

LAPORAN TUGAS BESAR 3
IF2211 STRATEGI ALGORITMA
PEMANFAATAN *PATTERN MATCHING* UNTUK MEMBANGUN SISTEM ATS
(*APPLICANT TRACKING SYSTEM*) BERBASIS CV DIGITAL



Disusun Oleh:

Kelompok 31 - Scoopy

Wardatul Khoiroh 13523001

Ranashahira Reztaputri 13523007

Diyah Susan Nugrahani 13523080

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG, 40132
2025

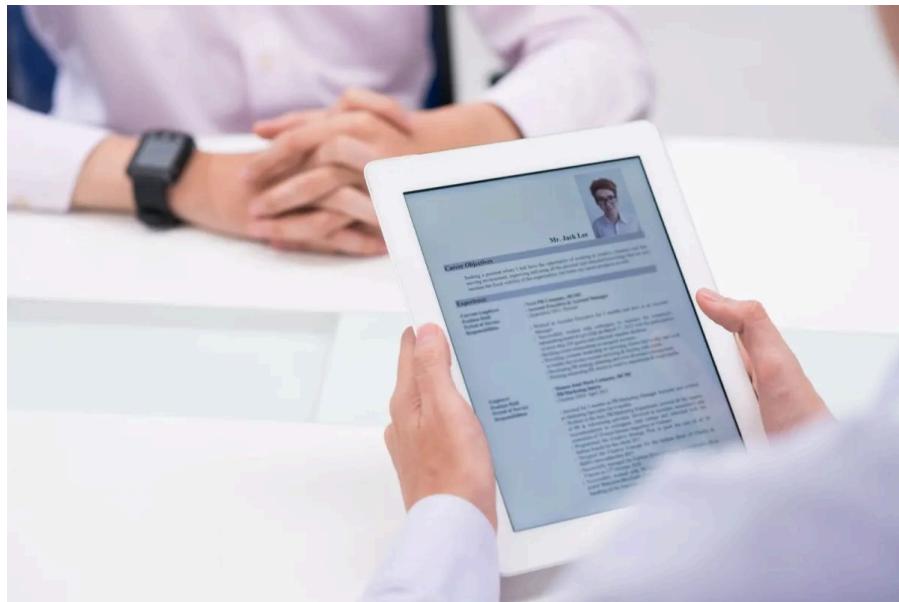
DAFTAR ISI

DESKRIPSI MASALAH.....	2
LANDASAN TEORI.....	6
2.1. Algoritma Knuth-Morris-Pratt (KMP).....	6
2.2. Algoritma Boyer-Moore.....	7
2.3. Algoritma Aho-Corasick.....	7
2.4. Pengembangan Interface GUI.....	8
ANALISIS PEMECAHAN MASALAH.....	10
3.1 Langkah-langkah Pemecahan Masalah.....	10
3.2 Proses Pemetaan Masalah.....	11
3.2.1 Pemetaan Masalah dengan KMP.....	11
3.2.2 Pemetaan Masalah dengan BM.....	12
3.2.3 Pemetaan Masalah dengan AC.....	14
3.3 Fitur Fungsional dan Arsitektur Aplikasi.....	14
3.3.1 Fitur Fungsional.....	14
3.3.2 Arsitektur Aplikasi.....	15
3.4 Contoh Ilustrasi Kasus.....	17
3.4.1 Ilustrasi Kasus KMP.....	17
3.4.2 Ilustrasi Kasus Boyer-Moore.....	18
3.4.3 Ilustrasi Kasus AC.....	19
IMPLEMENTASI DAN PENGUJIAN.....	21
4.1 Implementasi Database.....	21
4.2 Implementasi Program: Algoritma KMP.....	26
4.3 Implementasi Program: Algoritma BM.....	31
4.4 Implementasi Program: Algoritma AC.....	32
4.5 Implementasi GUI.....	36
4.6 Tata Cara Penggunaan Program.....	42
4.7 Hasil Pengujian dan Analisis.....	44
4.7.1 Algoritma Knuth-Morris-Pratt.....	44
4.7.1.1 Test Case 1: Exact Match - Single Keyword [“microsoft office”].....	44
4.7.1.2 Test Case 2.....	45
4.7.1.3 Test Case 3: Multiple Keywords - Exact Match [“science, technology”].....	46
4.7.1.4 Test Case 4: Multiple Keywords - Fuzzy Match [“reacx, hmml”].....	47
4.7.1.5 Test Case 5: Multiple Keywords - Exact & Fuzzy Match [“express, exxel”].....	48
4.7.1.6 Test Case 6: Random Keyword.....	49
4.7.2 Algoritma Boyer-Moore.....	50
4.7.2.1 Test Case 1: Exact Match - Single Keyword [“microsoft office”].....	50
4.7.2.2 Test Case 2: Fuzzy Match - Single Keyword [“programer”].....	50

4.7.2.3 Test Case 3: Multiple Keywords - Exact Match [“science, technology”].....	51
4.7.2.4 Test Case 4: Multiple Keywords - Fuzzy Match [“reacx, hmml”].....	52
4.7.2.5 Test Case 5: Multiple Keywords - Exact & Fuzzy Match [“express, exxel”].....	53
4.7.2.6 Test Case 6: Random Keyword.....	54
4.7.3 Algoritma Aho-Corasick.....	55
4.7.3.1 Test Case 1: Exact Match - Single Keyword [“microsoft office”].....	55
4.7.3.2 Test Case 2: Fuzzy Match - Single Keyword [“programer”].....	55
4.7.3.3 Test Case 3: Multiple Keywords - Exact Match [“science, technology”].....	56
4.7.3.4 Test Case 4: Multiple Keywords - Fuzzy Match [“reacx, hmml”].....	57
4.7.3.5 Test Case 5: Multiple Keywords - Exact & Fuzzy Match [“express, exxel”].....	58
4.7.3.6 Test Case 6: Random Keyword.....	59
4.7.4 Analisis Hasil Pengujian.....	60
KESIMPULAN DAN SARAN.....	61
5.1 Kesimpulan.....	61
5.2 Saran.....	61
LAMPIRAN.....	62
DAFTAR PUSTAKA.....	63

BAB I

DESKRIPSI MASALAH



Gambar 1. CV ATS dalam Dunia Kerja
(Sumber: <https://www.antaranews.com/>)

Di era digital ini, keamanan data dan akses menjadi semakin penting. Perkembangan proses rekrutmen tenaga kerja telah mengalami perubahan signifikan dengan memanfaatkan teknologi untuk meningkatkan efisiensi dan akurasi. Salah satu inovasi yang menjadi solusi utama adalah Applicant Tracking System (ATS), yang dirancang untuk mempermudah perusahaan dalam menyaring dan mencocokkan informasi kandidat dari berkas lamaran, khususnya Curriculum Vitae (CV). ATS memungkinkan perusahaan untuk mengelola ribuan dokumen lamaran secara otomatis dan memastikan kandidat yang relevan dapat ditemukan dengan cepat.

Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan riwayat pendidikan secara efisien. Pattern matching menjadi solusi ideal dalam menghadapi tantangan ini.

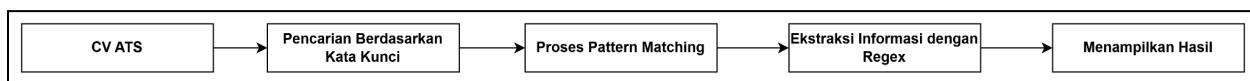
Pattern matching adalah teknik untuk menemukan dan mencocokkan pola tertentu dalam teks. Dalam konteks ini, algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) sering digunakan karena keduanya menawarkan efisiensi tinggi untuk pencarian teks di dokumen besar. Algoritma

ini memungkinkan sistem ATS untuk mengidentifikasi informasi penting dari CV pelamar dengan kecepatan dan akurasi yang optimal.

Di dalam Tugas Besar 3 ini, Anda diminta untuk mengimplementasikan sistem yang dapat melakukan deteksi informasi pelamar berbasis dokumen CV digital. Metode yang akan digunakan untuk melakukan deteksi pola dalam CV adalah Boyer-Moore dan Knuth-Morris-Pratt. Selain itu, sistem ini akan dihubungkan dengan identitas kandidat melalui basis data sehingga harapannya terbentuk sebuah sistem yang dapat mengenali profil pelamar secara lengkap hanya dengan menggunakan CV digital.

Penjelasan Implementasi

Dalam tugas ini, Anda akan mengembangkan sebuah sistem ATS (Applicant Tracking System) berbasis CV Digital dengan memanfaatkan teknik Pattern Matching. Implementasi sistem ini akan menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt (*Aho-Corasick* apabila mengerjakan bonus) untuk menganalisis dan mencocokkan pola dalam dokumen CV digital, sesuai dengan konsep yang telah dipelajari dalam materi dan slide perkuliahan.



Gambar 2. Skema Implementasi *Applicant Tracking System*

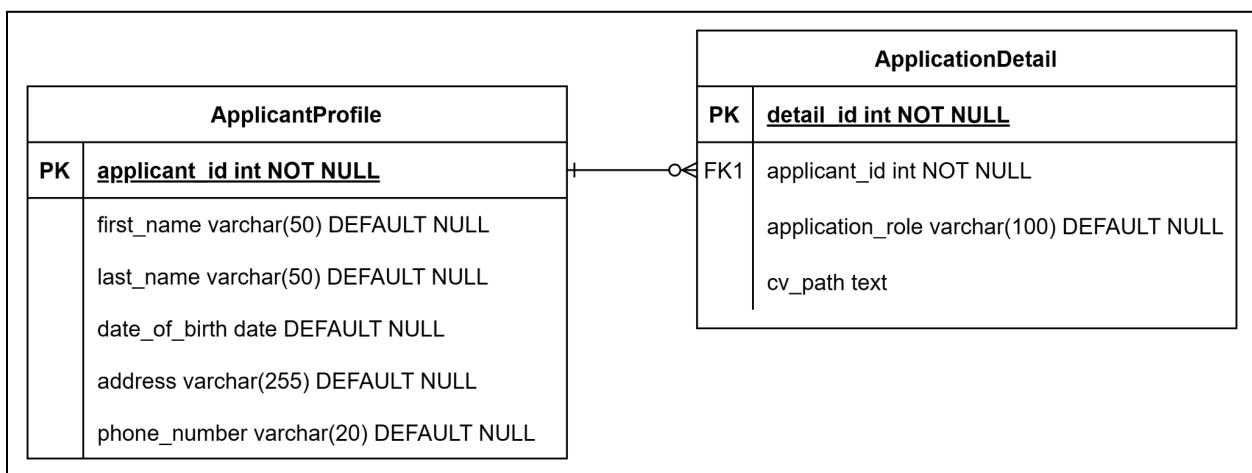
Sistem ini bertujuan untuk mencocokkan kata kunci dari user terhadap isi CV pelamar kerja dengan pendekatan pattern matching menggunakan algoritma KMP (Knuth-Morris-Pratt) atau BM (Boyer-Moore). Semua proses dilakukan secara in-memory, tanpa menyimpan hasil pencarian—hanya data mentah (raw) CV yang disimpan. Pengguna (HR atau rekruter) akan memberikan input berupa daftar kata kunci yang ingin dicari (misalnya: "python", "react", dan "sql") serta jumlah CV yang ingin ditampilkan (misalnya Top 10 matches). Setiap file CV dalam format PDF akan dikonversi menjadi satu string panjang yang memuat seluruh teks dari dokumen tersebut. Proses konversi ini bertujuan untuk mempermudah pencocokan pola menggunakan algoritma string matching, sehingga setiap kata kunci dapat dicari secara efisien dalam satu representasi data linear.

Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan. Dataset yang digunakan dalam contoh ini merupakan dataset CV ATS yang tercantum pada bagian referensi.

Tabel 1. Hasil ekstraksi teks dari CV ATS

CV ATS	Ekstraksi Text untuk Regex	Ekstraksi Text untuk <i>Pattern Matching</i> (KMP & BM)
--------	----------------------------	---

Pada tahap implementasi ini, setiap CV yang telah dikonversi menjadi string panjang untuk mempermudah proses pencocokan. Representasi ini menjadi dasar dalam mencari CV yang paling relevan dengan kata kunci yang dimasukkan oleh pengguna. Proses pencarian dilakukan dengan menggunakan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) untuk menemukan CV yang memiliki kemiripan tertinggi dengan kebutuhan yang ditentukan. Apabila tidak ditemukan satupun CV dalam basis data yang memiliki kecocokan kata kunci secara exact match menggunakan algoritma KMP maupun Boyer-Moore, maka sistem akan mencari CV yang paling mirip berdasarkan tingkat kemiripan di atas ambang batas tertentu (threshold). Hal ini mempertimbangkan kemungkinan adanya kesalahan pengetikan (typo) oleh pengguna atau HR saat memasukkan kata kunci. Anda diberikan **keleluasaan untuk menentukan nilai ambang batas persentase** kemiripan tersebut, dengan syarat dilakukan pengujian terlebih dahulu untuk menemukan nilai tuning yang optimal dan **dijelaskan secara rinci dalam laporan**. Metode perhitungan tingkat kemiripan harus diterapkan menggunakan algoritma **Levenshtein Distance**.



Gambar 3. Skema basis data CV ATS

Dalam skema basis data ini, tabel **ApplicantProfile** menyimpan informasi pribadi pelamar, sedangkan tabel **ApplicationDetail** menyimpan detail aplikasi yang diajukan oleh pelamar tersebut. Relasi antara tabel **ApplicantProfile** dan **ApplicationDetail** adalah one-to-many, karena seorang pelamar dapat mengajukan lamaran untuk beberapa posisi dalam perusahaan yang sama, atau bahkan perusahaan yang berbeda. Setiap lamaran mungkin memerlukan dokumen yang berbeda, seperti CV yang telah disesuaikan untuk peran tertentu.

Untuk keperluan pengembangan awal, basis data silahkan **di-seeding secara mandiri** menggunakan data simulasi. Mendekati tenggat waktu pengumpulan tugas, asisten akan menyediakan seeding resmi yang akan digunakan untuk Demo Tugas Besar.

Atribut **cv_path** pada tabel **ApplicationDetail** digunakan untuk menyimpan lokasi berkas CV digital pelamar di dalam repositori sistem. Lokasi penyimpanan mengikuti struktur folder di direktori **data/**, sebagaimana dijelaskan dalam struktur *repository* pada bagian [pengumpulan tugas](#). Berkas CV yang tersimpan akan dianalisis oleh sistem ATS (Applicant Tracking System) yang dikembangkan dalam Tugas Besar ini.

