggris. Prosedur pengujiannya adalah memasukkan kata bahasa Inggris ke dalam aplikasi kemudian dicocokkan dengan tabel klasifikasi konsonan bahasa Inggris.

Pengujian dilakukan untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada awal kata, tengah kata, dan akhir kata. Untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris dilakukan pengujian sebanyak 10 kali. Daftar kata bahasa Inggris yang digunakan dalam pengujian akurasi adalah sebagai berikut:

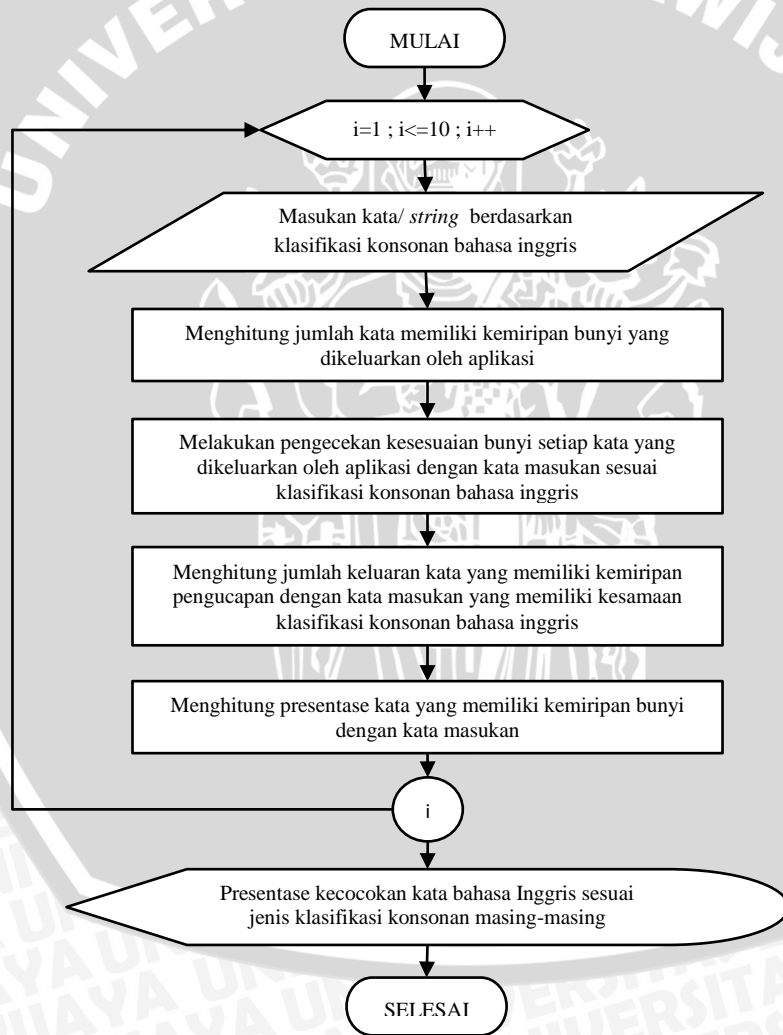
- Daftar kata untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada awal kata dapat dilihat pada **tabel 1** halaman **lampiran 1**.
- Daftar kata untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada tengah kata dapat dilihat pada **tabel 2** halaman **lampiran 1**.
- Daftar kata untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada akhir kata dapat dilihat pada **tabel 3** halaman **lampiran 1**.



Penghitungan Presentase kecocokan kemiripan kata dihitung dengan perhitungan sebagai berikut:

$$\text{Presentase kecocokan} = \frac{\text{jumlah keluaran kata}}{\text{jumlah kata yang mirip pengucapan}} \times 100\%$$

Jumlah keluaran kata adalah banyaknya kata bahasa Inggris yang memiliki kode fonetis yang sama dengan kode fonetis kata masukan. Sedangkan jumlah kata yang mirip pengucapan adalah banyaknya kata bahasa Inggris hasil keluaran kata yang memiliki kecocokan konsonan bahasa Inggris dengan golongan klasifikasi konsonan bahasa Inggrisnya. Diagram alir pengujian akurasi dapat dilihat pada gambar 6.1.



Gambar 6.1 Diagram Alir Pengujian Akurasi
Sumber: [Pengujian dan Analisis]



Hasil pengujian akurasi untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris adalah sebagai berikut:

- Hasil pengujian akurasi untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada awal kata dapat dilihat pada **tabel 1** halaman **lampiran 2**.
- Hasil pengujian akurasi untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada tengah kata dapat dilihat pada **tabel 1** halaman **lampiran 3**.
- Hasil pengujian akurasi untuk tiap-tiap jenis klasifikasi konsonan bahasa Inggris pada akhir kata dapat dilihat pada **tabel 1** halaman **lampiran 4**.

Hasil pengujian yang didapatkan pada setelah melakukan pengujian akurasi dapat disimpulkan pada tabel 6.8.

Tabel 6.8 Hasil Pengujian Akurasi Aplikasi

Klasifikasi konsonan	Awal	Tengah	Akhir
Plosive bilabial	96.42 %	62.55 %	92.27 %
Plosive alveolar	83.90 %	80.22 %	77.76 %
Plosive velar	44.22 %	56.22 %	67.67 %
Affricate palato-alveolar	72.88 %	36.81 %	74.02 %
Nasal bilabial	88.19 %	54.61 %	22.63 %
Nasal alveolar	86.95 %	79.11 %	100 %
Lateral alveolar	98.89 %	51.05 %	35.63 %
Fricative labio-dental	92.13 %	63.31 %	98.26 %
Fricative dental	28.36 %	30.96 %	18.80 %
Fricative alveolar	69.81 %	58.58 %	79.32 %
Fricative palato-alveolar	28.69 %	34.21 %	35.08 %
Fricative glottal	46.28 %	16.44 %	53.76 %
Semi-vowel bilabial	100 %	91.07 %	97.95 %
Semi-vowel palatal	37.63 %	33.90 %	61.58 %

Sumber: [Pengujian dan Analisis]

6.2 Analisis

Berdasarkan hasil pengujian akurasi di atas, dapat disimpulkan beberapa hal sebagai berikut:

1. Implementasi algoritma *Caverphone 2.0* pada aplikasi kamus Inggris-Indonesia dapat digunakan untuk menemukan kata bahasa Inggris lain yang memiliki kemiripan pengucapan.
2. Akurasi paling rendah di awal kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 28,36% untuk jenis klasifikasi konsonan *fricative dental* (konsonan th). Hal ini disebabkan karena konsonan th tidak diproses selama perubahan kata menjadi kode fonetis sehingga hasil kode disamakan dengan konsonan t.
3. Akurasi paling tinggi di awal kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 100% untuk jenis klasifikasi konsonan *semi-vowel bilabial* (konsonan m). Hal ini disebabkan karena konsonan m hanya memiliki kemiripan bunyi dengan konsonan m itu sendiri.
4. Akurasi paling rendah di tengah kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 16,44% untuk jenis klasifikasi konsonan *fricative glottal* (konsonan h). Hal ini disebabkan karena konsonan h selain pada karakter awal dihilangkan sehingga dianggap tidak memiliki bunyi.
5. Akurasi paling tinggi di tengah kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 91,07% untuk jenis klasifikasi konsonan *semi-vowel bilabial* (konsonan m). Hal ini disebabkan karena konsonan m hanya memiliki kemiripan bunyi dengan konsonan m itu sendiri.
6. Akurasi paling rendah di akhir kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 18,80% untuk jenis klasifikasi konsonan *fricative dental* (konsonan th). Hal ini disebabkan karena konsonan th tidak diproses selama perubahan kata menjadi kode fonetis sehingga hasil kode disamakan dengan konsonan t.
7. Akurasi paling tinggi di akhir kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 100% untuk jenis klasifikasi konsonan *nassal alveolar*

(konsonan n). Hal ini disebabkan karena konsonan n pada akhir kata hanya memiliki kemiripan bunyi dengan konsonan n itu sendiri.

