

Laporan Tugas Besar 3

IF2211 Strategi Algoritma 2025/2026



Disusun oleh:

Kelas K1 - Kelompok - Chisli

Yonatan Edward Njoto (13523036)

Benedict Presley (13523067)

Daniel Pedrosa Wu (13523099)

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

KATA PENGANTAR

Dengan penuh rasa syukur, kami menyusun dan mempersembahkan makalah ini sebagai bagian dari Tugas Besar 3 mata kuliah IF2211 Strategi Algoritma. Topik yang diangkat dalam tugas besar ini adalah “*Pemanfaatan Pattern Matching untuk Membangun Sistem ATS (Applicant Tracking System) Berbasis CV Digital*”, yang bertujuan untuk menerapkan dan mengeksplorasi efektivitas algoritma pattern matching dalam mengotomatisasi proses seleksi dan pemeringkatan kandidat.

Melalui tugas ini, kami mengembangkan sebuah sistem ATS fungsional yang mampu mem-parsing, menganalisis, dan mencocokkan informasi dari CV digital dengan kualifikasi pekerjaan yang telah ditentukan. Dengan memanfaatkan berbagai teknik pattern matching, sistem ini dapat mengekstraksi data kunci secara akurat seperti keahlian, pengalaman kerja, dan latar belakang pendidikan untuk kemudian memberikan skor relevansi pada setiap kandidat secara otomatis.

Pengerjaan proyek ini tidak hanya memberikan kesempatan bagi kami untuk memperdalam pemahaman mengenai teori algoritma pattern matching, tapi juga menantang kami dalam mengaplikasikannya untuk mengelola data tidak terstruktur dalam konteks nyata. Melalui proses ini, kami belajar untuk menggabungkan aspek teoretis dan praktis secara seimbang, serta mengeksplorasi penerapan algoritma dalam ranah teknologi sumber daya manusia (HR Tech) yang inovatif.

Kami mengucapkan terima kasih kepada dosen pengampu serta seluruh pihak yang telah membimbing dan memberikan dukungan selama penyusunan tugas ini. Semoga laporan ini dapat memberikan gambaran yang jelas mengenai hubungan antara strategi algoritma dan pengembangan aplikasi nyata, serta menjadi referensi yang bermanfaat bagi pembaca yang tertarik dalam bidang algoritma, pengolahan data, dan rekayasa perangkat lunak.

Bandung, 12 Mei 2025

Yonatan Edward Njoto (13523036)

Benedict Presley (13523067)

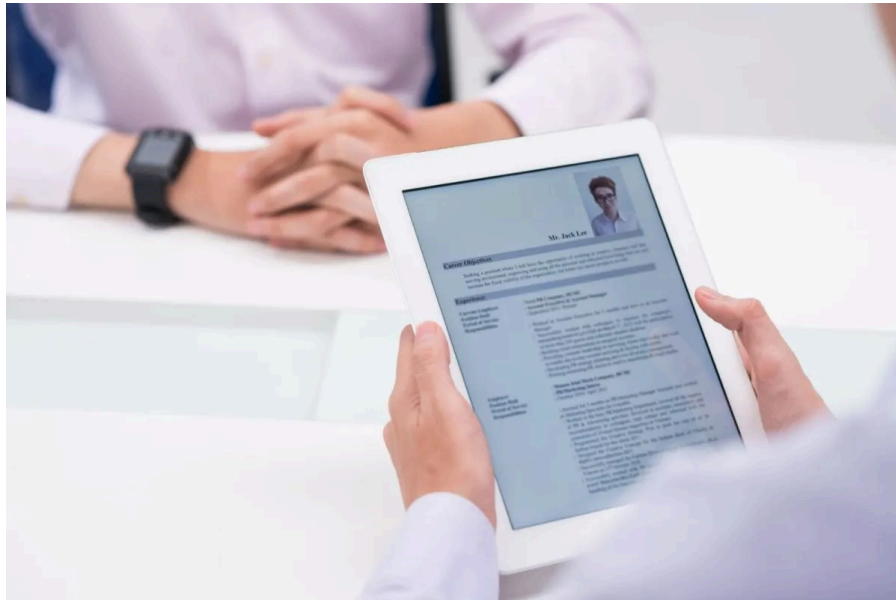
Daniel Pedrosa Wu (13523099)

DAFTAR ISI

KATA PENGANTAR.....	2
DAFTAR ISI.....	3
BAB I: Deskripsi Tugas.....	5
BAB II: Landasan Teori.....	7
2.1. String Matching.....	7
2.1.1. Algoritma Knuth-Morris-Pratt.....	7
2.1.2. Algoritma Boyer-Moore.....	8
2.1.3. Algoritma Aho-Corasick.....	9
2.2. Regex (Regular Expression).....	10
2.3. Metode Pencarian.....	11
2.3.1. Exact Matching.....	11
2.3.2. Fuzzy Matching.....	11
2.4. Teknologi Pendukung.....	12
2.4.1. Antarmuka Pengguna dengan Flet.....	12
2.4.2. Pemrosesan dan Ekstraksi CV.....	13
2.4.3. Manajemen Basis Data dengan MySQL dan SQLAlchemy.....	13
2.4.4. Mesin Pencari.....	14
BAB III: Analisis Pemecahan Masalah.....	16
3.1. Langkah-Langkah Pemecahan Masalah.....	16
3.1.1. Strukturisasi Basis Data.....	16
3.1.2. Transformasi Data.....	17
3.1.3. Enkripsi Data.....	18
3.1.4. Pengembangan Mesin Pencari.....	19
3.1.5. Pemeringkatan Hasil.....	19
3.1.6. Inisialisasi dan Pemeliharaan Sistem.....	19
3.2. Pemetaan Masalah Menjadi Elemen Algoritma Pencocokan String.....	19
3.3. Fitur Fungsional dan Arsitektur Aplikasi.....	21
3.3.1. Arsitektur Perangkat Lunak Berbasis MVC (Model-View-Controller) yang Dikembangkan.....	21
3.3.2. Fitur Fungsional.....	22
3.4. Ilustrasi Kasus.....	23
3.4.1. Ilustrasi Kasus Knuth-Morris-Pratt.....	23
3.4.2. Ilustrasi Kasus Boyer-Moore.....	25
3.4.3. Ilustrasi Kasus Aho-Corasick.....	26
BAB IV: Implementasi dan Pengujian.....	29
4.1. Spesifikasi Teknik Program.....	29
4.2. Tata Cara Penggunaan Program.....	34
4.3. Hasil Pengujian.....	36
4.3.1 Menggunakan Knuth-Morris-Pratt.....	36

4.3.2 Menggunakan Boyer-Moore.....	40
4.3.3 Menggunakan Aho-Corasick.....	44
4.4. Analisis.....	47
BAB V: Kesimpulan, Saran, dan Refleksi.....	49
5.1. Kesimpulan.....	49
5.2. Saran.....	50
5.3. Refleksi.....	50
LAMPIRAN.....	51
DAFTAR PUSTAKA.....	52

BAB I: Deskripsi Tugas



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.

BAB II: Landasan Teori

2.1. String Matching

String matching adalah proses pencarian satu atau lebih kemunculan sebuah string pendek atau *pattern* (P) di dalam sebuah string yang lebih panjang atau *text* (T). Secara formal, *string matching* dapat didefinisikan sebagai berikut:

- Diberikan sebuah *text* T yang merupakan sebuah (*long*) array karakter atau *string* yang terdiri dari n karakter $T[1... n]$.
- Diberikan sebuah *pattern* P yang merupakan sebuah array karakter atau *string* yang terdiri dari m karakter $P[1... m]$, di mana $m \leq n$.
- Tujuannya adalah untuk menemukan sebuah indeks S (*valid shift*) di dalam *text* T di mana $T[s + 1... s + m]$ sama dengan $P[1... m]$.

