

**PERBANDINGAN ALGORITMA STRING MATCHING ON ORDERED
ALPHABET DAN ALGORITMA SIMON PADA KAMUS
ISTILAH ARSITEKTUR**

SKRIPSI

PUTRI CHALISKA

131401069



**PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2017**

PERBANDINGAN ALGORITMA *STRING MATCHING ON
ORDERED ALPHABETS* DAN ALGORITMA SIMON
PADA KAMUS ISTILAH ARSITEKTUR

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh
ijazah Sarjana Ilmu Komputer

PUTRI CHALISKA

131401069



PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

MEDAN

2017

PERSETUJUAN

Judul : PERBANDINGAN ALGORITMA STRING
MATCHING ON ORDERED ALPHABETS DAN
ALGORITMA SIMON DALAM KAMUS ISTILAH
ARSITEKTUR

Kategori : SKRIPSI

Nama : PUTRI CHALISKA

Nomor Induk Mahasiswa : 131401069

Program Studi : SARJANA (S1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Komisi Pembimbing

Pembimbing 2

Pembimbing 1

Amalia, S.T, M.T

Dian Rachmawati, S.Si, M.Kom

NIP. 197812212014042001

NIP. NIP. 197510082008011011

Diketahui/disetujui oleh

Program Studi S1 Ilmu Komputer

Ketua,

Dr. Poltak Sihombing, M.Kom

NIP. 196203171991021001

PERNYATAAN

PERBANDINGAN ALGORITMA *STRING MATCHING ON ORDERED ALPHABETS* DAN ALGORITMA SIMON DALAM KAMUS
ISTILAH ARSITEKTUR

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, Oktober 2017

Putri Chaliska

131401069

PENGHARGAAN

Puji dan syukur kehadirat Allah SWT, yang hanya dengan rahmat dan izin-Nya penulis dapat menyelesaikan penyusunan skripsi ini, sebagai syarat untuk memperoleh gelar Sarjana Komputer, pada Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.

Pada penggerjaan skripsi dengan judul perbandingan algoritma *String Matching on Ordered Alphabets* dan algoritma Simon dalam kamus istilah Arsitektur penulis menyadari bahwa banyak pihak yang turut membantu, baik dari pihak keluarga, sahabat dan orang-orang tercinta yang mendukung dalam penggerjaan skripsi ini. Dalam kesempatan ini, penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Runtung Sitepu, SH, M. Hum selaku Rektor Universitas Sumatera Utara.
2. Bapak Prof. Dr. Opim Salim Sitompul, M.si sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara
3. Bapak Dr. Poltak Sihombing, M.Kom sebagai Ketua Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Bapak Herryance, S.T., M.Kom sebagai Sekretaris Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Ibu Dian Rachmawati, S.Si., M.Kom selaku dosen pembimbing I yang telah memberikan memberikan kritik dan saran, serta dukungan kepada penulis dalam menyelesaikan skripsi ini.

6. Ibu Amalia, S.T, M.T selaku dosen pembimbing II yang telah memberikan kritik dan saran, serta dukungan kepada penulis dalam menyelesaikan skripsi ini.
7. Ibu Sri Melvani Hardi, S.Kom, M.Kom selaku dosen pembanding II yang telah memberikan kritik dan saran, serta dukungan kepada penulis dalam menyelesaikan skripsi ini.
8. Keluarga tercinta yang selalu menjadi motivasi dan semangat penulis Ayahanda Del Ikhsansyah, Ibunda Cut Nurhayati yang selama ini telah memberikan kasih sayang, doa, dukungan moral maupun materil yang selalu diberikan tanpa pamrih untuk penulis dan terima kasih juga kepada abangda Dhifa Udayana Putra dan adinda Rezky Nadira.
9. Teman Dekat Penulis Muhammad Gusti yang telah menjadi penyemangat dan tempat berbagi keluh kesah bagi penulis dalam menyelesaikan skripsi.
10. Kakak Sri wulandari yang telah menjadi tempat berbagi keluh kesah dan selalu memberikan motivasi kepada penulis.
11. Sahabat dari awal masuk kuliah Khairunissa Nasution yang selalu setia membantu dan memberikan dorongan dalam penulisan skripsi.
12. Sahabat-sahabat yang selalu setia membantu dan memberikan dorongan untuk lebih semangat dari awal penulisan sampai akhir penulisan skripsi Winda Aprianti Harahap, Raviza Sitepu dan Dodo Muhammad.
13. Sahabat dari SMA Nur Fitria Anggraini dan Anggi Sylvia yang telah memberikan dukungan, menghibur dan motivasi.
14. Anak Umi Frozen atas semangat yang telah diberikan kepada penulis
15. Teman-Teman kuliah stambuk 2013 terkhusus KOM C 2013 yang telah menjadi teman selama penulis menjalani perkuliahan ini.
16. Semua pihak yang terlibat langsung maupun tidak langsung yang tidak dapat penulisucapkan satu demi satu yang telah membantu penyelesaian Skripsi ini

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan karena kesempurnaan hanyalah milik Allah SWT semata. Oleh karena itu penulis menerima kritik dan saran dari semua pihak yang bersifat membangun dan menyempurnakan skripsi ini. Penulis berharap semoga skripsi ini bermanfaat bagi penulis sendiri pada khususnya dan pembaca pada umumnya.

Medan , Oktober 2017

Putri Chaliska

ABSTRAK

Arsitektur adalah seni dan ilmu dalam merancang bangunan. Dalam artian luas, arsitektur mencakup merancang dan membangun keseluruhan lingkungan binaan, mulai dari level makro yaitu perencanaan kota, perencanaan perkotaan, arsitektur lansekap hingga level mikro yaitu desain perabot dan desain produk. Kegiatan manusia, apalagi kebutuhan terhadap ruang selalu berkaitan dengan ilmu arsitektur, Di era yang semakin modern ini kamus yang tebal, mahal dan tidak *update* semakin mempersulit dalam mencari istilah arsitektur.Untuk membantu masyarakat atau penguna dalam menemukan arti dari istilah Arsitektur maka dibuatlah kamus istilah Arsitektur berbasis desktop yang menerapkan algoritma *String Matching on Ordered Alphabets* dan algoritma Simon. Algoritma *String Matching on Ordered Alphabets* ialah algoritma yang melakukan pengecekan kata satu-satu dari kiri ke kanan, Sedangkan algoritma Simon fase pencarian algoritmanya dilakukan dari kiri ke kanan dengan tahapan inisialisasi tiap indeks pada pola yang telah diberikan. Penelitian ini membandingkan kompleksitas waktu (Θ) dan *running time* dari Algoritma *String Matching on Ordered Alphabets* dan algoritma Simon. Berdasarkan uji coba yang dilakukan dan algoritma *String Matchin on Ordered Alphabets* memiliki rata rata waktu pencarian $1.6928ms$ dan algoritma Simon 1.5134 ms , yang berarti algoritma Simon memiliki rata-rata waktu pencarian yang lebih cepat dan kompleksitas yang didapatkan dari kedua algoritma adalah $\Theta(n)$.

Kata Kunci : Desain dan Analisis Algoritma, Kamus istilah Arsitektur, *String Matching on Ordered Alphabets*, Simon, Kompleksitas.

COMPARISON OF STRING MATCHING ALGORITHMS ON ORDERED ALPHABETS AND SIMON ALGORITHMS IN THE DICTIONARY Of ARCHITECTURE TERMS

ABSTRACT

Architecture is the art and science of designing buildings. In a broad sense, the architecture includes designing and building the entire built environment, ranging from the macro level of urban planning, urban planning, landscape architecture to the micro level of furniture design and product design. Human activities, let alone the need for space is always related to the science of architecture, In this increasingly modern era dictionary is thick, expensive and does not update the more difficult in finding the term architecture. To help the community or users in finding the meaning of the term Architecture then made a dictionary desktop-based architecture that implements the algorithm String Matchin on Ordered Alphabets and Simon algorithm . String Matching Algorithm on Ordered Alphabets is an algorithm that checks one-to-one from left to right, while the algorithm of Simon's algorithm search phase is done from left to right with the initialization stage of each index in the given pattern. This study compares the time complexity (Θ) and running time of the String Matchin Algorithm on Ordered Alphabets and the Simon algorithm. Based on the experiments performed and the String Matchin algorithm on Ordered Alphabets has an average search time of 1.6928ms and the Simon algorithm 1.5134 ms, which means the Simon algorithm has a lower average search time. The complexity obtained from both algorithms is $\Theta(n)$

Keywords: Design and Analysis Algorithm, Dictionary of Architecture term, String Matching, String Matchin on Ordered Alphabets, Simon, Complexity.

DAFTAR ISI

| | Halaman |
|---|----------------|
| Persetujuan | ii |
| Pernyataan | iii |
| Penghargaan | iv |
| Abstrak | vii |
| Abstrack | viii |
| Daftar Isi | ix |
| Daftar Tabel | xi |
| Daftar Gambar | xii |
| Daftar Lampiran | xiii |
| Bab I Pendahuluan | |
| 1.1 Latar Belakang | 1 |
| 1.2 Rumusan Masalah | 2 |
| 1.3 Batasan Masalah | 2 |
| 1.4 Tujuan Penelitian | 3 |
| 1.5 Manfaat Penelitian | 3 |
| 1.6 Metodologi Penelitian | 3 |
| 1.7 Sistematika Penulisan | 4 |
| Bab II Landasan Teori | |
| 2.1 Definisi Kamus | 6 |
| 2.2 Definisi Algoritma | 6 |
| 2.3 Definis Algoritma <i>String Matching</i> | 7 |
| 2.4 Algoritma <i>String Matching on Ordered Alphabets</i> | 8 |
| 2.5 Algoritma Simon | 15 |
| 2.6 Penelitian yang Relavan | 16 |
| Bab III Analisis dan Perancangan Sistem | |
| 3.1 Analisis Sistem | 18 |
| 3.1.1 Analisis Masalah | 18 |
| 3.1.2 Analisis Kebutuhan | 20 |
| 3.1.2.1 Kebutuhan Fungsional | 20 |
| 3.1.2.2 Kubutahan NonFungsional | 20 |
| 3.1.3 Analisis Proses | 21 |
| 3.2 Perancangan Sistem | 21 |
| 3.2.1 General Arsitektur Umum | 21 |

| | | |
|-----------------------|---|----|
| | <i>3.2.2 Use-Case Diagram</i> | 22 |
| | <i>3.2.3 Activity Diagram</i> | 23 |
| | <i>3.2.4 Sequence Diagram</i> | 24 |
| | 3.2.5 Kamus Data | 25 |
| | 3.2.6 Flowchart | 25 |
| 3.3 | Pseudocode | 28 |
| 3.4 | Perancangan Sistem | 31 |
| | 3.4.1 Rancangan Halaman Utama | 31 |
| | 3.4.2 Rancangan Halaman Menu | 32 |
| | 3.4.3 Rancangan Halaman Bantuan | 33 |
| | 3.4.4 Rancangan Halaman Tentang | 34 |
| | 3.4.5 Rancangan Halaman Update Kata | 35 |
| Bab IV | Implementasi dan Pengujian | |
| | 4.1 Implementasi | 36 |
| | 4.1.1 Tampilan Halaman Utama | 36 |
| | 4.1.2 Tampilan Halaman Menu | 37 |
| | 4.1.3 Tampilan Halaman Update Kata | 37 |
| | 4.1.4 Tampilan Halaman Bantuan | 38 |
| | 4.1.5 Tampilan Halaman Tentang | 39 |
| | 4.2 Pengujian Sistem | 39 |
| | 4.2.1 Pengujian Algoritma <i>String Matching on Ordered Alphabets</i> dan Algoritma Simon | 39 |
| | 4.3 Kompleksitas Algoritma | 45 |
| Bab V | Kesimpulan dan Saran | |
| | 5.1 Kesimpulan | 50 |
| | 5.2 Saran | 51 |
| DAFTAR PUSTAKA | | 51 |

DAFTAR TABEL

| | | |
|------------|---|----|
| Tabel 2.1 | Pergeseran <i>Pattern</i> SMOA 1 | 9 |
| Tabel 2.2 | Pergeseran <i>Pattern</i> SMOA 2 | 9 |
| Tabel 2.3 | Pergeseran <i>Pattern</i> SMOA 3 | 10 |
| Tabel 2.4 | Pergeseran <i>Pattern</i> SMOA 4 | 10 |
| Tabel 2.5 | Pergeseran <i>Pattern</i> SMOA 5 | 10 |
| Tabel 2.6 | Pergeseran <i>Pattern</i> SMOA 6 | 10 |
| Tabel 2.7 | Pergeseran <i>Pattern</i> SMOA 7 | 11 |
| Tabel 2.8 | Pergeseran <i>Pattern</i> SMOA 8 | 11 |
| Tabel 2.9 | Pergeseran <i>Pattern</i> SMOA 9 | 11 |
| Tabel 2.10 | Pergeseran <i>Pattern</i> SMOA10 | 11 |
| Tabel 2.11 | Pergeseran <i>Pattern</i> Simon 1 | 12 |
| Tabel 2.12 | Pergeseran <i>Pattern</i> Simon 2 | 13 |
| Tabel 2.13 | Pergeseran <i>Pattern</i> Simon 3 | 13 |
| Tabel 2.14 | Pergeseran <i>Pattern</i> Simon 4 | 13 |
| Tabel 2.15 | Pergeseran <i>Pattern</i> Simon 5 | 13 |
| Tabel 2.16 | Pergeseran <i>Pattern</i> Simon 6 | 14 |
| Tabel 2.17 | Pergeseran <i>Pattern</i> Simon 7 | 14 |
| Tabel 2.18 | Pergeseran <i>Pattern</i> Simon 8 | 14 |
| Tabel 2.19 | Pergeseran <i>Pattern</i> Simon 9 | 14 |
| Tabel 2.20 | Pergeseran <i>Pattern</i> Simon 10 | 15 |
| Tabel 2.21 | Pergeseran <i>Pattern</i> Simon 11 | 15 |
| Tabel 2.22 | Pergeseran <i>Pattern</i> Simon 12 | 15 |
| Tabel 2.23 | Pergeseran <i>Pattern</i> Simon 13 | 15 |
| Tabel 3.1 | Kamus Data | 25 |
| Tabel 3.2 | Keterangan rancangan halaman Utama | 31 |
| Tabel 3.3 | Keterangan rancangan Menu | 32 |
| Tabel 3.4 | Keterangan rancangan halaman Bantuan | 33 |
| Tabel 3.5 | Keterangan rancangan halaman Tentang | 34 |
| Tabel 3.6 | Keterangan rancangan halaman Update Kata | 35 |
| Tabel 4.1 | Perbandingan waktu pencarian <i>pattern</i> pada sistem | 40 |
| Tabel 4.2 | Kompleksitas Fungsi <i>SMOA</i> | 45 |
| Tabel 4.3 | Kompleksitas algoritma <i>String Matching Ordered Alphabets</i> | 46 |
| Tabel 4.4 | Kompleksitas algoritma Simon | 48 |

DAFTAR GAMBAR

| | | |
|-------------|--|----|
| Gambar 2.1 | Percobaan pada algoritma <i>String Matching on Ordered Alphabets</i> | 8 |
| Gambar 3.1 | Diagram ishikawa untuk Analisis Masalah | 19 |
| Gambar 3.2 | General arsitektur umum | 21 |
| Gambar 3.3 | <i>Use-case Diagram</i> | 22 |
| Gambar 3.4 | <i>Activity Diagram</i> | 23 |
| Gambar 3.5 | <i>Sequence Diagram</i> Sistem | 24 |
| Gambar 3.6 | <i>Flowchart</i> gambaran umum sistem | 26 |
| Gambar 3.7 | <i>Flowchart String Matching on Ordered Alphabets</i> | 27 |
| Gambar 3.8 | <i>Flowchart</i> algoritma Simon | 28 |
| Gambar 3.9 | Rancangan Halaman Utama | 31 |
| Gambar 3.10 | Rancangan Halaman Menu | 32 |
| Gambar 3.11 | Rancangan Halaman Bantuan | 33 |
| Gambar 3.12 | Rancangan Halaman Tentang | 34 |
| Gambar 3.13 | Rancangan Halaman Update Kata | 35 |
| Gambar 4.1 | Halaman Utama | 36 |
| Gambar 4.2 | Halaman Menu | 37 |
| Gambar 4.3 | Halaman Update Kata | 38 |
| Gambar 4.4 | Halaman Bantuan | 38 |
| Gambar 4.5 | Halaman Tentang | 39 |
| Gambar 4.6 | Hasil pencarian <i>pattern</i> Add | 40 |
| Gambar 4.7 | Hasil pencarian <i>pattern</i> Action | 41 |
| Gambar 4.8 | Hasil pencarian <i>pattern</i> Acrylic | 41 |
| Gambar 4.9 | Hasil pencarian <i>pattern</i> Basement | 42 |
| Gambar 4.10 | Hasil pencarian <i>pattern</i> Cabin | 42 |
| Gambar 4.11 | Hasil pencarian <i>pattern</i> Basement Window | 43 |
| Gambar 4.12 | Hasil pencarian <i>pattern</i> Arsitek | 43 |
| Gambar 4.13 | Perbandingan <i>running time</i> algoritma <i>String Matching on Ordered Alphabets</i> dan algoritma Simon | 44 |

DAFTAR LAMPIRAN

| | |
|----------------------|-----|
| Listing Program | A-1 |
| Daftar Riwayat Hidup | B-1 |

BAB I

PENDAHULUAN

1.1. Latar Belakang

Arsitektur adalah seni dan ilmu dalam merancang bangunan. Dalam artian luas, arsitektur mencakup merancang dan membangun keseluruhan lingkungan binaan, mulai dari level makro yaitu perencanaan kota, perencanaan perkotaan, arsitektur lansekap hingga level mikro yaitu desain perabot dan desain produk. Arsitektur juga merujuk kepada hasil-hasil proses perancangan tersebut .Kegiatan manusia, apalagi kebutuhan terhadap ruang selalu berkaitan dengan ilmu arsitektur, maka dari itu masyarakat sebaiknya mengerti dan sedikit memahami tentang arsitektur.