Pada proses pengujian juga masih ditemukan beberapa kesalahan dan kelemahan sebagai berikut:

1. Terdapat 5 klasifikasi konsonan bahasa Inggris yang memiliki akurasi kurang dari 50 % pada awal kata yaitu *plossive velar*, *fricative dental*, *fricative palato-alveolar*, *fricative glottal*, dan *semi-vowel palatal*.
2. Terdapat 5 klasifikasi konsonan bahasa Inggris yang memiliki akurasi kurang dari 50% pada tengah kata yaitu *affricate palato-alveolar*, *fricative dental*, *fricative palato-alveolar*, *fricative glottal*, dan *semi-vowel palatal*.
3. Terdapat 4 klasifikasi konsonan bahasa Inggris yang memiliki akurasi kurang dari 50% pada akhir kata yaitu *nassal bilabial*, *lateral alveolar*, *fricative dental*, dan *fricative palato-alveolar*.



BAB VII

PENUTUP

Bab ini akan membahas mengenai kesimpulan dan saran yang dapat diambil dari pembuatan aplikasi kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0*.

1.1 Kesimpulan

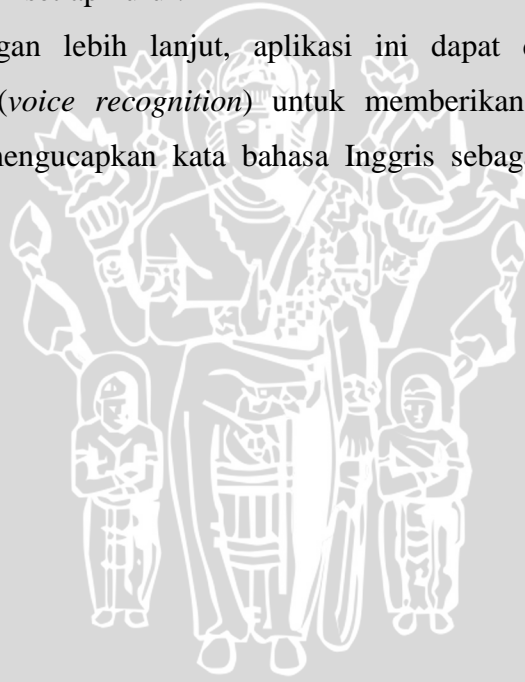
Berdasarkan penelitian yang dilakukan dalam pembuatan aplikasi kamus Inggris-Indonesia menggunakan algoritma *Caverphone 2.0* dapat ditarik beberapa kesimpulan sebagai berikut:

1. Implementasi algoritma *Caverphone 2.0* pada aplikasi kamus Inggris-Indonesia dapat digunakan untuk menemukan kata bahasa Inggris lain yang memiliki kemiripan pengucapan.
2. Akurasi paling tinggi di awal kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 100% untuk jenis klasifikasi konsonan *semi-vowel bilabial* (konsonan m). Hal ini disebabkan karena konsonan m hanya memiliki kemiripan bunyi dengan konsonan m itu sendiri.
3. Akurasi paling tinggi di tengah kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 91,07% untuk jenis klasifikasi konsonan *semi-vowel bilabial* (konsonan m). Hal ini disebabkan karena konsonan m hanya memiliki kemiripan bunyi dengan konsonan m itu sendiri.
4. Akurasi paling tinggi di akhir kata pada pengujian algoritma *Caverphone 2.0* didapatkan sebesar 100% untuk jenis klasifikasi konsonan *nassal alveolar* (konsonan n). Hal ini disebabkan karena konsonan n pada akhir kata hanya memiliki kemiripan bunyi dengan konsonan n itu sendiri.

1.2 Saran

Berdasarkan penelitian yang dilakukan dalam implementasi algoritma *Caverphone 2.0* pada aplikasi kamus Inggris-Indonesia dapat diberikan beberapa saran sebagai berikut:

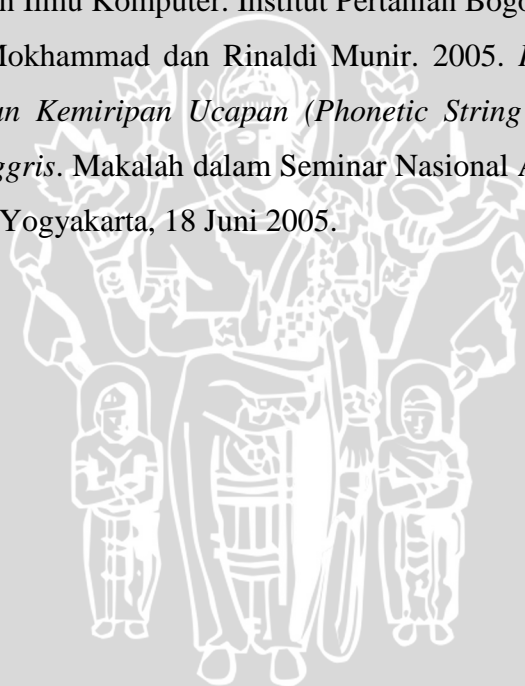
1. Untuk pengembangan lebih lanjut, proses perubahan kata menjadi kode fonetis pada algoritma *Caverphone 2.0* dapat lebih ditingkatkan khususnya pada jenis klasifikasi konsonan *plossive velar*, *affricate palato-alveolar*, *nassal bilabial*, *lateral alveolar*, *fricative dental*, *fricative palato-alveolar*, *fricative glottal*, dan *semi-vowel palatal*. Namun, jenis klasifikasi konsonan yang harus diberikan perhatian khusus adalah *fricative dental* dan *fricative palato-alveolar* dikarenakan 2 jenis klasifikasi konsonan ini memiliki akurasi kurang dari 50% pada awal, tengah, dan akhir kata.
2. Untuk pengembangan lebih lanjut, proses perubahan kata menjadi kode fonetis pada algoritma *Caverphone2.0* harus dapat memperhitungkan pula hubungan antar huruf dalam suatu kata, sehingga dapat ditentukan dengan jelas perubahan bunyi dari setiap huruf.
3. Untuk pengembangan lebih lanjut, aplikasi ini dapat ditambahkan fitur pengenalan suara (*voice recognition*) untuk memberikan kemudahan bagi pengguna dalam mengucapkan kata bahasa Inggris sebagai masukan untuk aplikasi.



DAFTAR PUSTAKA

- [ASH-12] Ashary, Fadhly. 2012. *Pengertian UML (Unified Modeling Language)*. <http://fadhlyashary.blogspot.com/2012/06/pengertian-uml-unified-modeling.html>. (diakses 9 September 2012).
- [HAM-12] Hamka, Dimas, Muhammad Sobri, dan Syahril Rizal. 2012. *Aplikasi Kamus Inggris Indonesia Indonesia Inggris pada Platform Android*. Universitas Bina Darma. Palembang.
- [HOO-04] Hood, David. 2004. *Caverphone Revisited*. Caversham Project Occasional Technical Paper.
- [JAT-11] Jatiblack. 2011. *Membuat Kamus Inggris-Indonesia dengan PHP*. <http://jatiblack.com/membuat-kamus-inggris-indonesia-dengan-php>. (diakses 9 September 2012).
- [KAD-09] Kadir, Abdul. 2009. *Dasar Pemrograman JAVA™ 2*. Andi: Yogyakarta.
- [MAD-09] Madaharsa Bernardino Dito Adiwidya. 2009. *Algoritma Levenshtein dalam Pendekatan Approximate String Matching*. Makalah IF3051 Strategi Algoritma.
- [MAN-09] Manning, Christopher D., Prabhakar Raghavan, dan Hinrich Schütze. 2009. *Introduction to Information Retrieval*. Cambridge University Press: Cambridge, England.
- [MUN-05] Munawar. 2005. *Pemodelan Visual dengan UML*. Graha Ilmu: Yogyakarta.
- [NOO-10] Noor, Mohamad Rizal. 2010. *Ortografi Bahasa Inggris dan Transkripsi Fonetik*. <http://mnrizal.wordpress.com/2010/06/16/ortografi-bahasa-inggris-dan-transkripsi-fonetik/>. (diakses 19 September 2012).
- [PAC-10] Pacidda, Masdin. 2010. *Pengantar English Pronunciation*. <http://letspeakenglish.info/2010/12/09/pengantar-english-pronunciation>. (diakses 19 september 2012).