String matching memiliki aplikasi di berbagai bidang seperti pencarian di dalam editor teks, *web search engine* (seperti Google), analisis citra, dan bioinformatika. Pendekatan *string matching* secara naif atau *brute-force* bekerja dengan cara memeriksa setiap kemungkinan posisi dari pola di dalam teks satu per satu. Jika teks sangat panjang, maka proses ini akan memakan waktu yang sangat lama sehingga algoritmanya tidak efisien. Untuk mengatasi kelemahan ini, terdapat berbagai algoritma yang dikembangkan, tiga di antaranya yakni:

2.1.1. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt mencari *pattern* P di dalam teks dari bagian kiri P ke bagian kanan P . Ide dasar dari algoritma KMP adalah fakta bahwa saat terjadi ketidakcocokan, maka sudah diketahui beberapa karakter yang akan diperiksa di jendela berikutnya. Informasi ini digunakan untuk menghindari proses pemrosesan redundan dengan menggeser pola lebih dari satu langkah. KMP menggunakan informasi *prefix* pola yang sudah cocok untuk menentukan pergeseran terjauh yang mungkin dilakukan tanpa melewatkan potensi kecocokan.

Algoritma ini menghindari perbandingan yang redundan dengan bantuan dari sebuah fungsi yang dikenal sebagai *border function* $b(k)$. Fungsi ini dihitung sebelum proses pencarian terjadi. Misalkan j adalah posisi terjadi ketidakcocokan dalam P dan k adalah posisi sebelum terjadi ketidakcocokan tersebut

($k = j - 1$), maka *border function* $b(k)$ didefinisikan sebagai ukuran dari *prefix* terbesar dari $P[0 \dots k]$, yang juga merupakan *suffix* dari $P[1 \dots k]$.

Saat terjadi ketidakcocokan antara karakter pada $T[i]$ dan $P[j]$, maka nilai dari *border function* $b(k)$ akan digunakan untuk posisi sebelum ketidakcocokan $k = j - 1$. Nilai dari $b(k)$ ini akan memberitahu posisi baru untuk penunjuk pola. Penunjuk pola j akan mundur sebesar $b(k)$, sementara penunjuk teks akan diam. Secara praktis, pola akan digeser sebesar $j - b(k)$ indeks. Langkah ini akan menggeser pola ke posisi terbaik berikutnya tanpa perlu memeriksa ulang karakter yang sudah pasti cocok sehingga mengurangi perbandingan yang sia-sia.

2.1.2. Algoritma Boyer-Moore

Berbeda dengan algoritma Knuth-Morris-Pratt, algoritma Boyer-Moore mencari pattern P di dalam teks dari bagian kanan P ke bagian kiri P . Ide dasar dari algoritma BM adalah ketika terjadi ketidakcocokan, informasi dari karakter di dalam teks dapat digunakan untuk melakukan lompatan yang besar. Lompatan yang dilakukan seringkali melebihi satu langkah sehingga pencariannya cukup cepat.

Algoritma ini memanfaatkan sebuah tabel bantu yang dikenal dengan nama *Last Occurrence Table*. Tabel ini dihitung sebelum proses pencarian terjadi dan menyimpan posisi kemunculan terakhir dari setiap karakter yang ada di dalam P . Pergeseran tetap dilakukan dari kiri ke kanan, tetapi perbandingan antara pola dan teks dilakukan dari kanan ke kiri. Saat terjadi ketidakcocokan antara karakter pada $T[i]$ dan $P[j]$, maka nilai dari tabel tersebut digunakan. Ada dua skenario umum yang terjadi, yaitu:

- Karakter penyebab ketidakcocokan ada di dalam pola

Jika $T[i]$ ada di dalam *Last Occurrence Table*, maka pola akan digeser ke kanan sedemikian sehingga kemunculan terakhir dari karakter $T[i]$ akan sejajar dengan $T[i]$ yang ada di dalam teks. Jika hal itu tidak memungkinkan (seperti kemunculan terakhir dari karakter $T[i]$ berada di sebelah kanan dari posisi ketidakcocokan saat ini, maka pola akan digeser ke kanan sebanyak satu karakter.

- Karakter penyebab ketidakcocokan tidak ada di dalam pola
Jika $T[i]$ tidak ada di dalam *Bad Character Table*, maka diketahui bahwa tidak mungkin ada kecocokan yang melibatkan posisi $T[i]$ saat ini. Oleh karena itu, seluruh pola dapat digeser melewati $T[i]$.

2.1.3. Algoritma Aho–Corasick

Berbeda dengan algoritma Knuth-Morris-Pratt dan Boyer-Moore yang dirancang untuk mencari satu pola, algoritma Aho-Corasick paling cocok digunakan untuk mencari banyak pola secara bersamaan di dalam sebuah teks dalam satu kali proses. Ide dasar dari algoritma Aho-Corasick adalah menggabungkan semua pola ke dalam sebuah struktur data tunggal yang menyerupai mesin pencari (*finite automaton*), dan kemudian memproses teks hanya satu kali untuk menemukan semua kemunculan dari semua pola.

Algoritma ini memanfaatkan sebuah struktur data gabungan yang terdiri dari Trie (Prefix Tree) dan Failure Links. Struktur ini dihitung sebelum proses pencarian terjadi. Trie digunakan untuk menyimpan semua pola secara efisien, di mana setiap jalur dari akar merepresentasikan *prefix* dari satu atau lebih pola. Sementara itu, *Failure Links* adalah cara cerdas yang menghubungkan node-node di dalam Trie. Saat proses pencarian, algoritma membaca teks karakter per karakter dan bergerak di sepanjang Trie. Ada dua skenario transisi utama yang terjadi, yaitu:

- Karakter Ditemukan di Jalur Trie (Success Transition)
Jika karakter berikutnya dari T memiliki jalur yang sah dari node saat ini, algoritma akan maju ke node selanjutnya. Setiap kali algoritma tiba di sebuah node, ia akan memeriksa apakah node tersebut menandai akhir dari sebuah pola (atau beberapa pola). Jika iya, sebuah kecocokan ditemukan dan dilaporkan, dan proses terus berjalan untuk menemukan kecocokan lainnya.
- Karakter Tidak Ditemukan di Jalur Trie (Failure Transition)
Jika karakter berikutnya dari T tidak memiliki jalur dari node saat ini, maka algoritma akan mengikuti *failure link* ke node lain yang merepresentasikan awalan terpanjang berikutnya yang mungkin cocok,

tanpa harus memundurkan penunjuk di dalam teks. Dari node baru ini, algoritma mencoba kembali melakukan transisi dengan karakter Teks yang sama. Proses ini memastikan tidak ada karakter di Teks yang terlewatkan dan pencarian tetap efisien.

2.2. Regex (Regular Expression)

Regular Expression atau Regex adalah sebuah bahasa formal yang digunakan untuk mengidentifikasi, mencocokkan, atau memanipulasi suatu *string* berdasarkan aturan sintaktis yang telah ditentukan. Regex dirancang untuk menangani pencarian yang bersifat kompleks dan abstrak, di mana struktur dari *string* diketahui namun kontennya bervariasi. Kekuatan ekspresif dari Regex berasal dari sintaksnya yang menggabungkan dua komponen, yaitu:

- Karakter Literal, adalah karakter standar yang merepresentasikan dirinya sendiri. Pola yang hanya terdiri dari karakter literal berfungsi selayaknya pencocokan string biasa.
- Metakarakter, merupakan karakter ataupun susunan karakter yang tidak merepresentasikan dirinya sendiri secara harfiah. Metakarakter memungkinkan pendefinisian pola yang dinamis. Terdapat empat klasifikasi metakarakter yaitu:
 - Kelas Karakter, yang mendefinisikan satu set karakter yang diizinkan untuk suatu posisi. Contohnya, `\d`, `\w`, `[abc]`, dan sebagainya.
 - Pengukur Kuantitas, yang menentukan jumlah kemunculan dari karakter, grup, atau kelas karakter yang terletak sebelum pengukur kuantitas tersebut. Contohnya, `*`, `+`, dan `?`.
 - Penanda Posisi, yang menentukan posisi spesifik dalam string di mana kecocokan terjadi. Contohnya, `^` dan `$`.
 - Pengelompokan, yang digunakan untuk mengelompokkan beberapa bagian dari pola menjadi satu unit.