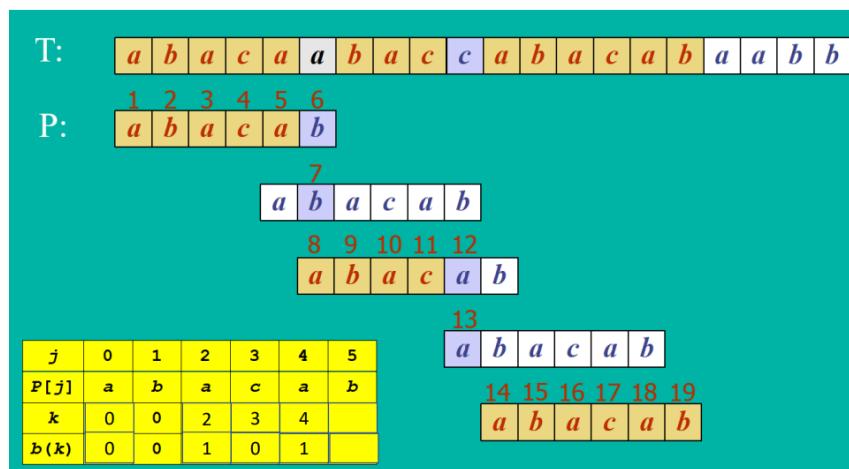
BAB II

LANDASAN TEORI

2.1. Algoritma *Knuth-Morris-Pratt* (KMP)

Algoritma Knuth-Morris-Pratt, atau lebih dikenal dengan KMP, adalah salah satu algoritma pencocokan string yang efisien dan banyak digunakan dalam bidang pengolahan teks. Algoritma ini diperkenalkan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris pada tahun 1977, dan secara khusus dirancang untuk mengatasi kelemahan dari metode pencocokan string sederhana (brute force) yang kerap melakukan pemeriksaan karakter berulang-ulang. Prinsip utama dari algoritma KMP adalah menghindari pencocokan ulang terhadap karakter yang sudah diketahui cocok sebelumnya. Untuk mencapai efisiensi tersebut, KMP melakukan dua tahap proses: preprocessing dan matching. Pada tahap preprocessing, pola (pattern) yang ingin dicocokkan dianalisis terlebih dahulu untuk membentuk array yang disebut LPS (Longest Prefix Suffix). LPS menyimpan informasi mengenai panjang prefiks yang juga merupakan sufiks dari bagian awal pola. Informasi ini digunakan untuk menentukan pergeseran (shift) pola saat terjadi ketidakcocokan tanpa harus kembali ke awal.

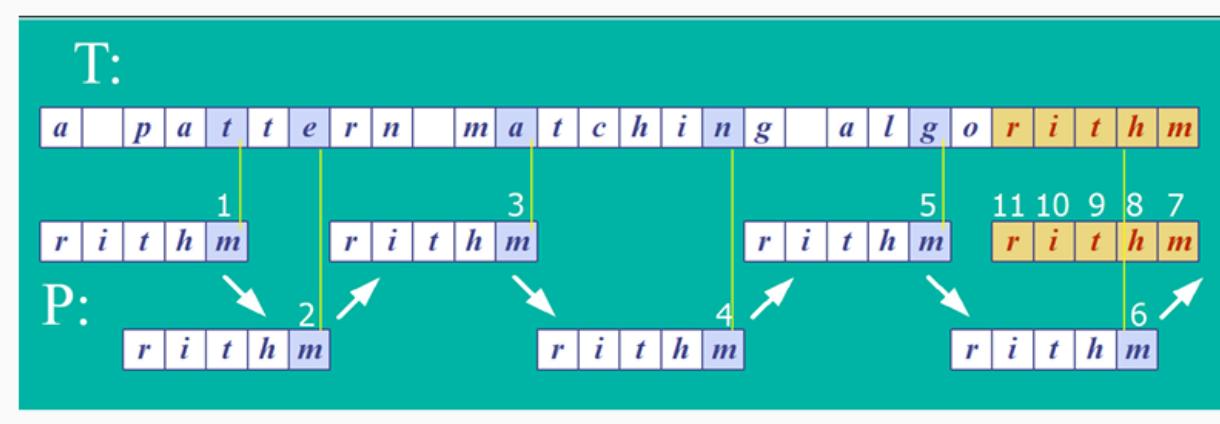
Sebagai contoh, misalkan kita ingin mencocokkan pola "ABABC" ke dalam teks "ABABABCAB". Saat algoritma menemukan bahwa karakter keempat tidak cocok, KMP tidak langsung menggeser pola satu langkah, melainkan memanfaatkan informasi dalam LPS untuk langsung lompat ke bagian yang berpotensi cocok, sehingga karakter yang sudah diperiksa tidak diperiksa ulang.



Jumlah perbandingan karakter: 19 kali

2.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah salah satu algoritma untuk pencarian string dengan mengoptimalkan pencocokan pola. Secara teoritis, algoritma ini menggabungkan dua teknik utama untuk meminimalkan jumlah perbandingan karakter yang diperlukan, yaitu *looking-glass technique* dan *character-jump technique*. Dalam pencocokan pola string digunakan pendekatan heuristik *bad character*. Ketika karakter dalam teks tidak cocok dengan karakter pada pola, informasi tersebut akan digunakan untuk menentukan pergeseran.



Untuk mempercepat proses penentuan pergeseran dapat dilakukan perhitungan *last occurrence function* yang memiliki informasi terkait lokasi kemunculan terakhir dari pola teks yang ingin dicari. Tabel ini menjelaskan cracter apa saja yang ada dalam pattern dan dimana kemunculan terakhir karakter tersebut.

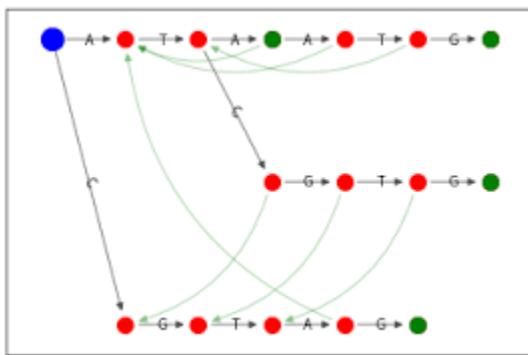
x	a	b	c	d
$L(x)$	4	5	3	-1

2.3. Algoritma Aho-Corasick

Algoritma Aho-Corasick adalah algoritma pencocokan string (string matching) yang dirancang untuk mencari banyak pola (multiple patterns) secara bersamaan dalam satu kali pemindaiannya terhadap teks. Algoritma ini diperkenalkan oleh Alfred V. Aho dan Margaret J. Corasick pada tahun 1975. Berbeda dengan algoritma seperti Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM) yang hanya cocok untuk satu pola dalam satu pencarian, Aho-Corasick unggul dalam kasus ketika terdapat banyak kata kunci yang ingin dicari sekaligus dalam dokumen teks yang besar, sehingga sangat cocok untuk aplikasi seperti sistem deteksi virus, spell-checker, atau

pencarian kata kunci pada teks CV dalam sistem Applicant Tracking System (ATS). Secara konseptual, algoritma ini bekerja dengan membangun sebuah struktur data bernama finite state machine atau biasa dikenal sebagai trie (pohon awalan), yang berisi seluruh pola yang ingin dicocokkan. Trie ini kemudian diperluas dengan failure links yang memungkinkan algoritma untuk berpindah antar node ketika terjadi ketidakcocokan, mirip dengan cara KMP menggunakan array LPS (Longest Prefix Suffix). Saat proses pencarian, teks akan dipindai satu karakter demi satu, dan automaton akan bergerak melalui trie untuk mencocokkan pola-pola yang tersimpan.

Sebagai ilustrasi sederhana, bayangkan kita ingin mencari tiga pola: he, she, dan his dalam sebuah teks. Algoritma Aho-Corasick akan terlebih dahulu menyusun pohon trie dari ketiga kata tersebut, lalu membangun jalur kegagalan (failure links) untuk kembali ke posisi yang mungkin cocok berikutnya jika karakter saat ini tidak cocok. Dengan pendekatan ini, algoritma mampu mendeteksi semua kemunculan semua pola secara bersamaan hanya dengan satu kali lintasan teks.



2.4. Pengembangan Interface GUI

Pengembangan antarmuka grafis atau Graphical User Interface (GUI) merupakan salah satu aspek penting dalam pengembangan perangkat lunak modern, khususnya pada aplikasi yang melibatkan interaksi langsung dengan pengguna. GUI adalah bentuk antarmuka pengguna yang memungkinkan pengguna untuk berinteraksi dengan sistem melalui elemen-elemen visual seperti tombol, menu, jendela, dan ikon, tanpa harus menuliskan perintah secara tekstual seperti pada antarmuka berbasis baris perintah (Command Line Interface / CLI). Tujuan utama dari pengembangan GUI adalah untuk meningkatkan kemudahan penggunaan (usability) dan pengalaman pengguna (user experience). Sebuah GUI yang dirancang dengan baik akan memandu pengguna untuk melakukan tugasnya secara efisien dan intuitif, bahkan tanpa memerlukan pelatihan teknis yang mendalam. Oleh karena itu, prinsip-prinsip desain antarmuka seperti konsistensi, kejelasan, hierarki visual, dan feedback langsung sangat penting dalam perancangan GUI.

Dalam praktik pengembangan perangkat lunak, GUI dibangun menggunakan pustaka atau framework tertentu yang menyediakan komponen-komponen grafis dan event handling. Beberapa pustaka populer yang digunakan dalam pengembangan GUI di bahasa Python adalah Tkinter, PyQt, dan Flet. Masing-masing pustaka ini memiliki pendekatan dan kelebihan tersendiri dalam membangun tampilan aplikasi desktop yang interaktif. Misalnya, Tkinter dikenal sederhana dan ringan untuk aplikasi dasar, sementara PyQt menawarkan kemampuan desain visual yang lebih kompleks dan modern. Pengembangan GUI melibatkan dua aspek utama: desain layout dan logika interaksi. Desain layout mencakup penempatan dan tampilan elemen-elemen antarmuka, sedangkan logika interaksi melibatkan pemrosesan input pengguna, pengaturan alur kerja aplikasi, dan integrasi dengan backend sistem.

Dalam konteks sistem Applicant Tracking System (ATS) yang dikembangkan, GUI digunakan sebagai jembatan antara pengguna (misalnya HR atau rekruter) dengan fitur-fitur pencarian, penyaringan, dan penampilan informasi kandidat. GUI dirancang agar pengguna dapat dengan mudah memasukkan kata kunci pencarian, memilih algoritma pencocokan, melihat hasil pencarian dalam bentuk daftar kandidat, dan mengakses file CV secara langsung. Dengan pendekatan berbasis GUI, sistem menjadi lebih ramah pengguna, mudah dioperasikan, dan efektif untuk proses rekrutmen, khususnya dalam pengolahan banyak data secara visual dan interaktif. Pengembangan GUI yang tepat dapat secara signifikan mempercepat proses pengambilan keputusan dalam perekrutan, sekaligus memberikan kesan profesional pada sistem yang dikembangkan.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-langkah Pemecahan Masalah

Langkah awal dalam pemecahan masalah pencocokan string pada sistem ini dimulai dengan mengolah data mentah berupa dokumen Curriculum Vitae (CV) pelamar kerja. Data CV yang digunakan umumnya berbentuk file berformat PDF, yang kemudian dikonversi menjadi representasi teks guna memungkinkan proses pencarian dan ekstraksi informasi lebih lanjut. Proses konversi dokumen PDF dilakukan dalam dua bentuk teks yang berbeda. Pertama, dokumen PDF diubah menjadi satu string panjang (flat text) yang memuat seluruh isi teks CV tanpa memperhatikan format baris, dengan semua huruf dikonversi ke huruf kecil (lowercase) dan karakter pemisah baris (newline) dihilangkan. Representasi ini bertujuan untuk memfasilitasi proses pencocokan string (pattern matching) yang dilakukan dengan algoritma seperti Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), atau Aho-Corasick. Dengan format string panjang dan homogen, algoritma pencarian dapat bekerja lebih cepat dan efisien dalam mendeteksi kecocokan kata kunci.

Kedua, dokumen juga dikonversi ke dalam bentuk teks yang lebih mempertahankan struktur aslinya, yaitu mempertahankan pemisah baris (newline), case sensitivity, dan urutan bagian seperti pada tampilan PDF. Representasi ini digunakan dalam proses ekstraksi informasi berbasis Regular Expression (Regex), untuk mengambil bagian-bagian penting dari CV seperti bagian Summary, Experience, Education, dan Skills secara otomatis. Agar informasi yang ditampilkan pada antarmuka pengguna (GUI) dapat ditelusuri dan diakses dengan baik, sistem ini menggunakan basis data MySQL untuk menyimpan metadata terkait setiap CV. Database ini terdiri dari dua entitas utama, yaitu ApplicantProfile dan ApplicationDetail. Tabel ApplicantProfile menyimpan informasi identitas pelamar secara umum, seperti nama, tanggal lahir, alamat, dan nomor telepon. Sementara itu, tabel ApplicationDetail menyimpan informasi riwayat aplikasi pekerjaan, termasuk jalur lokasi file CV dan posisi pekerjaan yang dilamar. Relasi antar tabel ini bersifat one-to-many, di mana satu pelamar dapat memiliki lebih dari satu CV yang dilamar ke berbagai posisi berbeda.

Proses pengisian awal (seeding) database dilakukan secara manual untuk memastikan keterkaitan antar pelamar dan file CV yang sesuai. Masing-masing file CV dikaitkan dengan role atau posisi yang dilamar, yang nantinya akan digunakan untuk proses pencarian kecocokan berbasis kata kunci dalam GUI. Informasi yang disimpan dalam database tidak hanya mendukung pengorganisasian data, tetapi juga memungkinkan tampilan informasi pelamar yang sesuai saat pengguna (HR) melihat hasil pencarian melalui antarmuka ScoopyHire.

Setelah data didapatkan format data regex data terkait informasi penting dalam CV, selanjutnya dilakukan ekstraksi informasi menggunakan regex untuk parsing data sesuai dengan kategori yang berhubungan. Data hasil parsing kemudian akan digunakan untuk ditampilkan dalam hasil pencarian. Informasi yang berkaitan dengan data profil akan diambil dari seeding database, sedangkan untuk data yang berkaitan dengan informasi CV diperoleh dari regex data.