- [PRI-97] Primasari, Dewi. 1997. *Metode Pencarian dan Temu-Kembali Nama Berdasarkan Kesamaan Fonetik*. Skripsi. Bogor: Institut Pertanian Bogor.
- [RAH-09] Raharjo, Budiman, Imam Heryanto, dan Arif Haryono. 2009. *Mudah Belajar JAVA*. Informatika: Bandung.
- [RIZ-10] Rizky, Fatardhi Andhika. 2010. *Penerapan String Suggestion dengan Algoritma Levenshtein Distance dan Alternatif Algoritma lain dalam Aplikasi*. Makalah IF3051 Strategi Algoritma – Sem. I. Bandung, 6 Desember 2010.
- [SUT-09] Sutisna, Utis dan Julio Adisantoso. 2009. *Koreksi Ejaan Query Bahasa Indonesia menggunakan Algoritme Damerau Levenshtein*. Departemen Ilmu Komputer. Institut Pertanian Bogor.
- [SYA-05] Syaroni, Mokhammad dan Rinaldi Munir. 2005. *Pencocokan String berdasarkan Kemiripan Ucapan (Phonetic String Matching) dalam Bahasa Inggris*. Makalah dalam Seminar Nasional Aplikasi Teknologi Informasi. Yogyakarta, 18 Juni 2005.



Lampiran 1. Daftar Kata yang Digunakan pada Pengujian Akurasi

Tabel 1 Daftar Kata untuk Pengujian Akurasi pada Awal Kata

Klasifikasi Konsonan	Kata yang digunakan dalam Pengujian Akurasi									
<i>Plossive bilabial</i>	Bad	Babe	Begin	Parody	Pellet	Benefit	Bleach	Page	Plant	Passage
<i>Plossive alveolar</i>	Table	Tackle	Tamper	Tear	Tendon	Damag e	Dealer	Degree	Destroy	Devil
<i>Plossive velar</i>	Kernel	Kid	Kepeer	Kimono	Killer	Gallery	Gallop	Gatewa y	Genetic	Gummy
<i>Africate palato-alveolar</i>	Cabbag e	Campu s	Cute	Custome r	Curious	Jacket	Jealous	Jingle	Job	Journal
<i>Nassal bilabial</i>	Wagon	Walk	Warehous e	Water	Wife	Wild	Winter	Women	Witness	Wonder
<i>Nassal alveolar</i>	Name	Nation	Near	Neutral	Nibble	Nice	Noise	None	Nut	Number
<i>Lateral alveolar</i>	Labial	Lady	Leaf	Lecture	Liberal	Limit	Load	Logic	Lucky	Lumber
<i>Fricative labio-dental</i>	Face	Feature	Fiber	Food	Furnitur e	Vacatio n	Vector	Violin	Vocal	Vulva
<i>Fricative dental</i>	Thankful	Theate r	Theme	Theolog y	Therapy	Thick	Thieve	Think	Thousan d	Thumbn a il
<i>Fricative alveolar</i>	Safari	Seal	Sick	Soldier	Zebra	Zombie	Zipper	Rabbit	Reactor	Ribbon
<i>Fricative palato-alveolar</i>	Shuttle	Shuffle	Showroom	Shower	Shop	Shirt	Shipme nt	Sheaf	Shampo o	Shame
<i>Fricative glottal</i>	Hack	Hair	Head	Heaven	Hibernat e	Hidden	Hobby	Honey	Human	Humor
<i>Semi-vowel bilabial</i>	Machin e	Magne t	Mesure	Medal	Midnigh t	Mileage	Mobile	Molecul e	Mumble	Muscle
<i>Semi-vowel palatal</i>	Yard	Yawn	Year	Yellow	Yield	Yogurt	Yoga	Young	Yummy	Yonder

Sumber: [Pengujian dan Analisis]

Tabel 2 Daftar Kata untuk Pengujian Akurasi pada Tengah Kata

Klasifikasi Konsonan	Kata yang digunakan dalam Pengujian Akurasi									
<i>Plossive bilabial</i>	Abode	Cabinet	Debate	Fibula	Labial	Amputate	Bipolar	Capital	Impale	Repair
<i>Plossive alveolar</i>	Batiste	Catnip	Detail	Fetal	Gateway	Intake	Bedtime	Codify	Deduce	Hideaway
<i>Plossive velar</i>	Bikini	Fakir	Joker	Naked	Poker	Begin	Digital	Jaguar	Magazine	Negate
<i>Africate palato-alveolar</i>	Ancient	Ascent	Bacon	Cactus	Decade	Pyjamas	Objector	Injury	Injunction	Dejection
<i>Nassal bilabial</i>	Beware	Coward	Dowel	Lawyer	Sawdust	Reward	Vowel	Jewry	Lawful	Rewrite
<i>Nassal alveolar</i>	Bandit	Canary	Dancer	Donkey	Fanatic	Funeral	Gender	Junior	Lineal	Manage
<i>Lateral alveolar</i>	Bolero	Caliber	Calorie	Delay	Delete	Filet	Folder	Galaxy	Helmet	Malady
<i>Fricative labio-dental</i>	Befit	Cafe	Deface	Default	Infection	Advise	Bevel	Cavalry	Device	Envelop
<i>Fricative dental</i>	Without	Ruthless	Python	Pathology	Orthopedic	Mother	Method	Lethal	Catholic	Bathing
<i>Fricative alveolar</i>	Absent	Basic	Casket	Destroy	Cozy	Daze	Gizmo	Virus	Terminal	Survivor
<i>Fricative palato-alveolar</i>	Cashew	Fashion	Cushion	Bishop	Masher	Pushcart	Pusher	Washable	Cashbox	Dashboard
<i>Fricative glottal</i>	Behalf	Behind	Coherent	Cohesion	Dahlia	Enhance	Exhaust	Fahrenheit	Mahogany	Rehearsal
<i>Semi-vowel bilabial</i>	Admiral	Bambo	Cemetery	Comedy	Damage	Family	Jumper	Lemonade	Member	Memory
<i>Semi-vowel palatal</i>	Anything	Anywa	Bayonet	Boyfriend	Ceylon	Crystal	Daylong	Dryness	Joyful	Joyless

Sumber: [Pengujian dan Analisis]

Tabel 3 Daftar Kata untuk Pengujian Akurasi pada Akhir Kata

Klasifikasi Konsonan	Kata yang digunakan dalam Pengujian Akurasi									
<i>Plosive bilabial</i>	Blob	Cab	Dub	Grab	Pedicab	Bishop	Clap	Drop	Flap	Heap
<i>Plosive alveolar</i>	Abort	Burnout	Cabinet	Debit	Edit	Field	Gold	Hard	Invalid	Keyboard
<i>Plosive velar</i>	Blank	Datebook	Feedback	Guidebook	Jerk	Analog	Belong	Catalog	Dialog	Firebug
<i>Africate palato-alveolar</i>	Aesthetic	Ballistic	Caloric	Demonic	Epidemic	Erotic	Fanatic	Garlic	Genetic	Heroic
<i>Nassal bilabial</i>	Bungalow	Cashew	Elbow	Eyebrow	Foresaw	Foreshadow	Interview	Meadow	Mellow	Narrow
<i>Nassal alveolar</i>	Ablution	Abortion	Balloon	Calculation	Deception	Erosion	Faction	Garden	Harpoon	Illumination
<i>Lateral alveolar</i>	Actual	Bacterial	Cannibal	Digital	Editorial	Factual	General	Historical	Immortal	Journal
<i>Fricative labio-dental</i>	Airproof	Brushoff	Cutoff	Debrief	Fluff	Slav	Golf	Herself	Kickoff	:Layoff
<i>Fricative dental</i>	Azimuth	Behemoth	Zenith	Youth	Width	Vermouth	Untruth	Undergrowth	Tolbooth	Thousandth
<i>Fricative alveolar</i>	Anonymous	Bacillus	Campus	Devious	Topaz	Waltz	Quartz	Actor	Bachelor	Calendar
<i>Fricative palato-alveolar</i>	Abolish	Backlash	Childish	Demolish	Diminish	English	Finish	Garnish	Hairbrush	Nourish
<i>Fricative glottal</i>	Ambush	Approach	Autograph	Banish	Behemoth	Clutch	Codfish	Danish	Dispatch	Epitaph
<i>Semi-vowel bilabial</i>	Acclaim	Ballroom	Calm	Decorum	Egoism	Film	Gloom	Harm	Idiom	Kilogram
<i>Semi-vowel palatal</i>	Abbey	Bakery	Canary	Daily	Every	Fairy	Glory	Harmony	Immunity	Jersey