.	Any character except newline.
\.	A period (and so on for *, \{, \\\, etc.)
^	The start of the string.
\$	The end of the string.
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.
\D,\W,\S	Anything except a digit, word character, or whitespace.
[abc]	Character a, b, or c.
[a-z]	a through z.
[^abc]	Any character except a, b, or c.
aa bb	Either aa or bb.
?	Zero or one of the preceding element.
*	Zero or more of the preceding element.
+	One or more of the preceding element.
{n}	Exactly n of the preceding element.
{n,}	n or more of the preceding element.
{m,n}	Between m and n of the preceding element.
??,*?,+?,{n}?, etc.	Same as above, but as few as possible.
(expr)	Capture expr for use with \1, etc.
(?:expr)	Non-capturing group.
(?=expr)	Followed by expr.
(?!expr)	Not followed by expr.

Gambar 2. Cheatsheet Regex

(Sumber: regexpal.com)

2.3. Metode Pencarian

2.3.1. Exact Matching

Exact matching adalah metode pencarian yang bertujuan untuk menemukan *string* yang identik dengan *string* kueri atau dalam kasus ini *P*. Metode ini tidak memiliki toleransi terhadap kesalahan sekecil apa pun, termasuk perbedaan huruf besar/kecil, spasi, atau tanda baca. Prinsipnya adalah perbandingan biner (benar/salah) karakter per karakter. Algoritma akan memeriksa apakah setiap karakter pada posisi yang sama di antara *T* dan *P* identik. Jika ada satu saja karakter yang berbeda, maka kedua *string* dianggap tidak cocok.

2.3.2. Fuzzy Matching

Fuzzy matching adalah metode pencarian yang dirancang untuk menemukan *string* yang mirip atau mendekati *string* kueri, meskipun tidak identik. Metode ini sangat toleran terhadap kesalahan seperti salah ketik (*typo*), variasi ejaan, atau perbedaan kecil lainnya. Ada berbagai metode yang dapat digunakan untuk menentukan kemiripan antara dua *string*. Standar emasnya adalah *Levenshtein Distance*.

Levenshtein Distance didefinisikan sebagai jumlah minimum operasi edit satu karakter yang diperlukan untuk mengubah satu kata menjadi kata lain. Ada tiga jenis operasi yang dihitung, yaitu *insertion* (penambahan satu karakter), *deletion* (penghapusan satu karakter), dan *substitution* (penggantian satu karakter dengan karakter lain). Dalam praktik implementasinya, ditetapkan sebuah ambang batas. Dua *string* dianggap cocok jika *Levenshtein Distancenya* kurang dari atau sama dengan nilai ambang batas tersebut.

2.4. Teknologi Pendukung

Aplikasi sistem ATS (Applicant Tracking System) ini dibangun menggunakan beberapa teknologi di dalam bahasa pemrograman Python. Python dipilih karena ekosistem pustakanya yang sangat luas dan kapabilitas bawaannya untuk mengelola Regex. Arsitekturnya dapat dipecah menjadi beberapa komponen utama, yaitu antarmuka pengguna, pemrosesan data, manajemen basis data, dan mesin pencari.

2.4.1. Antarmuka Pengguna dengan Flet

Antarmuka pengguna grafis untuk aplikasi desktop ini dibangun menggunakan *framework* Flet. Flet adalah sebuah *framework* Python modern yang memungkinkan pembuatan antarmuka pengguna berbasis Flutter tanpa menulis kode Dart. Dalam aplikasi ini, Flet digunakan untuk:

- Membangun Jendela Utama
Jendela utama memuat komponen dasar seperti judul, dimensi, dan tema, serta mendefinisikan struktur visual utama aplikasi.
- Struktur Navigasi
Sistem navigasi yang digunakan cukup sederhana, yaitu menggunakan suatu sidebar yang memungkinkan pengguna berpindah antar dua halaman yaitu halaman Applicants yang berisi data semua pelamar dan Search yang berisi modul pencarian.
- Manajemen Halaman dan Komponen
Seperti yang disebutkan sebelumnya, aplikasi diorganisir menjadi beberapa halaman, yaitu halaman pelamar *ApplicantsPage*, halaman pencarian *SearchPage*, dan halaman detail *DetailPage*.

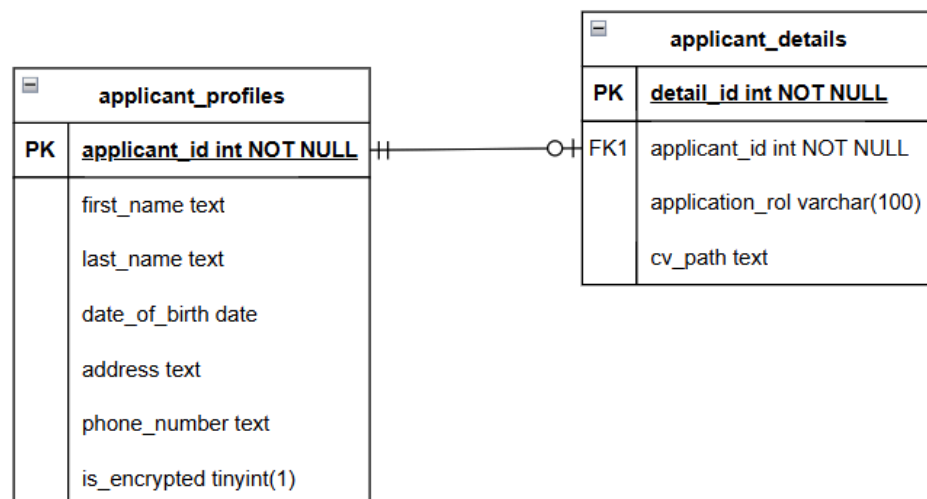
2.4.2. Pemrosesan dan Ekstraksi CV

Data yang berada di dalam CV pelamar akan diproses pada tahap ini. Semua file CV menggunakan format PDF. Untuk mendapatkan informasi dari PDF tersebut, aplikasi memanfaatkan pustaka pdfplumber. Pustaka ini dipilih karena kemampuannya untuk mengekstrak teks dari PDF dengan tetap berusaha mempertahankan tata letak dan urutan baca aslinya. Teks yang didapatkan kemudian akan disimpan di sebuah file .txt untuk analisis lebih lanjut.

Setelah teks diekstrak, CVProcessor akan menggunakan pustaka standar Python yang memungkinkan Regex untuk mengidentifikasi dan memisahkan bagian penting dari CV. Ini bertujuan mengubah data yang tidak terstruktur yang ada di dalam teks menjadi format yang lebih terstruktur dan dapat diindeks oleh mesin pencari.

2.4.3. Manajemen Basis Data dengan MySQL dan SQLAlchemy

Seluruh data pelamar yang telah diproses akan disimpan ke dalam sebuah sistem manajemen basis data, yang diimplementasikan menggunakan MySQL dan SQLAlchemy. Skema yang digunakan melibatkan dua tabel yaitu tabel applicant_profiles dan applicant_details. Berikut adalah skema basis data relasionalnya.