Pada tugas besar ini terdapat tiga opsi algoritma yang diimplementasikan untuk pencocokan string. Setiap algoritma memiliki pendekatan perhitungan heuristic yang berbeda-beda yang akan dijelaskan pada bagian proses pemetaan masalah. Namun secara umum, algoritma ini memiliki objektif untuk mencari jumlah kemunculan pola terbanyak diantara data CV yang ada. Pencocokan dilakukan dengan *exact match* dan *fuzzy match*. Pencarian *fuzzy match* hanya akan dilakukan apabila pencarian *exact match* tidak memberikan solusi. Pencarian *fuzzy match* akan memberikan hasil berupa CV dengan yang memiliki kata kunci yang mirip dengan kata kunci yang dicari. Tingkat kemiripan pada *fuzzy match* dikalkulasikan dengan menggunakan algoritma Levenshtein Distance. Levenshtein Distance merupakan algoritma yang mengukur kemiripan antara dua buah string dengan cara menghitung jumlah perubahan yang dibutuhkan untuk mendapatkan string tujuan dari string asal. Satu perubahan, baik itu insersi, delesi, maupun substitusi dihitung sebagai satu *distance*, hal ini menyebabkan nilai dari Levenshtein Distance pasti berupa bilangan bulat non-negatif. Pada pencarian yang dilakukan di tugas besar ini, kami memilih *threshold* atau ambang batas dari *distance* tidak boleh >1 . Sudah dilakukan percobaan pada beberapa nilai *distance*, tetapi nilai *distance* yang melebihi 1 menyebabkan terlalu banyak *kata kunci* lain yang ditoleransi sehingga semakin jauh dari konteks yang sebenarnya diinginkan.

3.2 Proses Pemetaan Masalah

3.2.1 Pemetaan Masalah dengan KMP

Langkah pertama dalam proses pencocokan string adalah menentukan terlebih dahulu kata atau frasa apa yang ingin dicari dalam teks. Dalam konteks sistem ini, pola yang dimaksud merupakan kata kunci yang relevan dengan kriteria pekerjaan tertentu, seperti "react", "html", atau "project management". Kata kunci tersebut akan menjadi acuan utama dalam proses pencarian terhadap isi CV yang telah dikonversi menjadi teks.

Sebelum proses pencarian dilakukan, terlebih dahulu dipersiapkan sebuah petunjuk yang akan membantu mempercepat proses pencarian. Petunjuk ini berisi informasi tentang bagaimana pola bisa "melompat" ke posisi berikutnya ketika terjadi ketidakcocokan, tanpa harus mengulang pencocokan dari awal. Logika ini didasarkan pada pengamatan bahwa dalam banyak kasus, bagian awal dari pola sering kali berulang dalam pola itu sendiri. Oleh karena itu, saat terjadi ketidakcocokan, pencarian bisa langsung dilanjutkan ke bagian yang masih mungkin cocok, berdasarkan informasi yang telah dipetakan sebelumnya.

Setelah petunjuk tersebut selesai dipersiapkan, proses pencarian dimulai dengan membandingkan huruf-huruf dari pola terhadap huruf-huruf dalam teks, satu per satu dari kiri ke kanan. Setiap kali huruf dalam pola cocok dengan huruf dalam teks, pencarian dilanjutkan ke huruf berikutnya. Namun, jika terjadi ketidakcocokan, maka pencarian tidak dimulai ulang dari awal pola, tetapi diarahkan ke posisi dalam pola yang masih memungkinkan untuk dicocokkan, sesuai petunjuk yang telah dibuat sebelumnya. Dengan cara ini, pencarian menjadi jauh lebih efisien karena tidak ada karakter yang diperiksa lebih dari sekali.

Setiap kali seluruh karakter dalam pola berhasil dicocokkan secara berurutan dengan bagian dari teks, maka posisi kemunculan pola dalam teks dicatat. Posisi ini mengindikasikan bahwa pola yang dicari memang terdapat dalam teks tersebut, dan informasi ini sangat penting untuk disimpan dan digunakan dalam tahap selanjutnya, seperti penilaian kecocokan kandidat terhadap kriteria pencarian.

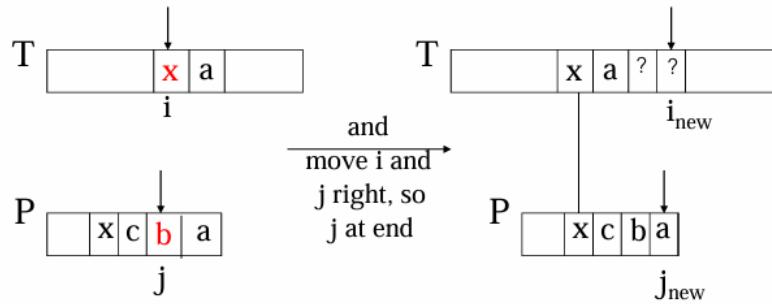
Setelah seluruh teks selesai dipindai, semua posisi di mana pola ditemukan dikumpulkan dan disusun sebagai hasil akhir. Informasi ini kemudian dapat disampaikan kepada pengguna, misalnya dalam bentuk tampilan daftar kandidat yang relevan dalam antarmuka sistem. Dari sini, pengguna dapat dengan mudah melihat siapa saja yang memenuhi kriteria kata kunci yang dicari, serta berapa kali kata tersebut muncul dalam dokumen CV mereka.

3.2.2 Pemetaan Masalah dengan BM

Tahap pertama untuk melakukan pencocokan string dengan algoritma Boyer-Moore adalah dengan menghitung fungsi *last occurrence* pada pattern yang ingin dicari. Fungsi ini akan mempermudah penentuan pergeseran yang perlu dilakukan. Dalam menentukan seberapa jauh pergeseran terjadi ketika ditemukan pola yang tidak sesuai dapat dilakukan dengan mengklasifikasikan kasus yang terjadi. Terdapat tiga kemungkinan pergeseran untuk algoritma ini, yaitu :

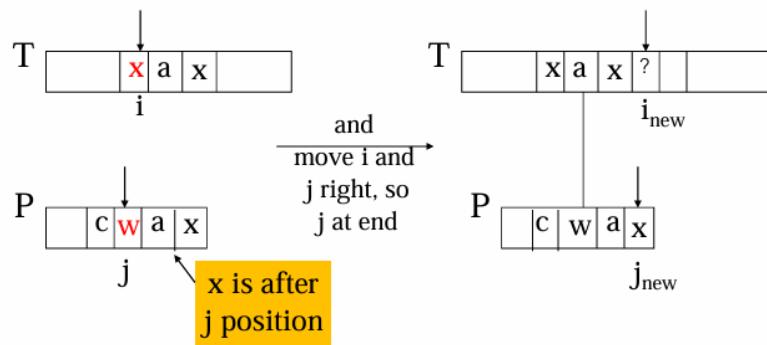
1. Kasus 1

Ketika terdapat x di dalam P maka geser P sehingga huruf yang sama menjadi sejajar dengan $T[i]$



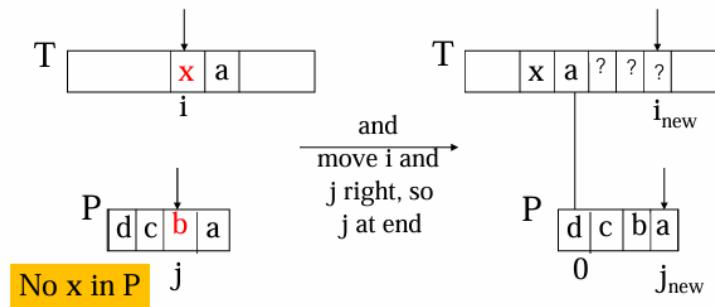
2. Kasus 2

Ketika x terdapat di P namun tidak bisa digeser menuju kemunculan terakhir, maka geser satu karakter menuju T[i]



3. Kasus 3

Ketika kasus 1 dan 2 tidak terpenuhi, maka geser P sehingga P[0] sejajar dengan T[i+1]



Setelah menentukan kondisi kasus yang memenuhi kemudian dilakukan pergeseran yang sesuai. Tahapan tersebut akan diulang hingga ditemukan *string matching* yang sesuai dengan pola yang dicari.

3.2.3 Pemetaan Masalah dengan AC

Dalam pencarian kemunculan kata kunci pada CV, kata kunci yang dicari tidak selalu tunggal. Terdapat kebutuhan untuk mencari banyak kata kunci sekaligus dalam satu kali proses pencarian. Jika pencarian dilakukan secara terpisah untuk setiap kata kunci, maka proses tersebut akan menjadi tidak efisien, terutama ketika jumlah CV dan kata kunci meningkat.

Masalah ini dapat dioptimasi dengan penggunaan algoritma Aho-Corasick. Algoritma ini secara garis besar memiliki kemiripan dengan algoritma Knuth-Morris-Pratt (KMP), yaitu sama-sama menggunakan informasi prefiks terpanjang untuk menghindari pencarian ulang dari awal saat terjadi ketidakcocokan karakter. Namun, keunggulan utama Aho-Corasick terletak pada kemampuannya dalam menangani banyak kata kunci secara simultan.

Aho-Corasick membangun sebuah struktur data berupa trie dari seluruh kata kunci yang ingin dicari. Trie ini kemudian dilengkapi dengan *failure links*, yang memungkinkan pencarian tetap berlanjut ke jalur lain dalam trie saat karakter berikutnya tidak cocok. Lebih jauh, algoritma ini menyimpan informasi karakter yang sudah ditelusuri dalam bentuk status (state), sehingga pencocokan terhadap satu kata kunci dapat sekaligus membantu pencocokan kata kunci lain yang memiliki prefiks atau sufiks yang sama.

3.3 Fitur Fungsional dan Arsitektur Aplikasi

3.3.1 Fitur Fungsional

ScoopyHire adalah aplikasi pencarian CV yang dirancang untuk membantu pengguna menemukan kandidat yang relevan berdasarkan kata kunci tertentu. Fitur-fitur fungsional utama dari aplikasi ini meliputi:

- Pencarian CV

Pengguna dapat memilih salah satu dari tiga algoritma pencarian teks, yaitu KMP, BM, dan AC. Selain dapat memilih algoritma, pencarian CV juga dapat memberikan hasil berupa exact match dan fuzzy match (jika tidak ditemukan exact match-nya). Hasil pencarian berupa cuplikan CV yang berisi kata kunci, identitas pelamar, dan opsi untuk melihat ringkasan atau CV keseluruhan. Hasil pencarian ditampilkan secara terurut menurun berdasarkan jumlah kemunculan kata kunci. Pengurutan ini bertujuan untuk memudahkan pencarian CV pelamar yang paling relevan.

- Penyajian Ringkasan CV Hasil Pencarian

Setelah pencarian dilakukan, pengguna diberi opsi untuk melihat ringkasan dari CV yang mengandung kata kunci. Ringkasan ini memberikan informasi berupa identitas, skill, pendidikan, serta riwayat kerja pelamar. Pengguna aplikasi dapat menilai relevansi kandidat secara cepat tanpa membuka keseluruhan file.

- Opsi untuk Melihat File Asli dari CV Hasil Pencarian

Jika ringkasan belum cukup, pengguna aplikasi juga dapat membuka file asli CV jika ingin melihat detail lebih lanjut. Fitur ini mempermudah proses evaluasi mendalam terhadap kandidat yang terjaring dari hasil pencarian awal.

3.3.2 Arsitektur Aplikasi

1. Database

Aplikasi ScoopyHire menggunakan MySQL sebagai backbone database dengan arsitektur client-server. Database server berjalan di localhost dengan user dedicated `ats_user` yang memiliki akses penuh ke database `cv_ats`. Arsitektur ini dipilih karena MySQL menyediakan ACID compliance yang diperlukan untuk integritas data profil pelamar dan mendukung relational queries yang efisien untuk operasi pencarian CV.

Sistem mengimplementasikan arsitektur hybrid storage yang menggabungkan:

- Relational Database Layer (MySQL): Menyimpan metadata profil pelamar dan relasi aplikasi
- File System Layer: Menyimpan dokumen PDF asli dan hasil ekstraksi teks
- Structured Data Layer: File JSON untuk informasi terstruktur hasil parsing CV
- Pattern Matching Layer: File teks yang telah dibersihkan untuk algoritma pencarian

Schema database dirancang dengan prinsip normalized relational design untuk menghindari redundansi data. Dua tabel utama (`ApplicantProfile` dan `ApplicationDetail`) dihubungkan melalui foreign key relationship yang memungkinkan one-to-many mapping antara satu pelamar dengan multiple aplikasi pekerjaan.

Pipeline ekstraksi data menggunakan ETL (Extract, Transform, Load) pattern:

- Extract: PyMuPDF library mengekstrak raw text dari PDF

- Transform: Text cleaning, section parsing, dan format standardization
- Load: Penyimpanan dalam multiple formats (MySQL, TXT, JSON)

2. GUI

Dalam aplikasi ini, kami menggunakan Flet sebagai framework untuk membangun antarmuka pengguna grafis (GUI). Flet, yang didukung oleh Flutter, memungkinkan pembuatan antarmuka yang kaya dan interaktif dengan pendekatan berbasis komponen. Dengan Flet, keseluruhan antarmuka, mulai dari tata letak hingga fungsionalitas, dapat ditulis sepenuhnya menggunakan bahasa pemrograman Python.

Beberapa komponen utama yang kami manfaatkan dalam antarmuka ini meliputi:

- Tata Letak dengan Column dan Row

Struktur utama aplikasi dibangun menggunakan `ft.Column` untuk menyusun elemen-elemen secara vertikal dan `ft.Row` untuk menyusun elemen secara horizontal. `Column` digunakan pada halaman utama untuk menata konten dari atas ke bawah, sementara `Row` digunakan untuk menempatkan tombol-tombol pilihan algoritma atau detail kandidat secara berdampingan.

- Styling dengan Container

Komponen `ft.Container` memegang peranan kunci dalam aspek visual. Kami menggunakannya untuk membungkus elemen lain, memberikan warna latar, mengatur padding dan margin serta menambahkan border. Penggunaan `Container` terlihat jelas dalam pembuatan kartu hasil pencarian dan sebagai dasar untuk elemen interaktif.

- Tombol dan Interaktivitas

Tombol utama seperti "Get Started" dan "Search CV" dibuat menggunakan `ft.ElevatedButton` yang gayanya diatur melalui `ft.ButtonStyle`. Untuk pilihan algoritma, kami menciptakan elemen interaktif menggunakan `ft.Container` yang diberi fungsi `on_click`. Saat salah satu pilihan diklik, warnanya akan berubah untuk memberikan umpan balik visual yang jelas tentang opsi mana yang sedang aktif.

- Teks dan Input

ft.Text digunakan untuk menampilkan semua informasi tekstual, mulai dari judul aplikasi "ScoopyHire" hingga detail pada ringkasan CV, dengan properti gaya seperti size, color, dan weight. Untuk input dari pengguna, kami menggunakan ft.TextField yang memungkinkan pengguna memasukkan kata kunci pencarian.