Dalam ilmu arsitektur banyak istilah istilah yang tidak dipahami oleh masyarakat, mahasiswa bahkan para arsitek profesional. Menurut seorang dosen dan konsultan dibidang arsitektur, mengatakan bahwa “saya sering mendapati para mahasiswa maupun klien kesulitan bahkan tidak mengenali istilah-istilah arsitektur yang sebenarnya sangat umum dan harus diketahui”. Sehingga mengharuskan mempelajari istilah memalui kamus. Di era yang semakin modern ini kamus yang tebal, mahal dan tidak *update* semakin mempersulit dalam mencari istilah arsitektur (Uniek Praptiningrum, 2008).

Maka dari itu dengan kemajuan teknologi yang semakin canggih penulis akan membangun aplikasi kamus istilah arsitektur digital yang dengan mudah dapat digunakan dalam mencari istilah tanpa harus membaca lembar perlembar kata yang ingin dicari. Penulis membuat aplikasi kamus istilah arsitektur berbasis desktop dengan menggunakan *String Matching on Ordered Alphabets* dan algoritma Simon. Dimana berdasarkan penelitian sebelumnya (siregar,2017) didapatkan hasil bahwa Algoritma *String Matching on Ordered Alphabets* lebih cepat dari algoritma *brute force* karna saat dilakukan pencocokan *string*, algoritma ini menghitung *suffix* maksimal dari *prefix pattern* yang sama dari *string* yang dicari tiap percobaannya. Ini berfungsi untuk menghindarkan perhitungan awal lagi setelah pergantian panjang

dilakukan. Dan penelitian (Megawaty, 2017) didapatkan hasil bahwa algoritma Simon lebih cepat dari algoritma *brute force* dikarenakan algoritma Simon fase pencariannya dilakukan dari kiri ke kanan dengan tahapan inisialisasi tiap indeks pada pola yang telah diberikan. Maka dari hasil yang didapatkan penulis akan membandingkan Algoritma *String Matching on Ordered Alphabets* dan Algoritma Simon untuk mengetahui mana yang lebih cepat dan efisien dalam pencarian *string*.

1.2. Rumusan Masalah

Berdasarkan latar belakang diatas dapat dilihat Algoritma *String Matching on Ordered Alphabets* dan Algoritma Simon merupakan algoritma memiliki karakteristik pencarian *string* yang sama yaitu dari kiri ke kanan sehingga apabila dibandingkan dengan algoritma lain maka algoritma tersebut menghasilkan waktu yang lebih cepat dalam proses pencarian *string*. Untuk itulah penulis tertarik membandingkan algoritma *String Matching on Ordered Alphabets* dan Algoritma Simon untuk menghitung *running time* dan kompleksitasnya.

1.3. Batasan Masalah

Batasan penelitian sebagai berikut :

1. Aplikasi kamus ini dapat menerjemahkan 500 kata dan istilah pada Arsitektur.
2. Parameter banding yang digunakan adalah *running time* (milisekon) dan kompleksitas dengan notasi Big Θ
3. Istilah yang dicari berasal dari dua referensi yang berbeda yang bersumber dari pada Kamus Arsitektur & Konstruksi penerbit Dahara Prize dan Kamus Istilah Arsitektur penerbit Andi.
4. Bahasa Pemograman yang digunakan adalah bahasa C# dan DBMS MySql

1.4. Tujuan penelitian

Tujuan penelitian ini untuk mengetahui perbandingan algoritma *String Matching on Ordered Alphabets* dan algoritma *Simon* pada aplikasi kamus istilah Arsitektur, dan mencari algoritma yang lebih efisien dalam pencocokan *string* dengan membandingkan kompleksitas teoretis (big Θ), *running time* pencocokan *pattern* (milisekon).

1.5. Manfaat Penelitian

1. Penulis maupun pembaca dapat memahami dan melihat perbandingan waktu yang mana lebih efisien antara algoritma *String Matching on Ordered Alphabets* dengan algoritma *Simon*.
2. Dapat membantu pengguna dalam memahami istilah Arsitektur.

1.6. Metodologi Penelitian

Penelitian ini menerapkan beberapa metode penelitian sebagai berikut:

1. Studi Literatur

Pada tahap ini penulisan dimulai dengan studi kepustakaan yaitu proses pengumpulan bahan-bahan referensi baik dari buku-buku, artikel-artikel, maupun dari hasil penelitian mengenai *String Matching*, algoritma String matching on ordered Alphabets dan algoritma *Simon*.

2. Pengumpulan dan Analisis Data

Pada tahap ini dilakukan pengumpulan dan analisis data yang berhubungan dengan penelitian ini, seperti fungsi algoritma String matching on ordered alphabets dan *Simon* bisa bekerja dalam sebuah aplikasi pencarian *string* agar penulis mengetahui karakter (*string*) yang akan dicari.

3. Perancangan Sistem

Merancang sistem sesuai dengan rencana yang telah ditentukan, yaitu meliputi perancangan desain awal seperti *button* maupun *font* yang lebih minimalis, flowchart, ishikawa dan general arsitektur . Proses perancangan ini berdasarkan pada batasan masalah dari penelitian ini.

4. Implementasi Sistem

Pada tahap ini pembuatan sistem telah selesai dilaksanakan dan menambahkan data hasil algoritma String matching on ordered Alphabets dan Simon ke dalam sistem.

5. Pengujian Sistem

Pada tahap ini akan dilakukan pengujian terhadap sistem yang telah dikembangkan.

6. Dokumentasi Sistem

Melakukan pembuatan dokumentasi sistem mulai dari tahap awal hingga pengujian sistem, untuk selanjutnya dibuat dalam bentuk laporan penelitian (skripsi)

1.7. Sistematika penulisan

Sistematika penulisan skripsi ini terdiri dari beberapa bagian utama yang dijelaskan sebagai berikut.

BAB 1 PENDAHULUAN

Bab ini menjelaskan latar belakang penelitian yang dilakukan, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab ini berisi tentang penjelasan singkat mengenai algoritma, pencocokan string (*string matching*), teori-teori dasar yang digunakan dalam algoritma *String Matching on Ordered Aplhabets* dan algoritma Simon.

BAB 3 ANALISIS DAN PERANCANGAN

Bab ini membahas analisis terhadap masalah penelitian, analisis kebutuhan dalam membangun sistem dan perancangan terhadap sistem yang akan dibangun.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini merupakan hasil penelitian yang dilakukan. Berisi tentang penjelasan implementasi sistem berdasarkan analisis dan perancangan, dilakukan pengujian terhadap sistem yang telah dibangun serta pembahasan hasil pengujian.

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran keseluruhan penelitian berdasarkan hasil pengujian yang diharapkan dapat bermanfaat untuk pengembangan berikutnya.

BAB II

LANDASAN TEORI

2.1. Definisi Kamus

Menurut Kamus Besar Bahasa Indonesia (KBBI) kamus merupakan buku acuan yang memuat kata dan ungkapan, biasanya disusun menurut abjad beserta keterangan dan makna, pemakaian atau terjemahan(Siregar, 2017). Selain itu, kamus merupakan buku yang memuat kumpulan istilah atau nama yang disusun menurut abjad berikut dengan penjelasan makna dan pemakaianya.

2.2. Definisi Algoritma

Algoritma adalah tahapan-tahapan logis penyelesaian masalah yang tersusun secara sistematis. Tahapan-tahapan yang dilakukan harus logis , berarti nilai kebenarannya harus dapat ditentukan, benar atau salah. Tahapan-tahapan yang tidak benar akan menghasilkan hasil yang salah (Siregar, 2017).

Menurut Donald E. Knuth, algoritma mempunyai lima ciri penting yang meliputi. (Wulan, 2011).

1. Algoritma mempunyai awal dan akhir. Suatu algoritma harus berhenti setelah mengerjakan serangkaian tugas atau dengan kata lain suatu algoritma harus memiliki langkah yang terbatas.
2. Setiap langkah harus didefinisikan dengan tepat sehingga tidak memiliki arti ganda.
3. Memiliki masukkan atau kondisi awal.
4. Memiliki keluaran atau kondisi akhir.
5. Algoritma harus efektif, bila diikuti dengan benar–benar akan menyelesaikan persoalan.

Dalam pembuatan kamus, diperlukannya algoritma untuk melakukan pencocokan string atau sering disebut dengan *string matching algorithm*.

2.3. Definisi Algoritma *String Matching*

Pencocokan *String* (*String Matching*) adalah proses pencarian kata (*pattern*) dalam sebuah teks. *String matching* bisa digunakan untuk menemukan satu *pattern* yang pertama kali ditemukan, atau yang lebih umum menampilkan semua *pattern* yang dapat ditemukan dalam teks. Jenis *string matching* bermacam-macam, dibedakan menurut hasil yang diinginkan. (Charras & Lecroq, 1997).

Prinsip kerja algoritma *string matching* adalah sebagai berikut :

1. Memindai teks dengan bantuan sebuah *window* yang ukurannya sama dengan *pattern*.
2. Menempatkan *window* pada awal teks
3. Membandingkan karakter pada *window* dengan karakter dari *pattern*. Setelah pencocokan (baik hasilnya cocok atau tidak cocok), dilakukan *shift* ke kanan pada *window*. Presedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks. Mekanisme ini disebut *sliding-window*.

String Matching dibagi menjadi dua, yaitu *exact string matching* dan *heuristic* (*statistical matching*). *Exact matching* digunakan untuk menemukan *pattern* yang berasal dari satu teks. Contoh pencarian *exact matching* adalah pencarian kata “pelajar” dalam kalimat “ saya seorang pelajar” atau “saya seorang siswa”. Sistem akan memberikan hasil bahwa kata pelajar dan siswa bersinonim. Algoritma *exact matching* diklasifikasikan menjadi tiga bagian menurut arah pencarinya, yaitu :

1. Arah pencarian dari kiri ke kanan.

Algoritma yang termasuk dalam kategori ini adalah *Brute Force*, *Morris* dan *Pratt* (yang kemudian dikembangkan oleh *Knuth, Morris* dan *Pratt*).

2. Arah pembacaan dari kanan ke kiri.

Algoritma yang termasuk dalam kategori ini adalah *Boyer Moore* yang kemudian dikembangkan menjadi algoritma *Turbo Boyer Moore*, *Tuned Boyer Moore* dan *Zhu Takaoka*.

3. Arah pencarian yang ditentukan oleh program.

Algoritma yang termasuk dalam kategori ini adalah algoritma *Colussi* dan *Crochemore-Perrin*.

Heuristic matching adalah teknik yang digunakan untuk menghubungkan dua data terpisah ketika *exact matching* tidak mampu mengatasi karena pembatasan pada data yang tersedia. *Heuristic matching* dapat dilakukan dengan perhitungan *distance* antara *pattern* dengan teks. *Exact* dan *heuristic matching* memiliki kemiripan makna tetapi berbeda tulisan.

2.4. Algoritma *String Matching on Ordered Alphabets*

Algoritma *String Matching on Ordered Alphabets* ialah algoritma yang mirip dengan algoritma pencarian satu-satu (*brute force*). Pengecekan pada setiap kedudukan dalam *text* dari karakter pertama hingga karakter akhir. Seusai pengecekan karakter pertama terjadi, maka proses *shift* dilakukan dengan berpindah tepat satu posisi ke arah kanan (berpindah ke karakter kedua, ketiga dan seterusnya). Perbedaanya adalah pada saat dilakukan percobaan untuk menyamakan *string* dimana ‘jendela’ diposisikan oleh *substring* $y[j..j+m-1]$, saat prefix *u* dari *x* telah ditemukan dan ketidaksamaan terjadi antara karakter *a* dalam *x* dengan *b* dalam kata *y*.

| | | | |
|---|---|---|--|
| y | u | B | |
| x | u | A | |

Gambar 2.1 Percobaan pada algoritma *String Matching on Ordered Alphabets*.

Algoritma ini akan menghitung periode *ub* seperti pada gambar diatas. Jika tidak berhasil dalam menemukan periode yang tepat, algoritma ini akan beralih untuk menghitung perkiraannya. Algoritma *String Matching on Ordered Alphabets* tidak membutuhkan fase untuk melakukan praproses.

Berikut ini akan dijelaskan langkah-langkah penyelesaian dengan Algoritma *String Matching On Ordered Alphabets*.

Contoh :

Teks (T) : Basement Window

Pattern (P) : Wind

Perhitungan Pergeseran : Untuk proses pergeseran dihitung berdasarkan banyaknya *string* yang sama ditambah dengan *string* yang berbeda. Jika karakter *string* teks dan *pattern* tidak sama, lakukan pengecekan ke *string* berikutnya

Tabel 2.1. Pergeseran Pattern SMOA1

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | W | I | N | D | | | | | | | | | | | |

Pada Langkah pertama *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.2. Pergeseran Pattern SMOA2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | W | I | N | D | | | | | | | | | | |

Pada Langkah ke-2 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.3. Pergeseran Pattern SMOA3

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | W | I | N | D | | | | | | | | | |

Pada Langkah ke-3 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.4. Pergeseran Pattern SMOA4

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | W | I | N | D | | | | | | | | |

Pada Langkah ke-4 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.5. Pergeseran Pattern SMOA5

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | W | I | N | D | | | | | | | |

Pada Langkah ke-5 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.6. Pergeseran Pattern SMOA6

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | W | I | N | D | | | | | | | |

Pada Langkah ke-6 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.7. Pergeseran Pattern SMOA7

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | W | I | N | D | | | | | |

Pada Langkah ke-7 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.8. Pergeseran Pattern SMOA8

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | W | I | N | D | | | | |

Pada Langkah ke-8 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.9. Pergeseran Pattern SMOA9

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | | W | I | N | D | | |

Pada Langkah ke-9 *pattern* belum cocok dengan karakter pada text ,selanjutnya *pattern* akan bergeser 1 ke kanan.

Tabel 2.10. Pergeseran Pattern SMOA10

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | | W | I | N | D | | |

Pada Langkah ke-10 *pattern* sudah cocok dengan karakter . Pada Langkah ke-10 *pattern* sudah cocok dengan karakter pada text ,selanjutnya *pattern* akan berhenti.

2.5. Algoritma Simon

Pada algoritma Simon akan dilakukan fase untuk pencarian string yang dilakukan dari kiri ke kanan dengan tahapan inisialisasi di setiap indeknya berdasarkan *pattern* yang akan diberikan, kemudian pada keadaan awal pencarian *string* diberi nilai -1 yang artinya *mismatch* atau tidak sesuai dengan *pattern*.

Berikut adalah contoh Algoritma Simon dalam melakukan pencarian *pattern* terhadap teks.

Inisialisasi dimulai dari -1.

Keadaan : -1 { W → 0 }

Keadaan : 0 { W → 0 I → 1 }

Keadaan : 1 { W → 0 N → 2 }

Keadaan : 2 { W → 0 D → 3 }

Keadaan : 3 { }

Berhenti di keadaan 3.

Tabel 2.11. Pergeseran Pattern Simon 1

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | W | | | | | | | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka B diberi nilai -1 yang berarti *mismatch*.

Tabel 2.12. Pergeseran Pattern Simon 2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | W | | | | | | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka A diberi nilai -1 yang berarti *mismatch*.

Tabel 2.13. Pergeseran Pattern Simon 3

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | W | | | | | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka S diberi nilai -1 yang berarti *mismatch*.

Tabel 2.14. Pergeseran Pattern Simon 4

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | W | | | | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka E diberi nilai -1 yang berarti *mismatch*.

Tabel 2.15. Pergeseran Pattern Simon 5

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | W | | | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka M diberi nilai -1 yang berarti *mismatch*.

Tabel 2.16. Pergeseran Pattern Simon 6

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | W | | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka E diberi nilai -1 yang berarti *mismatch*.

Tabel 2.17. Pergeseran Pattern Simon 7

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | W | | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka N diberi nilai -1 yang berarti *mismatch*.

Tabel 2.18. Pergeseran Pattern Simon 8

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | W | | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka T diberi nilai -1 yang berarti *mismatch*.

Tabel 2.19. Pergeseran Pattern Simon 9

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | W | | | | | | |

-1

Karna *pattern* tidak sesuai dengan text maka (*space*) diberi nilai -1 yang berarti *mismatch*.

Tabel 2.20. Pergeseran Pattern Simon 10

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | | W | | | | | |

0

Karena *pattern* sesuai dengan text maka W diberi nilai 0 yang artinya *match*.

Tabel 2.21. Pergeseran Pattern Simon 11

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | | | I | | | | |

1

Karena *pattern* sesuai dengan text maka I diberi nilai 1 yang artinya *match*.

Tabel 2.22. Pergeseran Pattern Simon 12

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | | | | N | | | |

2

Karena *pattern* sesuai dengan text maka N diberi nilai 2 yang artinya *match*.

Tabel 2.23. Pergeseran Pattern Simon 13

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|
| T | B | A | S | E | M | E | N | T | | W | I | N | D | O | W |
| P | | | | | | | | | | | | D | | | |

3

Karena *pattern* sesuai dengan text maka D diberi nilai 3 yang artinya *match*. Keadaan 3 sudah terpenuhi , sesuai insialisasi awal pencarian akan berhenti pada keadaan 3.