Gambar 3. Skema basis data

applicant_profiles menyimpan informasi pribadi pelamar, sedangkan tabel application_details menyimpan detail lamaran yang diajukan. Relasi antara tabel

applicant_profiles dan applicant_details adalah *one-to-many*, karena seorang pelamar dapat mengajukan beberapa lamaran dengan konten yang berbeda pula. Interaksi antara aplikasi Python dan basis data MySQL dijemput oleh SQLAlchemy. SQLAlchemy memungkinkan interaksi dengan tabel basis data, seolah-olah tabel tersebut adalah objek dalam Python biasa. Ini menyederhanakan operasi CRUD (Create, Read, Update, dan Delete).

2.4.4. Mesin Pencari

Mesin pencari yang digunakan menggabungkan dua metode pencarian utama, yaitu Exact Matching dan Fuzzy Matching. Pengguna dapat memasukkan kata kunci yang akan dicari di dalam data CV yang ada. Pengguna dapat memilih salah satu dari tiga algoritma yang ada untuk pencocokan string, yaitu algoritma Knuth-Morris-Pratt, algoritma Boyer-Moore, dan algoritma Aho-Corasick. Jika metode pencarian yang digunakan adalah exact matching, maka teks yang dicari harus persis mengandung kata kunci yang dimasukkan.

Berbeda dengan metode exact matching, metode fuzzy matching memberikan toleransi terhadap kesalahan ketik dan variasi kata. Jika kata kunci tidak ditemukan dengan metode exact matching, maka mesin pencari akan menggunakan metode fuzzy matching yang memanfaatkan Levenshtein Distance untuk menghitung rasio kemiripan antara kata kunci.

Untuk membuat pencocokan fuzzy lebih cerdas, sistem ini menerapkan ambang batas dinamis (dynamic threshold). Ambang batas ini secara otomatis menyesuaikan tingkat toleransi kesalahan berdasarkan panjang dan jumlah kata dari kata kunci yang dimasukkan.

Untuk kata kunci yang pendek (1-3 huruf), ambang batasnya sangat ketat (1.0 atau 100%), yang berarti harus ada kecocokan yang hampir sempurna. Seiring dengan bertambahnya panjang kata kunci, ambang batas menjadi lebih longgar (misalnya, 0.9 hingga 0.7), memungkinkan adanya beberapa kesalahan ketik.

Untuk frasa (terdiri dari beberapa kata), ambang batasnya bahkan lebih fleksibel (0.7 hingga 0.55) untuk mengakomodasi variasi yang lebih besar dalam kalimat yang panjang. Pendekatan dinamis ini memastikan bahwa kata kunci

pendek memerlukan presisi tinggi, sementara kata kunci atau frasa yang lebih panjang dapat ditemukan bahkan jika mengandung lebih banyak kesalahan ketik.

Hasil akhir pencarian adalah kombinasi dari kedua metode tersebut. Search engine akan menggabungkan skor dari exact matches dan fuzzy matches untuk menghasilkan skor keseluruhan yang digunakan untuk mengurutkan pelamar.

BAB III: Analisis Pemecahan Masalah

3.1. Langkah-Langkah Pemecahan Masalah

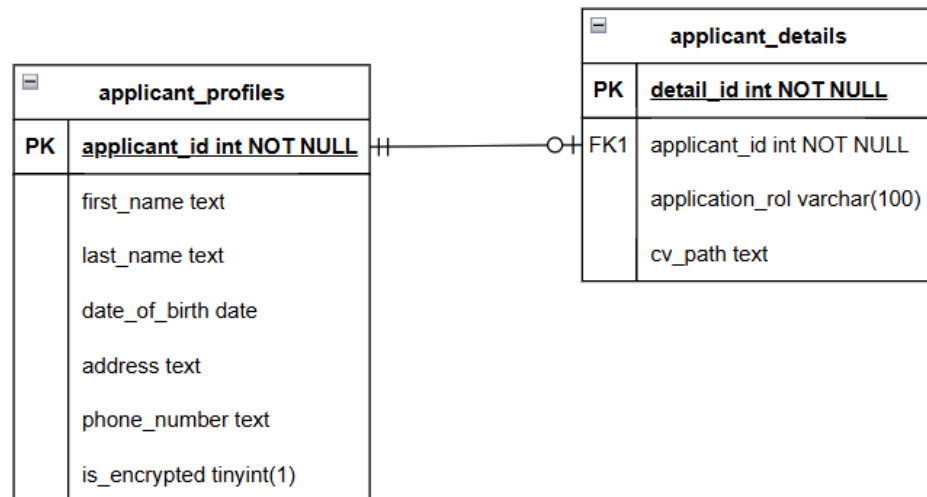
3.1.1. Strukturisasi Basis Data

Untuk memecahkan masalah ini, data perlu disimpan di sebuah basis data. Karena seorang pelamar dapat mengajukan lamaran untuk beberapa posisi dan setiap lamaran tersebut mungkin memerlukan dokumen yang berbeda, maka data pelamar dan data dokumen CV perlu dipisah. Hubungan yang diilustrasikan pada kasus sebelumnya menggambarkan hubungan *one-to-many*.

- Skema Basis Data

Skema basis data melibatkan dua tabel utama, yaitu `applicant_profiles` dan `applicant_details`. `applicant_profiles` menyimpan data pribadi dari pelamar seperti nama, tanggal lahir, kontak pribadi, sementara `applicant_details` menyimpan informasi terkait lamaran spesifik, yang juga menyimpan *path* ke file CV. Desain *one-to-many* digunakan karena memungkinkan satu pelamar untuk memiliki banyak CV, serta mengisolasi data sensitif ke tabelnya tersendiri untuk mengurangi redundansi.

Relasi *one-to-many* pada MySQL biasanya diimplementasikan dengan *foreign key*. Tabel yang berperan sebagai *parent* memiliki *primary key* yang berperan sebagai identitas unik. Tabel yang berperan sebagai *child* akan memiliki suatu *foreign key* yang mengacu pada *primary key* dari tabel *parent*. Berikut adalah skema basis data relasional yang digunakan dalam pemecahan masalah ini.



Gambar 4. Skema basis data relasional

- **Pengunggahan File**

Untuk memasukkan data lamaran baru ke dalam sistem, terdapat fitur pengunggahan CV melalui fungsionalitas antarmuka grafis. Sebelum mengunggah, pengguna dapat mengaitkan CV baru dengan profil pelamar yang sudah ada melalui sebuah *dropdown*. Jika tidak ada pelamar yang dipilih (opsi "Random Applicant"), sistem akan secara cerdas menugaskan CV tersebut ke salah satu profil yang ada secara acak. Ini memastikan setiap aplikasi memiliki pemilik profil. File yang diunggah kemudian akan diproses dan ditransformasi ke bentuk yang lebih terbaca.

3.1.2. Transformasi Data

CV dalam format PDF, meskipun tampak terstruktur bagi mata manusia, pada dasarnya adalah dokumen semi-terstruktur dari sudut pandang mesin. Format PDF (*Portable Document Format*) dirancang utamanya untuk presentasi visual yang konsisten di berbagai platform, bukan untuk representasi semantik data. Ini berarti, PDF tidak menyimpan teks sebagai sebuah alur paragraf yang koheren seperti pada file .txt. Sebaliknya, PDF menyimpan elemen-elemen secara terpisah:

- **Objek Teks**, di mana setiap karakter atau potongan teks disimpan sebagai objek independen dengan koordinat X dan Y yang presisi di atas halaman.

- Grafis dan Gambar, yakni elemen visual seperti garis, logo, atau foto yang tersimpan sebagai objek terpisah.