- Konten yang Dapat Digulir

Untuk area yang menampilkan konten panjang, seperti daftar hasil pencarian, kami menerapkan properti scroll=ft.ScrollMode.AUTO pada ft.Column. Ini secara otomatis menambahkan scrollbar ketika konten melebihi ukuran tampilan, memastikan semua informasi tetap dapat diakses oleh pengguna.

3.4 Contoh Ilustrasi Kasus

Ilustrasi contoh pencocokan kata kunci pada CV, dalam hal ini diambil satu sampel dari seluruh dataset yang ada dengan nomor file 10041713.pdf. File pdf ini akan diekstrak informasinya hingga menghasilkan file txt yang nantinya akan digunakan untuk pencarian menggunakan regex. Dalam contoh ilustrasi kasus kali ini diambil sebagian text dari file text CV untuk mempersingkat dan mempermudah ilustrasi kasus.

Untuk ilustrasi ini akan dicari kata “sales” dari dengan menggunakan ketiga algoritma yang telah diimplementasikan. Pembahasan ilustrasi akan dijelaskan dalam setiap sub bagian setelah ini.

sr. estimator-bas construction sales summary
--

3.4.1 Ilustrasi Kasus KMP

k = j-1	0	1	2	3
b(k)	0	0	0	1

Berikut ilustrasinya:

T : sr. estimator-bas construction sales summary
--

P : sa

S

S

s
s
sa
s
s
S
S
S
S
S
S
S
S
Sa
S
S
S
S
S
S
S
S
S
S
S
S
S
S
Sales

Jumlah perbandingan karakter : 40

3.4.2 Ilustrasi Kasus Boyer-Moore

T : sr. estimator-bas construction sales summary
P : sales
sales
sales
sales
sales
sales

sales
sales
sales

Jumlah perbandingan karakter : 13

3.4.3 Ilustrasi Kasus AC

Text: sr.estimator-bas construction sales summary

State: 0

i = 0: 's' - State 0 → 1

T: sr.estimator-bas construction sales summary

P: s

State: 1

i = 1: 'r' - State 1, mismatch, failure ke 0

T: sr.estimator-bas construction sales summary

P: s

State: 0

i = 2: '!' - State 0, tidak ada transisi

T: sr.estimator-bas construction sales summary

P: s

State: 0

i = 3: '"' - State 0, tidak ada transisi

T: sr.estimator-bas construction sales summary

P: s

State: 0

i = 4: 'e' - State 0, tidak ada transisi

T: sr.estimator-bas construction sales summary

P: s

State: 0

i = 5: 's' - State 0 → 1

T: sr.estimator-bas construction sales summary

P: s

State: 1

```
i = 6: 't' - State 1, mismatch, failure ke 0
T: sr. estimator-bas construction sales summary
P:      s
State: 0

...
i = 31: 's' - State 0→1
T: sr. estimator-bas construction sales summary
P:          s
State: 1

i = 32: 'a' - State 1→2
T: sr. estimator-bas construction sales summary
P:          sa
State: 2

i = 33: 'l' - State 2→3
T: sr. estimator-bas construction sales summary
P:          sal
State: 3

i = 34: 'e' - State 3→4
T: sr. estimator-bas construction sales summary
P:          sale
State: 4

i = 35: 's' - State 4→5 (kata ditemukan)
T: sr. estimator-bas construction sales summary
P:          sales
State: 5, Output: "sales" di posisi 31
```

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Database

4.1.1 Struktur Data

Struktur Data	Penjelasan
<pre>db_config = { "host": "localhost", "user": "ats_user", "password": "Ats_Pass11", "database": "cv_ats" }</pre>	Struktur database configuration digunakan untuk menyimpan konfigurasi koneksi database dalam format dictionary, untuk memisahkan credential dari logic aplikasi untuk security, dan dapat di-import oleh module lain tanpa hardcoding connection string.
<pre>CREATE TABLE ApplicantProfile (applicant_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, first_name VARCHAR(50) DEFAULT NULL, last_name VARCHAR(50) DEFAULT NULL, date_of_birth DATE DEFAULT NULL, address VARCHAR(255) DEFAULT NULL, phone_number VARCHAR(20) DEFAULT NULL);</pre>	Struktur tabel ApplicantProfile untuk seeding 100 profil pelamar dengan data personal yang beragam.
<pre>CREATE TABLE ApplicationDetail (detail_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, applicant_id INT NOT NULL, application_role VARCHAR(100) DEFAULT NULL, cv_path TEXT, FOREIGN KEY (applicant_id) REFERENCES ApplicantProfile(applicant_id));</pre>	Struktur tabel ApplicationDetail untuk seeding 480 detail aplikasi yang dipetakan ke file CV dalam 24 kategori pekerjaan dan sebagai relational mapping antara applicant_id dengan cv files.

4.1.2 Fungsi dan Prosedur

1. connect()

Fungi ini membuat koneksi ke MYSQL database menggunakan konfigurasi dari db_config.

```
def connect(): return mysql.connector.connect(**db_config)
```

2. insert_applicant(first_name, last_name, dob, address, phone)

Fungi ini untuk insert data profil pelamar baru ke tabel ApplicantProfile.

```
def insert_applicant(first_name, last_name, dob, address, phone):
    conn = connect()
    cursor = conn.cursor()
    query = """
        INSERT INTO ApplicantProfile
        (first_name, last_name, date_of_birth, address,
        phone_number)
        VALUES (%s, %s, %s, %s, %s)
    """
    cursor.execute(query, (first_name, last_name, dob, address,
                           phone))
    conn.commit()
    applicant_id = cursor.lastrowid
    cursor.close()
    conn.close()
    return applicant_id
```

3. insert_applicant_detail

Fungi ini untuk insert detail aplikasi yang menghubungkan pelamar dengan posisi dan CV file.

```
def insert_application_detail(applicant_id, role, cv_path):
    conn = connect()
    cursor = conn.cursor()
    query = """
        INSERT INTO ApplicationDetail
        (applicant_id, application_role, cv_path)
        VALUES (%s, %s, %s)
    """
    cursor.execute(query, (applicant_id, role, cv_path))
    conn.commit()
    cursor.close()
    conn.close()
```

```
conn.commit()
applicant_id = cursor.lastrowid
cursor.close()
conn.close()
return applicant_id
```

4. extract_text_from_pdf

Fungi ini ekstraksi raw text dari PDF menggunakan PyMuPDF library.

```
def extract_text_from_pdf(pdf_path):
    """Extract text from PDF"""
    try:
        doc = fitz.open(pdf_path)
        text = "\n".join([page.get_text() for page in doc])
        doc.close()
        return text
    except Exception as e:
        print(f"Error extracting PDF {pdf_path}: {e}")
        return ""
```

5. save_pattern_text

Fungi ini membersihkan text dan menyimpan untuk pattern matching algorithms.

```
def save_pattern_text(text, output_path):
    """Save cleaned text for pattern matching"""
    cleaned = " ".join(text.replace("\n", " ").lower().split())
    with open(output_path, "w", encoding="utf-8") as f:
        f.write(cleaned)
```

6. save_regex_text

Fungi ini menyimpan original text untuk regex processing.

```
def save_regex_text(text, output_path):
    """Save original text for regex processing"""
    with open(output_path, "w", encoding="utf-8") as f:
        f.write(text)
```

7. extract_sections_flexible

Fungi ini parse CV text menjadi sections (Skills, Experience, Education, dll).

```
def extract_sections_flexible(text):
    """Extract sections from CV text"""
    headings = [
```

```

        "Skills", "Summary", "Highlights", "Accomplishments",
        "Experience", "Education",
        "Certifications", "Projects", "Technical Skills",
        "Languages", "Awards"
    ]
    pattern = rf"(?P<header>^({'|'.join(map(re.escape,
headings))})\s*\$)"
    matches = list(re.finditer(pattern, text,
flags=re.MULTILINE | re.IGNORECASE))
    result = {}
    for i, match in enumerate(matches):
        header = match.group("header").strip().title()
        start = match.end()
        end = matches[i + 1].start() if i + 1 < len(matches)
else len(text)
        result[header] = text[start:end].strip()
    return result

```

8. process_single_file

Fungi ini complete processing pipeline untuk satu file CV.

```

def process_single_file(pdf_path, role):
    """Process single PDF file"""
    filename = os.path.splitext(os.path.basename(pdf_path))[0]

    # Extract text from PDF
    text = extract_text_from_pdf(pdf_path)
    if not text.strip():
        print(f"⚠️ No text extracted from {pdf_path}")
        return False

    # Create output directories
    os.makedirs("data/pattern_matching", exist_ok=True)
    os.makedirs("data/regex_data", exist_ok=True)
    os.makedirs("data/structured_info", exist_ok=True)

    # Save pattern matching text (cleaned)
    save_pattern_text(text,
f"data/pattern_matching/{filename}.txt")

    # Save regex text (original)
    save_regex_text(text, f"data/regex_data/{filename}.txt")

    # Extract sections and save as JSON
    sections = extract_sections_flexible(text)
    structured_info = {

```

```
        "filename": filename,
        "role": role,
        "sections": sections,
        "text_length": len(text),
        "processed_at": datetime.now().isoformat()
    }

    with open(f"data/structured_info/{filename}.json", "w",
encoding="utf-8") as jf:
    json.dump(structured_info, jf, ensure_ascii=False,
indent=2)

    print(f"✓ Processed {filename} ({role}) - {len(text)} characters")
    return True
```

9. process_all_files

Fungi ini batch processing untuk semua PDF files dalam 24 role folders.

```
def process_all_files():
    """Process all PDF files in all role folders"""
    base_data_dir = "data"

    # Define all roles (folders)
    roles = [
        'ACCOUNTANT', 'ADVOCATE', 'AGRICULTURE', 'APPAREL',
        'ARTS', 'AUTOMOBILE',
        'AVIATION', 'BANKING', 'BPO', 'BUSINESS-DEVELOPMENT',
        'CHEF', 'CONSTRUCTION',
        'CONSULTANT', 'DESIGNER', 'DIGITAL-MEDIA',
        'ENGINEERING', 'FINANCE',
        'FITNESS', 'HEALTHCARE', 'HR',
        'INFORMATION-TECHNOLOGY', 'PUBLIC-RELATIONS',
        'SALES', 'TEACHER'
    ]

    total_processed = 0
    total_files = 0

    for role in roles:
        role_dir = os.path.join(base_data_dir, role)

        if not os.path.exists(role_dir):
            print(f"⚠️ Directory not found: {role_dir}")
            continue
```

```

# Get all PDF files in the role directory
pdf_files = [f for f in os.listdir(role_dir) if
f.endswith('.pdf')]

if not pdf_files:
    print(f"⚠️ No PDF files found in {role_dir}")
    continue

print(f"\n📁 Processing {role} ({len(pdf_files)} files):")

role_processed = 0
for pdf_file in sorted(pdf_files):
    pdf_path = os.path.join(role_dir, pdf_file)
    total_files += 1

    if process_single_file(pdf_path, role):
        role_processed += 1
        total_processed += 1

    print(f"✓ {role}: {role_processed}/{len(pdf_files)} files processed")

print(f"\n🎉 SUMMARY:")
print(f"  Total files found: {total_files}")
print(f"  Successfully processed: {total_processed}")
print(f"  Failed: {total_files - total_processed}")

# Display output structure
print(f"\n📁 Output directories created:")
print(f"  - data/pattern_matching/ (cleaned text for pattern matching)")
print(f"  - data/regex_data/ (original text for regex)")
print(f"  - data/structured_info/ (JSON with extracted sections)")

```

4.2 Implementasi Program: Algoritma KMP

4.2.1 Struktur Data

Struktur Data	Penjelasan
lps = [0] * len(pattern)	Struktur data list of integers tersebut berukuran sama dengan panjang pattern

	untuk menyimpan informasi tentang prefix terpanjang yang juga merupakan suffix untuk setiap posisi dalam pattern.
positions = []	Struktur data list of integers untuk menyimpan semua posisi di mana pattern ditemukan dalam text.

4.2.2 Fungsi dan Prosedur

1. build_lps

Fungsi ini membangun tabel LPS (failure function) yang digunakan untuk optimasi pencarian dengan kompleksitas $O(m)$ dimana m adalah panjang pattern.

```
def build_lps(pattern):
    lps = [0] * len(pattern)
    length = 0
    i = 1
    while i < len(pattern):
        if pattern[i] == pattern[length]:
            length += 1
            lps[i] = length
            i += 1
        else:
            if length != 0:
                length = lps[length - 1]
            else:
                lps[i] = 0
                i += 1
    return lps
```

2. kmp_search

Fungi ini mencari semua kemunculan pattern dalam text menggunakan algoritma KMP dengan kompleksitas $O(n + m)$ di mana n adalah panjang text, m adalah panjang pattern.

```
def kmp_search(text, pattern):
    lps = build_lps(pattern)
    i = j = 0
    positions = []
    while i < len(text):
        if pattern[j] == text[i]:
            i += 1
            j += 1
        else:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1
    positions.append(i - j)
```

```
        if j == len(pattern):
            positions.append(i - j)
            j = lps[j - 1]
        elif i < len(text) and pattern[j] != text[i]:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1
    return positions
```

3. search_keyword_kmp

Fungi ini mencari keyword dalam multiple files menggunakan KMP.

```
def search_keyword_kmp(keyword, files, applicant_db,
cv_mapping):
    """Search keyword using KMP algorithm"""
    result = []
    keyword_lower = keyword.lower()

    print(f"🔍 Searching for '{keyword_lower}' in {len(files)} files...")

    for filename, text in files:
        positions = kmp_search(text, keyword_lower)
        if positions:
            name = get_applicant_name_by_cv(applicant_db,
cv_mapping, filename.replace(".txt", ""))
            result.append({
                "name": name,
                "filename": filename,
                "count": len(positions),
                "positions": positions[:5]
            })

    print(f"⌚ Found {len(result)} matches")
    return sorted(result, key=lambda x: x["count"],
reverse=True)
```