Sumber: [Pengujian dan Analisis]

Lampiran 2. Hasil Pengujian Akurasi pada Awal Kata

Tabel 1 Pengujian Akurasi Konsonan Plosive Bilabial pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive bilabial	Bad	Babe	Begin	Parody	Pellet	Benefit	Bleach	Page	Plant	Passage	
Jumlah keluaran kata	62	19	14	15	36	2	16	63	13	7	
Jumlah kata mirip	60	18	14	13	36	2	15	63	12	7	
Presentase	96.77	94.74	100	86.67	100	100	93.75	100	92.31	100	96.42

Sumber: [Pengujian dan Analisis]

Tabel 2 Pengujian Akurasi Konsonan Plosive Alveolar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive alveolar	Table	Tackle	Tamper	Tear	Tendon	Damage	Dealer	Degree	Destroy	Devil	
Jumlah keluaran kata	25	30	13	75	5	4	19	7	3	14	
Jumlah kata mirip	24	27	12	52	5	2	16	4	3	14	
Presentase	96	90	92.31	69.33	100	50	84.21	57.14	100	100	83.90

Sumber: [Pengujian dan Analisis]

Tabel 3 Pengujian Akurasi Konsonan Plosive Velar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive velar	Kernel	Kid	Keper	Kimono	Killer	Gallery	Gallop	Gateway	Genetic	Gummy	
Jumlah keluaran kata	32	74	25	7	32	5	9	2	3	12	
Jumlah kata mirip	13	26	7	4	15	2	4	1	2	4	
Presentase	40.63	35.14	28	57.14	46.88	40	44.44	50	66.67	33.33	44.22

Sumber: [Pengujian dan Analisis]

Tabel 4 Pengujian Akurasi Konsonan Affricate Palato-Alveolar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Affricate palato-alveolar	Cabbage	Campus	Cute	Customer	Curious	Jacket	Jealous	Jingle	Job	Journal	
Jumlah keluaran kata	5	3	74	2	26	5	2	7	7	5	
Jumlah kata mirip	3	3	24	2	5	3	2	6	5	5	
Presentase	60	100	32.43	100	19.23	60	100	85.71	71.43	100	72.88

Sumber: [Pengujian dan Analisis]

Tabel 5 Pengujian Akurasi Konsonan Nassal Bilabial pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Nassal bilabial	Wagon	Walk	Warehouse	Water	Wife	Wild	Winter	Women	Witness	Wonder	
Jumlah keluaran kata	3	16	5	24	13	33	5	2	7	5	
Jumlah kata mirip	3	14	3	19	11	28	5	2	6	5	
Presentase	100	87.50	60	79.17	84.62	84.85	100	100	85.71	100	88.19

Sumber: [Pengujian dan Analisis]

Tabel 6 Pengujian Akurasi Konsonan Nassal Alveolar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Nassal alveolar	Name	Nation	Near	Neutral	Nibble	Nice	Noise	None	Nut	Number	
Jumlah keluaran kata	4	3	15	3	7	14	8	7	16	2	
Jumlah kata mirip	3	3	12	3	7	11	5	6	14	2	
Presentase	75	100	80	100	100	78.57	62.50	85.71	87.50	100	86.95

Sumber: [Pengujian dan Analisis]

Tabel 7 Pengujian Akurasi Konsonan Lateral Alveolar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Lateral alveolar	Labial	Lady	Leaf	Lecture	Liberal	Limit	Load	Logic	Lucky	Lumber	
Jumlah keluaran kata	8	20	9	3	3	2	28	3	22	4	
Jumlah kata mirip	8	20	8	3	3	2	28	3	22	4	
Presentase	100	100	88.89	100	100	100	100	100	100	100	98.89

Sumber: [Pengujian dan Analisis]

Tabel 8 Pengujian Akurasi Konsonan Fricative Labio-Dental pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative labio-dental	Face	Feature	Fiber	Food	Furniture	Vacation	Vector	Violin	Vocal	Vulva	
Jumlah keluaran kata	26	42	13	38	11	7	7	7	26	8	
Jumlah kata mirip	25	37	11	35	10	7	7	6	25	7	
Presentase	96.15	88.10	84.62	92.11	90.91	100	100	85.71	96.15	87.50	92.13

Sumber: [Pengujian dan Analisis]

Tabel 9 Pengujian Akurasi Konsonan Fricative Dental pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative dental	Thankful	Theater	Theme	Theology	Therapy	Thick	Thieve	Think	Thousand	Thumbnail	
Jumlah keluaran kata	2	46	23	2	13	43	12	18	7	3	
Jumlah kata mirip	1	4	4	1	2	3	2	5	2	1	
Presentase	50	8.70	17.39	50	15.38	35.78	16.67	27.78	28.57	33.33	28.36

Sumber: [Pengujian dan Analisis]

Tabel 10 Pengujian Akurasi Konsonan Fricative Alveolar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative alveolar	Safari	Seal	Sick	Soldier	Zebra	Zombi e	Zipper	Rabb it	Reac tor	Ribbo n	
Jumlah keluaran kata	6	48	34	52	12	8	31	17	7	5	
Jumlah kata mirip	6	30	26	18	2	6	12	16	7	5	
Presentase	100	62.50	76.47	34.62	16.67	75	38.71	94.1	100	100	69.81

Sumber: [Pengujian dan Analisis]

Tabel 11 Pengujian Akurasi Konsonan Fricative Palato-Alveolar pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative palato-alveolar	Shuutl e	Shufl e	Showr oom	Showe r	Shop	Shirt	Shipme nt	Shea f	Sha mpo o	Sham e	
Jumlah keluaran kata	52	30	2	11	17	52	2	16	8	8	
Jumlah kata mirip	11	5	1	2	5	17	1	5	1	2	
Presentase	21.15	16.67	50	18.18	29.41	32.69	50	31.2 5	12.5 0	25	28.69

Sumber: [Pengujian dan Analisis]

Tabel 12 Pengujian Akurasi Konsonan Fricative Glottal pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative glotal	Hack	Hair	Head	Heave n	Hibern ate	Hidden	Hobby	Hone y	Hum an	Hum or	
Jumlah keluaran kata	46	56	44	5	6	16	23	20	9	10	
Jumlah kata mirip	19	32	27	3	2	5	11	5	5	5	
Presentase	41.30	57.14	61.36	60	33.33	31.25	47.83	25	55.5 6	50	46.28

Sumber: [Pengujian dan Analisis]

Tabel 13 Pengujian Akurasi Konsonan Semi-Vowel Bilabial pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Semi-vowel bilabial	Machine	Magnet	Mesure	Medal	Midnight	Mileage	Mobile	Molecule	Mumble	Muscle	
Jumlah keluaran kata	2	3	19	37	2	2	4	2	2	5	
Jumlah kata mirip	2	3	19	37	2	2	4	2	2	5	
Presentase	100	100	100	100	100	100	100	100	100	100	100

Sumber: [Pengujian dan Analisis]