Tantangan ini membuat ekstraksi teks menjadi tugas yang kompleks. Setelah mengevaluasi beberapa alternatif, pustaka pdfplumber dipilih karena fokusnya pada analisis tata letak dan rekonstruksi alur baca yang cerdas. Sistem juga dioptimalkan untuk CV modern yang dibuat secara digital, di mana data teks benar-benar ada sebagai objek teks, bukan gambar sehingga tidak memerlukan proses OCR (*Optical Character Recognition*). Proses yang dilalui untuk melakukan transformasi data adalah sebagai berikut:

- Ekstraksi Teks Mentah: Dokumen PDF diubah menjadi teks mentah menggunakan library pdfplumber.
- Parsing: Teks mentah ini kemudian diuraikan oleh serangkaian fungsi untuk mengenali header-header bagian umum. Konten yang berada di bawah header ini akan diekstrak.
- Penyimpanan Teks: Teks yang berhasil diekstraksi akan disimpan di dalam sebuah file .txt.

3.1.3. Enkripsi Data

Data pelamar diamankan menggunakan mekanisme enkripsi custom yang berlapis dan bersifat reversibel. Artinya, data yang sudah dienkripsi dapat dikembalikan ke bentuk aslinya melalui proses dekripsi. Sistem ini dirancang untuk melindungi informasi sensitif saat disimpan di database, sekaligus memastikan data tersebut dapat diakses kembali saat dibutuhkan.

- Salting: *Salt* acak ditambahkan di awal untuk mencegah serangan tabel pelangi.
- Caesar Cipher: Penggeseran karakter sederhana sebagai lapisan obfuskasi pertama.
- Vignere Cipher: Enkripsi polialfabetik yang menggunakan kunci internal sebagai lapisan perlindungan selanjutnya.
- XOR Cipher: Operasi XOR *bit-by-bit* dengan kunci yang sama.
- Base64 Encoding: Hasilnya diubah menjadi format URL-safe Base64.

Proses enkripsi dan dekripsi diintegrasikan dalam model ApplicantProfile. Fungsi untuk mengenkripsi data dipanggil sebelum menyimpan data baru, dan fungsi untuk menampilkan data akan mendekripsi data (untuk data yang dienkripsi) setiap kali ada data yang akan ditampilkan.

3.1.4. Pengembangan Mesin Pencari

Pencarian CV dibagi menjadi dua tahap, yaitu tahap *exact matching* dan tahap *fuzzy matching*. Pada tahap *exact matching*, sistem akan mencari kecocokan persis dengan *keyword* yang dicari. Pengguna memiliki fleksibilitas untuk memilih salah satu dari tiga algoritma yaitu KMP, Boyer-Moore, dan Aho-Corasick. Untuk *keyword* yang tidak menghasilkan kecocokan tepat, sistem akan beralih ke *fuzzy matching*. Rasio kemiripan akan dihitung menggunakan *Levenshtein Distance*. Sistem akan mencari kata-kata yang mirip dengan *keyword* yang dicari.

3.1.5. Pemeringkatan Hasil

Sistem memberikan peringkat pada kandidat-kandidat dengan menghitung skor dari setiap pelamar. Skor yang dimaksud ini adalah agregat dari jumlah total kemunculan *exact matches* dan total skor kesamaan dari *fuzzy matches*. Hasil akhir tersebut akan diurutkan secara menurun sehingga kandidat paling relevan akan ditempatkan di urutan teratas.

3.1.6. Inisialisasi dan Pemeliharaan Sistem

Untuk mempermudah penggunaan dan memelihara sistem, maka terdapat beberapa fitur yang diimplementasikan. Salah satunya adalah fungsi `init_db`. Fungsi `init_db` dipanggil di awal dan secara otomatis akan memeriksa keberadaan basis data dan tabel. Jika tidak ada, ia akan membuatnya. Fitur lain adalah fitur `auto_migrate_schema` yang akan membandingkan skema yang ada di model SQLAlchemy dengan skema basis data aktual dan secara otomatis mencoba menambahkan kolom yang hilang.

3.2. Pemetaan Masalah Menjadi Elemen Algoritma Pencocokan String

Proses pencarian dalam sistem ini pada dasarnya adalah serangkaian pemetaan (transformasi) yang menerjemahkan kebutuhan abstrak pengguna menjadi hasil yang terukur dan dapat ditindaklanjuti. Konsep teoretis Teks dalam algoritma, yang merupakan

sebuah sekuens karakter untuk dicari, dipetakan menjadi sebuah *string* tunggal yang merupakan gabungan dari seluruh data relevan seorang pelamar. Fungsi `_get_searchable_text` bertanggung jawab untuk transformasi ini, mengambil data dari berbagai sumber seperti nama, peran, keterampilan, dan ringkasan, lalu menyatukannya menjadi satu string linear. Sejalan dengan itu, konsep teoretis Pola dipetakan menjadi satu atau lebih *keywords* yang dimasukkan pengguna. *String* input seperti "java, python" dari antarmuka secara langsung dipetakan menjadi sebuah himpunan pola formal, yaitu $K = \{"java", "Python"\}$.

Langkah pemetaan selanjutnya adalah mengubah setiap Pola *string* menjadi struktur data algoritmik yang dapat membantu dalam proses pencarian. Untuk algoritma KMP, setiap *keyword* dipetakan menjadi sebuah list integer di Python yang merepresentasikan Tabel *Border Function* melalui fungsi `_build_failure_function`. Untuk Boyer-Moore, string *keyword* dipetakan menjadi sebuah *dictionary* Python yang berfungsi sebagai Tabel Kemunculan Terakhir (*Last Occurrence Table*) oleh fungsi `_build_last_occurrence_table`. Sementara itu, untuk Aho-Corasick, keseluruhan daftar *keywords* pengguna dipetakan menjadi satu objek `AhoCorasickMatcher` tunggal yang kompleks, yang berisi *automaton* lengkap dari struktur `TrieNode` dengan *failure links*-nya, melalui fungsi `build_automaton`.

Setelah proses pencarian selesai, output mentah dari algoritma, yang biasanya berupa daftar indeks posisi kemunculan, harus dipetakan kembali menjadi informasi yang bermakna. SearchEngine memetakan daftar indeks ini menjadi sebuah hitungan frekuensi. Hitungan ini kemudian menjadi bagian dari metrik bisnis yang terukur untuk setiap kandidat.

Pada tahap akhir, semua metrik yang terkumpul, seperti frekuensi kecocokan tepat dan skor similaritas dari kecocokan kabur, dipetakan menjadi satu nilai numerik tunggal yang disebut `overall_score`. Nilai inilah yang digunakan untuk mengurutkan para kandidat. Dengan demikian, sistem telah menyelesaikan siklus pemetaan penuh: dari kebutuhan pengguna yang abstrak, ke data formal, ke representasi matematis, ke hasil komputasi mentah, dan akhirnya kembali ke sebuah daftar peringkat yang sederhana dan dapat ditindaklanjuti.

3.3. Fitur Fungsional dan Arsitektur Aplikasi

3.3.1. Arsitektur Perangkat Lunak Berbasis MVC (Model-View-Controller) yang Dikembangkan

Aplikasi ini mengadopsi arsitektur berlapis (*layered architecture*), yang secara logis memisahkan fungsionalitas ke dalam beberapa lapisan independen. Pendekatan ini mirip dengan pola desain Model-View-Controller (MVC).