4. search_multiple_keywords_kmp

Fungi ini mencari multiple keywords dengan logika AND (semua keyword harus ada).

```
def search_multiple_keywords_kmp(keywords, files, applicant_db,
cv_mapping):
    """Search multiple keywords using KMP algorithm"""
```

```

result = []
keywords_lower = [kw.strip().lower() for kw in keywords if
kw.strip()]

print(f"🔍 Searching for {len(keywords_lower)} keywords:
{', '.join(keywords_lower)}")
print(f"📝 Scanning {len(files)} CV files...")

for filename, text in files:
    keyword_results = {}
    total_matches = 0
    all_positions = []

    # Search each keyword in the current file
    for keyword in keywords_lower:
        positions = kmp_search(text, keyword)
        if positions:
            keyword_results[keyword] = {
                'count': len(positions),
                'positions': positions[:3] # Store first 3
positions for context
            }
            total_matches += len(positions)
            all_positions.extend(positions)

    # Only include if ALL keywords are found (AND logic)
    if len(keyword_results) == len(keywords_lower):
        name = get_applicant_name_by_cv(applicant_db,
cv_mapping, filename.replace(".txt", ""))
        result.append({
            "name": name,
            "filename": filename,
            "total_count": total_matches,
            "keyword_details": keyword_results,
            "keywords_found": len(keyword_results),
            "all_positions": sorted(all_positions)[:10] # First 10 positions for context
        })

    print(f"⌚ Found {len(result)} CVs containing ALL
keywords")

    # Sort by total matches (higher = better match)
    return sorted(result, key=lambda x: x["total_count"],
reverse=True)

```

5. search_multiple_keywords_or

Fungi ini mencari multiple keywords dengan logika OR (minimal satu keyword harus ada).

```
def search_multiple_keywords_or(keywords, files, applicant_db,
cv_mapping):
    """Search multiple keywords with OR logic (any keyword found)"""
    result = []
    keywords_lower = [kw.strip().lower() for kw in keywords if kw.strip()]

    print(f"🔍 Searching for ANY of {len(keywords_lower)} keywords: {', '.join(keywords_lower)}")
    print(f"📄 Scanning {len(files)} CV files...")

    for filename, text in files:
        keyword_results = {}
        total_matches = 0
        all_positions = []

        # Search each keyword in the current file
        for keyword in keywords_lower:
            positions = kmp_search(text, keyword)
            if positions:
                keyword_results[keyword] = {
                    'count': len(positions),
                    'positions': positions[:3]
                }
                total_matches += len(positions)
                all_positions.extend(positions)

        # Include if ANY keyword is found (OR logic)
        if keyword_results:
            name = get_applicant_name_by_cv(applicant_db,
            cv_mapping, filename.replace(".txt", ""))
            result.append({
                "name": name,
                "filename": filename,
                "total_count": total_matches,
                "keyword_details": keyword_results,
                "keywords_found": len(keyword_results),
                "all_positions": sorted(all_positions)[:10]
            })

    print(f"⌚ Found {len(result)} CVs containing ANY of the keywords")
```

```
# Sort by number of different keywords found, then by total
# matches
    return sorted(result, key=lambda x: (x["keywords_found"],
x["total_count"])), reverse=True)
```

4.3 Implementasi Program: Algoritma BM

4.3.1 Fungsi dan Prosedur

1. bad_char_heuristic

Fungsi bad_char_heuristic adalah fungsi untuk menghitung *last occurrence* yang berisi informasi posisi terakhir dari setiap karakter dalam pola.

```
def bad_char_heuristic(pattern):
    m = len(pattern)
    bad_char = {}

    for i in range(m):
        bad_char[pattern[i]] = i

    return bad_char
```

2. boyer_moore_search

Fungsi ini digunakan untuk mencari kata dalam teks yang sesuai dengan pattern. Fungsi ini akan mengembalikan daftar indeks di mana pola ditemukan dalam teks.

```
def boyer_moore_search(text, pattern):
    n = len(text)
    m = len(pattern)

    if m == 0:
        return []
    if n == 0 or n < m:
        return []

    bad_char = bad_char_heuristic(pattern)

    s = 0 # shift dari pola terhadap teks
    matches = []

    while s <= (n - m):
        j = m - 1
```

```

while j >= 0 and pattern[j] == text[s + j]:
    j -= 1

if j < 0:
    matches.append(s)
    if s + m < n:
        next_char = text[s + m]
        s += m - bad_char.get(next_char, -1)
    else:
        s += 1 # Jika sudah di akhir teks
else:
    mismatched_char = text[s + j]

shift = j - bad_char.get(mismatched_char, -1)

s += max(1, shift)

return matches

```

4.4 Implementasi Program: Algoritma AC

4.4.1 Struktur Data

Struktur Data	Penjelasan
Dictionary	Struktur data dictionary digunakan untuk dua hal, yaitu goto dan failure table. Dictionary digunakan untuk memudahkan pemetaan char ke state yang dituju, yaitu state selanjutnya pada goto table dan state sebelumnya/fallback pada failure table.
List	Struktur data list digunakan untuk menyimpan beberapa pattern yang akan dicari.
Deque	Struktur data deque digunakan untuk membangun failure table, yakni untuk memastikan failure link mengacu ke state dengan level yang sama atau lebih rendah (mendekati root).

4.4.2 Fungsi dan Prosedur

Berikut adalah fungsi dan prosedur yang digunakan.

1. add_patterns

Fungsi untuk melakukan iterasi pada kumpulan pattern sekaligus memanggil proses pembuatan trie.

```
def add_patterns(self, patterns):
    for pattern in patterns:
        if pattern:
            self.patterns.append(pattern)

            self._build_trie()
            self._built = False
```

2. _build_trie

Fungsi untuk membangun trie dari kumpulan patterns, yaitu menetapkan state, goto table, dan menginisialisasi output table.

```
def _build_trie(self):
    self.goto_table = {0: {}}
    self.output_table = {}
    self.state_count = 1

    for pattern in self.patterns:
        state = 0

        for char in pattern:
            if char not in self.goto_table[state]:
                self.goto_table[state][char] =
self.state_count
                self.goto_table[self.state_count] = {}
                self.state_count += 1

            state = self.goto_table[state][char]

        if state not in self.output_table:
```

```
        self.output_table[state] = []
        self.output_table[state].append(pattern)
```

3. _build_failure_and_output

Fungsi untuk membangun failure table dan menyalin output dari simpul failure (simpul fallbacknya).

```
def _build_failure_and_output(self):
    if self._built:
        return

    self.failure_table = {0: 0}
    queue = deque()

    for char, state in self.goto_table[0].items():
        self.failure_table[state] = 0
        queue.append(state)

    while queue:
        r = queue.popleft()

        for char, u in self.goto_table[r].items():
            queue.append(u)

            state = self.failure_table[r]
            while state != 0 and char not in
self.goto_table[state]:
                state = self.failure_table[state]

                if char in self.goto_table[state]:
                    self.failure_table[u] =
self.goto_table[state][char]
                else:
                    self.failure_table[u] = 0

    failure_state = self.failure_table[u]
```

```

        if failure_state in self.output_table:
            if u not in self.output_table:
                self.output_table[u] = []

        self.output_table[u].extend(self.output_table[failure_state])

        self._built = True
    
```

4. search

Fungsi inti dari algoritma Aho-Corasick, yaitu melakukan pencarian. Fungsi ini dipanggil setelah semua pattern sudah dimasukkan ke list dan trie sudah dibangun. Fungsi ini memanggil pembangunan failure table dan penyalinan output dari simpul failure kemudian melakukan pencarian mulai dari pattern pertama sampai pattern terakhir dalam list.

```

def search(self, text):
    if not self.patterns or not text:
        return []

    self._build_failure_and_output()

    results = []
    state = 0

    for i, char in enumerate(text):
        while state != 0 and char not in
self.goto_table[state]:
            state = self.failure_table[state]

        if char in self.goto_table[state]:
            state = self.goto_table[state][char]

        if state in self.output_table:
            for pattern in self.output_table[state]:
                start_pos = i - len(pattern) + 1
                results.append((pattern, start_pos, i))
    
```

```
    return results
```

5. aho_corasick_search

Fungsi yang menghubungkan pemanggilan dari frontend/GUI dengan kelas Aho Corasick sehingga memungkinkan penggunaan fungsi beserta prosedur privat di dalam kelas Aho Corasick.

```
def aho_corasick_search(text, patterns):
    ac = AhoCorasick()
    ac.add_patterns(patterns)

    return ac.search(text)
```

4.5 Implementasi GUI

4.5.1 Prosedur dan Fungsi

1. search_with_algorithm

Fungsi yang menjadi jembatan untuk memanggil algoritma pencocokan string spesifik (KMP, Boyer-Moore, atau Aho-Corasick) yang dipilih oleh pengguna.

```
def search_with_algorithm(text, keyword, algorithm='kmp'):
    if algorithm.lower() == 'kmp':
        return kmp_search(text, keyword)
    elif algorithm.lower() in ['boyer-moore', 'bm']:
        return boyer_moore_search(text, keyword)
    elif algorithm.lower() in ['ahocorasick', 'ac']:
        matches = aho_corasick_search(text, [keyword])
        return [start for pattern, start, end in matches if pattern ==
               keyword]
    else:
        return []
```

2. search_keywords

Fungsi inti yang mengorkestrasikan seluruh alur pencarian, menggabungkan exact dan fuzzy match, lalu mengurutkan hasilnya berdasarkan tingkat relevansi.

```
def search_keywords(keywords_input, algorithm, max_results_count):
    if not pattern_files:
        return {"results": [], "exact_time_ms": 0, "fuzzy_time_ms": 0,
"cv_count": 0}

    keywords = [kw.strip().lower() for kw in
keywords_input.split(',') if kw.strip()]
    if not keywords:
        return {"results": [], "exact_time_ms": 0, "fuzzy_time_ms": 0,
"cv_count": len(pattern_files)}

    results = []
    found_keywords_exact = set()
    exact_start_time = time.time()

    for filename, text in pattern_files:
        text_lower = text.lower()
        keyword_results = {}
        total_matches = 0
        for keyword in keywords:
            positions = search_with_algorithm(text_lower, keyword,
algorithm)
            if positions:
                found_keywords_exact.add(keyword)
                if keyword not in keyword_results:
                    keyword_results[keyword] = {'count': 0, 'type': 'exact'}
                    keyword_results[keyword]['count'] += len(positions)
                    total_matches += len(positions)
            if keyword_results:
                name, role = get_applicant_name_by_cv(filename)
                results.append({
                    "name": name,
                    "role": role,
                    "filename": filename,
```

```

        "total_matches": total_matches,
        "keyword_details": keyword_results,
        "match_type": "exact"
    })

exact_time_ms = int((time.time() - exact_start_time) * 1000)
keywords_for_fuzzy = [kw for kw in keywords if kw not in
found_keywords_exact]
fuzzy_time_ms = 0

if keywords_for_fuzzy:
    fuzzy_start_time = time.time()
    for filename, text in pattern_files:
        text_lower = text.lower()
        all_fuzzy_matches_for_cv = {}
        total_fuzzy_matches_for_cv = 0

        for keyword in keywords_for_fuzzy:
            found_matches = fuzzy_search(text_lower, keyword)
            if found_matches:
                for found_word, count in found_matches.items():
                    if found_word == keyword:
                        continue
                    if found_word not in all_fuzzy_matches_for_cv:
                        all_fuzzy_matches_for_cv[found_word] = {'count': 0,
'type': 'fuzzy'}
                        all_fuzzy_matches_for_cv[found_word]['count'] += count
                    total_fuzzy_matches_for_cv += count

            if all_fuzzy_matches_for_cv:
                existing_result = next((r for r in results if
r['filename'] == filename), None)
                if existing_result:
                    for word, details in all_fuzzy_matches_for_cv.items():
                        if word in existing_result['keyword_details']:
                            existing_result['keyword_details'][word]['count'] +=

```

```
details['count']
    else:
        existing_result['keyword_details'][word] = details
        existing_result['total_matches'] +=
total_fuzzy_matches_for_cv
        existing_result['match_type'] = 'mixed'
    else:
        name, role = get_applicant_name_by_cv(filename)
        results.append({
            "name": name,
            "role": role,
            "filename": filename,
            "total_matches": total_fuzzy_matches_for_cv,
            "keyword_details": all_fuzzy_matches_for_cv,
            "match_type": "fuzzy"
        })
fuzzy_time_ms = int((time.time() - fuzzy_start_time) * 1000)

results.sort(key=lambda x: x["total_matches"], reverse=True)
if max_results_count > 0:
    results = results[:max_results_count]

return {
    "results": results,
    "exact_time_ms": exact_time_ms,
    "fuzzy_time_ms": fuzzy_time_ms,
    "cv_count": len(pattern_files)
}
```

3. load_applicant_by_exact_filename_from_db

Fungsi untuk mengambil data profil lengkap seorang pelamar dari database menggunakan nama file sebagai kunci pencarian.

```
def load_applicant_by_exact_filename_from_db(file_number):
    applicant_data = {}
```

```

DB_CONFIG = {
    "host": "localhost",
    "user": "ats_user",
    "password": "Ats_Pass11",
    "database": "cv_ats"
}

try:
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor(dictionary=True)
    query = """
        SELECT
            ap.applicant_id, ap.first_name, ap.last_name,
            ap.date_of_birth,
            ap.address, ap.phone_number, ad.application_role,
            ad.cv_path
        FROM ApplicantProfile ap
        JOIN ApplicationDetail ad ON ap.applicant_id =
            ad.applicant_id
        WHERE REPLACE(SUBSTRING_INDEX(ad.cv_path, '/', -1), '.pdf',
        '') = %s
        LIMIT 1;
    """
    cursor.execute(query, (file_number,))
    result = cursor.fetchone()

    if result:
        applicant_data = {
            'applicant_id': result['applicant_id'],
            'first_name': result['first_name'],
            'last_name': result['last_name'],
            'full_name': f"{result['first_name']} {result['last_name']}",
            'date_of_birth': str(result['date_of_birth']),
            'address': result['address'],
            'phone_number': result['phone_number'],
        }

```

```
'role': result['application_role'],
'cv_path': result['cv_path'],
'cv_filename': file_number
}

except Exception as e:
    print(f"Database/Unexpected error: {e}")

finally:
    if 'conn' in locals() and conn.is_connected():
        cursor.close()
        conn.close()

return applicant_data
```

4. parse_cv_text_file

Fungsi untuk membaca file .txt berisi CV dan membaginya ke dalam beberapa bagian terstruktur seperti identitas, keterampilan, pendidikan, dan riwayat pekerjaan.

```
def parse_cv_text_file(filename: str) -> dict:
    parsed_data = {"Experience": "", "Education": "", "Skills": ""}
    filepath = os.path.join("data", "regex_data", filename)

    if not os.path.exists(filepath):
        print(f"File not found: {filepath}")
        return parsed_data

    try:
        with open(filepath, 'r', encoding='utf-8') as f:
            content = f.read()

            current_section = None
            for line in content.split('\n'):
                line_stripped = line.strip().lower().replace(":", "")
```

```
        if any(keyword in line_stripped for keyword in
            ["experience", "work history", "employment"]) and
            len(line_stripped.split()) < 4:
            current_section = "Experience"
        elif any(keyword in line_stripped for keyword in
            ["education", "training"]) and len(line_stripped.split()) < 4:
            current_section = "Education"
        elif "skills" in line_stripped and
            len(line_stripped.split()) < 4:
            current_section = "Skills"
        elif current_section:
            parsed_data[current_section] += line + "\n"

    except Exception as e:
        print(f"Error parsing {filename}: {e}")

    for key in parsed_data:
        parsed_data[key] = parsed_data[key].strip()

return parsed_data
```

4.6 Tata Cara Penggunaan Program

4.6.1 Interface Program dan Fitur

Ketika pertama kali membuka aplikasi ScoopyHire CV Analyzer, user akan disambut dengan Landing Page yang menampilkan judul besar "Welcome to ScoopyHire" beserta tagline aplikasi. Pada halaman ini, user hanya perlu mengklik tombol "Get Started" yang terletak di tengah layar dengan desain button coklat berukuran 200x50 pixel. Setelah tombol ini diklik, aplikasi akan otomatis beralih ke halaman pencarian utama.