2.6. Penelitian yang Relavan

Berikut ini beberapa penelitian yang terkait dengan Algoritma *String Matching on Ordered Alphabets* dan Algoritma Simon :

1. Pada penelitian terdahulu yang dilakukan oleh Iqbal, 2016 tentang “Analisis dan Perbandingan Algortima Colussi dan Algoritma Simon dalam Pencarian Kata pada Jurnal”, menyimpulkan bahwa algoritma *Colussi* memiliki beberapa tahap sebelum pemrosesan yaitu dengan menggunakan fungsi $x[i]$, $kmin[i]$, $h[i]$, $next[i]$, $shift[i]$, $hmax[i]$, $rmin[i]$ dan juga $ndh0[i]$ sedangkan algoritma *Simon* hanya menggunakan inisialisasi untuk memberikan indeks pada tiap pola dan inisialisasi setiap *state* pada pola.
2. Pada penelitian terdahulu yang dilakukan oleh Nurmaidah, 2017 tentang“ Perbandingan Algoritma Turbo Boyer Moore dan Algoritma String Matching on Ordered Alphabets untuk Aplikasi Kamus Fisika Berbahasa Indonesia Berbasis Android”, menyimpulkan bahwa Algoritma Turbo Boyer Moore melakukan pencarian kata lebih cepat bila dibandingkan dengan String Matching on Ordered Alphabets , begitu juga dengan running timenya.
3. Pada penelitian terdahulu yang dilakukan oleh Megawaty,2017 tentang “Implementasi dan Perbandingan Algoritma *Brute Force* dengan Algoritma Simon dalam Pembuatan Kamus Istilah Kebidanan”, menyimpulkan bahwa Algoritma *Simon* melakukan pencarian kata lebih cepat bila dibandingkan dengan *Brute Force* , begitu juga dengan running timenya.
4. Pada Jurnal Dr.Deepak Garg dan Nimisha Singla tentang “ String Matching Algorithms and their Applicability in various Application “ manyimpulkan dalam skenario online saat ini, menemukan konten yang sesuai dalam waktu minimum adalah yang terpenting. Algoritma string memainkan peran penting untuk ini. Kelompok yang berbeda bekerja pada tingkat perangkat lunak dan perangkat keras

untuk membuat pola pencarian lebih cepat. Dengan menerapkan berbagai algoritma dalam berbagai aplikasi, perkiraan algoritma terbaik untuk berbagai aplikasi ditentukan. Algoritma yang disarankan memberikan kompleksitas yang berkurang dan juga mengurangi waktu komputasi. Algoritma yang ditugaskan ke berbagai aplikasi mungkin bukan algoritma optimal terbaik namun lebih baik daripada algoritma umum. Alih-alih menerapkan setiap algoritma ke setiap aplikasi satu aplikasi dijelaskan dengan algoritma optimal tertentu.

BAB III

ANALISIS DAN PERANCANGAN

3.1. Analisis Sistem.

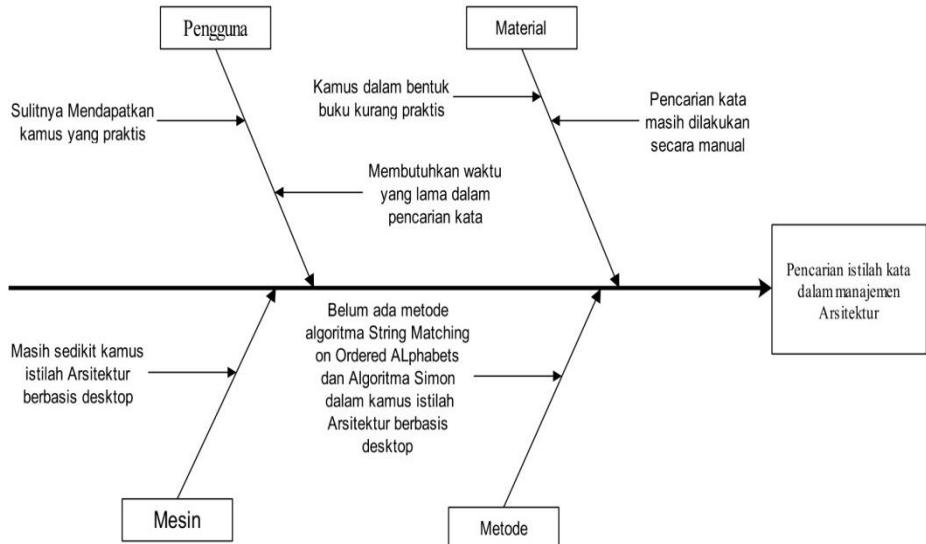
Analisis sistem ialah teknik untuk menyelesaikan suatu masalah di dalam sistem menjadi suatu komponen kecil dengan cara mengidentifikasi kebutuhan, mengevaluasi konsep sistem, melakukan analisis teknis dan ekonomis, mengalokasikan fungsi-fungsi untuk perangkat keras, perangkat lunak, manusia, database dan elemen yang lain, membuat batasan biaya dan jadwal serta menciptakan definisi sistem. Ada dua tahapan analisis sistem yaitu analisis masalah dan analisis persyaratan. Analisis masalah untuk memahami kelayakan masalah dan analisis persyaratan untuk menjelaskan fungsi-fungsi yang ditawarkan dan mampu dikerjakan sistem.

3.1.1. Analisis Masalah

Analisis masalah merupakan proses mengidentifikasi sebab dan akibat dibangunnya sebuah sistem agar sistem yang akan dibangun dapat berjalan sesuai dengan tujuan dari sistem tersebut. Seperti yang kita ketahui, jika seseorang ingin mencari pengertian istilah Arsitektur, hal yang pertama kali digunakan yaitu dengan menggunakan kamus. Kamus identik dengan ukuran yang tebal dan berat sehingga tidak efektif dan efisien dalam penggunaannya karena menghabiskan banyak waktu dalam proses pencarian. Untuk mengatasi permasalahan tersebut dibutuhkan sebuah aplikasi kamus Arsitektur yang bertujuan untuk mempermudah proses pencarian kata.

Pencarian kata adalah masalah yang akan diselesaikan dengan menggunakan sistem. Untuk mengidentifikasi masalah tersebut digunakan diagram *Ishikawa* (*fishbone* diagram). Diagram *ishikawa* berbentuk seperti ikan dimana strukturnya terdiri dari kepala ikan (*fish'shead*) dan tulang-tulang ikan (*fish's bones*). Nama dan judul dari masalah yang akan diidentifikasi terletak pada bagian kepala ikan. Sedangkan tulang-

tulang ikan menggambarkan penyebab-penyebab masalah tersebut. Diagram *Ishikawa* pada sistem dapat dilihat pada **Gambar 3.1**



Gambar 3.1. Diagram Ishikawa untuk Analisis Masalah.

Pada gambar 3.1 dapat dilihat bahwa terdapat empat kategori penyebab masalah pada penelitian pencarian istilah kata dalam kamus arsitektur yang digambarkan dengan tanda panah yang mengarah ke tulang utama, yaitu berkaitan dengan pengguna, bahan (*material*), metode, dan media atau alat yang terlibat mesin. Setiap detail penyebab masalah tersebut digambarkan dengan tanda panah yang mengarah kemasing-masing kategori. Pada Pengguna yaitu manusia, kurang efisien nya waktu dalam mencari kata/istilah pada kebidanan Masalah dari Material adalah bahan acuan untuk mencari kata/istilah adalah masih berbentuk buku (*hardcopy*) yang membuat kurang efisien. Untuk masalah dari Metode, pada pembuatan kamus istilah kebidanan ini menggunakan dua algoritma dari pencocokan kata, yaitu algoritma *String Matching on Ordered Alphabets* dengan algoritma Simon. Dan untuk masalah dari Mesin, karena belum adanya aplikasi untuk mencari kata/istilah Arsitektur yang berbasis *desktop*.

3.1.2. Analisis Kebutuhan

Tahap analisis kebutuhan terbagi atas dua bagian, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kedua kebutuhan ini harus ada untuk membangun sebuah sistem yang baik.

3.1.2.1. Kebutuhan Fungsional

Kebutuhan fungsional yang dimiliki kamus istilah Arsitektur ini adalah sebagai berikut:

1. Sistem dapat menampilkan pencocokan *string* dari kata yang dimasukkan oleh pengguna.
2. Sistem dapat menampilkan *running time* pencarian dalam satuan waktu *milisecon* dan jumlah kata yang cocok dari algoritma String Matching on ordered Alphabets dan algoritma Simon
3. Sistem memiliki fitur untuk menambahkan data istilah Arsitektur

3.1.2.2. Kebutuhan Non-Fungsional

Kebutuhan nonfungsional yang dimiliki kamus istilah Arsitektur ini adalah sebagai berikut:

1. Kinerja
Sistem yang dibangun memiliki *running time* yang cepat dan tampilan yang menarik.
2. Mudah Dipelajari dan Digunakan
Tampilan dari sistem yang dibangun bersifat sederhana dan ramah pengguna (*user friendly*).
3. Hemat Biaya
Sistem yang dibangun bersifat hemat biaya karena dapat digunakan tanpa menggunakan koneksi internet dan tidak berbayar.

3.1.3. Analisis Proses

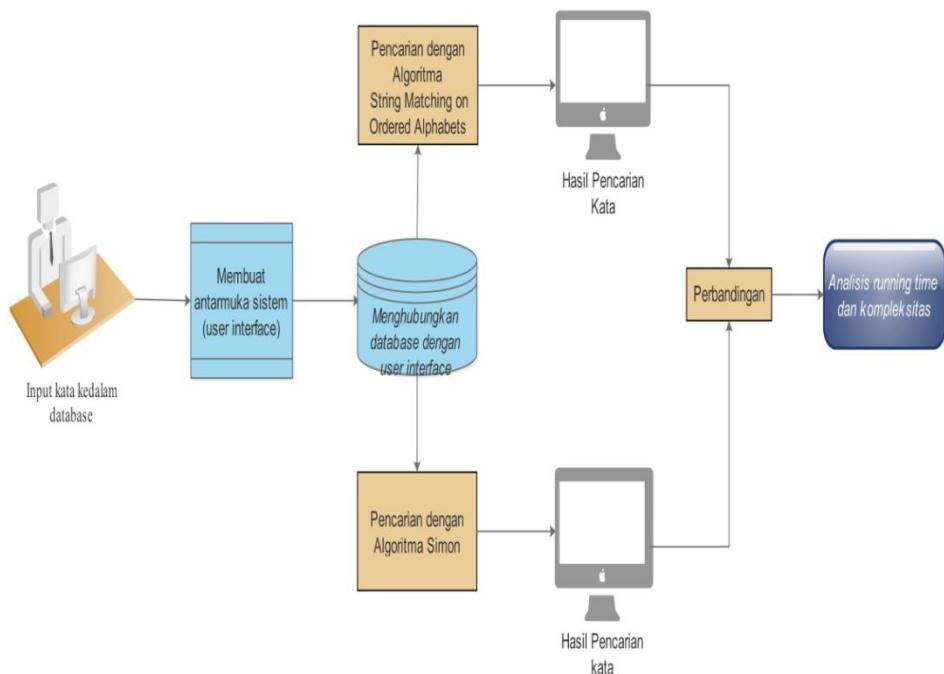
Sistem yang akan dibangun menggunakan algoritma String Matching on ordered Alphabets dan algoritma Simon untuk proses pencocokan *string*. Sistem akan melakukan pencocokan *string* berdasarkan kata yang dimasukkan oleh pengguna (*pattern*) dan kata yang ada di dalam database. Kemudian, kata di dalam database yang mengandung *pattern* yang dimasukkan oleh *user* akan ditampilkan di sistem.

3.2. Perancangan Sistem.

Perancangan sistem merupakan cara untuk mendapatkan gambaran dengan jelas tentang apa yang akan dikerjakan pada analisis sistem dan dilanjutkan dengan memikirkan bagaimana membuat sistem tersebut. Pada perancangan sistem terdapat *flowchart*, *use case diagram*, *activity diagram*, *sequence diagram* dan perancangan antarmuka (*interface*).

3.2.1. General Arsitektur Sistem.

General arsitektur sistem adalah gambaran sistem secara keseluruhan yang meliputi proses *input* data, proses sistem berlangsung, dan *output* data yang dikeluarkan dari sistem. Ada beberapa hal yang dapat dilakukan pengguna kepada sistem.

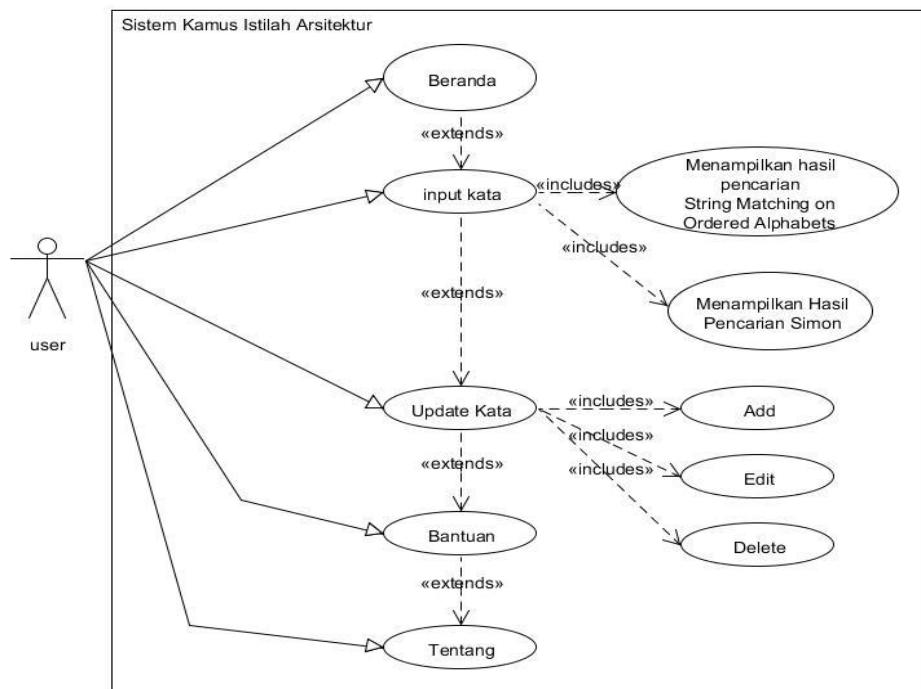


Gambar 3.2. General Arsitektur Sistem

*General arsitektur sistem seperti yang ditunjukkan pada gambar 3.2. Diatas, menunjukkan interaksi yang dilakukan pembuat sistem kepada sistem. Langkah pertama, dilakukan pengisian kamus kata di dalam database. Database yang digunakan adalah MySQL. Selanjutnya, pembuat sistem membuat antarmuka sistem (*user interface*) untuk memudahkan pengguna dalam menggunakan sistem. Sistem dibuat menarik sehingga bersifat *user friendly*. Selanjutnya, database yang berisi kamus kata dihubungkan dengan antarmuka sistem dengan menggunakan bahasa pemrograman C#. Kemudian, pengguna memasukkan kata yang ingin dicari dan sistem akan melakukan pencarian dan membandingkan hasil kecepatan waktu pencarinya. Setelah dilakukan pencarian, maka sistem akan mengeluarkan hasil pencarian di dalam antarmuka sistem sehingga pengguna bisa melihat hasilnya*

3.2.2. Use Case Diagram

*Usecase atau diagram *usecase* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. Usecase mendekripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Usecase digunakan untuk mengetahui fungsi apa yang ada di dalam sebuah sistem informasi dan siapa yang berhak menggunakan fungsi-fungsi tersebut (Siregar, 2017).*



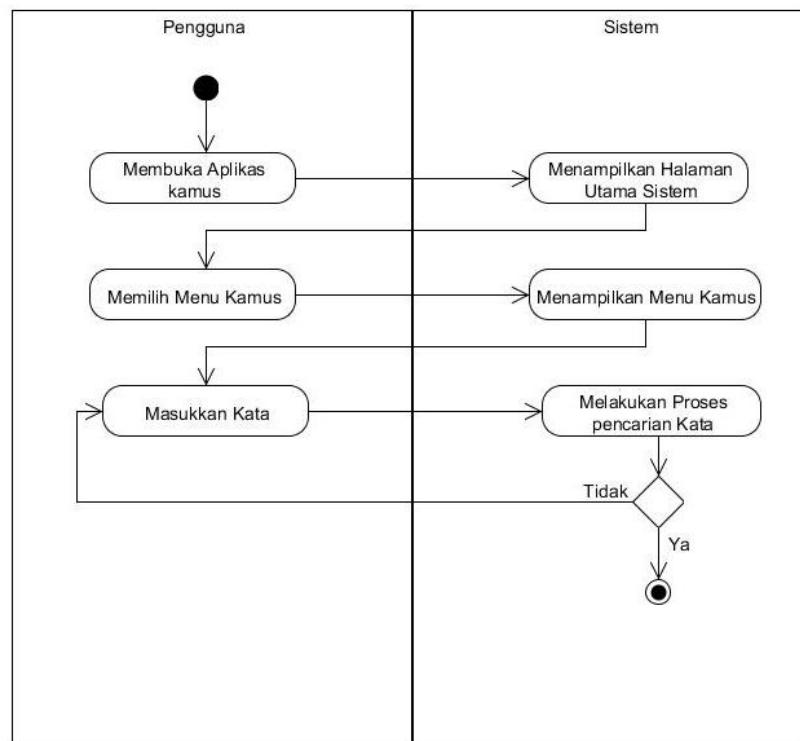
Gambar 3.3. Use Case Diagram sistem

Terlihat pada gambar 3.3. bahwa *actor* pada *use case* ada satu, yaitu pengguna. Selanjutnya, pengguna dapat melakukan interaksi dengan sistem sebanyak empat interaksi. Pertama pengguna masuk kemenu utama, Kedua pengguna memasukkan kata yang ingin dicari. Ketiga, sistem akan menampilkan hasil pencarian dan ditampilkan di antarmuka sistem. Keempat pengguna dapat mengakses menu tentang.

Untuk menampilkan hasil pencarian, penulis menambahkan bagian *includes* pada *use case*, artinya bahwa hasil pencarian kata hanya bisa ditampilkan jika sebelumnya pengguna telah memasukkan kata yang dicari.