- Layer Presentasi (View - src/gui):
Layer ini dibuat dengan menggunakan *framework* Flet, yang bertanggung jawab penuh atas rendering antarmuka dan menangkap interaksi pengguna. Halaman (pages) dan komponen (components) hanya menampilkan data dan meneruskan *event* (seperti klik tombol) ke *controller*.
- Layer Controller (Controller - src/gui/main_window.py):
Layer ini bertindak sebagai perantara. MainWindow mengelola navigasi dan status UI global. Logika di dalam setiap halaman menerima *event* dari View, memanggil metode yang sesuai di lapisan Core, dan kemudian memperbarui View dengan data baru.
- Layer Logika Bisnis (Model/Core - src/core):
Layer ini adalah otak dari aplikasi, terdiri dari dua fungsionalitas yaitu Search Engine dan CVProcessor. SearchEngine dan CVProcessor berisi semua aturan bisnis dan logika pemrosesan. Lapisan ini tidak tahu apa-apa tentang antarmuka pengguna, hanya menerima data, memprosesnya, dan mengembalikan hasilnya.
- Layer Akses Data (Model/Database - src/database):
Layer ini mengelola semua komunikasi dengan database MySQL melalui SQLAlchemy ORM. Ini mengabstraksi detail kueri SQL dan menyediakan antarmuka berorientasi objek untuk berinteraksi dengan data.
- Lapisan Pendukung (Algorithms & Utils): Menyediakan layanan horizontal seperti implementasi algoritma murni dan utilitas enkripsi yang digunakan oleh lapisan lain.

3.3.2. Fitur Fungsional

- Manajemen Profil Pelamar

Sistem menyediakan halaman khusus untuk manajemen profil pelamar. Di halaman ini, pengguna dapat membuat entitas pelamar baru melalui sebuah formulir yang mencakup field-field esensial seperti nama depan, nama belakang, alamat, dan nomor telepon. Setelah formulir diisi dan tombol "Create Applicant" ditekan, sistem melakukan validasi dasar sebelum membuat objek ApplicantProfile. Sebagai umpan balik langsung kepada pengguna, daftar pelamar di sisi kanan layar akan diperbarui secara dinamis, menampilkan kartu informatif untuk profil yang baru saja dibuat.

- Manajemen Pengunggahan CV

Halaman "Applications" dirancang sebagai pusat untuk mengelola setiap lamaran atau CV yang masuk. Alur kerja dimulai dengan kemampuan pengguna untuk mengaitkan CV yang akan diunggah dengan profil pelamar yang sudah ada melalui sebuah *dropdown*. Tombol "Select PDF File" memicu `ft.FilePicker` yang telah disaring khusus untuk file `.pdf`, dan setelah file dipilih, label teks akan menampilkan nama file dengan warna hijau sementara tombol "Upload CV" diaktifkan. Setelah `CVProcessor` selesai memproses dan menyimpan file, sebuah *callback* akan memicu `ft.SnackBar` di antarmuka untuk memberikan notifikasi sukses atau gagal, dan daftar aplikasi akan langsung dimuat ulang untuk menampilkan entri yang baru.

- Mesin Pencari

Komponen `SearchSection` menyediakan formulir pencarian yang kaya fitur. Pengguna dapat memasukkan *keywords* dalam `ft.TextField` yang mendukung multi-baris untuk kueri yang panjang. Pemilihan algoritma *exact match* (KMP, BM, AC) dilakukan melalui `ft.Dropdown` yang jelas. Selain itu, pengguna dapat menyempurnakan pencarian dengan menentukan jumlah maksimal hasil melalui `ft.TextField` numerik. Saat pencarian dieksekusi, semua parameter dari kontrol UI ini dikumpulkan, divalidasi, dan diteruskan ke metode `search_engine.search()` untuk diproses.

- Visualisasi

Komponen ResultsSection secara dinamis me-render hasil pencarian ke dalam format kartu yang mudah dicerna. Setiap kartu dirancang untuk memberikan informasi berupa sebuah rencana peringkat, nama dan peran pelamar, skor relevansi keseluruhan, dan suatu "chip" berwarna yang menyoroti *keyword* yang cocok. Saat pengguna mengklik sebuah kartu, aplikasi akan beralih ke DetailView, yang menyajikan semua data yang telah diparsing dari CV dalam format yang terstruktur rapi, dikelompokkan ke dalam kategori seperti Info Pribadi, Ringkasan, Keterampilan, Pengalaman Kerja, dan Pendidikan.

3.4. Ilustrasi Kasus

3.4.1. Ilustrasi Kasus Knuth-Morris-Pratt

- Skenario

Tujuan: Mencari kandidat dengan keahlian spesifik di bidang *engineering*.

Keyword: "engineering"

Algoritma yang Dipilih: KMP

Data Kandidat: "like tinkering with engine, studied electrical engineering..."

- Pemetaan ke Bentuk Matematis

$T = \text{"likes tinkering engine, studied electrical engineering..."}$

$P = \text{"engineering"}$

j	0	1	2	3	4	5	6	7	8	9	10
$P[j]$	<i>e</i>	<i>n</i>	<i>g</i>	<i>i</i>	<i>n</i>	<i>e</i>	<i>e</i>	<i>r</i>	<i>i</i>	<i>n</i>	<i>g</i>
k	0	1	2	3	4	5	6	7	8	9	
$b(k)$	0	0	0	0	0	1	1	0	0	0	

- Simulasi

[illegible]

engineering

Pada titik $T[18]$ hingga $T[23]$, terjadi kecocokan antara P dan T dalam bentuk “engine...”. Pada $T[24]$ dan $P[6]$, terjadi ketidakcocokan antara karakter ‘r’ dan karakter ‘e’. Pada saat ini, nilai dari $j = 6$, sehingga nilai $k = j - 1 = 5$. Dengan melihat nilai pada tabel *border function*, nilai dari $b(5) = 1$. Pergeseran dilakukan sebesar $j - b(k) = 6 - 1 = 5$.

3.4.2. Ilustrasi Kasus Boyer-Moore

- Skenario

Tujuan: Mencari kandidat dengan peran spesifik backend

Keyword: “backend”

Algoritma yang Dipilih: BM

Data Kandidat: “adi 10 years of backend experience”

- Pemetaan ke Bentuk Matematis

$T = \text{"adi 10 years of backend experience"}$

$P = \text{"backend"}$

<i>char</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>k</i>	<i>e</i>	<i>n</i>	<i>d</i>
$l[char]$	0	1	2	3	4	5	6

- Simulasi

```

adi 10 years of backend experience
backend
      backend
            backend
                  backend

```

Pola “backend” akan disejajarkan dengan awal teks “adi 10 years of backend experience”. Perbandingan dimulai dari kanan. Karena ketidakcocokan langsung terjadi pada karakter paling kanan dan saat diperiksa bahwa karakter tersebut tidak ada di *Last Occurrence Table*, maka semua pola digeser melewati karakter tersebut. Saat tiba di $T[14]$, maka ketika diperiksa bahwa karakter tersebut ada di *Last Occurrence Table*,

maka pola akan digeser sehingga karakter pada teks tersebut sejajar dengan kemunculan terakhirnya di tabel. Setelah traversal terakhir, tampak bahwa pola ditemukan di teks.

3.4.3. Ilustrasi Kasus Aho-Corasick

- Skenario

Tujuan: Mencari kandidat dengan multiple keahlian spesifik di bidang teknologi.