Di halaman pencarian (Search Page), user akan melihat interface yang lebih lengkap dengan beberapa elemen input. Langkah pertama yang harus dilakukan adalah mengisi field "Keywords" dengan kata kunci yang ingin dicari, misalnya "python, django, mysql" (multiple keywords dapat dipisahkan dengan koma). Selanjutnya, user dapat memilih algoritma pencarian dengan mengklik salah satu dari tiga tombol: KMP (yang sudah

terpilih secara default), BM (Boyer-Moore), atau AC (Aho-Corasick). User juga dapat mengatur jumlah hasil maksimal yang ingin ditampilkan melalui field "Top results to show" (default 10). Setelah semua input diisi sesuai keinginan, user harus mengklik tombol "Search CV" untuk memulai proses pencarian. Tombol ini akan berubah menjadi "Search Again" setelah pencarian dilakukan.

Setelah proses pencarian selesai, aplikasi akan menampilkan hasil pencarian dalam bentuk kartu-kartu (result cards) yang tersusun dalam grid layout. Setiap kartu menampilkan informasi kandidat seperti nama, posisi, jenis matching (exact/fuzzy/mixed), jumlah keywords yang ditemukan, dan detail breakdown per keyword. Pada setiap kartu hasil, user memiliki dua pilihan aksi: mengklik tombol "View Details" untuk melihat ringkasan lengkap CV kandidat, atau mengklik tombol "View CV" untuk membuka file PDF CV dengan aplikasi viewer eksternal. Jika user ingin melihat detail kandidat lain, mereka dapat mengklik tombol "View Details" pada kartu hasil yang berbeda tanpa perlu kembali ke halaman pencarian.

Ketika user memilih "View Details", aplikasi akan membuka halaman Summary yang menampilkan informasi komprehensif tentang kandidat terpilih. Halaman ini dibagi menjadi beberapa section: Personal Information (nama, tanggal lahir, alamat, telepon), Skills & Expertise, Professional Experience, dan Educational Background. Semua data ini diambil dari kombinasi database MySQL dan hasil parsing CV. Dari halaman Summary, user dapat mengklik tombol "Back to Search" yang terletak di bagian atas untuk kembali ke halaman hasil pencarian sebelumnya, sehingga mereka dapat melihat kandidat lain atau melakukan pencarian baru.

Jika user ingin melakukan pencarian dengan kata kunci berbeda, mereka dapat mengklik tombol "Search Again" yang akan mereset seluruh form pencarian dan mengembalikan aplikasi ke state awal halaman pencarian. User kemudian dapat mengisi keywords baru, mengubah algoritma pencarian, atau menyesuaikan jumlah hasil maksimal sebelum melakukan pencarian ulang. Seluruh alur ini dirancang secara intuitif sehingga user dapat dengan mudah berpindah antar halaman, membandingkan hasil kandidat yang berbeda, dan melakukan multiple pencarian dalam satu sesi penggunaan aplikasi tanpa kehilangan hasil pencarian sebelumnya sampai mereka memilih untuk melakukan pencarian baru.



4.7 Hasil Pengujian dan Analisis

4.7.1 Algoritma Knuth-Morris-Pratt

4.7.1.1 Test Case 1: Exact Match - Single Keyword [“microsoft office”]

Pencarian	<p>ScoopyHire CV ANALYZER APP</p> <p>Keywords microsoft office</p> <p>Choose your preferred algorithm KMP BM AC</p> <p>Top results to show 10</p> <p>Search Again</p> <p>Found 10 Results</p> <p>Exact Match: 479 CVs scanned in 11859ms. Fuzzy Match: 0 CVs scanned in 0ms.</p>												
Hasil Pencarian	<p>Found 10 Results</p> <p>Exact Match: 479 CVs scanned in 11859ms. Fuzzy Match: 0 CVs scanned in 0ms.</p> <table border="1"> <tbody> <tr> <td>Lisa Miller 7 total matches Matched keywords: 1. Microsoft office: 7 occurrences < Summary View CV ></td> <td>Samantha Roberts 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV ></td> <td>John Smith 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV ></td> </tr> <tr> <td>Stephen Washington 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV ></td> <td>Jessica Jackson 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV ></td> <td>Daniel Lee 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV ></td> </tr> <tr> <td>Paul Henderson 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV ></td> <td>Amanda Reed 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV ></td> <td>David Wilson 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV ></td> </tr> <tr> <td colspan="2"></td> <td>Benjamin Sanchez 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV ></td> </tr> </tbody> </table>	Lisa Miller 7 total matches Matched keywords: 1. Microsoft office: 7 occurrences < Summary View CV >	Samantha Roberts 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	John Smith 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	Stephen Washington 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	Jessica Jackson 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	Daniel Lee 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >	Paul Henderson 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >	Amanda Reed 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >	David Wilson 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >			Benjamin Sanchez 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >
Lisa Miller 7 total matches Matched keywords: 1. Microsoft office: 7 occurrences < Summary View CV >	Samantha Roberts 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	John Smith 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >											
Stephen Washington 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	Jessica Jackson 4 total matches Matched keywords: 1. Microsoft office: 4 occurrences < Summary View CV >	Daniel Lee 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >											
Paul Henderson 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >	Amanda Reed 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >	David Wilson 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >											
		Benjamin Sanchez 3 total matches Matched keywords: 1. Microsoft office: 3 occurrences < Summary View CV >											

4.7.1.2 Test Case 2

Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. At the top, it says "ScoopyHire" and "CV ANALYZER APP". Below that is a "Keywords" input field containing "programmer". Underneath the input field are buttons for "Choose your preferred algorithm" (KMP, BM, AC) and "Top results to show" (5). A large brown button at the bottom says "Search Again". Below the search area, a message says "Found 2 Results". Underneath this, it says "Exact Match: 479 CVs scanned in 1382ms." and "Fuzzy Match: 479 CVs scanned in 15058ms."</p>
Hasil Pencarian	<p>The screenshot shows the search results for "programmer". It displays two cards: one for "Joshua Hall" and one for "Zachary Collins". Both cards show "2 total matches" and "Matched keywords: 1. Programmer (fuzzy): 2 occurrences" or "1. Programmer (fuzzy): 1 occurrences". Each card has a "View CV" button.</p>

4.7.1.3 Test Case 3: Multiple Keywords - Exact Match ["science, technology"]

Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. At the top, it says "ScoopyHire" and "CV ANALYZER APP". Below that is a "Keywords" input field containing "science, technology". Underneath the input field are buttons for "Choose your preferred algorithm" (KMP, BM, AC) and "Top results to show" (50). A large brown button at the bottom says "Search Again". Below the search area, a message says "Found 50 Results". Underneath this, it says "Exact Match: 479 CVs scanned in 12477ms." and "Fuzzy Match: 0 CVs scanned in 0ms."</p>
-----------	--

Hasil Pencarian	Found 50 Results Exact Match: 479 CVs scanned in 12477ms. Fuzzy Match: 0 CVs scanned in 0ms.		
	Matthew Clark 27 total matches Exact Match Matched keywords: 1. Science: 8 occurrences 2. Technology: 19 occurrences < Summary View CV >	Christopher White 19 total matches Exact Match Matched keywords: 1. Science: 4 occurrences 2. Technology: 15 occurrences < Summary View CV >	Lisa Miller 14 total matches Exact Match Matched keywords: 1. Technology: 14 occurrences < Summary View CV >
	Donald Barnes 13 total matches Exact Match Matched keywords: 1. Science: 10 occurrences 2. Technology: 3 occurrences < Summary View CV >	Eric Rivera 12 total matches Exact Match Matched keywords: 1. Technology: 12 occurrences < Summary View CV >	James Thomas 12 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 10 occurrences < Summary View CV >
	Elizabeth Watson 5 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 3 occurrences < Summary View CV >	Sarah Wood 5 total matches Exact Match Matched keywords: 1. Science: 1 occurrences 2. Technology: 4 occurrences < Summary View CV >	Larry Foster 5 total matches Exact Match Matched keywords: 1. Science: 1 occurrences 2. Technology: 4 occurrences < Summary View CV >
	Dorothy Patterson 4 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 2 occurrences < Summary View CV >	Crystal Nelson 4 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 2 occurrences < Summary View CV >	

Note: hasil yang ditampilkan hanya cuplikan bagian teratas dan terbawah

4.7.1.4 Test Case 4: Multiple Keywords - Fuzzy Match [“reacx, hmml”]

Pencarian	 CV ANALYZER APP		
	Keywords <input type="text" value="reacx, hmml"/> Choose your preferred algorithm Top results to show <input checked="" type="button" value="KMP"/> <input type="button" value="BM"/> <input type="button" value="AC"/> <input type="button" value="8"/> Search Again		
	Found 8 Results Exact Match: 479 CVs scanned in 1292ms. Fuzzy Match: 479 CVs scanned in 9869ms.		

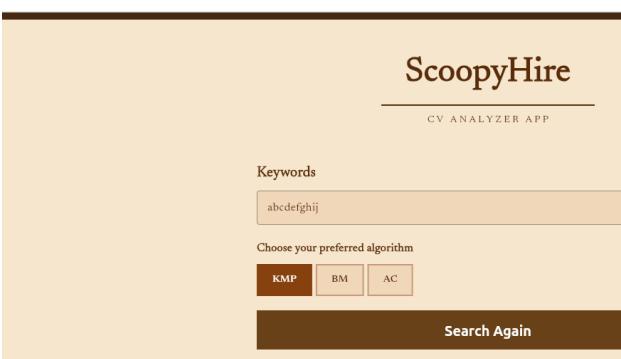
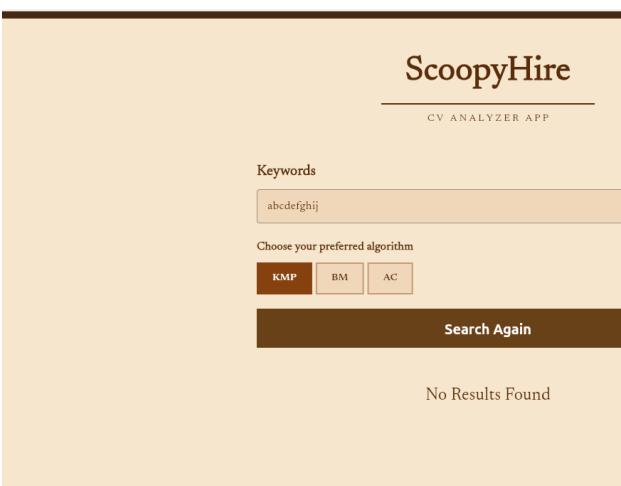
Hasil Pencarian						
	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Alexander Richardson 9 total matches Matched keywords: 1. Html (fuzzy): 9 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Amanda Taylor 3 total matches Matched keywords: 1. Html (fuzzy): 3 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Ashley Harris 3 total matches Matched keywords: 1. Html (fuzzy): 3 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>			
	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Frank Butler 3 total matches Matched keywords: 1. Reach (fuzzy): 1 occurrences 2. Html (fuzzy): 2 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Jennifer Martinez 3 total matches Matched keywords: 1. Reach (fuzzy): 1 occurrences 2. Html (fuzzy): 2 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Jeffrey Alexander 2 total matches Matched keywords: 1. Html (fuzzy): 2 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>			
	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Elizabeth Watson 2 total matches Matched keywords: 1. Html (fuzzy): 2 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> Rachel Wright 2 total matches Matched keywords: 1. Html (fuzzy): 2 occurrences </div> <div style="text-align: right; margin-top: -10px;"> View CV > </div> </div>				

4.7.1.5 Test Case 5: Multiple Keywords - Exact & Fuzzy Match [“express, exxel”]

Pencarian						
	 <p>The screenshot shows the ScoopyHire CV Analyzer App interface. The search bar contains the keywords "express, exxel". Below the search bar are buttons for "KMP", "BM", and "AC" (algorithm selection). To the right is a dropdown for "Top results to show" set to 10. A large "Search Again" button is at the bottom. Below the search area, a section titled "Found 10 Results" displays the following information: "Exact Match: 479 CVs scanned in 2393ms." and "Fuzzy Match: 479 CVs scanned in 10860ms.".</p>					

	<div style="display: flex; justify-content: space-around;"> <div style="width: 45%;"> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Alexander Richardson</p> <p>4 total matches</p> <p>Exact Match</p> <p>Matched keywords:</p> <ul style="list-style-type: none"> 1. Express: 4 occurrences <p>< Summary View CV ></p> </div> <div style="width: 45%;"> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Nancy Coleman</p> <p>4 total matches</p> <p>Mixed Match</p> <p>Matched keywords:</p> <ul style="list-style-type: none"> 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences <p>< Summary View CV ></p> </div> </div> </div> <div style="margin-top: 20px;"> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Ralph Cole</p> <p>4 total matches</p> <p>Mixed Match</p> <p>Matched keywords:</p> <ul style="list-style-type: none"> 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences <p>< Summary View CV ></p> </div> </div> </div>
--	---