3.2.3. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan aliran kerja (*workflow*) atau aktivitas dari sebuah sistem atau proses bisnis. Manfaat dari *activity diagram* ialah untuk membantu memahami proses secara keseluruhan dari sistem tersebut. *activity diagram* dibuat berdasarkan sebuah atau beberapa *use case* diagram. Proses pencarian dapat dilihat pada *activity diagram* gambar 3.4 berikut:



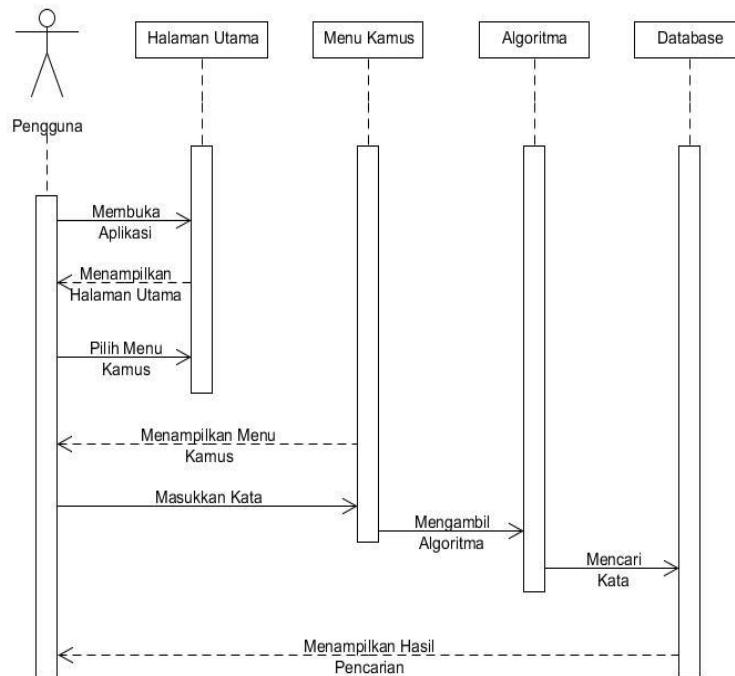
Gambar 3.4 Activity Diagram

Pada *activity diagram* gambar 3.4. dijelaskan interaksi antara pengguna dengan sistem di dalam aplikasi kamus. Dimulai dari pengguna membuka aplikasi,

maka sistem akan menampilkan menu utama. Lalu, pengguna memilih menu kamus dan sistem menampilkan menu kamus. Kemudian, pengguna memasukkan kata yang ingin dicari dan sistem akan melakukan proses pencarian terhadap masukan dari pengguna. Dalam tahap prosesnya, akan ada dua kondisi, yaitu jika kata yang dicari sesuai, maka sistem akan menampilkan kata tersebut, sebaliknya, jika kata yang dicari tidak sesuai, maka sistem akan menampilkan pesan kata tidak ditemukan kemudian pengguna diminta untuk kembali memasukkan kata. Ketika sistem berhasil menampilkan hasil pencarian kata, maka sistem berhasil berjalan dan selesai.

3.2.4. Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan interaksi antara objek dengan proses yang terkait pada kelas diagram melalui *message* dalam eksekusi *operation*, untuk satu buah *use case*. *Sequence* membantu untuk menggambarkan data yang masuk dan keluar dari sistem, seperti yang terlihat pada 3.5 berikut :



Gambar 3.5 *Sequence Diagram*

Pada gambar 3.5 *sequence diagram* sistem memiliki empat objek yaitu pengguna, menu utama, menu kamus, algoritma dan database yang digambarkan dengan simbol objek UML. Referensi pada use-case digambarkan dengan *lifeline-garis* vertikal putus-putus. Kotak persegi empat yang berada pada masing-masing

objek merupakan *behavior* atau operasi yang perlu dilakukan oleh masing-masing objek. Kotak-kotak tersebut menggambarkan kode program. Kemudian, anak panah antara garis menggambarkan interaksi atau pesan yang telah dikirim ke objek tertentu untuk menginvokasi (memanggil) salah satu operasinya untuk memenuhi permintaan.

3.2.5. *Kamus Data*

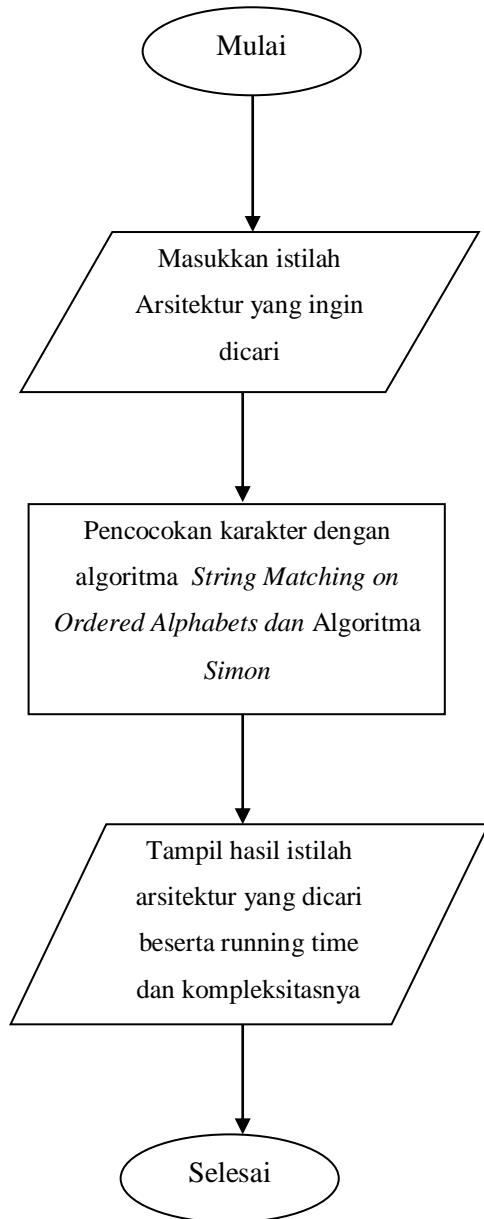
Kamus data merupakan sebuah daftar yang mengatur semua komponen data yang berhubungan terhadap sistem dengan definisi singkat dan sejelas-jelasnya sehingga pengguna dan analisis sistem dapat sama-sama mengerti tentang data masukan, keluaran, komponen penyimpanan, dan kalkulasi lanjutan (Setiawan, 2012). Kamus data dari sistem yang akan dibangun dapat dilihat pada Tabel 3.1.

Tabel 3.1 kamus data

| Data | Filed | Type | Deskripsi |
|--------------------|-------|-----------------------------|----------------------------------|
| Tabel_kamus_istiah | ID | Integer, <i>Primary key</i> | Kode unik, <i>Auto increment</i> |
| | Kata | Varchar(300) | Teks kata istilah Arsitektur |
| | Arti | Text | Teks arti istilah Arsitektur |

3.2.6. *Flowchart*.

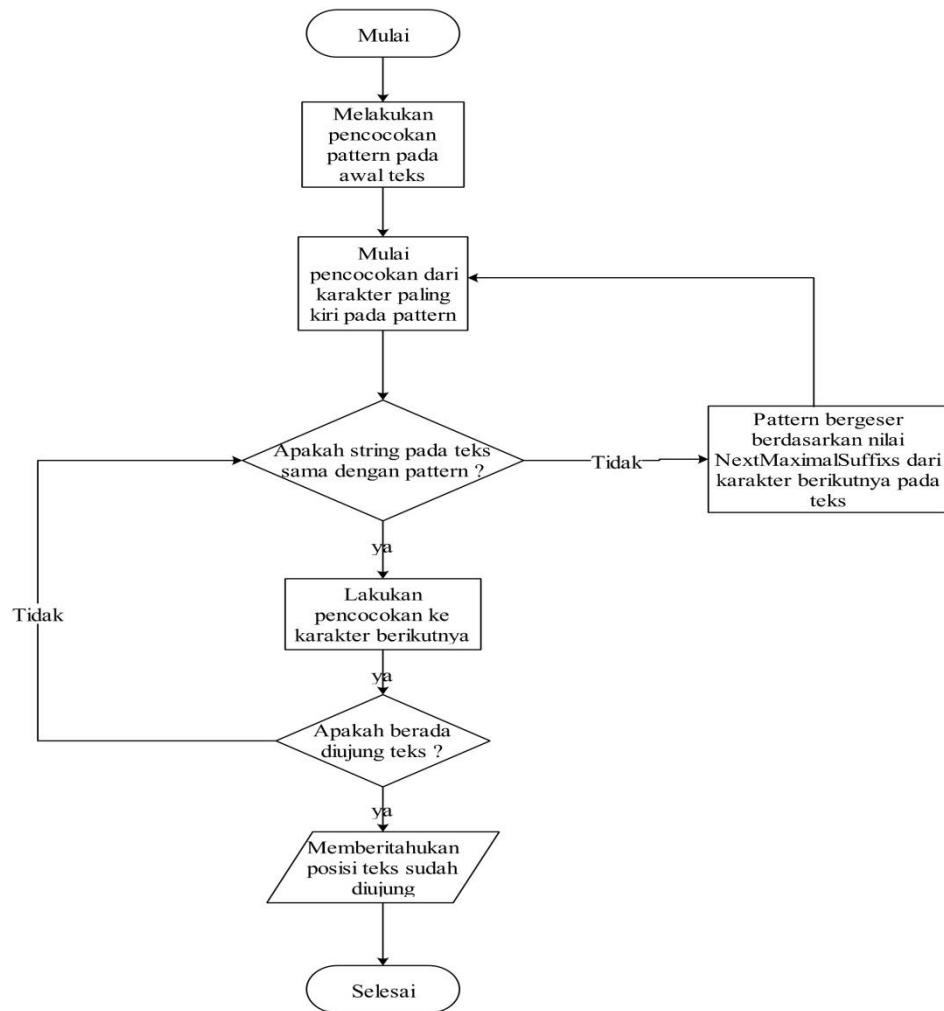
Flowchart merupakan gambar atau bagan yang menunjukkan aliran data didalam sistem secara logika. *Flowchart* sistem dapat didefinisikan sebagai bagan yang menunjukkan aliran pekerjaan secara keseluruhan dari sistem. *Flowchart* dari sistem yang akan dibangun dapat dilihat pada gambar 3.6



Gambar 3.6 Flowchart Gambaran Umum Sistem

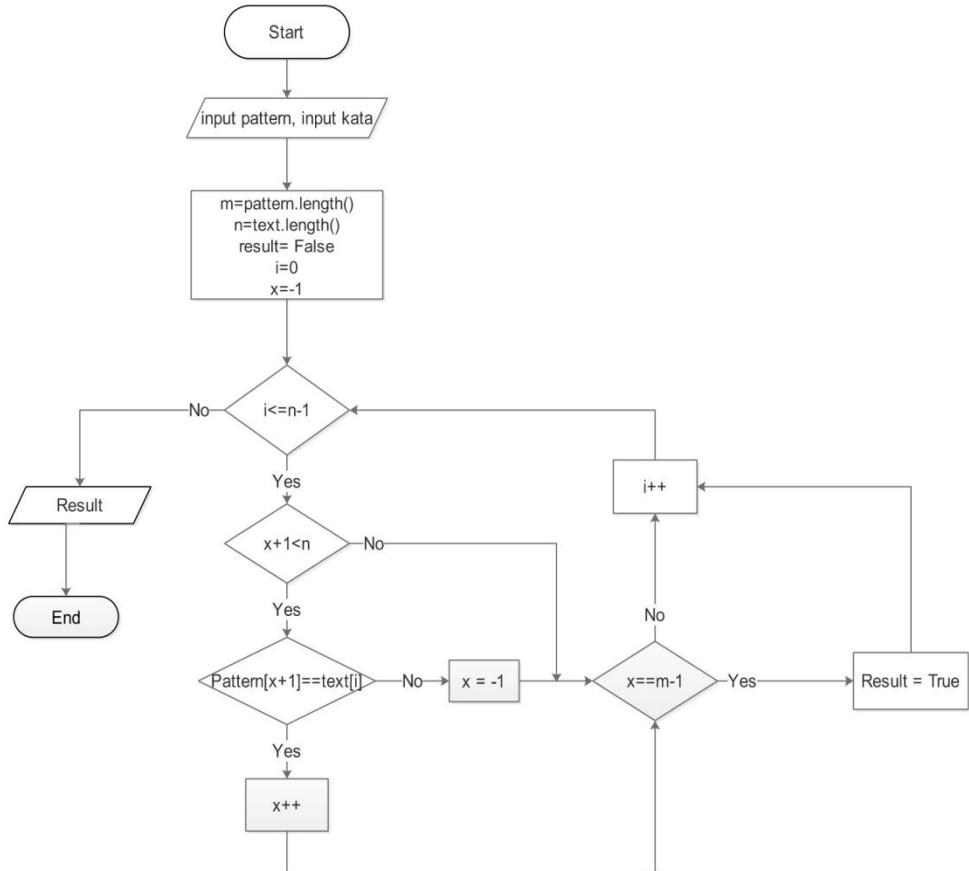
Pada gambar 3.6 dijelaskan mengenai *flowchart* (alur sistem). Pengguna memasukkan istilah Arsitektur akan dicari, kemudian sistem melakukan pencocokan string menggunakan algoritma *String Matching on Ordered Alphabets* dan algoritma Simon, kemudian sistem akan menampilkan hasil pencarian istilah Arsitektur.

Untuk algoritma *String Matching on Ordered Alphabets* memiliki satu *flowchart*, yaitu fase pencarian yang akan ditunjukkan pada gambar 3.7.



Gambar 3.7 Flowchart fase pencarian algoritma String Matching on ordered alphabets

Untuk algoritma Simon terdapat satu buah *flowchart*, yang akan ditunjukkan pada gambar 3.8 berikut.



Gambar 3.8 Flowchart algoritma Simon

3.3. Pseudocode

Pseudocode adalah sebuah kode yang digunakan untuk menulis sebuah algoritma dengan cara yang bebas yang tidak terikat dengan bahasa pemrograman tertentu. *Pseudocode* berisikan langkah-langkah sistematis untuk menyelesaikan permasalahan tertentu. *Pseudocode* menggunakan bahasa yang hampir menyerupai bahasa pemrograman. Selain itu biasanya *pseudocode* menggunakan bahasa yang mudah dipahami secara *universal* dan juga lebih ringkas dari pada algoritma. Berikut *pseudocode* untuk algoritma *String matching on ordered alphabets* dan algoritma Simon.

Pseudocode untuk algoritma String Matching on Ordered Alphabets

```

bool SMOA (string x, string y){
    int i, ip, j, jp, k, p;
    ip=-1;
    i=j=jp=0;
    k=p=1;
    while (j<=y.Length-x.Length) {
        while(i+j<y.Length && i<x.Length && x[i] == y[i+j])
            ++i;
        if (i==0){
            ++j;
            ip = -1;
            jp = 0;
            k=p=1;
        }
        else {
            if (i>= x.Length)
                return true;
            nextMaximalSuffix(y+j, i+1, ip, jp, k, p);
            if (ip< 0 || ip<p && Equals((y+j).Substring(0, ip+1), (y+j+p).Substring(0, ip+1))){
                j+=p;
                i-=p;
                if (i<0)
                    i=0;
                if (jp-ip>p)
                    jp-=p;
                else{
                    ip=-1;
                    jp=0;
                    k=p=1;
                }
            }
            else {
                j+=(Math.Max(ip+1, Math.Min(i-ip-1, jp+1))+1);
            }
        }
    }
}

```

```

    i=jp=0;
    ip=-1;
    k=p=1;
}
}
}

```

Pseudocode untuk algoritma Simon

```

bool simon (string x, string y){
    int j,m;
    j=0;
    m=-1;
    do {
        if(m+1 <=x.Length-1){
            if(x[m+1] == y[j])
                m++;
            else
                m=-1;
        }
        if(m ==x.Length-1){
            return true;
            m=-1;
        }
        j++;
    }
    while (j<=y.Length-1);
    return false;
}

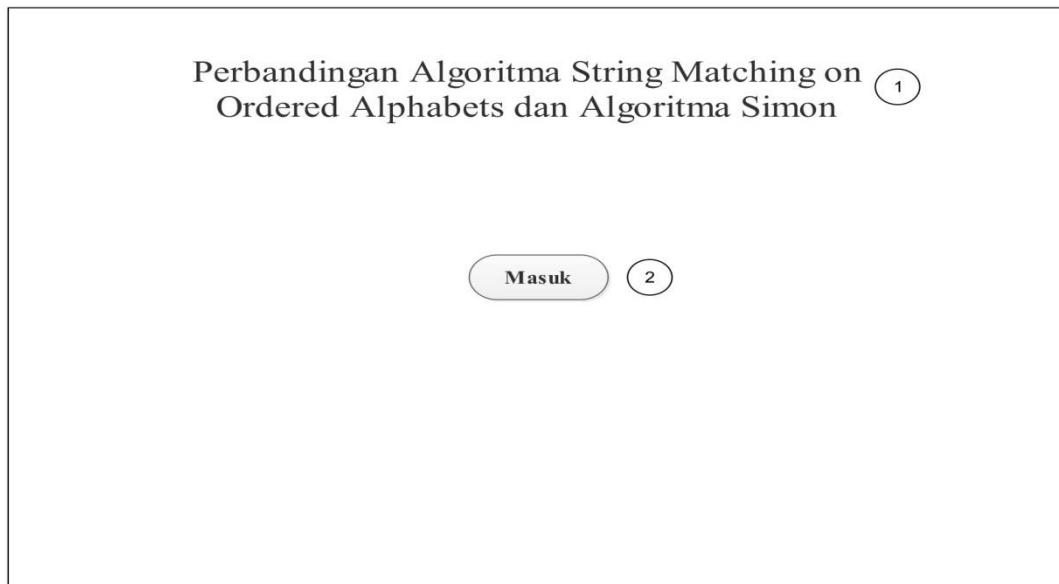
```

3.4. Perancangan Sistem.

Proses perancangan *interface* sistem adalah sebuah proses yang penting. Sebuah *interface* sistem dirancang untuk memudahkan pengguna dalam menggunakan sistem tersebut sehingga memberikan kenyamanan dan kemudahan dalam penggunaannya. Proses perancangan sistem sepenuhnya harus memperhatikan faktor pengguna yang akan menggunakan sistem tersebut sehingga harus bersifat *user friendly*.

3.4.1. Rancangan Halaman Utama

Halaman utama merupakan halaman yang pertama kali muncul pada saat sistem dibuka. Rancangan halaman utama dapat dilihat pada gambar 3.9. berikut.