Keywords: "java", "python", "data"

Algoritma yang Dipilih: Aho-Corasick

Data Kandidat: "skilled in java, python and data analysis"

- Pemetaan ke Bentuk Matematis

$T = \text{"skilled in java, python and data analysis"}$

$P_1 = \text{"java"}$

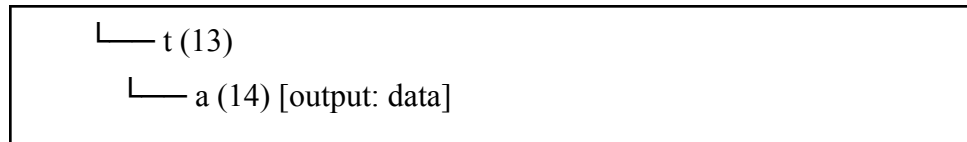
$P_2 = \text{"python"}$

$P_3 = \text{"data"}$

- Konstruksi Trie

```

Root (0)
├── j (1)
│   ├── a (2)
│       ├── v (3)
│           └── a (4) [output: java]
├── p (5)
│   ├── y (6)
│       ├── t (7)
│           ├── h (8)
│               ├── o (9)
│                   └── n (10) [output: python]
└── d (11)
    └── a (12)
  
```



- Kontruksi *failure links*

Node	Karakter	<i>Failure link</i>	Keterangan
0	root	-	Root node
1	j	0	Kembali ke root
2	a	0	Kembali ke root
3	v	0	Kembali ke root
4	a	2	Suffix "a" cocok dengan node 2
5	p	0	Kembali ke root
6	y	0	Kembali ke root
7	t	0	Kembali ke root
8	h	0	Kembali ke root
9	o	0	Kembali ke root
10	n	0	Kembali ke root
11	d	0	Kembali ke root
12	a	2	Suffix "a" cocok dengan node 2
13	t	0	Kembali ke root

14	a	2	Suffix "a" cocok dengan node 2
----	---	---	--------------------------------

- Pencarian pattern pertama

```

T = "skilled in java, python and data analysis"
0123456789012345678901234567890123
      ^^^^

```

Posisi 11-14: Karakter 'j', 'a', 'v', 'a'

i=11, j=0: T[11]='j' cocok dengan P₁[0]='j', transisi ke node 1

i=12, j=1: T[12]='a' cocok dengan P₁[1]='a', transisi ke node 2

i=13, j=2: T[13]='v' cocok dengan P₁[2]='v', transisi ke node 3

i=14, j=3: T[14]='a' cocok dengan P₁[3]='a', transisi ke node 4

MATCH FOUND: Pattern "java" ditemukan pada posisi 11-14

- Pencarian pattern kedua

```

T = "skilled in java, python and data analysis"
0123456789012345678901234567890123
      ^^^^^^

```

Posisi 17-22: Karakter 'p', 'y', 't', 'h', 'o', 'n'

i=17, j=0: T[17]='p' cocok dengan P₂[0]='p', transisi ke node 5

i=18, j=1: T[18]='y' cocok dengan P₂[1]='y', transisi ke node 6

i=19, j=2: T[19]='t' cocok dengan P₂[2]='t', transisi ke node 7

i=20, j=3: T[20]='h' cocok dengan P₂[3]='h', transisi ke node 8

i=21, j=4: T[21]='o' cocok dengan P₂[4]='o', transisi ke node 9

i=22, j=5: T[22]='n' cocok dengan P₂[5]='n', transisi ke node 10

MATCH FOUND: Pattern "python" ditemukan pada posisi 17-22

- Pencarian pattern ketiga

```

T = "skilled in java, python and data analysis"
0123456789012345678901234567890123
      ^^^^

```

Posisi 28-31: Karakter 'd', 'a', 't', 'a'

i=28, j=0: T[28]='d' cocok dengan P₃[0]='d', transisi ke node 11

i=29, j=1: T[29]='a' cocok dengan P₃[1]='a', transisi ke node 12

i=30, j=2: T[30]='t' cocok dengan P₃[2]='t', transisi ke node 13

i=31, j=3: T[31]='a' cocok dengan P₃[3]='a', transisi ke node 14

MATCH FOUND: Pattern "data" ditemukan pada posisi 28-31

- Skenario Mismatch

Ketika mencari "python" di posisi 6-7

T[6-7] = "in"

P₂ = "python"

i=6, j=0: T[6]='i' ≠ P₂[0]='p', tetap di root (node 0)

i=7, j=0: T[7]='n' ≠ P₂[0]='p', tetap di root (node 0)

Karena tidak ada partial match yang terjadi, failure function tidak diperlukan dan pencarian berlanjut dari karakter berikutnya.

BAB IV: Implementasi dan Pengujian

4.1. Spesifikasi Teknik Program

4.1.1. Struktur Basis Data

Sistem menggunakan basis data MySQL untuk menyimpan data pelamar dan detail aplikasi mereka, yaitu:

- Tabel `applicant_profiles`: menyimpan data pribadi setiap pelamar.

No.	Atribut	Tipe	Kegunaan
1.	<code>applicant_id</code>	int	ID unik untuk setiap profil pelamar.
2.	<code>first_name</code>	text	Nama depan pelamar.
3.	<code>last_name</code>	text	Nama belakang pelamar.
4.	<code>date_of_birth</code>	date	Tanggal lahir pelamar.
5.	<code>address</code>	text	Alamat pelamar.
6.	<code>phone_number</code>	text	Nomor telepon pelamar.
7.	<code>is_encrypted</code>	bool	Flag yang menandakan apakah data pada baris ini terenkripsi

- Tabel `applicant_details`: menyimpan informasi terkait setiap CV.

No.	Atribut	Tipe	Kegunaan
1.	<code>detail_id</code>	int	ID unik untuk setiap aplikasi/CV.
2.	<code>applicant_id</code>	int	Merujuk ke <code>applicant_id</code> di tabel <code>applicant_profiles</code> .
3.	<code>application_role</code>	varchar (50)	Posisi yang dilamar.

4.	cv_path	text	Path atau lokasi file CV yang disimpan dalam sistem.
----	---------	------	--

4.1.2. Skema

Untuk memastikan keamanan tipe data (*type safety*) di seluruh aplikasi, digunakan data class sebagai skema data:

- Skema ApplicantDataSchema

Merepresentasikan data gabungan seorang pelamar yang akan ditampilkan atau diteruskan antar komponen. Ini mencakup semua field dari applicant_profiles dan applicant_details, ditambah dengan data yang diekstrak dari CV seperti ringkasan, keahlian, pengalaman kerja, dan pendidikan.

- Skema SearchResultSchema

Merepresentasikan hasil dari operasi pencarian, berisi daftar kandidat yang cocok, kata kunci yang dicari, serta metrik waktu eksekusi

4.1.3. Kelas

Fungsionalitas inti dari program dibangun atas beberapa kelas, yaitu:

- CVProcessor: bertanggung jawab untuk semua tugas yang berkaitan dengan pemrosesan file CV.

No.	Fungsi/Prosedur	Kegunaan
1.	extract_text_from_pdf(file_path)	Mengekstrak teks mentah dari file PDF menggunakan pdfplumber.
2.	process_cv_file(file_path, original_filename, applicant_id)	Prosedur utama yang mengambil file CV, menyimpannya ke direktori penyimpanan, mengekstrak teksnya, dan mencatat informasinya ke basis data

		sebagai entri baru di tabel applicant_details.
3.	compute_cv_fields(cv_path)	Menghitung dan mengekstrak semua field penting dari CV
4.	extract_all(text)	Mengekstrak informasi terstruktur dari teks CV, termasuk informasi pribadi, ringkasan, keahlian, pengalaman kerja, dan pendidikan menggunakan pola regex.
5.	extract_personal_info(text)	Mengekstrak detail pribadi seperti email, nomor telepon, dan nama dari teks.
6.	extract_summary(text)	Menemukan dan mengekstrak bagian ringkasan atau profile dari teks CV.
7.	extract_skills(text)	Mengekstrak daftar keahlian dari teks CV.
8.	extract_work_experience(text)	Mengurai dan mengekstrak riwayat pekerjaan dari teks CV.
9.	extract_education(text)	Mengurai dan mengekstrak riwayat pendidikan dari teks

		CV.
--	--	-----

- SearchEngine: melakukan pencarian CV.