Tabel 14 Pengujian Akurasi Konsonan Semi-Vowel Palatal pada Awal Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Semi-vowel palatal	Yard	Yawn	Year	Yellow	Yield	Yogurt	Yoga	Young	Yummy	Yonder	
Jumlah keluaran kata	11	10	22	7	11	5	10	4	2	4	
Jumlah kata mirip	4	4	11	2	4	2	2	2	1	1	
Presentase	36.36	40	50	28.57	36.36	40	20	50	50	25	37.63

Sumber: [Pengujian dan Analisis]

Lampiran 3. Hasil Pengujian Akurasi pada Tengah Kata

Tabel 1 Pengujian Akurasi Konsonan Plosive Bilabial pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive bilabial	Abode	Cabinet	Debate	Fibula	Labial	Amputate	Bipolar	Capital	Impale	Repair	
Jumlah keluaran kata	16	3	20	2	8	3	4	7	18	16	
Jumlah kata mirip	14	2	10	1	6	2	2	3	10	13	
Presentase	87.5	66.67	50	50	75	66.67	50	42.86	55.56	81.25	62.55

Sumber: [Pengujian dan Analisis]

Tabel 2 Pengujian Akurasi Konsonan Plosive Alveolar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive alveolar	Batiste	Catnip	Detail	Fetal	Gateway	Intake	Bedtime	Codify	Deduce	Hideaway	
Jumlah keluaran kata	7	2	46	42	2	7	3	2	5	3	
Jumlah kata mirip	5	2	26	20	2	7	3	2	3	2	
Presentase	71.43	100	56.52	47.62	100	100	100	100	60	66.67	80.22

Sumber: [Pengujian dan Analisis]

Tabel 3 Pengujian Akurasi Konsonan Plosive Velar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive velar	Bikini	Fakir	Joker	Naked	Poker	Begin	Digital	Jaguar	Magazine	Negative	
Jumlah keluaran kata	5	26	10	4	40	14	8	10	3	4	
Jumlah kata mirip	3	11	7	3	12	5	3	7	2	3	
Presentase	60	42.31	70	75	30	35.71	37.5	70	66.67	75	56.22

Sumber: [Pengujian dan Analisis]

Tabel 4 Pengujian Akurasi Konsonan Affricate Palato-Alveolar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Affricate palato-alveolar	Ancient	Ascend	Bacon	Cactus	Decade	Pyjamas	Objector	Injury	Injunction	Dejection	
Jumlah keluaran kata	4	5	14	2	19	3	5	4	6	4	
Jumlah kata mirip	1	2	5	1	9	2	1	1	2	1	
Presentase	25	40	35.71	50	47.37	66.67	20	25	33.33	25	36.81

Sumber: [Pengujian dan Analisis]

Tabel 5 Pengujian Akurasi Konsonan Nasal Bilabial pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Nasal bilabial	Beware	Coward	Dowel	Lawyer	Sawdust	Reward	Vowel	Jewelry	Lawful	Rewrite	
Jumlah keluaran kata	8	2	11	4	3	3	3	4	14	4	
Jumlah kata mirip	8	2	3	4	1	2	2	1	1	2	
Presentase	100	100	27.27	100	33.33	66.67	66.67	25	7.14	50	54.61

Sumber: [Pengujian dan Analisis]

Tabel 6 Pengujian Akurasi Konsonan Nasal Alveolar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Nasal alveolar	Bandit	Canary	Dancer	Donkey	Fanatic	Funeral	Gender	Junior	Lineal	Manage	
Jumlah keluaran kata	4	6	5	15	4	4	20	5	8	9	
Jumlah kata mirip	4	5	5	14	3	4	16	2	6	4	
Presentase	100	83.33	100	93.33	75	100	80	40	75	44.44	79.11

Sumber: [Pengujian dan Analisis]

Tabel 7 Pengujian Akurasi Konsonan Lateral Alveolar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Lateral alveolar	Bolero	Caliber	Calorie	Delay	Delete	Filet	Folder	Gala xy	Hel met	Mala dy	
Jumlah keluaran kata	5	8	5	19	8	21	42	9	5	3	
Jumlah kata mirip	2	2	2	15	7	7	8	6	1	3	
Presentase	40	25	40	78.95	87.5	33.33	19.05	66.67	20	100	51.05

Sumber: [Pengujian dan Analisis]

Tabel 8 Pengujian Akurasi Konsonan Fricative Labio-Dental pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative labio-dental	Befit	Cafe	Deface	Default	Infection	Advise	Bevel	Cavalry	Device	Envelope	
Jumlah keluaran kata	9	23	6	19	3	3	16	3	6	2	
Jumlah kata mirip	3	7	5	10	2	2	8	2	5	2	
Presentase	33.33	30.43	83.33	52.63	66.67	66.67	50	66.67	83.33	100	63.31

Sumber: [Pengujian dan Analisis]

Tabel 9 Pengujian Akurasi Konsonan Fricative Dental pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative dental	Without	Ruthless	Python	Pathology	Orthopedic	Mother	Method	Lethal	Catholic	Bathing	
Jumlah keluaran kata	4	2	13	3	2	37	3	20	5	7	
Jumlah kata mirip	2	1	1	1	1	1	2	3	1	1	
Presentase	50	50	7.69	33.33	50	2.70	66.67	15	20	14.29	30.96

Sumber: [Pengujian dan Analisis]

Tabel 10 Pengujian Akurasi Konsonan Fricative Alveolar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative alveolar	Absent	Basic	Casket	Destroy	Cozy	Daze	Gizmo	Virus	Terminal	Survivor	
Jumlah keluaran kata	2	7	3	3	22	22	2	14	4	3	
Jumlah kata mirip	2	5	3	3	3	2	1	7	1	2	
Presentase	100	71.43	100	100	13.64	9.09	50	50	25	66.67	58.58

Sumber: [Pengujian dan Analisis]

Tabel 11 Pengujian Akurasi Konsonan Fricative Palato-Alveolar pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative palato-alveolar	Cashew	Fashion	Cushion	Bishop	Masher	Pushcart	Pusher	Wasable	Cashbox	Dashboard	
Jumlah keluaran kata	22	7	13	2	19	5	26	4	2	6	
Jumlah kata mirip	6	1	1	2	3	1	4	3	1	1	
Presentase	27.27	14.29	7.69	100	15.79	20	15.39	75	50	16.67	34.21

Sumber: [Pengujian dan Analisis]

Tabel 12 Pengujian Akurasi Konsonan Fricative Glottal pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative glotal	Behalf	Behind	Coherent	Cohesion	Dahlia	Enhance	Exhaust	Fahrenheit	Mahogany	Rehearsal	
Jumlah keluaran kata	10	25	12	13	19	7	11	7	2	14	
Jumlah kata mirip	3	1	1	1	1	2	1	1	1	1	
Presentase	30	4	8.33	7.69	5.26	28.57	9.09	14.29	50	7.14	16.44

Sumber: [Pengujian dan Analisis]

Tabel 13 Pengujian Akurasi Konsonan Semi-vowel bilabial pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Semi-vowel bilabial	Admiral	Bambo	Cemetery	Comedy	Damag e	Family	Jumper	Lem onad e	Mem ber	Mem ory	
Jumlah keluaran kata	2	7	2	6	4	4	4	2	2	3	
Jumlah kata mirip	2	6	2	6	2	3	4	2	2	3	
Presentase	100	85.71	100	100	50	75	100	100	100	100	91.07

Sumber: [Pengujian dan Analisis]

Tabel 14 Pengujian Akurasi Konsonan Semi-vowel palatal pada Tengah Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Semi-vowel palatal	Anything	Anyw ay	Bayonet	Boyfri end	Ceylon	Crystal	Daylon g	Dryn ess	Joyf ul	Joyle ss	
Jumlah keluaran kata	3	4	25	3	7	6	3	3	6	2	
Jumlah kata mirip	2	2	2	1	1	1	1	1	2	1	
Presentase	66.67	50	8	33.33	14.29	16.67	33.33	33.33	33.33	50	33.90

Sumber: [Pengujian dan Analisis]