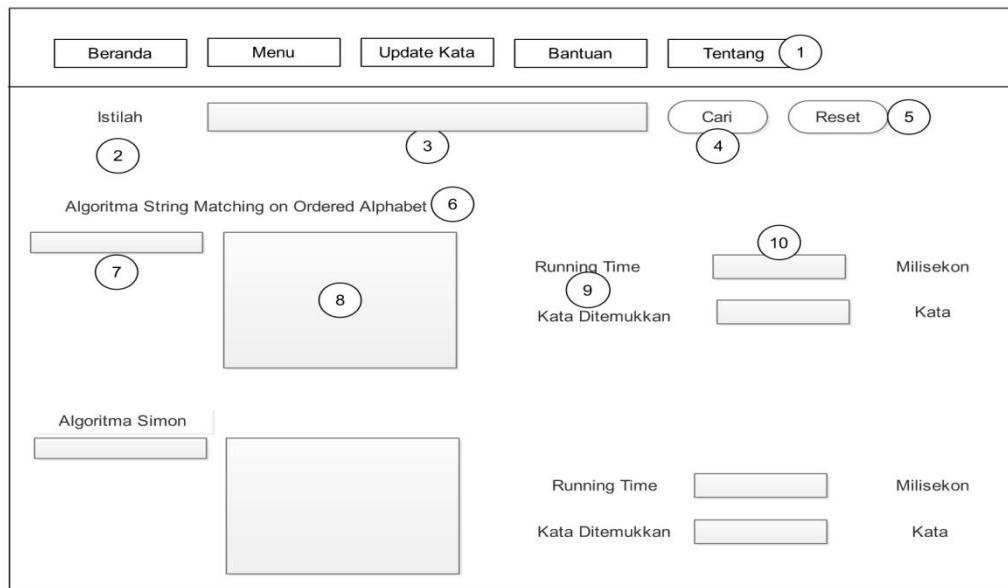
Gambar 3.9. Rancangan tampilan halaman utama

Tabel 3.2. Keterangan rancangan halaman utama

| No. | Keterangan |
|-----|--|
| 1 | <i>Label</i> untuk menampilkan judul skripsi |
| 2 | <i>Button</i> untuk masuk kemu menu kamus |

3.4.2. Rancangan Halaman Menu

Halaman Menu merupakan halaman pencarian kata, dimana proses pencarian kata dengan algoritma berlangsung. Halaman menu berisi program pencarian kata yang dihubungkan dengan *database*. Rancangan halaman menu dapat dilihat pada gambar 3.10. berikut.



Gambar 3.10. Rancangan tampilan halaman Menu

Tabel 3.3. Keterangan rancangan halaman menu

| No. | Keterangan |
|-----|---|
| 1 | <i>Menu strip</i> untuk menyajikan menu printah menu utama, menu printah cari kata , menu printah tentang dan menu printah bantuan. |
| 2 | <i>Label</i> untuk menampilkan judul skripsi |
| 3 | <i>Text box</i> untuk memasukkan kata yang ingin dicari |
| 4 | <i>Button cari</i> untuk mencari kata yang telah dimasukkan |
| 5 | <i>Button hapus</i> untuk masuk kemu menu kamus |
| 6 | <i>Label</i> untuk meampilkhan tulisan |
| 7 | <i>List Box</i> untuk meampilkhan hasil pencarian |
| 8 | <i>Text box</i> untuk menampilkan arti kata |
| 9 | <i>Label</i> untuk menampilkan tulisan |
| 10 | <i>Text box</i> untuk menampilkan waktu pencarian |

3.4.3. Rancangan Halaman Bantuan

Pada Halaman Bantuan merupakan halaman yang berisi deskripsi pembuat program dalam penelitian ini. Rancangan halaman tentang dapat dilihat pada gambar 3.11. berikut



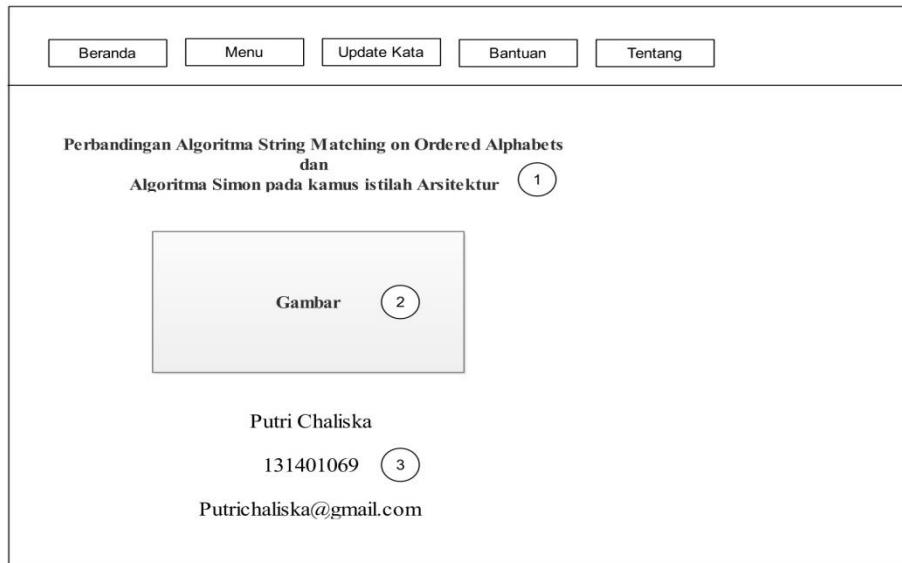
Gambar 3.11. Rancangan tampilan halaman bantuan

Tabel 3.4. Keterangan rancangan halaman bantuan

| No. | Keterangan |
|-----|---|
| 1 | <i>Menu strip</i> untuk menyajikan menu printah menu utama, menu printah cari kata , menu printah tentang dan menu printah bantuan. |
| 2 | <i>Label</i> untuk menampilkan tulisan cara menggunakan sistem |
| 3 | <i>Rich Text Box</i> untuk menampilkan panduan penggunaan sistem |

3.4.4. Rancangan Halaman Tentang

Halaman tentang merupakan halaman yang berisi deskripsi pembuat program dalam penelitian ini. Rancangan halaman tentang dapat dilihat pada gambar 3.12. berikut



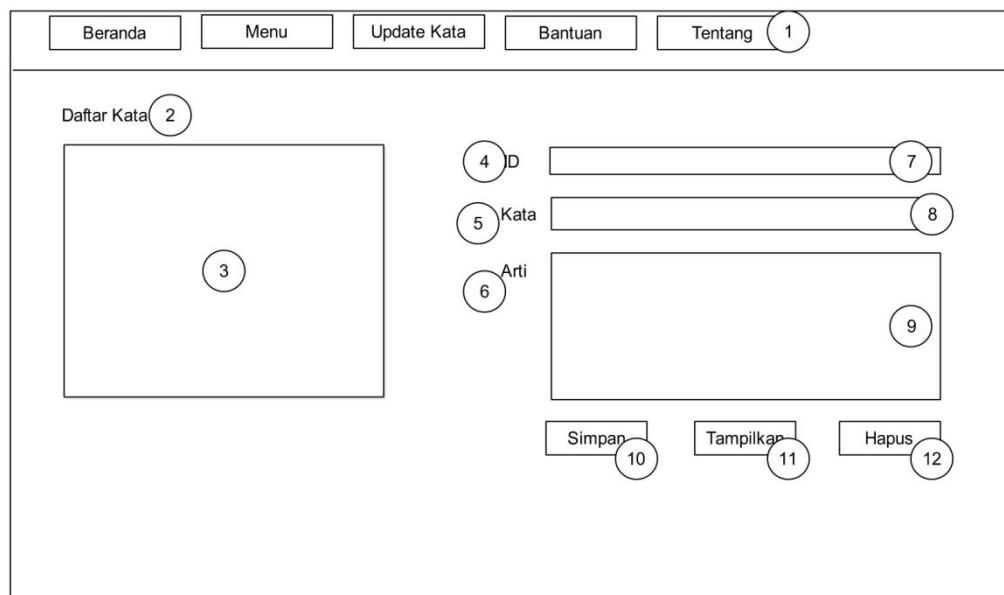
Gambar 3.12. Rancangan tampilan halaman tentang

Tabel 3.5. Keterangan rancangan halaman tentang

| No. | Keterangan |
|-----|---|
| 1 | <i>Menu strip</i> untuk menyajikan menu printah menu utama, menu printah cari kata , menu printah tentang dan menu printah bantuan. |
| 2 | <i>Picture box</i> untuk menampilkan foto pembuat |
| 4 | <i>Label</i> untuk menampilkan nama, nim, email pembuat |

3.4.5. Rancangan Halaman *Update* kata

Halaman *update* kata merupakan halaman yang dapat menambah kata dan menghapus kata dalam penelitian ini. Rancangan halaman *update* kata dapat dilihat pada gambar 3.13. berikut



Gambar 3.13. Rancangan tampilan halaman *update kata*

Tabel 3.6. Keterangan rancangan halaman *update kata*

| No. | Keterangan |
|-----|---|
| 1 | <i>Menu strip</i> untuk menyajikan menu printah menu utama, menu printah cari kata , menu printah tentang dan menu printah bantuan. |
| 2 | <i>Label</i> untuk menampilkan tulisan daftar kata |
| 3 | <i>Text box</i> untuk menampilkan kata |
| 4 | <i>Label</i> untuk menampilkan tulisan no. |
| 5 | <i>Label</i> untuk menampilkan tulisan kata |
| 6 | <i>Label</i> untuk menampilkan tulisan arti kata |
| 7 | <i>Text box</i> untuk memasukkan angka |
| 8 | <i>Text box</i> untuk memasukkan istilah kata baru |
| 9 | <i>Text box</i> untuk memasukkan arti istilah |
| 10 | <i>Button simpan</i> untuk menyimpan istilah baru |
| 11 | <i>Button tampilkan kata</i> untuk menampilkan kata yang ada dalam <i>database</i> |
| 12 | <i>Button hapus kata</i> untuk menghapus kata dari <i>database</i> |
| 13 | <i>Button reset</i> untuk menghapus istilah baru yang belum disimpan |

BAB IV

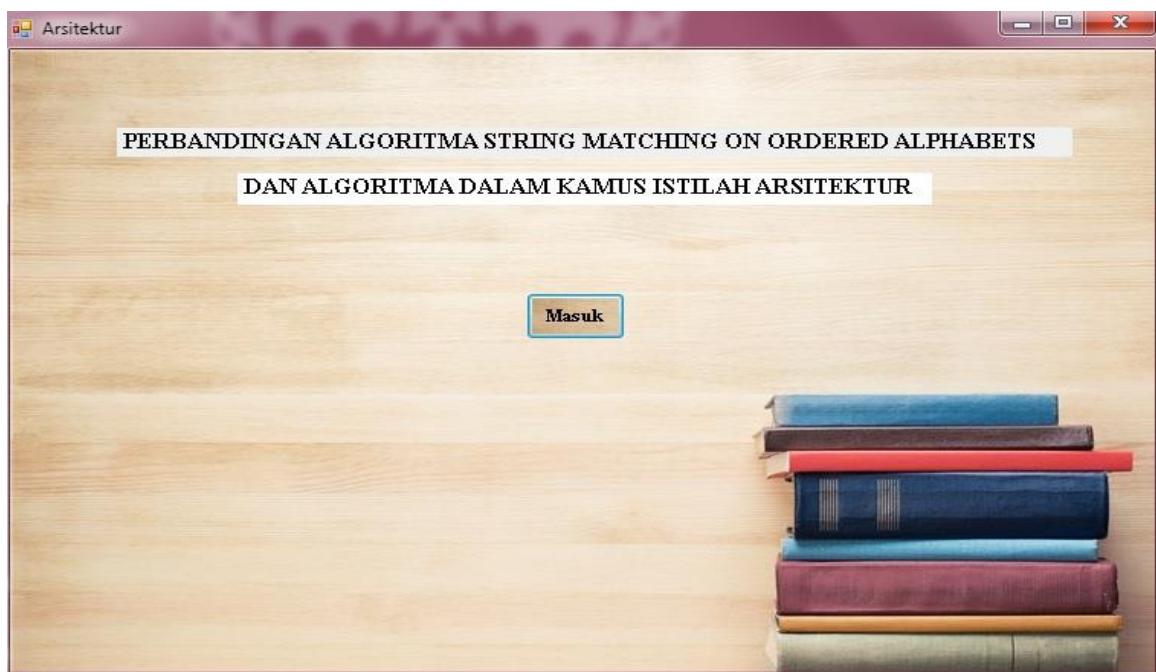
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi Sistem.

Implementasi sistem merupakan tahapan lanjutan dari analisis dan perancangan sistem. Sistem ini akan dibangun dengan menggunakan bahasa pemrograman C# dan menggunakan *database* MySQL sebagai penyimpan kamus kata. Pada sistem ini terdapat 4 tampilan halaman, yaitu Halaman Utama, Halaman Menu, Halaman Update kata dan Halaman Tentang.

4.1.1. Halaman Utama

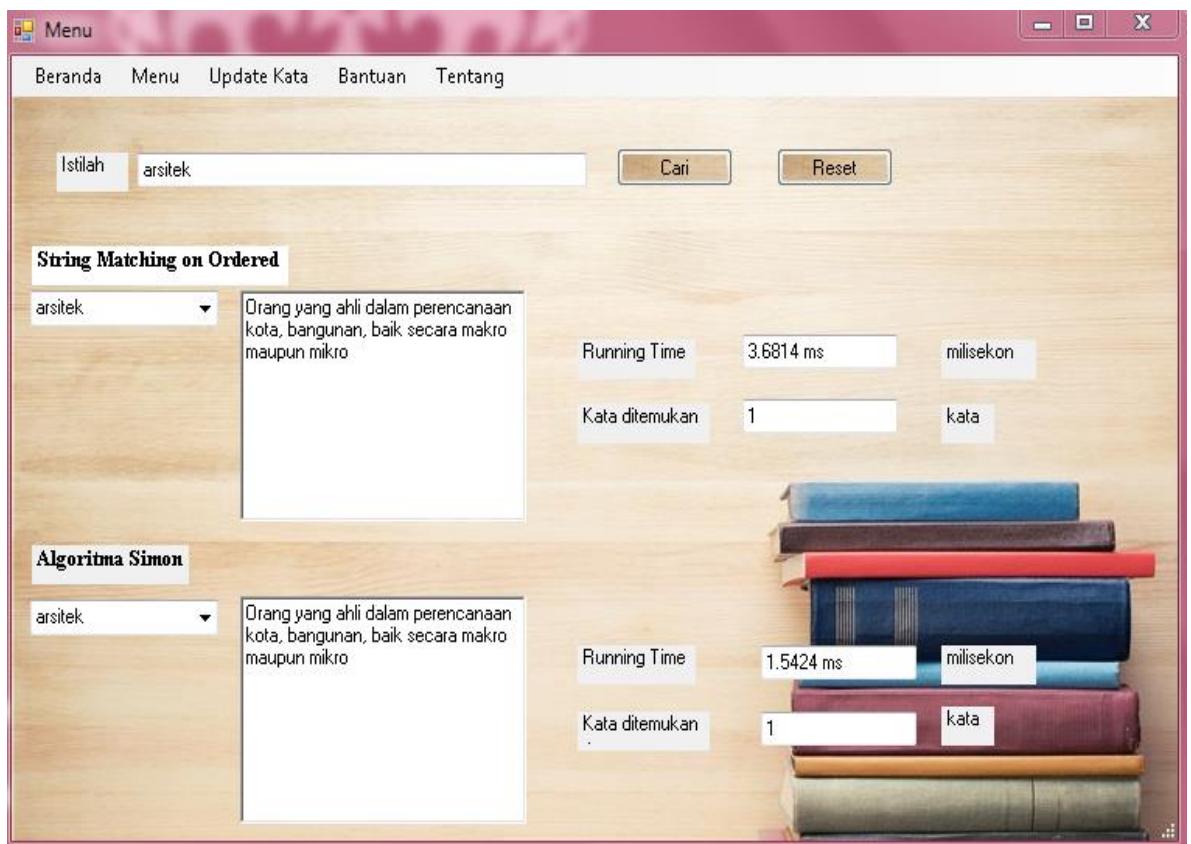
Ketika kita menjalankan program maka halaman utama yang akan muncul halaman utama merupakan halaman yang pertama kali muncul ketika sistem dibuka. Halaman Utama dapat dilihat pada Gambar 4.1. berikut.



Gambar 4.1. Halaman Utama

4.1.2. Halaman Menu

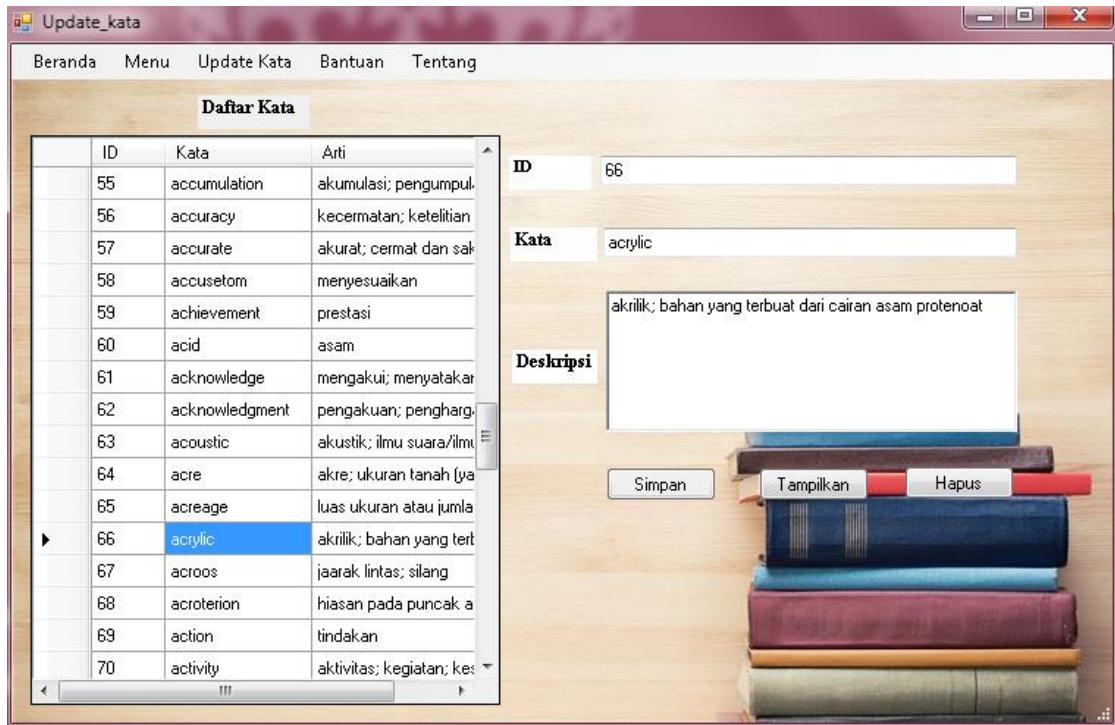
Pada halaman Menu adalah halaman kamus di dalam sistem. Di dalam halaman ini sistem akan melakukan pencarian kata dari *pattern* yang telah dimasukkan oleh pengguna lalu mencarinya dengan kedua algoritma pencocokan *string*. Halaman Menu dapat dilihat pada Gambar 4.2. berikut.