4.7.1.6 Test Case 6: Random Keyword

Pencarian	
Hasil Pencarian	

4.7.2 Algoritma Boyer-Moore

4.7.2.1 Test Case 1: Exact Match - Single Keyword [“microsoft office”]

Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. In the 'Keywords' input field, 'microsoft office' is entered. Below it, there are three algorithm selection buttons: KMP, BM (which is selected), and AC. To the right, a 'Top results to show' dropdown is set to 10. A large 'Search Again' button is at the bottom. The results section below is titled 'Found 10 Results' and includes a note: 'Exact Match: 479 CVs scanned in 9457ms. Fuzzy Match: 0 CVs scanned in 0ms.' The results list contains ten entries, each with a name, total matches, and an 'Exact Match' badge.</p>
Hasil Pencarian	<p>The screenshot shows the results page from the ScoopyHire CV Analyzer App. It displays 10 job profiles, each with a name, total matches, and an 'Exact Match' badge. The profiles are: Lisa Miller (7 total matches), Samantha Roberts (4 total matches), John Smith (4 total matches), Stephen Washington (4 total matches), Jessica Jackson (4 total matches), Daniel Lee (3 total matches), Paul Henderson (3 total matches), Amanda Reed (3 total matches), David Wilson (3 total matches), and Benjamin Sanchez (3 total matches). Each profile has a 'View CV' button.</p>

4.7.2.2 Test Case 2: Fuzzy Match - Single Keyword [“programmer”]

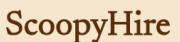
Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. At the top, it says "ScoopyHire" and "CV ANALYZER APP". Below that is a "Keywords" input field containing "programer". Underneath the input field are three algorithm selection buttons: KMP, BM (which is highlighted in dark red), and AC. To the right of these buttons is a "Top results to show" dropdown set to "5". A large brown "Search Again" button is below the algorithm selection. At the bottom, a message says "Found 2 Results" followed by two entries: "Exact Match: 479 CVs scanned in 392ms." and "Fuzzy Match: 479 CVs scanned in 14503ms."</p>
Hasil Pencarian	<p>The screenshot shows the search results for the keyword "programer". It displays two results in a grid format:</p> <ul style="list-style-type: none"> Joshua Hall: Shows 2 total matches and 1. Programmer (fuzzy): 2 occurrences. Buttons for "View CV" and "Summary" are present. Zachary Collins: Shows 1 total matches and 1. Programmer (fuzzy): 1 occurrences. Buttons for "View CV" and "Summary" are present. <p>At the top of the results, it says "Found 2 Results" and provides execution times for exact and fuzzy matches.</p>

4.7.2.3 Test Case 3: Multiple Keywords - Exact Match ["science, technology"]

Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. At the top, it says "ScoopyHire" and "CV ANALYZER APP". Below that is a "Keywords" input field containing "science, technology". Underneath the input field are three algorithm selection buttons: KMP, BM (highlighted in dark red), and AC. To the right of these buttons is a "Top results to show" dropdown set to "50". A large brown "Search Again" button is below the algorithm selection. At the bottom, a message says "Found 50 Results" followed by two entries: "Exact Match: 479 CVs scanned in 12002ms." and "Fuzzy Match: 0 CVs scanned in 0ms."</p>
-----------	--

Hasil Pencarian	Found 50 Results <small>Exact Match: 479 CVs scanned in 12002ms. Fuzzy Match: 0 CVs scanned in 0ms.</small>		
	<div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Matthew Clark Exact Match 27 total matches Matched keywords: 1. Science: 8 occurrences 2. Technology: 19 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Christopher White Exact Match 19 total matches Matched keywords: 1. Science: 4 occurrences 2. Technology: 15 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Lisa Miller Exact Match 14 total matches Matched keywords: 1. Technology: 14 occurrences < Summary View CV > </div>	<div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Donald Barnes Exact Match 13 total matches Matched keywords: 1. Science: 10 occurrences 2. Technology: 3 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Eric Rivera Exact Match 12 total matches Matched keywords: 1. Technology: 12 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> James Thomas Exact Match 12 total matches Matched keywords: 1. Science: 2 occurrences 2. Technology: 10 occurrences < Summary View CV > </div>	<div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Elizabeth Watson Exact Match 5 total matches Matched keywords: 1. Science: 2 occurrences 2. Technology: 3 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Sarah Wood Exact Match 5 total matches Matched keywords: 1. Science: 1 occurrences 2. Technology: 4 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Larry Foster Exact Match 5 total matches Matched keywords: 1. Science: 1 occurrences 2. Technology: 4 occurrences < Summary View CV > </div>
	<div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Dorothy Patterson Exact Match 4 total matches Matched keywords: 1. Science: 2 occurrences 2. Technology: 2 occurrences < Summary View CV > </div> <div style="background-color: #f2e0c7; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;"> Crystal Nelson Exact Match 4 total matches Matched keywords: 1. Science: 2 occurrences 2. Technology: 2 occurrences < Summary View CV > </div>		
	Note: hasil yang ditampilkan hanya cuplikan bagian teratas dan terbawah		

4.7.2.4 Test Case 4: Multiple Keywords - Fuzzy Match [“reacx, hmml”]

Pencarian	 <small>CV ANALYZER APP</small>		
	<div style="margin-bottom: 10px;"> Keywords <input type="text" value="reacx, hmml"/> </div> <div style="display: flex; justify-content: space-between;"> Choose your preferred algorithm Top results to show </div> <div style="display: flex; justify-content: space-around; align-items: center;"> KMP BM AC 8 </div> <div style="text-align: center; margin-top: 10px;"> Search Again </div> <div style="text-align: center; margin-top: 10px;"> Found 8 Results <small>Exact Match: 479 CVs scanned in 606ms. Fuzzy Match: 479 CVs scanned in 9932ms.</small> </div>		

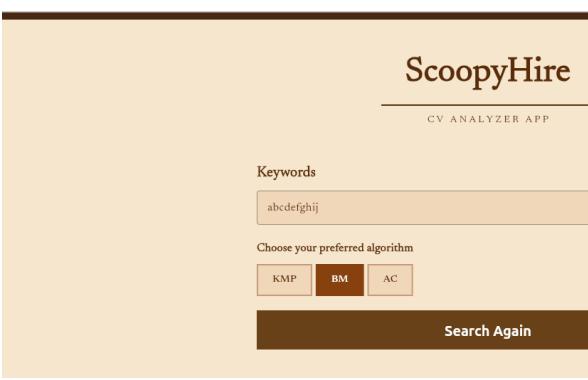
Hasil Pencarian	Alexander Richardson 9 total matches Matched keywords: 1. Html (fuzzy): 9 occurrences < Summary View CV >	Amanda Taylor 3 total matches Matched keywords: 1. Html (fuzzy): 3 occurrences < Summary View CV >	Ashley Harris 3 total matches Matched keywords: 1. Html (fuzzy): 3 occurrences < Summary View CV >
	Frank Butler 3 total matches Matched keywords: 1. Reach (fuzzy): 1 occurrences 2. Html (fuzzy): 2 occurrences < Summary View CV >	Jennifer Martinez 3 total matches Matched keywords: 1. Reach (fuzzy): 1 occurrences 2. Html (fuzzy): 2 occurrences < Summary View CV >	Jeffrey Alexander 2 total matches Matched keywords: 1. Html (fuzzy): 2 occurrences < Summary View CV >
	Elizabeth Watson 2 total matches Matched keywords: 1. Html (fuzzy): 2 occurrences < Summary View CV >	Rachel Wright 2 total matches Matched keywords: 1. Html (fuzzy): 2 occurrences < Summary View CV >	

4.7.2.5 Test Case 5: Multiple Keywords - Exact & Fuzzy Match [“express, exxl”]

Pencarian	<p>ScoopyHire CV ANALYZER APP</p> <p>Keywords: express, exxl</p> <p>Choose your preferred algorithm: KMP, BM, AC</p> <p>Top results to show: 10</p> <p>Search Again</p> <p>Found 10 Results</p> <p>Exact Match: 479 CVs scanned in 1801ms. Fuzzy Match: 479 CVs scanned in 11240ms.</p>						
Hasil Pencarian	<p>Found 10 Results</p> <p>Exact Match: 479 CVs scanned in 1801ms. Fuzzy Match: 479 CVs scanned in 11240ms.</p> <table border="1"> <tbody> <tr> <td>Sarah Johnson 9 total matches Matched keywords: 1. Express: 3 occurrences 2. Excel (fuzzy): 6 occurrences < Summary View CV ></td> <td>Jose Howard 7 total matches Matched keywords: 1. Excel (fuzzy): 7 occurrences < Summary View CV ></td> <td>Virginia Hamilton 6 total matches Matched keywords: 1. Express: 2 occurrences 2. Excel (fuzzy): 4 occurrences < Summary View CV ></td> </tr> <tr> <td>Heather Green 6 total matches Matched keywords: 1. Excel (fuzzy): 6 occurrences < Summary View CV ></td> <td>Christina Ward 5 total matches Matched keywords: 1. Excel (fuzzy): 5 occurrences < Summary View CV ></td> <td>Justin Adams 5 total matches Matched keywords: 1. Excel (fuzzy): 5 occurrences < Summary View CV ></td> </tr> </tbody> </table>	Sarah Johnson 9 total matches Matched keywords: 1. Express: 3 occurrences 2. Excel (fuzzy): 6 occurrences < Summary View CV >	Jose Howard 7 total matches Matched keywords: 1. Excel (fuzzy): 7 occurrences < Summary View CV >	Virginia Hamilton 6 total matches Matched keywords: 1. Express: 2 occurrences 2. Excel (fuzzy): 4 occurrences < Summary View CV >	Heather Green 6 total matches Matched keywords: 1. Excel (fuzzy): 6 occurrences < Summary View CV >	Christina Ward 5 total matches Matched keywords: 1. Excel (fuzzy): 5 occurrences < Summary View CV >	Justin Adams 5 total matches Matched keywords: 1. Excel (fuzzy): 5 occurrences < Summary View CV >
Sarah Johnson 9 total matches Matched keywords: 1. Express: 3 occurrences 2. Excel (fuzzy): 6 occurrences < Summary View CV >	Jose Howard 7 total matches Matched keywords: 1. Excel (fuzzy): 7 occurrences < Summary View CV >	Virginia Hamilton 6 total matches Matched keywords: 1. Express: 2 occurrences 2. Excel (fuzzy): 4 occurrences < Summary View CV >					
Heather Green 6 total matches Matched keywords: 1. Excel (fuzzy): 6 occurrences < Summary View CV >	Christina Ward 5 total matches Matched keywords: 1. Excel (fuzzy): 5 occurrences < Summary View CV >	Justin Adams 5 total matches Matched keywords: 1. Excel (fuzzy): 5 occurrences < Summary View CV >					

	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Alexander Richardson 4 total matches Matched keywords: 1. Express: 4 occurrences View CV > </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Nancy Coleman 4 total matches Matched keywords: 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences View CV > </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Ralph Cole 4 total matches Matched keywords: 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences View CV > </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Alexander Richardson 4 total matches Matched keywords: 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences View CV > </div> </div>
--	--

4.7.2.6 Test Case 6: Random Keyword

Pencarian	
Hasil Pencarian	

4.7.3 Algoritma Aho-Corasick

4.7.3.1 Test Case 1: Exact Match - Single Keyword [“microsoft office”]

Pencarian	<p>The screenshot shows the ScoopyHire CV ANALYZER APP interface. In the 'Keywords' input field, 'microsoft office' is entered. Below it, under 'Choose your preferred algorithm', 'AC' (Aho-Corasick) is selected. To the right, 'Top results to show' is set to 10. A large brown 'Search Again' button is at the bottom. The results section below says 'Found 10 Results' and includes a note: 'Exact Match: 479 CVs scanned in 11105ms. Fuzzy Match: 0 CVs scanned in 0ms.'</p>
Hasil Pencarian	<p>The screenshot shows the search results page with 10 job profiles listed. Each profile card includes the name, total matches (all exact), and a green 'Exact Match' badge. The profiles are: Lisa Miller (7 matches), Samantha Roberts (4 matches), John Smith (4 matches), Stephen Washington (4 matches), Jessica Jackson (4 matches), Daniel Lee (3 matches), Paul Henderson (3 matches), Amanda Reed (3 matches), David Wilson (3 matches), and Benjamin Sanchez (3 matches). Each card has a 'View CV' button.</p>

4.7.3.2 Test Case 2: Fuzzy Match - Single Keyword [“programer”]

Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. In the search bar, the keyword 'programmer' is entered. Below the search bar, there are buttons for choosing an algorithm: KMP, BM, and AC (which is selected). A dropdown menu for 'Top results to show' is set to 5. A large brown button at the bottom says 'Search Again'. Below the search area, a message says 'Found 2 Results'. Underneath, it states 'Exact Match: 479 CVs scanned in 338ms.' and 'Fuzzy Match: 479 CVs scanned in 6933ms.' Two search results are displayed in boxes: 'Joshua Hall' and 'Zachary Collins', each with a 'View CV' button.</p>
Hasil Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. In the search bar, the keywords 'science, technology' are entered. Below the search bar, there are buttons for choosing an algorithm: KMP, BM, and AC (which is selected). A dropdown menu for 'Top results to show' is set to 50. A large brown button at the bottom says 'Search Again'. Below the search area, a message says 'Found 50 Results'. Underneath, it states 'Exact Match: 479 CVs scanned in 11157ms.' and 'Fuzzy Match: 0 CVs scanned in 0ms.' A single search result is displayed in a box: 'science, technology' with a 'View CV' button.</p>

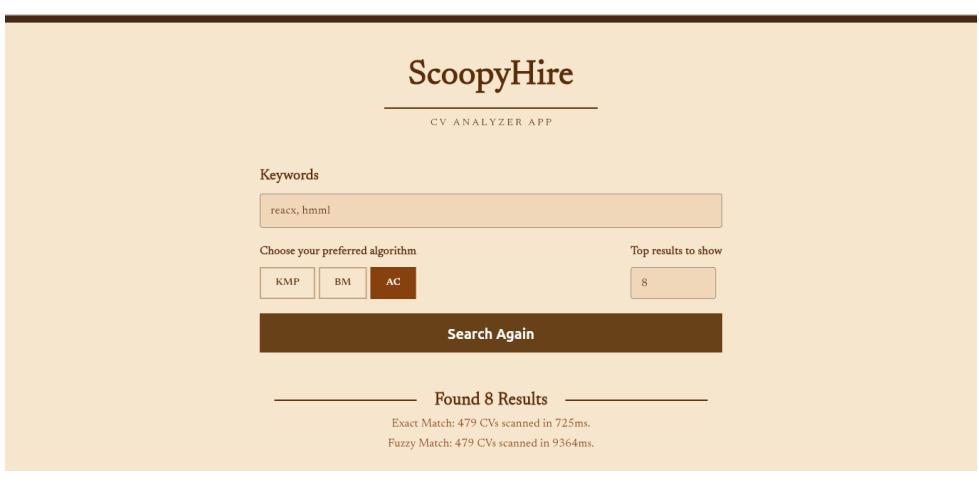
4.7.3.3 Test Case 3: Multiple Keywords - Exact Match [“science, technology”]