Lampiran 4. Hasil Pengujian Akurasi pada Akhir Kata

Tabel 1 Pengujian Akurasi Konsonan Plosive Bilabial pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive bilabial	Blob	Cab	Dub	Grab	Pedica b	Bishop	Clap	Drop	Flap	Heap	
Jumlah keluaran kata	6	33	19	16	2	2	9	14	6	15	
Jumlah kata mirip	5	27	16	16	2	2	9	14	6	11	
Presentase	83.33	81.81	84.21	100	100	100	100	100	100	73.33	92.27

Sumber: [Pengujian dan Analisis]

Tabel 2 Pengujian Akurasi Konsonan Plosive Alveolar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive alveolar	Abort	Burno ut	Cabin et	Debit	Edit	Field	Gold	Hard	Inval id	Keyb oard	
Jumlah keluaran kata	16	25	3	20	10	38	74	44	2	12	
Jumlah kata mirip	12	24	2	16	8	29	61	35	1	11	
Presentase	75	96	66.67	80	80	76.32	82.43	79.5 5	50	91.67	77.76

Sumber: [Pengujian dan Analisis]

Tabel 3 Pengujian Akurasi Konsonan Plosive Velar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Plosive velar	Blank	Dateb ook	Feedb ack	Guide book	Jerk	Analog	Belong	Catal og	Dial og	Fireb ug	
Jumlah keluaran kata	10	2	2	5	12	6	10	5	4	2	
Jumlah kata mirip	5	2	2	4	7	2	5	4	3	1	
Presentase	50	100	100	80	58.33	33.33	50	80	75	50	67.67

Sumber: [Pengujian dan Analisis]

Tabel 4 Pengujian Akurasi Konsonan Affricate Palato-Alveolar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Affricate palato-alveolar	Aesthetic	Ballistic	Caloric	Demoniac	Epidemic	Erotic	Fanatic	Garlic	Genetic	Heroic	
Jumlah keluaran kata	2	2	2	2	2	4	4	17	3	4	
Jumlah kata mirip	2	2	2	2	2	3	2	4	2	1	
Presentase	100	100	100	100	100	75	50	23.53	66.67	25	74.02

Sumber: [Pengujian dan Analisis]

Tabel 5 Pengujian Akurasi Konsonan Nassal Bilabial pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Nasal bilabial	Bungalow	Cashew	Elbow	Eyebrow	Foresaw	Foreshadow	Interview	Meadow	Mellow	Narrow	
Jumlah keluaran kata	2	22	23	7	5	6	5	37	8	2	
Jumlah kata mirip	1	1	1	3	1	1	1	2	1	1	
Presentase	50	4.55	4.35	42.86	20	16.67	20	5.41	12.5	50	22.63

Sumber: [Pengujian dan Analisis]

Tabel 6 Pengujian Akurasi Konsonan Nassal Alveolar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Nasal alveolar	Ablution	Abortion	Balloon	Calculation	Deception	Erosion	Faction	Garden	Harpoon	Illumination	
Jumlah keluaran kata	3	2	12	2	3	5	7	14	11	2	
Jumlah kata mirip	3	2	12	2	3	5	7	14	11	2	
Presentase	100	100	100	100	100	100	100	100	100	100	100

Sumber: [Pengujian dan Analisis]

Tabel 7 Pengujian Akurasi Konsonan Lateral Alveolar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Lateral alveolar	Actual	Bacterial	Cannibal	Digital	Editorial	Factual	General	Historical	Immortal	Journal	
Jumlah keluaran kata	8	7	4	8	4	7	6	2	8	5	
Jumlah kata mirip	3	3	2	3	1	1	1	2	1	1	
Presentase	37.5	42.86	50	37.5	25	14.29	16.67	100	12.5	20	35.63

Sumber: [Pengujian dan Analisis]

Tabel 8 Pengujian Akurasi Konsonan Fricative Labio-Dental pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative labio-dental	Airproof	Brushoff	Cutoff	Debrief	Fluff	Slav	Golf	Herself	Kickoff	Layoff	
Jumlah keluaran kata	2	3	2	3	2	3	23	2	2	9	
Jumlah kata mirip	2	3	2	3	2	3	19	2	2	9	
Presentase	100	100	100	100	100	100	82.61	100	100	100	98.26

Sumber: [Pengujian dan Analisis]

Tabel 9 Pengujian Akurasi Konsonan Fricative Dental pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative dental	Azimuth	Behemoth	Zenith	Youth	Width	Vermouth	Untruth	Undergrowth	Tolbooth	Thousandth	
Jumlah keluaran kata	6	4	18	11	33	5	9	2	20	7	
Jumlah kata mirip	1	1	1	1	5	1	2	1	2	1	
Presentase	16.67	25	5.56	9.10	15.15	20	22.22	50	10	14.29	18.80

Sumber: [Pengujian dan Analisis]

Tabel 10 Pengujian Akurasi Konsonan Fricative Alveolar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative alveolar	Anon ymous	Bacillus	Campus	Devio us	Topaz	Waltz	Quartz	Acto r	Bach elor	Calen dar	
Jumlah keluaran kata	2	2	3	7	5	2	10	8	5	2	
Jumlah kata mirip	2	2	3	6	5	1	9	3	4	1	
Presentase	100	100	100	85.71	100	50	90	37.5	80	50	79.32

Sumber: [Pengujian dan Analisis]

Tabel 11 Pengujian Akurasi Konsonan Fricative Palato-Alveolar pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative palato-alveolar	Aboli sh	Backl ash	Childi sh	Demol ish	Dimini sh	English	Finish	Garn ish	Hair brus h	Nouri sh	
Jumlah keluaran kata	5	2	10	2	3	3	12	8	3	3	
Jumlah kata mirip	1	1	1	1	1	1	5	1	1	2	
Presentase	20	50	10	50	33.33	33.33	41.67	12.5	33.33	66.67	35.08

Sumber: [Pengujian dan Analisis]

Tabel 12 Pengujian Akurasi Konsonan Fricative Glottal pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Fricative glottal	Ambu sh	Appro ach	Autog raph	Banish	Behem oth	Clutch	Codfis h	Dani sh	Disp atch	Epita ph	
Jumlah keluaran kata	7	2	2	6	4	17	3	9	2	2	
Jumlah kata mirip	2	1	2	3	1	2	3	2	2	1	
Presentase	28.57	50	100	50	25	11.77	100	22.22	100	50	53.76

Sumber: [Pengujian dan Analisis]

Tabel 13 Pengujian Akurasi Konsonan Semi-Vowel Bilabial pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Semi-vowel bilabial	Acclaim	Ballroom	Calm	Decorum	Egoism	Film	Gloom	Harmon	Idiom	Kilogram	
Jumlah keluaran kata	3	8	16	3	3	10	7	13	5	2	
Jumlah kata mirip	3	8	15	3	3	10	6	13	5	2	
Presentase	100	100	93.75	100	100	100	85.71	100	100	100	97.95

Sumber: [Pengujian dan Analisis]

Tabel 14 Pengujian Akurasi Konsonan Semi-Vowel Palatal pada Akhir Kata

Klasifikasi konsonan	Kata 1	Kata 2	Kata 3	Kata 4	Kata 5	Kata 6	Kata 7	Kata 8	Kata 9	Kata 10	Rata-rata
Semi-vowel palatal	Abbe y	Baker y	Canar y	Daily	Every	Fairy	Glory	Harmon y	Immunity	Jerse y	
Jumlah keluaran kata	23	2	6	19	5	27	5	5	3	4	
Jumlah kata mirip	4	2	3	10	4	11	3	2	3	3	
Presentase	17.39	100	50	52.63	80	40.74	60	40	100	75	61.58

Sumber: [Pengujian dan Analisis]