Gambar 4.2. Halaman Menu

4.1.3 Halaman Update Kata

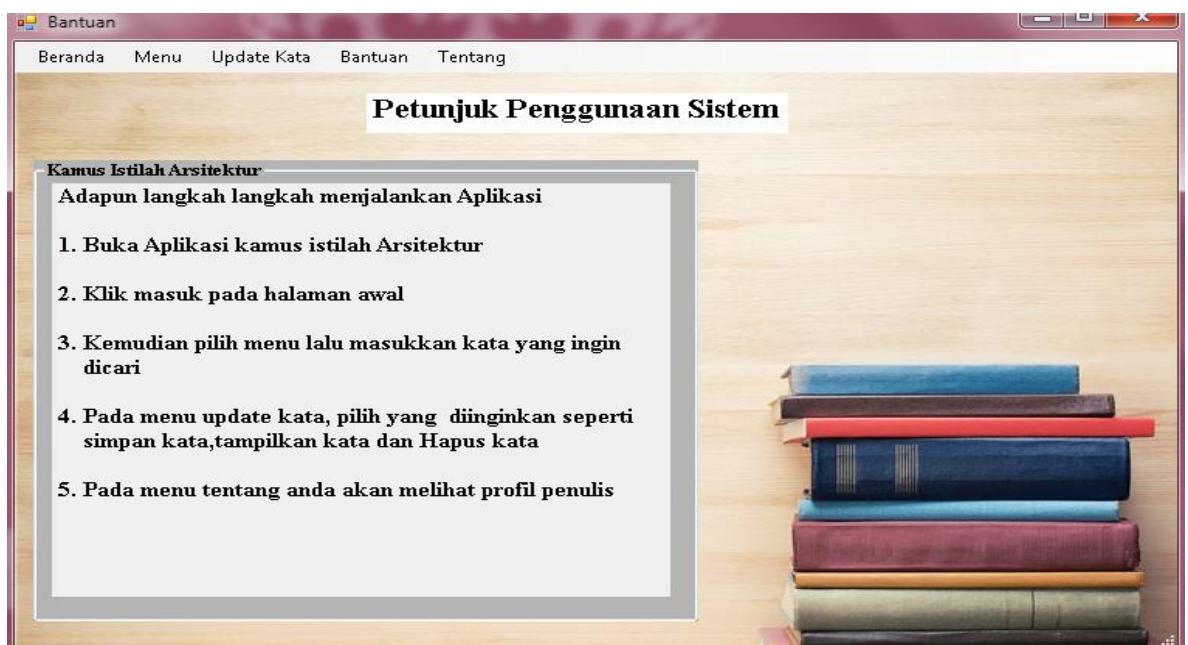
Pada Halaman *update* kata merupakan halaman yang berfungsi untuk menambahkan istilah baru, menghapus istilah pada *database* dan melihat istilah yang ada dalam *database*. Halaman ini dapat dilihat pada Gambar 4.3. berikut.



Gambar 4.3. Halaman *Update Kata*

4.1.4 Halaman Bantuan

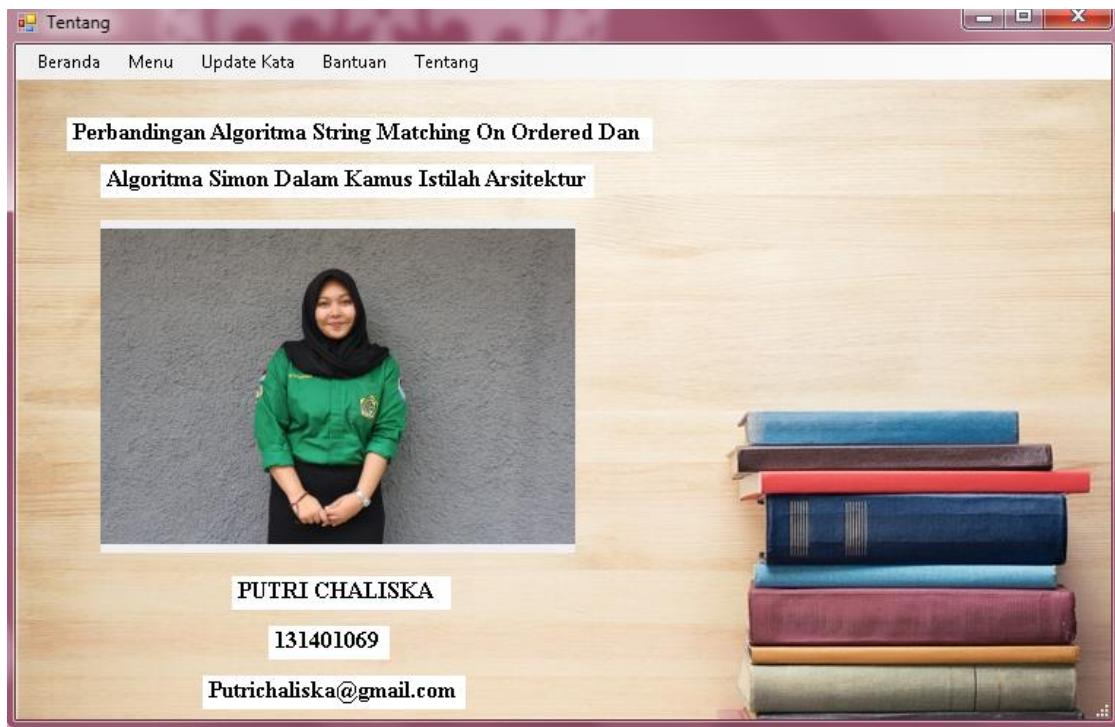
Pada Halaman bantuan merupakan halaman yang menampilkan cara penggunaan sistem kamus istilah Arsitektur . Halaman ini dapat dilihat pada Gambar 4.4. berikut.



Gambar 4.4. Halaman Bantuan

4.1.5 Halaman Tentang

Pada halaman Tentang berisi tentang pembuat sistem beserta fotonya. Halaman ini dapat dilihat pada Gambar 4.4. berikut.



Gambar 4.5. Halaman Tentang

4.2 Pengujian Sistem

Pengujian sistem dilakukan Setelah implementasi sistem selesai dilakukan. Hal ini dilakukan untuk mengetahui apakah sistem dapat berjalan dengan baik dan sesuai dengan perancangan sebelumnya.

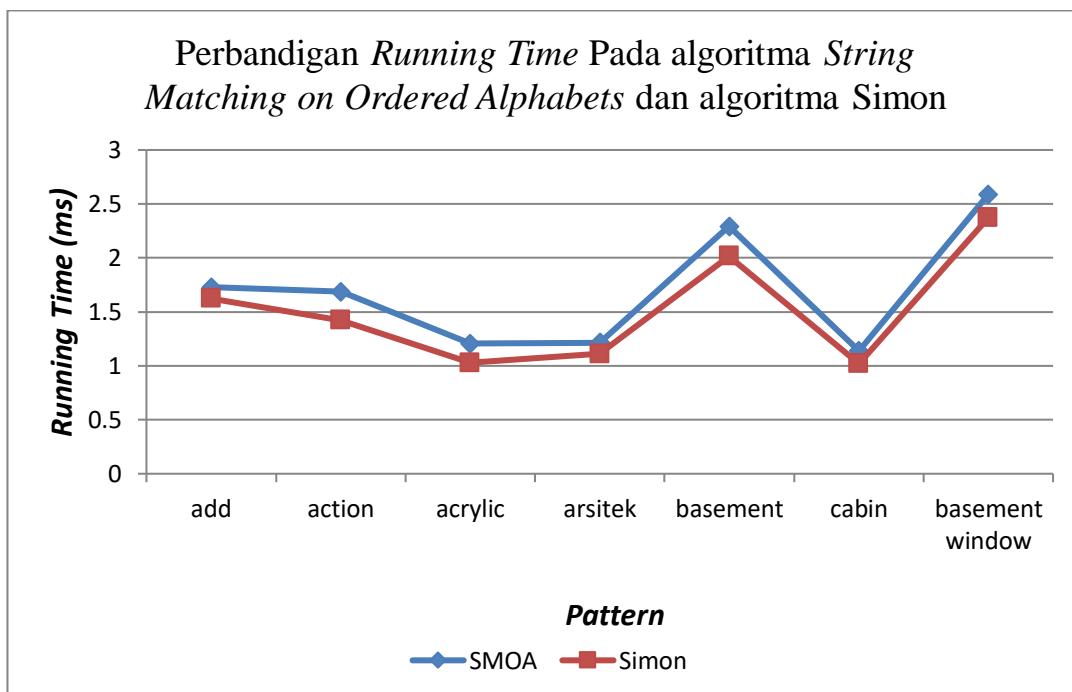
4.2.1. Pengujian Algoritma *String Matching on Ordered Alphabets* dan Simon

Pengujian sistem dilakukan dengan memasukan kata yang ingin dicari kemudian dibandingakan waktu pencarinya. Hal ini dilakukan untuk mengetahui kinerja dari sistem apakah dapat berjalan dengan benar. Kecocokan *input*, *output* dan *running time* dapat diihat pada tabel 4.1 berikut ini.

Tabel 4.1 Perbandingan waktu pencarian *pattern* pada sistem

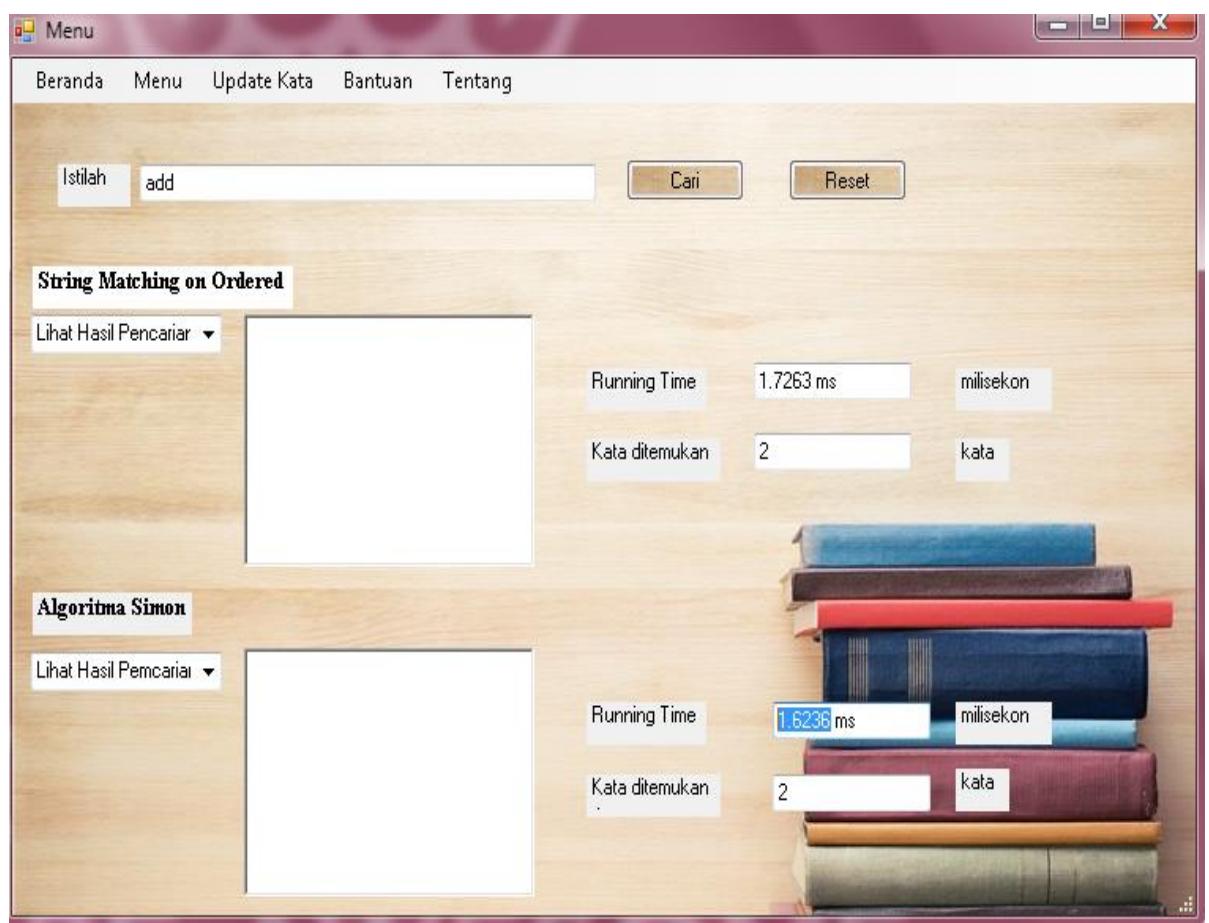
| No. | Pattern | Panjang <i>pattern</i> | Gambar hasil Pencarian | Waktu pencarian ms | |
|----------------------------------|--------------------|-----------------------------------|-----------------------------------|---------------------------|--------------|
| | | | | SMOA | Simon |
| 01. | Add | 3 | Gambar 4.2.6 | 1.7263 | 1.6236 |
| 02. | Action | 6 | Gambar 4.2.7 | 1.6861 | 1.4227 |
| 03. | Acrylic | 7 | Gambar 4.2.8 | 1.2054 | 1.0271 |
| 04. | Arsitek | 7 | Gambar 4.2.12 | 1.2148 | 1.1105 |
| 05. | Basement | 8 | Gambar 4.2.9 | 2.2916 | 2.0149 |
| 06. | Cabin | 5 | Gambar 4.2.10 | 1.1383 | 1.0194 |
| 07. | Basement window | 15 | Gambar 4.2.11 | 2.5875 | 2.3716 |
| Rata-rata waktu pencarian | | | | 1.6928 | 1.5134 |

Selanjutnya, perbandingan *running time* dimasukkan dalam bentuk grafik dapat dilihat pada Gambar 4.13 berikut.

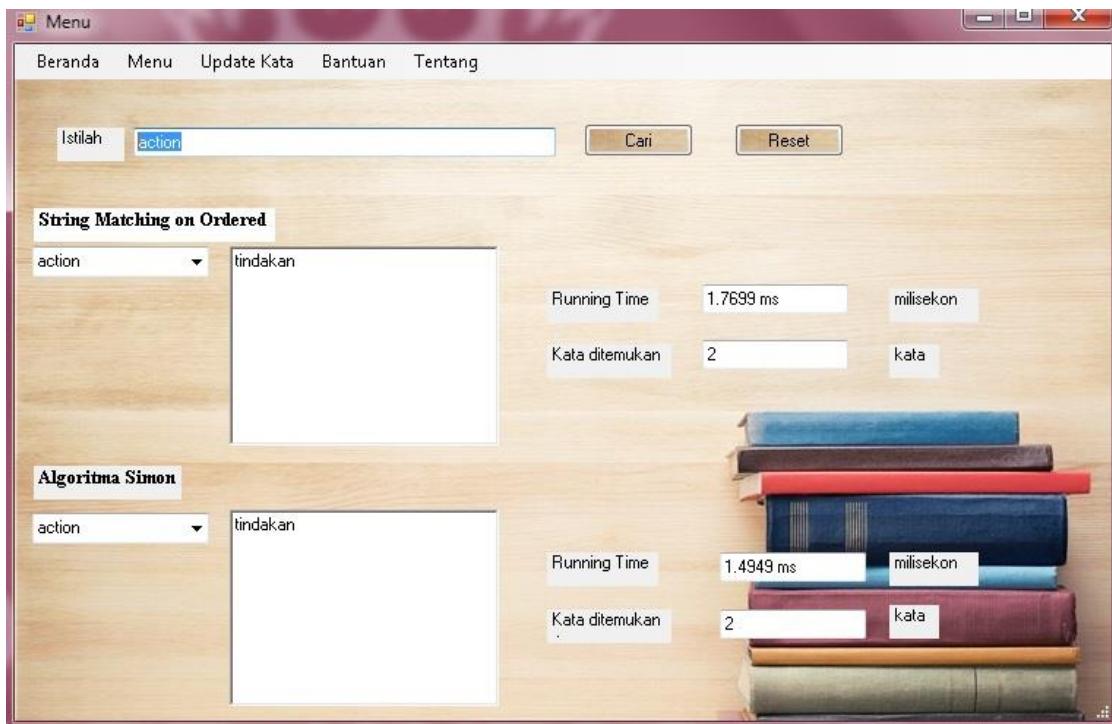
**Gambar 4.13. Perbandingan *Running Time* Algoritma SMOA dan Algoritma Simon**

Dari grafik diatas dapat dijelaskan bahwa Algoritma Simon mendapatkan Hasil *Running Time* yang relatif rendah dibandingkan dengan Algoritma *String Matching on Ordered Alphabets*. Artinya bahwa Algoritma Simon lebih cepat untuk menemukan pencocokan kata dibandingkan dengan Algoritma *String Matching on Ordered Alphabets*.

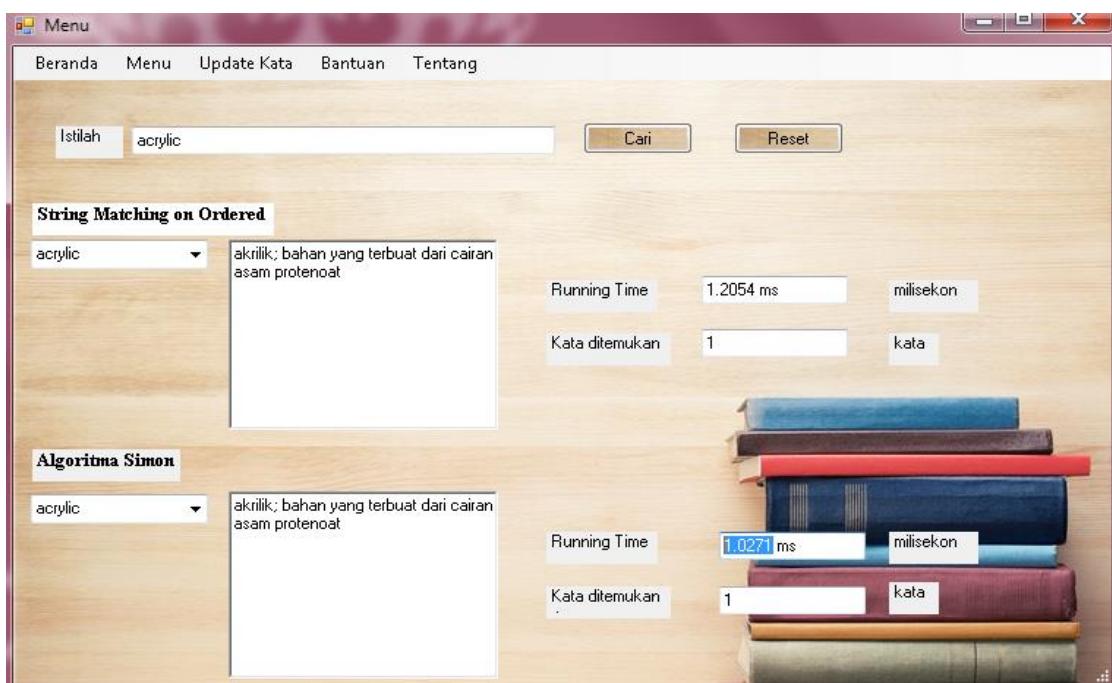
Gambar hasil pencarian berdasarkan tabel 4.1. dapat dilihat pada gambar 4.6 sampai dengan 4.12 berikut.