Pencarian	<p>The screenshot shows the ScoopyHire CV Analyzer App interface. In the search bar, the keywords 'science, technology' are entered. Below the search bar, there are buttons for choosing an algorithm: KMP, BM, and AC (which is selected). A dropdown menu for 'Top results to show' is set to 50. A large brown button at the bottom says 'Search Again'. Below the search area, a message says 'Found 50 Results'. Underneath, it states 'Exact Match: 479 CVs scanned in 11157ms.' and 'Fuzzy Match: 0 CVs scanned in 0ms.' A single search result is displayed in a box: 'science, technology' with a 'View CV' button.</p>
-----------	---

Hasil Pencarian	Found 50 Results <small>Exact Match: 479 CVs scanned in 11157ms. Fuzzy Match: 0 CVs scanned in 0ms.</small>		
	Matthew Clark 27 total matches Exact Match Matched keywords: 1. Science: 8 occurrences 2. Technology: 19 occurrences View CV >	Christopher White 19 total matches Exact Match Matched keywords: 1. Science: 4 occurrences 2. Technology: 15 occurrences View CV >	Lisa Miller 14 total matches Exact Match Matched keywords: 1. Technology: 14 occurrences View CV >
	Donald Barnes 13 total matches Exact Match Matched keywords: 1. Science: 10 occurrences 2. Technology: 3 occurrences View CV >	Eric Rivera 12 total matches Exact Match Matched keywords: 1. Technology: 12 occurrences View CV >	James Thomas 12 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 10 occurrences View CV >
	Elizabeth Watson 5 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 3 occurrences View CV >	Sarah Wood 5 total matches Exact Match Matched keywords: 1. Science: 1 occurrences 2. Technology: 4 occurrences View CV >	Larry Foster 5 total matches Exact Match Matched keywords: 1. Science: 1 occurrences 2. Technology: 4 occurrences View CV >
	Dorothy Patterson 4 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 2 occurrences View CV >	Crystal Nelson 4 total matches Exact Match Matched keywords: 1. Science: 2 occurrences 2. Technology: 2 occurrences View CV >	

Note: hasil yang ditampilkan hanya cuplikan bagian teratas dan terbawah

4.7.3.4 Test Case 4: Multiple Keywords - Fuzzy Match [“reacx, hmml”]

Pencarian	
	ScoopyHire <small>CV ANALYZER APP</small> <div style="margin-top: 20px;"> Keywords <input type="text" value="reacx, hmml"/> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> Choose your preferred algorithm <div style="display: flex;"> <input type="button" value="KMP"/> <input type="button" value="BM"/> <input style="background-color: #800000; color: white; font-weight: bold; border: 1px solid black; padding: 2px 5px; margin-left: 10px;" type="button" value="AC"/> </div> Top results to show <div style="border: 1px solid #ccc; padding: 2px 5px; margin-left: 10px;">8</div> </div> <div style="text-align: center; margin-top: 10px;"> <input style="background-color: #800000; color: white; font-weight: bold; border: 1px solid black; padding: 5px 20px;" type="button" value="Search Again"/> </div> <div style="text-align: center; margin-top: 20px;"> Found 8 Results <small>Exact Match: 479 CVs scanned in 725ms. Fuzzy Match: 479 CVs scanned in 9364ms.</small> </div>

Hasil Pencarian							
Alexander Richardson	9 total matches	Fuzzy Match	< Summary	View CV >	Amanda Taylor	3 total matches	Fuzzy Match
Matched keywords:	1. Html (fuzzy): 9 occurrences				Matched keywords:	1. Html (fuzzy): 3 occurrences	
Frank Butler	3 total matches	Fuzzy Match	< Summary	View CV >	Jennifer Martinez	3 total matches	Fuzzy Match
Matched keywords:	1. Reach (fuzzy): 1 occurrences 2. Html (fuzzy): 2 occurrences				Matched keywords:	1. Reach (fuzzy): 1 occurrences 2. Html (fuzzy): 2 occurrences	
Elizabeth Watson	2 total matches	Fuzzy Match	< Summary	View CV >	Rachel Wright	2 total matches	Fuzzy Match
Matched keywords:	1. Html (fuzzy): 2 occurrences				Matched keywords:	1. Html (fuzzy): 2 occurrences	
Ashley Harris	3 total matches	Fuzzy Match	< Summary	View CV >	Jeffrey Alexander	2 total matches	Fuzzy Match
Matched keywords:	1. Html (fuzzy): 3 occurrences				Matched keywords:	1. Html (fuzzy): 2 occurrences	

4.7.3.5 Test Case 5: Multiple Keywords - Exact & Fuzzy Match [“express, exxel”]

Pencarian																																																																			
	<p align="center">ScoopyHire CV ANALYZER APP</p> <p>Keywords express, exxel</p> <p>Choose your preferred algorithm <input type="checkbox"/> KMP <input type="checkbox"/> BM <input checked="" type="checkbox"/> AC</p> <p>Top results to show 10</p> <p align="center">Search Again</p> <p align="center">Found 10 Results</p> <p>Exact Match: 479 CVs scanned in 1817ms. Fuzzy Match: 479 CVs scanned in 10374ms.</p>																																																																		
Hasil Pencarian	<p align="center">Found 10 Results</p> <p>Exact Match: 479 CVs scanned in 1817ms. Fuzzy Match: 479 CVs scanned in 10374ms.</p> <table border="1"> <tbody> <tr> <td>Sarah Johnson</td><td>9 total matches</td><td>Mixed Match</td><td>< Summary</td><td>View CV ></td><td>Jose Howard</td><td>7 total matches</td><td>Fuzzy Match</td><td>< Summary</td><td>View CV ></td><td>Virginia Hamilton</td><td>6 total matches</td><td>Mixed Match</td><td>< Summary</td><td>View CV ></td></tr> <tr> <td>Matched keywords:</td><td>1. Express: 3 occurrences 2. Excel (fuzzy): 6 occurrences</td><td></td><td></td><td></td><td>Matched keywords:</td><td>1. Excel (fuzzy): 7 occurrences</td><td></td><td></td><td></td><td>Matched keywords:</td><td>1. Express: 2 occurrences 2. Excel (fuzzy): 4 occurrences</td><td></td><td></td><td></td></tr> <tr> <td>Heather Green</td><td>6 total matches</td><td>Fuzzy Match</td><td>< Summary</td><td>View CV ></td><td>Christina Ward</td><td>5 total matches</td><td>Fuzzy Match</td><td>< Summary</td><td>View CV ></td><td>Justin Adams</td><td>5 total matches</td><td>Fuzzy Match</td><td>< Summary</td><td>View CV ></td></tr> <tr> <td>Matched keywords:</td><td>1. Excel (fuzzy): 6 occurrences</td><td></td><td></td><td></td><td>Matched keywords:</td><td>1. Excel (fuzzy): 5 occurrences</td><td></td><td></td><td></td><td>Matched keywords:</td><td>1. Excel (fuzzy): 5 occurrences</td><td></td><td></td><td></td></tr> </tbody> </table>							Sarah Johnson	9 total matches	Mixed Match	< Summary	View CV >	Jose Howard	7 total matches	Fuzzy Match	< Summary	View CV >	Virginia Hamilton	6 total matches	Mixed Match	< Summary	View CV >	Matched keywords:	1. Express: 3 occurrences 2. Excel (fuzzy): 6 occurrences				Matched keywords:	1. Excel (fuzzy): 7 occurrences				Matched keywords:	1. Express: 2 occurrences 2. Excel (fuzzy): 4 occurrences				Heather Green	6 total matches	Fuzzy Match	< Summary	View CV >	Christina Ward	5 total matches	Fuzzy Match	< Summary	View CV >	Justin Adams	5 total matches	Fuzzy Match	< Summary	View CV >	Matched keywords:	1. Excel (fuzzy): 6 occurrences				Matched keywords:	1. Excel (fuzzy): 5 occurrences				Matched keywords:	1. Excel (fuzzy): 5 occurrences			
Sarah Johnson	9 total matches	Mixed Match	< Summary	View CV >	Jose Howard	7 total matches	Fuzzy Match	< Summary	View CV >	Virginia Hamilton	6 total matches	Mixed Match	< Summary	View CV >																																																					
Matched keywords:	1. Express: 3 occurrences 2. Excel (fuzzy): 6 occurrences				Matched keywords:	1. Excel (fuzzy): 7 occurrences				Matched keywords:	1. Express: 2 occurrences 2. Excel (fuzzy): 4 occurrences																																																								
Heather Green	6 total matches	Fuzzy Match	< Summary	View CV >	Christina Ward	5 total matches	Fuzzy Match	< Summary	View CV >	Justin Adams	5 total matches	Fuzzy Match	< Summary	View CV >																																																					
Matched keywords:	1. Excel (fuzzy): 6 occurrences				Matched keywords:	1. Excel (fuzzy): 5 occurrences				Matched keywords:	1. Excel (fuzzy): 5 occurrences																																																								

	<div style="background-color: #f0e6d2; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Alexander Richardson 4 total matches Exact Match Matched keywords: 1. Express: 4 occurrences View CV > </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Nancy Coleman 4 total matches Mixed Match Matched keywords: 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences View CV > </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Ralph Cole 4 total matches Mixed Match Matched keywords: 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences View CV > </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Alexander Richardson 4 total matches Mixed Match Matched keywords: 1. Express: 1 occurrences 2. Excel (fuzzy): 3 occurrences View CV > </div> </div>
--	--

4.7.3.6 Test Case 6: Random Keyword

Pencarian	
Hasil Pencarian	

4.7.4 Analisis Hasil Pengujian

Dari hasil pengujian, meskipun terdapat keragaman waktu eksekusi akibat faktor eksternal, dapat disimpulkan bahwa algoritma Boyer-Moore secara konsisten menunjukkan kinerja paling efisien dalam skenario pencarian satu kata kunci. Kinerja ini kemudian disusul oleh algoritma Aho-Corasick, yang sedikit lebih unggul dibandingkan Knuth-Morris-Pratt. Hal ini karena Boyer-Moore mengadopsi pendekatan heuristik yang cerdas, yaitu dengan menggunakan tabel Bad Character atau dikenal juga sebagai *last occurrence*, algoritma ini mampu melakukan lompatan-lompatan besar pada teks, melewati segmen yang sudah pasti tidak akan cocok. Mekanisme ini secara drastis mengurangi jumlah perbandingan karakter yang diperlukan sehingga sangat efektif untuk pencarian pola tunggal.

Sebaliknya, dalam kasus pencarian lebih dari satu kata kunci, Aho-Corasick menjadi algoritma lebih superior. Keunggulannya menjadi semakin dominan seiring dengan bertambahnya jumlah kata kunci yang dicari. Hal ini menunjukkan kekuatan Aho-Corasick sebagai algoritma multi-pattern matching. Prinsip kerjanya yang hanya memerlukan satu kali pemindaian teks (*single pass*) untuk menemukan semua pola secara simultan, membuatnya jauh lebih efisien daripada Boyer-Moore dan Knuth-Morris-Pratt yang harus melakukan pemindaian berulang untuk setiap kata kunci.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Aplikasi ScoopyHire berhasil diimplementasikan dengan baik untuk dapat melakukan pattern matching dalam membangun sistem ATS (*Applicant Tracking System*) berbasis CV digital. Aplikasi ini mampu mengolah data CV dalam bentuk pdf dan mengekstrak informasi di dalamnya menjadi file txt yang nantinya akan diolah dengan algoritma pattern matching dan regex untuk mencari applicant paling relevan dengan kata kunci yang dicari. Dalam tugas besar ini diimplementasikan tiga macam algortima string matching. Algoritma dengan performa paling optimal adalah Aho-Corasick dengan kompleksitas waktu $O(n + m + z)$ dimana n adalah panjang teks, m adalah total panjang semua pattern, dan z adalah jumlah kemunculan pattern. Untuk pencarian multiple keywords, Aho-Corasick menunjukkan performa superior karena dapat mencari semua pattern secara simultan dalam satu pass, sedangkan KMP dan Boyer-Moore harus melakukan pencarian terpisah untuk setiap keyword.

Untuk detail pribadi applicant juga telah diimplementasikan dengan database sql yang akan di mapping dengan data CV untuk ditampilkan di resume CV. Tampilan interface aplikasi dibuat dengan menggunakan flet. Alur penggunaan yang cukup sederhana dan UI yang informatif akan mempermudah pengguna dalam menggunakan aplikasi ini.

5.2 Saran

Terdapat beberapa hal yang penulis sadari dan dapat dijadikan sebagai bahan pembelajaran ke depan, diantaranya :

1. Implementasi regex untuk mencari kata yang relevan dengan kategori sebaiknya dibuat lebih detail supaya mampu menangani *edge case* dimana kategori dalam cv tersebut memiliki nama yang berbeda namun masih dalam satu konteks.
2. Parsing file txt sebaiknya bisa dilakukan dengan lebih presisi supaya data yang diambil memang merupakan data yang paling relevan untuk ditampilkan.
3. Optimasi memory usage dengan streaming processing untuk file CV berukuran besar
4. Implementasi advanced search filters (berdasarkan pengalaman kerja, pendidikan, lokasi, dll.)

LAMPIRAN

- Tautan Repository GitHub : https://github.com/rararana/Tubes3_Scoopy.git
- Tautan Video Youtube : <https://youtu.be/v24K0SIIxdk?si=PTRA5-RfAGTXUdY1>

No	Poin	Ya	Tidak
1	Aplikasi dapat dijalankan.	v	
2	Aplikasi menggunakan basis data berbasis SQL dan berjalan dengan lancar.	v	
3	Aplikasi dapat mengekstrak informasi penting menggunakan Regular Expression (Regex).	v	
4	Algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) dapat menemukan kata kunci dengan benar.	v	
5	Algoritma Levenshtein Distance dapat mengukur kemiripan kata kunci dengan benar.	v	
6	Aplikasi dapat menampilkan <i>summary CV applicant</i> .	v	
7	Aplikasi dapat menampilkan <i>CV applicant</i> secara keseluruhan.	v	
8	Membuat laporan sesuai dengan spesifikasi.	v	
9	Membuat bonus enkripsi data profil <i>applicant</i> .		v
10	Membuat bonus algoritma Aho-Corasick.	v	
11	Membuat bonus video dan diunggah pada Youtube.	v	

DAFTAR PUSTAKA

Munir, Rinaldi. 2025. “Pencocokan String (String/Pattern Matching).” [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf).

Munir, Rinaldi. 2025. “String Matching dengan Regular Expression.” [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-(2025).pdf).