Lampiran 5. *Sourcecode* Tahap Perubahan Kata menjadi Kode Fonetis

```

1 public String caverphone(String kata){
2     this.kata = kata;
3
4     word = word.toLowerCase();
5
6     akhiran = word.substring(word.length()-1,word.length());
7     if (akhiran.equals("e") )
8     {
9         word= word.substring(0, word.length()-1);
10    }
11    if(word.length()>=5)
12    {
13        if (word.substring(0, 5).equals("cough") )
14        {
15            word="cou2f"+word.substring(5, word.length());
16        }
17        if (word.substring(0, 5).equals("rough") )
18        {
19            word="rou2f"+word.substring(5, word.length());
20        }
21    }
22    if (word.substring(0, 5).equals("tough") )
23    {
24        word="tou2f"+word.substring(5, word.length());
25    }
26    }
27    if(word.length()>=6)
28    {
29        if (word.substring(0, 6).equals("enough") )
30        {
31            word="enou2f"+word.substring(6, word.length());
32        }
33    }
34    if(word.length()>=7)
35    {
36        if (word.substring(0, 7).equals("through") &&
37        word.length()>=7 )
38        {
39            word="throu2f"+word.substring(7, word.length());
40        }
41    }
42    if(word.length()>=2)
43    {
44        if (word.substring(0, 2).equals("gn") && word.length()>=2)
45        {
46            word="2n"+word.substring(2, word.length());
47        }
48    }
49    if (word.substring(word.length()-2,
50    word.length()).equals("mb") && word.length()>=2)
51    {
52        word=word.substring(0, word.length()-2)+"m2";
53    }
54    }

```

Gambar 1 Tampilan *sourcecode* Proses Pencarian Kode Fonetis

Sumber: [Implementasi]

```

55     word = word.replaceAll("cq", "2q");
56     word = word.replaceAll("ci", "si");
57     word = word.replaceAll("ce", "se");
58     word = word.replaceAll("cy", "sy");
59     word = word.replaceAll("tch", "2ch");
60     word = word.replaceAll("c", "k");
61     word = word.replaceAll("q", "k");
62     word = word.replaceAll("x", "k");
63     word = word.replaceAll("v", "f");
64     word = word.replaceAll("dg", "2g");
65     word = word.replaceAll("tio", "sio");
66     word = word.replaceAll("tia", "sia");
67     word = word.replaceAll("d", "t");
68     word = word.replaceAll("ph", "fh");
69     word = word.replaceAll("b", "p");
70     word = word.replaceAll("sh", "s2");
71     word = word.replaceAll("z", "s");
72     if (word.substring(0, 1).equals("a") || word.substring(0,
73 1).equals("i") || word.substring(0, 1).equals("u") ||
74 word.substring(0, 1).equals("e") || word.substring(0, 1).equals("0")
75 && word.length()>=2)
76     {
77         word="A"+word.substring(1, word.length());
78     }
79     word = word.replaceAll("a", "3");
80     word = word.replaceAll("i", "3");
81     word = word.replaceAll("u", "3");
82     word = word.replaceAll("e", "3");
83     word = word.replaceAll("o", "3");
84     word = word.replaceAll("j", "y");
85     if (word.substring(0, 2).equals("y3") && word.length()>=2)
86     {
87         word="Y3"+word.substring(2, word.length());
88     }
89     if (word.substring(0, 1).equals("y") && word.length()>=2)
90     {
91         word="A"+word.substring(1, word.length());
92     }
93     word = word.replaceAll("y", "3");
94     word = word.replaceAll("3gh3", "3kh3");
95     word = word.replaceAll("gh", "22");
96     word = word.replaceAll("g", "k");
97     word = word.replaceAll("ss", "S");
98     word = word.replaceAll("s", "S");
99     word = word.replaceAll("tt", "T");
100    word = word.replaceAll("t", "T");
101    word = word.replaceAll("pp", "P");
102    word = word.replaceAll("p", "P");
103    word = word.replaceAll("kk", "K");
104    word = word.replaceAll("k", "K");
105    word = word.replaceAll("ff", "F");
106    word = word.replaceAll("f", "F");
107    word = word.replaceAll("mm", "M");
108    word = word.replaceAll("m", "M");
109    word = word.replaceAll("nn", "N");
110    word = word.replaceAll("n", "N");
111    word = word.replaceAll("w3", "W3");
112    word = word.replaceAll("wh3", "Wh3");

```

Gambar 2 Tampilan *sourcecode* Proses Pencarian Kode Fonetis (lanjutan)

Sumber: [Implementasi]

```

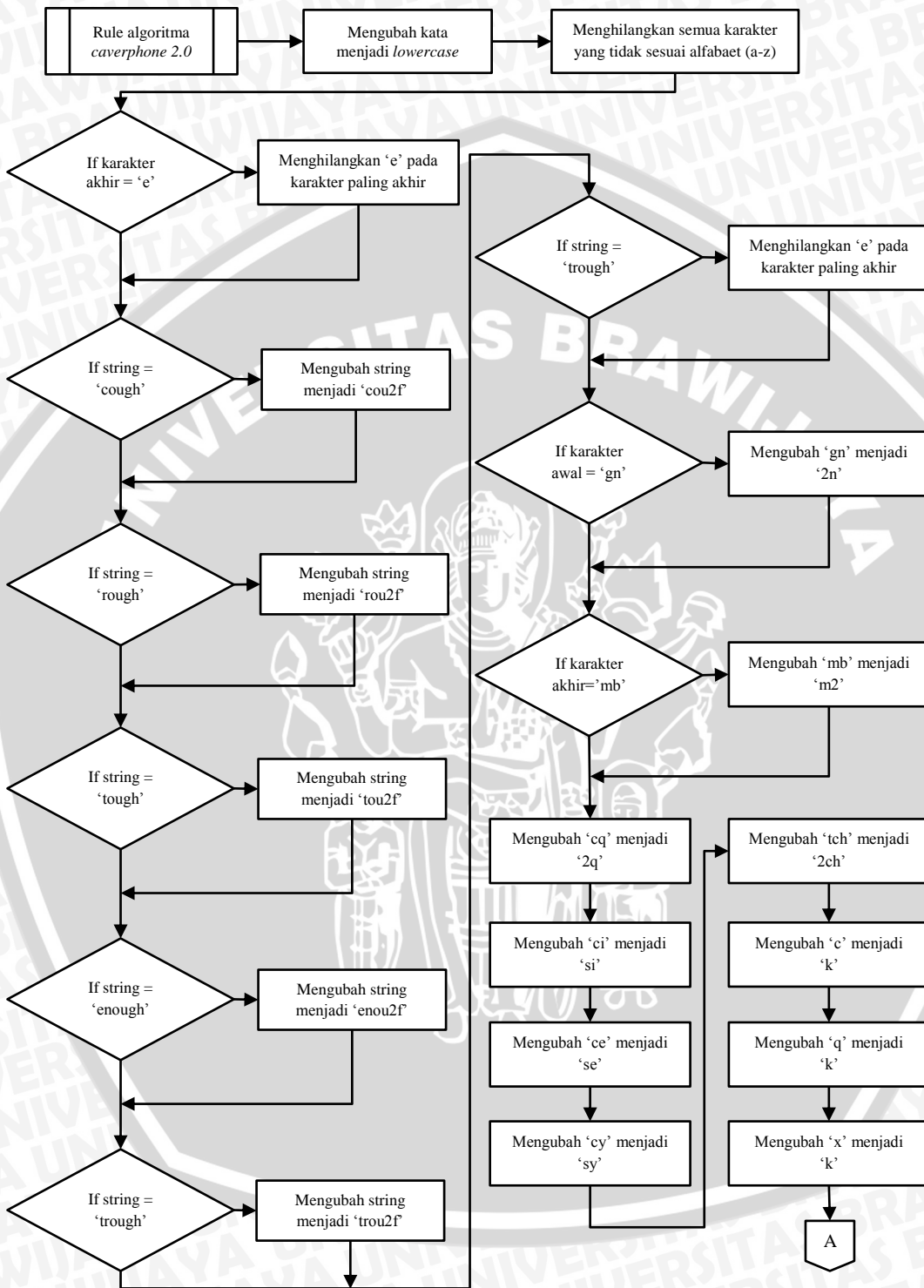
113     if (word.substring(word.length()-1,
114 word.length()).equals("w") && word.length()>=2)
115     {
116         word=word.substring(0, word.length()-1)+"3";
117     }
118
119     word = word.replaceAll("w", "2");
120
121
122     if (word.substring(0, 1).equals("h") && word.length()>=2)
123     {
124         word="A"+word.substring(1, word.length());
125     }
126
127     word = word.replaceAll("h", "2");
128     word = word.replaceAll("r3", "R3");
129
130     if (word.substring(word.length()-1,
131 word.length()).equals("r") && word.length()>=2)
132     {
133         word=word.substring(0, word.length()-1)+"3";
134     }
135
136     word = word.replaceAll("r", "2");
137     word = word.replaceAll("l3", "L3");
138
139
140     if (word.substring(word.length()-1,
141 word.length()).equals("l") && word.length()>=2)
142     {
143         word=word.substring(0, word.length()-1)+"3";
144     }
145
146     word = word.replaceAll("l", "2");
147     word = word.replaceAll("2", "");
148
149     if (word.substring(word.length()-1,
150 word.length()).equals("3") && word.length()>=2)
151     {
152         word=word.substring(0, word.length()-1)+"A";
153     }
154     word = word.replaceAll("3", "");
155     word = word+"1111111111";
156     word = word.substring(0, 10);
157 }
158 return word;
159 }
160 }