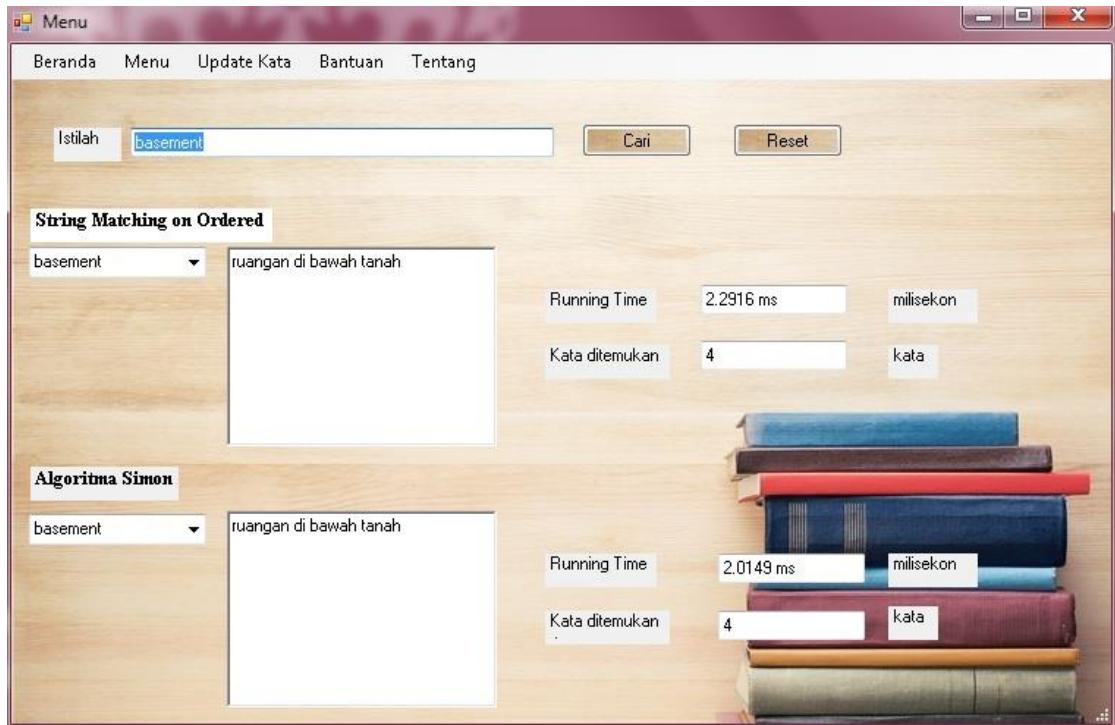
Gambar 4.6. Hasil pencarian pattern “Add”



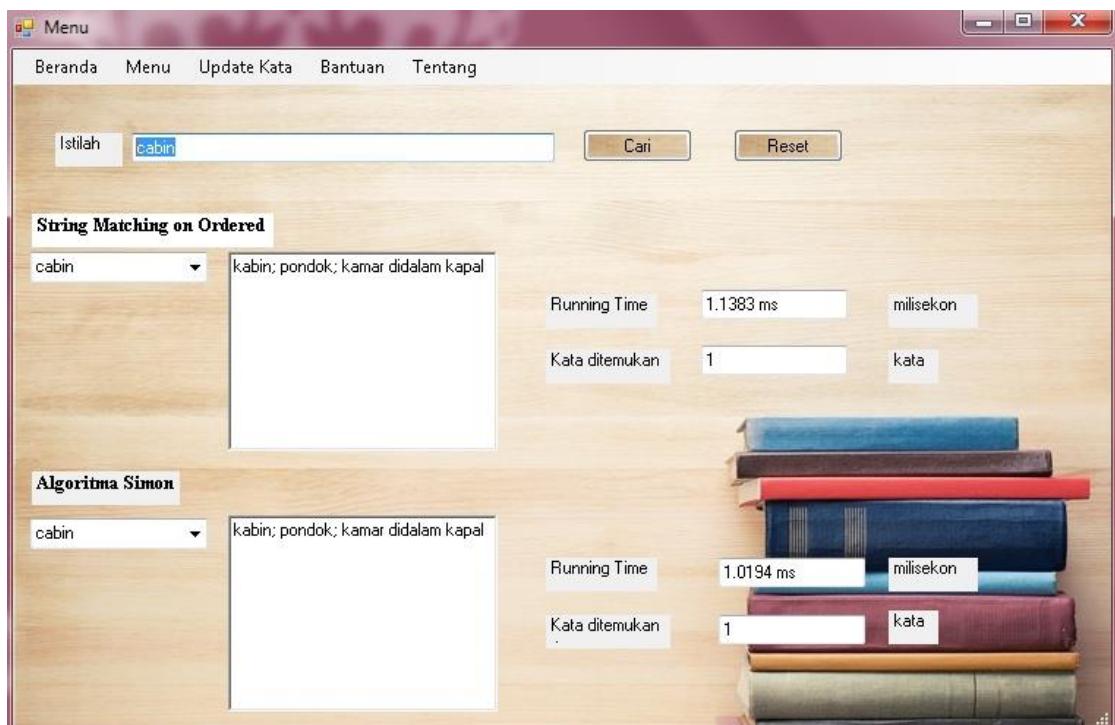
Gambar 4.7. Hasil pencarian pattern “Action”



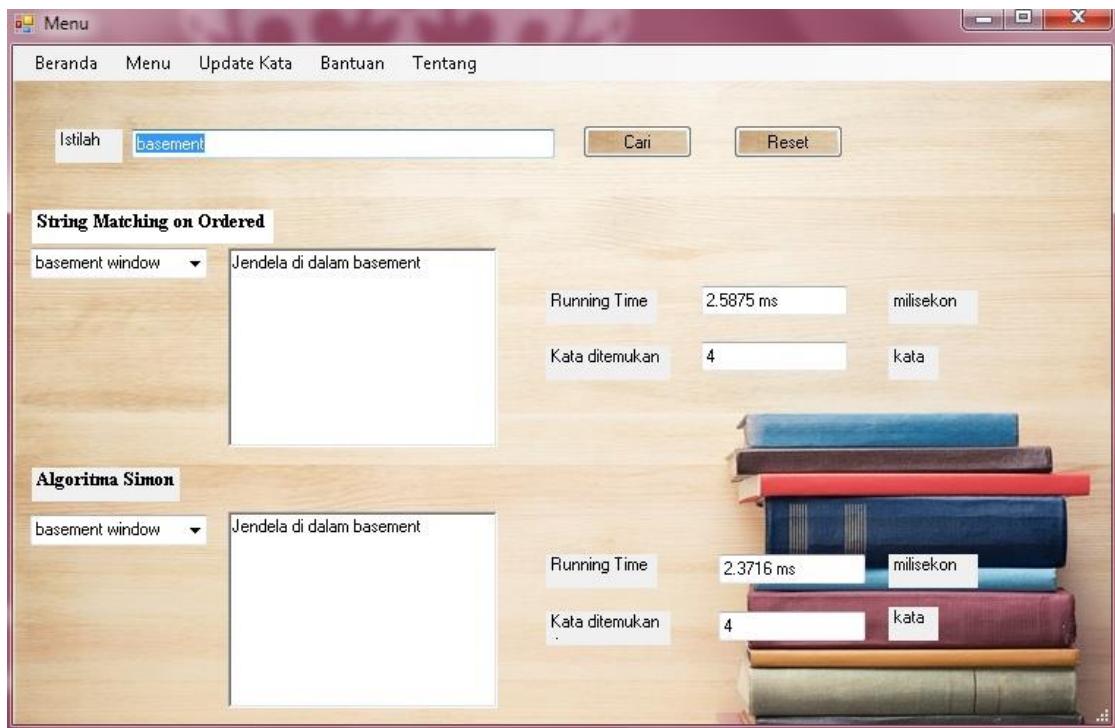
Gambar 4.8. Hasil pencarian pattern “Acrylic”



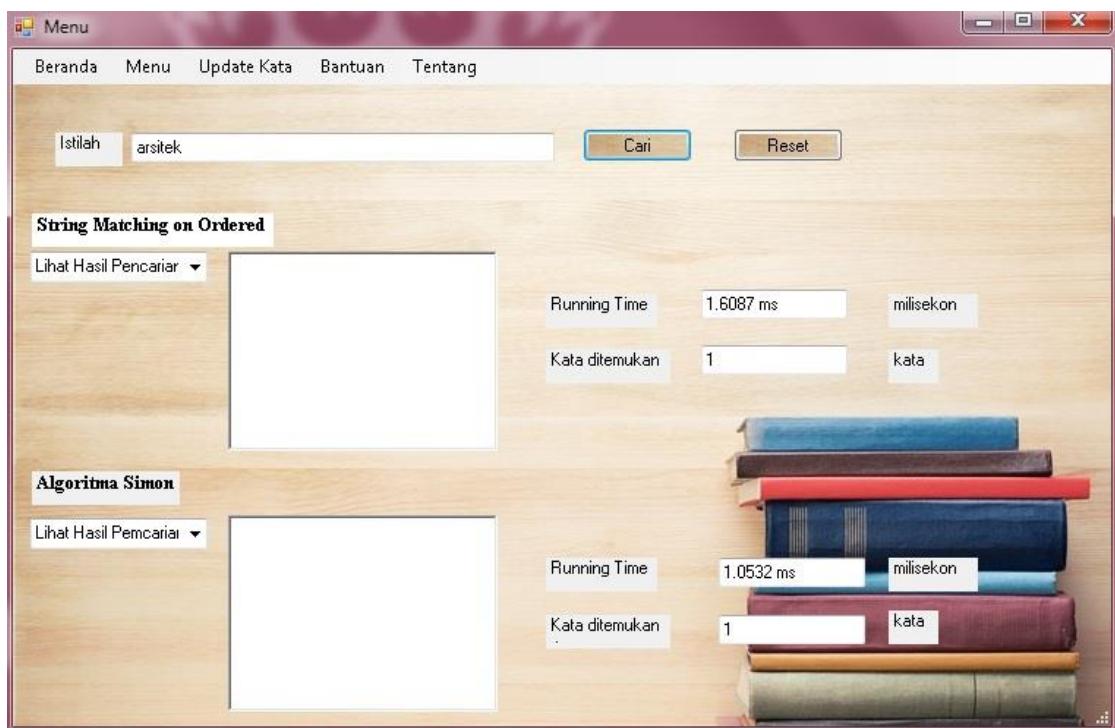
Gambar 4.9. Hasil pencarian *pattren* “Basement”



Gambar 4.10. Hasil pencarian *pattren* “Cabin”



Gambar 4.11. Hasil pencarian pattern “Basement Window”



Gambar 4.12. Hasil pencarian pattern “Arsitek”

4.3 Kompleksitas Algoritma

Pada penelitian ini, kompleksitas yang akan diuji adalah kompleksitas algoritma *String Matching on Ordered Alphabets* dan algoritma Simon pencarian dengan menggunakan notasi Big- Θ .

Tabel 4.2 Kompleksitas Fungsi SMOA

| Kode Program | C | # | C# |
|--|-----|---|-----|
| void nextMaximalSuffix (string x, int m, int i, int j, int k, int p){ | C1 | 1 | C1 |
| char a, b; | C2 | 1 | C2 |
| while (j+k < m){ | C3 | n | C3n |
| a=x[i+k]; | C4 | n | C4n |
| b=x[i+k]; | C4 | n | C4n |
| if (a==b){ | C5 | n | C5n |
| if(k==p){ | C5 | n | C5n |
| j+=p; | C6 | n | C6n |
| k=1; | C6 | n | C6n |
| } | | | |
| Else | | | |
| ++k; | C7 | 1 | C7 |
| } | | | |
| else if (a >b){ | C8 | 1 | C8 |
| j+=k; | C9 | 1 | C9 |
| k= 1; | C6 | 1 | C6 |
| p =j-1; | C10 | 1 | C10 |
| } | | | |
| else { | | | |
| i=j; | C11 | 1 | C11 |

| | | | |
|---------------------|-----|---|-----|
| <code>++j;</code> | C12 | 1 | C12 |
| <code>K=p=1;</code> | C13 | 1 | C13 |
| <code>}</code> | | | |
| <code>}</code> | | | |
| <code>}</code> | | | |

Kompleksitas fungsi *String matching on ordered alphabets*:

$$\begin{aligned}
 T(n) &= c_1 + c_2 + c_3n + c_4n + c_4n + c_5n + c_5n + c_6n + c_6n + c_7 + c_8 + c_9 + c_6 + c_{10} + c_{11} + c_{12} + c_{13} \\
 &= c_1 + c_2 + c_6 + c_7 + c_8 + c_9 + c_{10} + c_{11} + c_{12} + c_{13} + (c_3 + 2c_4 + 2c_5 + 2c_6)n \\
 &= \theta(n)
 \end{aligned}$$

Tabel 4.3 Kompleksitas Algoritma *String Matching on Ordered Alphabets*

| Kode Program | C | # | C# |
|--|----|-------|---------|
| <code>bool SMOA (string x, string y) {</code> | C1 | 1 | C1 |
| <code> int i, ip, j, jp, k, p;</code> | C2 | 1 | C2 |
| <code> ip=-1;</code> | C3 | 1 | C3 |
| <code> i=j=jp=0;</code> | C3 | 1 | C3 |
| <code> k=p=1;</code> | C3 | 1 | C3 |
| <code> while (j<=y.Length-x.Length) {</code> | C4 | n | C4n |
| <code> while(i+j<y.Length && i<x.Length && x[i] == y[i+j])</code> | C4 | n^2 | $C4n^2$ |
| <code> ++i;</code> | C5 | n^2 | $C5n^2$ |
| <code> if(i==0){</code> | C6 | 1 | C6n |
| <code> ++j;</code> | C5 | 1 | C5 |
| <code> ip=-1;</code> | C3 | 1 | C3 |
| <code> jp=0;</code> | C3 | 1 | C3 |
| <code> k=p=1</code> | C3 | 1 | C3 |
| <code>}</code> | | | |

| | | | |
|---|-----|-------|----------|
| else{ | | | |
| if (i>=x.Length) | C7 | 1 | C7 |
| return true; | C8 | 1 | C8 |
| nextMaximalSuffix(y+j, i+1, ip, jp, k, p); | C9 | 1 | C9 |
| If (ip< 0 ip<p && Equals((y+j).Substring(0, ip+1), (y+j+p).Substring(0, ip+1))){ | C10 | n^2 | $C10n^2$ |
| j+=p; | C11 | n^2 | $C11n^2$ |
| i-=p; | C11 | n^2 | $C11n^2$ |
| if (i<0) | C12 | n^2 | $C12n^2$ |
| i=0; | C13 | n^2 | $C13n^2$ |
| if (jp-ip>p) | C14 | n^2 | $C14n^2$ |
| jp-=p; | C15 | n^2 | $C15n^2$ |
| else{ | | | |
| ip=-1; | C3 | n^2 | $C3n^2$ |
| jp=0; | C3 | n^2 | $C3n^2$ |
| k=p=1; | C16 | n^2 | $C16n^2$ |
| } | | | |
| } | | | |
| else { | | | |
| j+=(Math.Max(ip+1, Math.Min(i-ip-1, jp+1))+1); | C17 | n^2 | $C17n^2$ |
| i=jp=0; | C18 | n^2 | $C18n^2$ |
| ip=-1; | C3 | n^2 | $C3n^2$ |
| k=p=1; | C16 | n^2 | $C16n^2$ |
| } | | | |
| } | | | |
| } | | | |

| | | | |
|---------------|----|---|----|
| Return false; | C8 | 1 | C8 |
| } | | | |

Kompleksitas pencarian algoritma *String Matching on Ordered Alphabets*

$$\begin{aligned}
 T(n) &= c_1 + c_2 + c_3 + c_3 + c_3 + c_4n + c_4n^2 + c_5n^2 + c_6n + c_5 + c_3 + c_3 + c_3 + c_7 + c_8 + c_9 + c_{10}n^2 \\
 &\quad + c_{11}n^2 + c_{11}n^2 + c_{12}n^2 + c_{13}n^2 + c_{14}n^2 + c_{15}n^2 + c_{3n^2} + c_{3n^2} + c_{16}n^2 \\
 &\quad + c_{17}n^2 + c_{18}n^2 + c_{3n^2} + c_{16}n^2 + c_8 \\
 &= c_1 + c_2 + 6c_3 + c_5 + c_7 + 2c_8 + c_9 + (c_4 + c_6)n + (3c_3 + c_4 + c_5 + c_{10} + 2c_{11} + c_{12} + c_{13} + \\
 &\quad c_{14} + c_{15} + 2c_{16} + c_{17} + c_{18})n^2 \\
 &= \theta(n)^2
 \end{aligned}$$

Tabel 4.4 Kompleksitas Algoritma Simon

| Kode Program | C | # | C# |
|-----------------------------------|----|---|-----|
| bool simon (string x, string y) { | C1 | 1 | C1 |
| int j,m; | C2 | 1 | C2 |
| j=0; | C3 | 1 | C3 |
| m=-1; | C3 | 1 | C3 |
| do { | | | |
| if(m+1 <=x.Length-1) { | C4 | n | C4n |
| if(x[m+1] == y[j]) | C5 | n | C5n |
| m++; | C6 | n | C6n |
| Else | | | |
| M=-1 | C3 | n | C3n |
| } | | | |
| if(m ==x.Length-1) { | C4 | n | C4n |

| | | | |
|------------------------|----|---|-----|
| return true; | C7 | n | C7n |
| m=-1; | C3 | n | C3n |
| } | | | |
| j++; | C8 | n | C8n |
| } | | | |
| while (j<=y.Length-1); | C9 | n | C9n |
| return false; | C7 | 1 | C7 |
| } | | | |

Kompleksitas fase pencarian algoritma Simon :

$$\begin{aligned}
 T(n) &= c_1 + c_2 + c_3 + c_3 + c_4 n + c_5 n + c_6 n + c_3 n + c_4 n + c_7 n + c_3 n + c_8 n + c_9 n + c_7 \\
 &= c_1 + c_2 + (2c_3 + c_7) + (2c_3 + 2c_4 + c_5 + c_6 + c_7 + c_8 + c_9)n \\
 &= \theta(n)
 \end{aligned}$$

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan dan Saran

Berdasarkan analisis, perancangan dan pengujian pada aplikasi kamus istilah Arsitektur dengan menggunakan algoritma *String Matching on Ordered Alphabets* dan Algoritma Simon, maka diperoleh beberapa kesimpulan :

1. Berdasarkan penelitian yang telah dilakukan, algoritma *String Matching on Ordered Alphabet* dan algoritma Simons berhasil diterapkan dalam aplikasi Kamus istilah Arsitektur.
2. Total rata-rata *running time* untuk algoritma *String Matching on Ordered Alphabet* adalah $1.6928ms$ dan algortima Simon adalah 1.5134 . Pada skenario pencarian kata terpanjang, algoritma *String Matching on Ordered Alphabet* memiliki rata-rata *running time* selama $2.5875ms$ dan algoritma Simon selama $2.3716ms$ dan untuk pencarian kata terpendek, algoritma *String Matching on Ordered Alphabet* memiliki *running time* selama $1.7263ms$ dan algoritma Simon selama $1.6236ms$. Hal ini membuktikan bahwa algoritma *String Matching on Ordered Alphabet* melakukan pencarian *string* lebih lama dibandingkan algoritma Simon, namun perbedaan *running time* tidak terlalu signifikan.
3. Algortima Simon lebih cepat dalam pencarian *string* dikarenakan dikarenakan algoritma Simon fase pencariannya dilakukan dari kiri ke kanan dengan tahapan inisialisasi tiap indeks pada pola yang telah diberikan

5.2.Saran

Adapun saran yang dapat penulis berikan untuk pengembangan penelitian selanjutnya, yaitu sebagai berikut:

1. Untuk pengembangan selanjutnya, diharapkan dapat mengembangkan sistem ke dalam basis *mobile* sehingga tidak hanya berbasis *desktop* saja.
2. Untuk pengembangan sistem selanjutnya diharapkan jumlah istilah Arsitektur yang ada dalam *database* dapat ditambahkan untuk semakin menambah pengetahuan dalam bidang Arsitektur.
3. Untuk pengembangan selanjutnya diharapkan pencarian kata dapat ditemukan dengan audio(suara).

DAFTAR PUSTAKA

- Charras, C. & Lecroq, T. 2004. *Handbook of Exact String-Matching Algorithms*. London: King's College Publications.
- Charras, C. & Lecroq, T. 1997. *Handbook of Exact String – Matching Algorithms*. Prancis
- Habibie, I. 2016. *Analisis dan Perbandingan Algoritma Colussi dan Algoritma Simon dalam Pencarian Kata pada Jurnal*. Skripsi. Universitas Sumatera Utara.
- Jogiyanto. 2005. Pengenalan Komputer. Yogyakarta: Andi.
- Kamara, G. H. 2008. *Visualisasi Beberapa Algoritma Pencocokan String Dengan Java*. Jurnal. Bandung: Institut Teknologi Bandung
- Levitin, A. 2012. *Introduction to The Design and Analysis of Algorithms: 3rd Edition*. Pearson: Villanova University
- Megawaty. 2017. *Implementasi dan Perbandingan Algoritma Brute Force dengan Algoritma Simon dalam Pembuatan Kamus Istilah Kebidanan*. Skripsi. Universitas Sumatera Utara
- Siregar, N. 2017. *Perbandingan Algoritma Turbo Boyer Moore dan Algoritma String Matching on Ordered Alphabets untuk Aplikasi Kamus Fisika Berbahasa Indonesia Berbasis Android*.
- Rasool, A., Tiwari, A., Singla, G., Khare, N. String Matching Methodologies:A Comparative Analysis *International Journal of Computer Science and Information Technologies*,3 (2) : 2012,3394 – 3397
- Sabrina, N. D. 2016. Perbandingan algoritma *string matching quick search* dengan algoritma Berry-Ravindran pada aplikasi kamus Indonesia-Prancis berbasis web. Skripsi. Universitas Sumatera Utara.
- Singla, Nimisha dan Garg, Deepak. 2012. String Matching Algorithms and their Applicability in various Applications *International Journal of Soft Computing and Engineering (IJSCE)* ISSN: 2231-2307, Volume-I, Issue-6,

LAMPIRAN**LISTING PROGRAM****1.MainForm.cs**

```
namespace Arsitektur
{
    /// <summary>
    /// Description of MainForm.
    /// </summary>
    public partial class MainForm : Form
    {
        public MainForm()
        {

        }

        void Button1Click(object sender, EventArgs e)
        {
            this.Hide();
            Menu tampil = new Menu();
            tampil.ShowDialog();
            this.Close();
        }

        void Label1Click(object sender, EventArgs e)
        {

        }

        void MainFormLoad(object sender, EventArgs e)
        {

        }
    }
}
```

2.Menu.cs

```

namespace Arsitektur
{
    /// <summarysummary>
    public partial class Menu : Form
    {
        public static List<string>[] lists
        = new List<string>[2];
        public Menu()
        {

            //
            // The InitializeComponent() call is required for
            Windows Forms designer support.
            //
            InitializeComponent();
            lists[0] = new List< string >();
            lists[1] = new List< string >();
            hasilsmoacombo.Text = "Lihat Hasil Pencarian";
            hasilsimoncombo.Text = "Lihat Hasil Pemcarian";

            //
            // TODO: Add constructor code after the
            InitializeComponent() call.
            //
        }

        void Button1Click(object sender, EventArgs e)
        {
            hasilsmoacombo.Items.Clear();
            hasilsimoncombo.Items.Clear();
            lists[0].Clear();
            lists[1].Clear();

            MySqlConnection connect = new MySqlConnection("server=localhost;
database=arsitektur; uid=root; password=;");
            connect.Open();
            MySqlCommand command = new MySqlCommand("select *

```

```

from kamus", connect);
    MySqlDataReader reader = command.ExecuteReader();
    Stopwatch a = new Stopwatch();
    Stopwatch b = new Stopwatch();
    a.Start();

    //int[,] brbc = new int[ASIZE,ASIZE];
    //preBrBc(input.Text.ToLower(), input.Text.ToLower().Length, ref
    brbc);

    while (reader.Read()) {
        if (SMOA(input.Text.ToLower(), reader[1].ToString().ToLower())) {
            hasilsmoacombo.Items.Add(reader[1].ToString());
            lists[0].Add(reader[2].ToString());
        }
    }

    a.Stop();
    waktusmoa.Text
    = (int.Parse(a.Elapsed.ToString("fffffff"))/10000f).ToString() +
    " ms ";
    reader.Close();
    reader.Dispose();
    reader = command.ExecuteReader();
    b.Start();

        while (reader.Read()) {
            if(simon(input.Text.ToLower(), reader[1].ToString().ToLower())){
                hasilsimoncombo.Items.Add(reader[1].ToString());
                lists[1].Add(reader[2].ToString());
            }
        }

    b.Stop();
    waktusimon.Text
    = (int.Parse(b.Elapsed.ToString("fffffff"))/10000f).ToString() +
    " ms ";
    reader.Close();
    reader.Dispose();
    connect.Close();

    katasmao.Text =
    hasilsmoacombo.Items.Count.ToString();
    katasimon.Text =

```

```

hasilsimoncombo.Items.Count.ToString();
}

void nextMaximalSuffix(string x, int m, int i, int j, int k, int p) {
    char a, b;
    while (j+k < m) {
        a=x[i+k];
        b=x[i+k];
        if (a==b) {
            if (k==p) {
                j+=p;
                k=1;
            }
            else
                ++k;
        }
        else if (a >b) {
            j+=k;
            k = 1;
            p = j-1;
        }
        else {
            i=j;
            ++j;
            k=p=1;
        }
    }
}

bool SMOA (string x, string y){
    int i, ip, j, jp, k, p;
    ip=-1;
    i=j=jp=0;
    k=p=1;
    while (j<=y.Length-x.Length) {
while(i+j<y.Length && i<x.Length && x[i] == y[i+j])
    ++i;
    if (i==0) {
        ++j;
        ip = -1;
        jp = 0;
    }
}
}

```

```

        k=p=1;
    }
    else {
        if (i>= x.Length)
            return true;
nextMaximalSuffix(y+j, i+1, ip, jp, k, p);
if (ip< 0 || ip<p && Equals((y+j).Substring(0, ip+1), (y+j+p).Sub
string(0, ip+1))) {
            j+=p;
            i-=p;
            if (i<0)
                i=0;
            if (jp-ip>p)
                jp-=p;
            else{
                ip=-1;
                jp=0;
                k=p=1;
            }
        }
        else {
            j+=(Math.Max(ip+1, Math.Min(i-ip-1, jp+1))+1);
            i=jp=0;
            ip=-1;
            k=p=1;
        }
    }
    return false;
}

bool simon (string x, string y) {
    int j,m;
    j=0;
    m=-1;
    do {
        if (m+1 <=x.Length-1) {
            if (x[m+1] == y[j])
                m++;
            else
                m=-1;
        }
        if (m ==x.Length-1) {

```

```

                return true;
                m=-1;
            }
            j++;
        }
    while (j<=y.Length-1);
    return false;
}

void HasilsimoncomboSelectedIndexChanged(object sender, EventArgs e)
{
    artiSimon.Text =
lists[1][hasilsimoncombo.SelectedIndex];
}
void HasilsmoacomboSelectedIndexChanged(object sender, EventArgs e)
{
    artismoa.Text =
lists[1][hasilsmoacombo.SelectedIndex];
}
void BerandaToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    MainForm tampil = new MainForm();
    tampil.ShowDialog();
    this.Close();
}
void UpdateKataToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Update_kata tampil1 = new Update_kata();
    tampil1.ShowDialog();
    this.Close();
}
void BantuanToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Bantuan tampil2 = new Bantuan();
    tampil2.ShowDialog();
    this.Close();
}
void TentangToolStripMenuItemClick(object sender, EventArgs e)

```

```

        {
            this.Hide();
            Tentang tampil3 = new Tentang();
            tampil3.ShowDialog();
            this.Close();
        }

        void Button2Click(object sender, System.EventArgs e)
        {
            hasilsmoacombo.Text = "Lihat Hasil Pencarian";
            hasilsimoncombo.Text = "Lihat Hasil Pemcarian";
            input.Text = "";
            artismoa.Text = "";
            artiSimon.Text = "";
            waktusimon.Text = "";
            waktusmoa.Text = "";
            katasimon.Text = "";
            katasmao.Text = "";
            hasilsmoacombo.Items.Clear();
            hasilsimoncombo.Items.Clear();
            lists[0].Clear();
            lists[1].Clear();
        }
    }
}

```

3.UpdateKata.cs

```

namespace Arsitektur
{
    /// <summary>
    /// Description of Update_kata.
    /// </summary>
    public partial class Update_kata : Form
    {
        public Update_kata()
        {
        }
    }

    void BerandaToolStripMenuItemClick(object sender, EventArgs e)
    {
        this.Hide();
        MainForm tampil = new MainForm();
        tampil.ShowDialog();
        this.Close();
    }
}

```

```

}

void MenuToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Menu tampill1 = new Menu();
    tampill1.ShowDialog();
    this.Close();
}
void BantuanToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Bantuan tampil2 = new Bantuan();
    tampil2.ShowDialog();
    this.Close();
}
void TentangToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Tentang tampil3 = new Tentang();
    tampil3.ShowDialog();
    this.Close();
}

//TAMPILKAN DATA
void Button2Click(object sender, EventArgs e)
{
    MySqlConnection conn = koneksi.getkoneksi();
    MySqlCommand command = conn.CreateCommand();
    command.CommandText = "Select * from kamus";
    conn.Open();
    DataSet ds = new DataSet();
    MySqlDataAdapter da = new MySqlDataAdapter(command);
    da.Fill(ds, "kamus");
    dataGridView1.DataSource = ds;
    dataGridView1.DataMember = "kamus";
    dataGridView1.Columns[0].HeaderText = "ID";
    dataGridView1.Columns[1].HeaderText = "Kata";
    dataGridView1.Columns[2].HeaderText = "Arti";
    dataGridView1.Columns[0].Width = 50;
    dataGridView1.Columns[1].Width = 100;
    dataGridView1.Columns[2].Width = 150;
}

```

```

dataGridView1.RowsDefaultCellStyle.Alignment=DataGridViewContentAlignment.A
lignment.MiddleLeft;
    conn.Close();

}

//SIMPAN DATA
void Button1Click(object sender, EventArgs e)
{
    try{
        MySqlConnection conn = koneksi.getkoneksi();
        conn.Open();
        MySqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "insert into kamus (id, kata, arti) values
(@id, @kata, @arti)";
        cmd.Parameters.AddWithValue("@id", tb_idkata.Text);
        cmd.Parameters.AddWithValue("@kata", tb_namakata.Text);
        cmd.Parameters.AddWithValue("@arti", tb_artikata.Text);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Kata berhasil ditambahkan");
        conn.Close();
    }
    catch {
        MessageBox.Show("ID sudah ada, masukkan ID yang lain");
    }
}

//HAPUSDATA
void Button3Click(object sender, EventArgs e)
{
    MySqlConnection conn = koneksi.getkoneksi();
    conn.Open();
    MySqlCommand cmd = conn.CreateCommand();
    cmd.Parameters.AddWithValue("@id", tb_idkata.Text);
    cmd.CommandText = "delete from kamus where id=
" + int.Parse(tb_idkata.Text) + "";
    cmd.ExecuteNonQuery();
    MessageBox.Show("Data berhasil dihapus");
    conn.Close();
}

void DataGridView1CellContentClick(object sender, DataGridViewCellEventArgs e)

```

```

        {
            tb_idkata.Text =
dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
            tb_namakata.Text =
dataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString();
            tb_artikata.Text =
dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();
        }
    }
}

```

4.Bantuan.cs

```

namespace Arsitektur
{
    public partial class Bantuan : Form
    {
        public Bantuan()
        {

    }

void BerandaToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    MainForm tampil3 = new MainForm();
    tampil3.ShowDialog();
    this.Close();
}

void MenuItemToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Menu tampil1 = new Menu();
    tampil1.ShowDialog();
    this.Close();
}

void UpdateKataToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
}

```

```
        Update_kata tampil2 = new Update_kata();
        tampil2.ShowDialog();
        this.Close();
    }

void TentangToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Tentang tampil3 = new Tentang();
    tampil3.ShowDialog();
    this.Close();
}

void Label2Click(object sender, EventArgs e)
{
}

void GroupBox1Enter(object sender, EventArgs e)
{
}

}
```

5.Tentang.cs

```
namespace Arsitektur
{
    /// <summary>
    /// Description of Tentang.
    /// </summary>
    public partial class Tentang : Form
    {
        public Tentang()
        {

        }

        void BerandaToolStripMenuItemClick(object sender, EventArgs e)
        {
            this.Hide();
            MainForm tampil = new MainForm();
            tampil.ShowDialog();
            this.Close();
        }
    }
}
```

```
}

void MenuStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Menu tampil2 = new Menu();
    tampil2.ShowDialog();
    this.Close();
}

void UpdateKataToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Update_kata tampill = new Update_kata();
    tampill.ShowDialog();
    this.Close();
}

void BantuanToolStripMenuItemClick(object sender, EventArgs e)
{
    this.Hide();
    Bantuan tampil3 = new Bantuan();
    tampil3.ShowDialog();
    this.Close();
}

void PictureBox1Click(object sender, EventArgs e)
{
}

void Label1Click(object sender, EventArgs e)
{
}

void Label3Click(object sender, EventArgs e)
{
}

}
```

DAFTAR RIWAYAT HIDUP

CURRICULUM VITAE



DATA PRIBADI

Nama Lengkap : Putri Chaliska
Tempat / Tanggal Lahir : Medan, 06 September 1994
Jenis Kelamin : Perempuan
Agama : Islam
Kebangsaan : Indonesia
Alamat : Jln. Permai VI no.127 Komplek BTN tanjung Permai, Sukadono
Telepon : 0857-6343-0366
Moto Hidup : Hidup ini Pilihan , apapun itu Jalanin
Tinggi / Berat : 159 cm / 65 kg
Email : Putrichaliska@gmail.com

PENDIDIKAN FORMAL

- [2010 – 2013]
SMA KEMALA BHAYANGKARI 1 MEDAN - SUMATERA UTARA
- [2007 – 2010]
SMP NEGERI 7 MEDAN - SUMATERA UTARA
- [2001 – 2007]
SD IKAL MEDAN - SUMATERA UTARA
- [2000 – 2001]
TK ANNISA MEDAN – SUMATERA UTARA

KEMAMPUAN

Bahasa : Bahasa Indonesia

Pemrograman : C, C#

Database : MySQL

Lainnya : Ms. Office

SEMINAR

- Peserta Seminar Nasional Literasi Informasi (SENARAI) Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Medan [2014]
 - Peserta Seminar Nasional Industri Kreatif IT Fest, Universitas Sumatera Utara, Medan [2017]
-

PENGALAMAN ORGANISASI

- Anggota WASKOM IMILKOM USU [2016 – 2017]
 - Anggota KEWIRAUSAHAAN PEMA FASILKOM-TI
-

PENGALAMAN KEPANTIAAN

- Anggota Acara PMB ILKOM 2015
- Anggota Konsumsi PORSENI ILKOM 2014