```

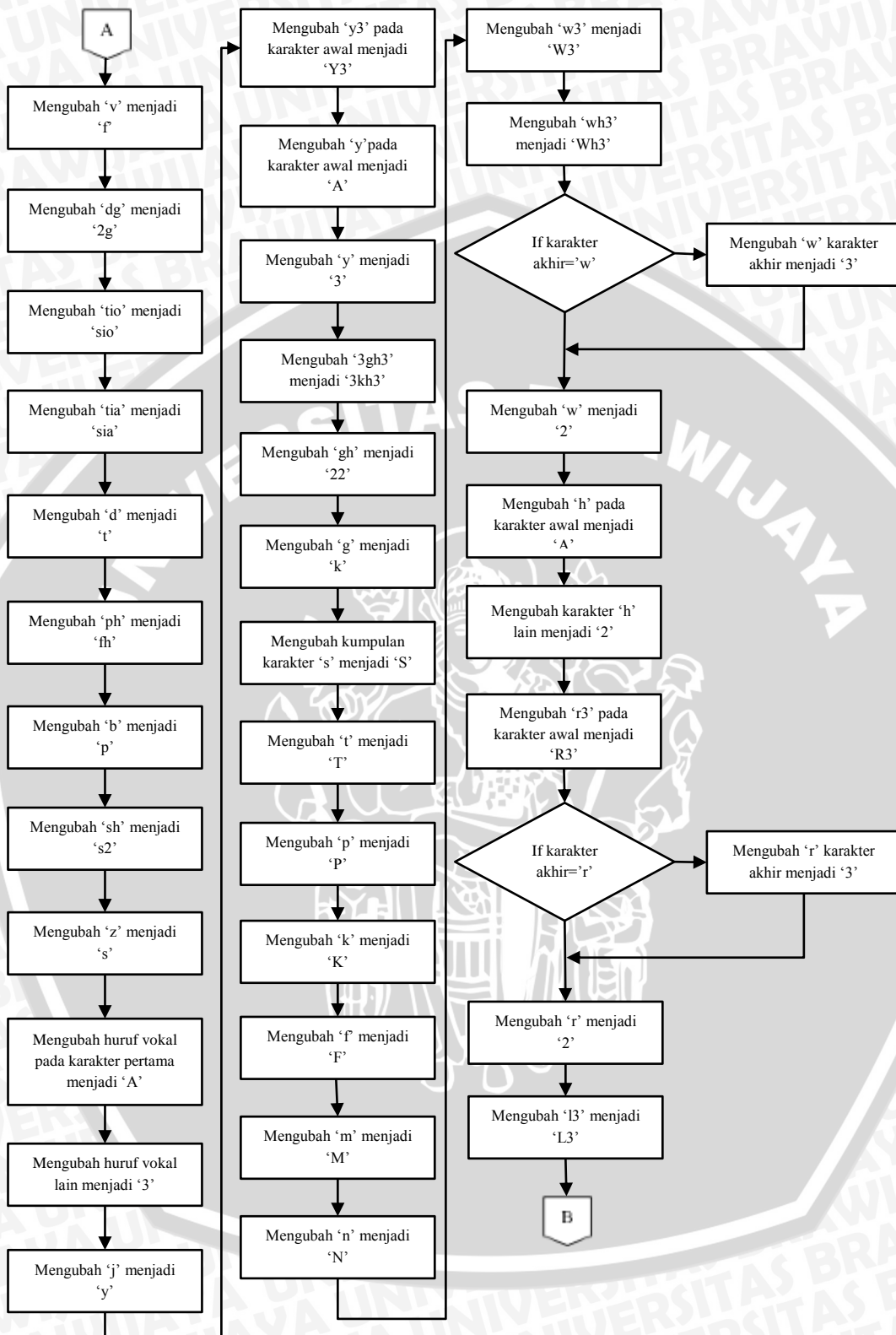
Gambar 3 Tampilan *sourcecode* Proses Pencarian Kode Fonetis (lanjutan)

Sumber: [Implementasi]

Lampiran 6. Diagram Alir Rule Algoritma Caverphone 2.0

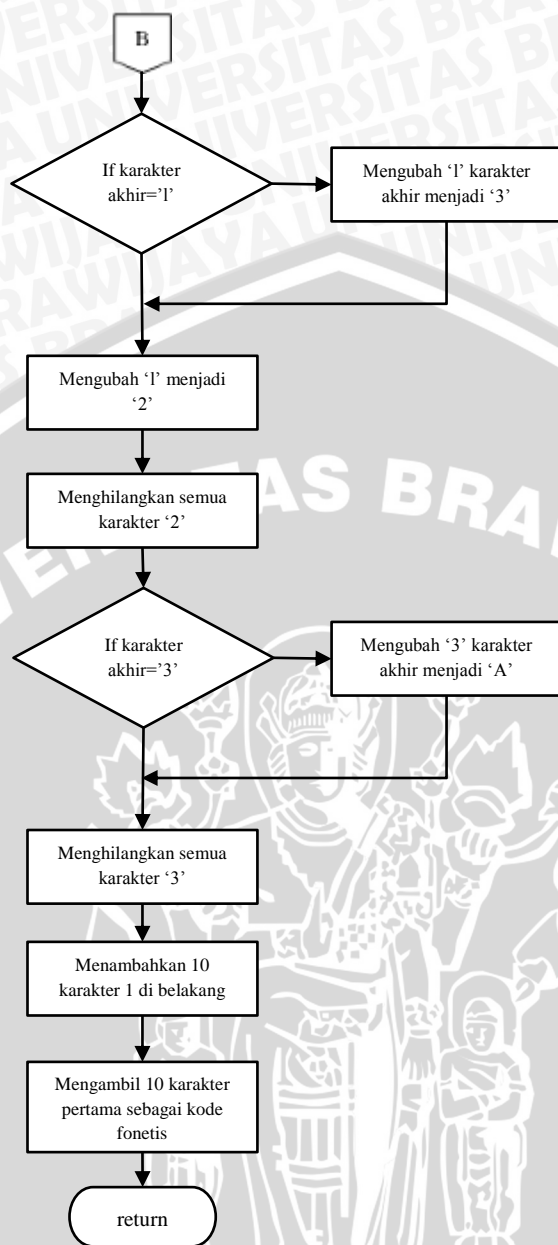


Gambar 1 Diagram Alir Rule Algoritma Caverphone 2.0
 Sumber: [Perancangan]



Gambar 2 Diagram Alir Rule Algoritma Caverphone 2.0 (lanjutan)

Sumber: [Perancangan]



Gambar 3 Diagram Alir Rule Algoritma Caverphone 2.0 (lanjutan)

Sumber: [Perancangan]