No.	Fungsi/Prosedur	Kegunaan
1.	search(keywords, algorithm, max_results, use_multiprocessing)	Menerima kata kunci, algoritma, dan parameter lainnya untuk melakukan pencarian, baik single-threaded ataupun multi-processing.
2.	get_applicant_details(detail_id)	Mengambil informasi detail dari seorang pelamar untuk mendapatkan data CV yang terurai.
3.	combine_and_rank_results(exact_results, fuzzy_results)	Menggabungkan hasil dari <i>exact matching</i> dan <i>fuzzy matching</i> , lalu mengurutkannya berdasarkan skor relevansi.

- Algoritma String Matching, dipisahkan ke dalam modulnya sendiri, yaitu:
 - KMPMatcher

No.	Fungsi/Prosedur	Kegunaan
1.	search(text, pattern)	Mencari semua kemunculan pattern dalam text menggunakan algoritma Knuth-Morris-Pratt.

- BoyerMooreMatcher

No.	Fungsi/Prosedur	Kegunaan
1.	search(text, pattern)	Mencari semua kemunculan pattern dalam text menggunakan algoritma Boyer-Moore.

- AhoCorasickMatcher

No.	Fungsi/Prosedur	Kegunaan
1.	search(text, pattern)	Mencari semua kemunculan pattern dalam text menggunakan algoritma Aho-Corasick.
2.	build_automaton(patterns)	Membangun <i>trie</i> dan <i>failure links</i> yang diperlukan untuk pencarian multi-pola yang efisien.

- FuzzyMatcher

No.	Fungsi/Prosedur	Kegunaan
1.	similarity_ratio(s1, s2)	Menghitung rasio similaritas (0.0 hingga 1.0) antara dua <i>string</i> menggunakan Levenshtein Distance.
2.	find_best_matches(query,	Menemukan beberapa

	candidates, threshold, max_results)	kandidat <i>string</i> terbaik yang paling mirip dengan kueri berdasarkan ambang batas similaritas.
--	-------------------------------------	---

- CustomEncryption: fungsionalitas mengamankan data sensitif pelamar melalui enkripsi berlapis.

No.	Fungsi/Prosedur	Kegunaan
1.	encrypt(plaintext)	Mengambil string teks biasa, menambahkan <i>salt</i> , lalu mengenkripsinya secara berlapis menggunakan <i>Caesar cipher</i> , <i>Vigenère cipher</i> , dan <i>XOR cipher</i> . Hasilnya di- <i>encode</i> dengan Base64.
2.	decrypt(ciphertext)	Mengambil string terenkripsi (dalam format Base64), mendekodenya, lalu membalikkan semua lapisan enkripsi untuk mendapatkan kembali teks aslinya.

4.2. Tata Cara Penggunaan Program

4.2.1. Halaman Utama.

Saat program dijalankan, jendela utama akan muncul dengan bilah sisi navigasi di sebelah kiri dan area konten di sebelah kanan.

- Applicants: Halaman untuk mengelola profil pelamar.

- Applications: Halaman untuk mengunggah CV dan menautkannya ke profil pelamar.
- Search: Halaman untuk mencari CV berdasarkan kata kunci.

4.2.2. Halaman Applicants

Halaman ini berfungsi untuk membuat dan melihat profil dasar para pelamar. Berikut adalah fitur dari halaman ini:

- Create New Applicant: Di sisi kiri, terdapat formulir untuk menambahkan pelamar baru. Terdapat *text field* yang dapat diisi dengan nama, tanggal lahir, alamat, dan nomor telepon.
- Applicants List: Di sisi kanan, ditampilkan daftar semua pelamar yang sudah ada di basis data. Daftar ini dilengkapi dengan paginasi untuk navigasi yang mudah.

4.2.3. Halaman Applications

Halaman Applications adalah tempat mengunggah CV dan mengaitkannya dengan seorang pelamar. Berikut adalah fitur dari halaman ini:

- Select Applicant: Sebelum mengunggah CV, pilih pelamar dari menu dropdown. Pelamar itu akan ditautkan ke CV.
- Upload CV: Saat mengklik tombol “Select PDF File”, akan muncul halaman untuk memilih file CV dari komputer. Setelah file dipilih, namanya akan muncul dan tombol “Upload CV” akan aktif. Saat tombol “Upload CV” diklik, proses akan dimulai.

4.2.4. Halaman Search

Halaman Search memungkinkan pengguna untuk mencari kandidat berdasarkan *keyword* yang ingin dicari dari CV mereka. Berikut adalah fitur dari halaman ini:

- Search Form:
 - Keywords: Satu atau lebih kata dimasukkan untuk dicari dalam seluruh basis data CV.
 - Search Algorithm: Memilih algoritma pencocokan string yang akan digunakan.
 - Max Results: Jumlah hasil teratas yang ingin ditampilkan.

- Use Multiprocessing: Switch yang akan mengaktifkan atau menonaktifkan multiprocessing.
- Results Section: Hasil pencarian akan ditambahkan sebagai daftar kartu yang diurutkan berdasarkan skor relevansi tertinggi.

4.2.5. Halaman Detail

Halaman ini menampilkan pandangan terhadap profile seorang pelamar beserta CV-nya. Berikut adalah fitur dari halaman ini:

- Informasi Pribadi: menampilkan data dasar seperti nama, kontak, dan alamat.
- Tombol Aksi CV: membuka file asli PDF atau file TXT yang diekstraksi.
- Analisis CV: Menampilkan bagian-bagian yang telah diuraikan seperti summary, skills, work experience, dan education.

4.3. Hasil Pengujian

4.3.1 Menggunakan Knuth-Morris-Pratt

The screenshot displays a web interface titled "Search Applicants". Inside, there's a section for "Search CVs" with a search bar containing the keyword "company". Below the search bar, the "Search Options" are configured: the "Search Algorithm" is set to "Knuth-Morris-Pratt (KMP)" and the "Max Results" is set to 10. A toggle switch for "Use Multiprocessing" is turned on, with a note stating "Enable for faster search on multi-core systems". At the bottom of the search options, there are buttons for "Search CVs" and "Clear". Below the search options, an "Algorithm Information" section provides details about the selected algorithm: KMP is good for single pattern matching, Boyer-Moore is efficient for longer patterns, and Aho-Corasick is best for multiple patterns (labeled as a bonus).

Search Results 2 results • KMP • Total: 0.035s (Exact: 0.004s, Fuzzy: 0.000s)

1
Sous Chef
SOUS CHEF

★
12.0

✓ company (12)

A highly- motivated, productive and customer-focused team player with strong communication, interpersonal, organizational, time management, analytical...

2
Career Focus
ENGINEERING LAB TECHNICIAN

★
6.0

✓ company (6)

in seeking employment with Triumph Actuation Systems Inc. is to work in a professional atmosphere where I can utilize my

Search: company

Total Time: 0.035s (Process pdf + search)

Exact Time: 0.004s

Fuzzy Time: 0.000s

Search Applicants

Search CVs

Keywords
Search Keywords

Search Options

Search Algorithm
Knuth-Morris-Pratt (KMP) ▼

Max Results

✓ ☒ Use Multiprocessing
Enable for faster search on multi-core systems

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasic: Best for multiple patterns (Bonus)

Search Results

2 results • KMP • Total: 0.069s (Exact: 0.007s, Fuzzy: 0.011s)

1

Career Focus
ENGINEERING LAB TECHNICIAN

compay→company (0.9)

in seeking employment with Triumph Actuation Systems Inc. is to work in a professional atmosphere where I can utilize my

★ 0.9

2

Sous Chef
SOUS CHEF

compay→company (0.9)

★ 0.9

Search: compay

Total Time: 0.069s (Process pdf + search)

Exact Time: 0.007s

Fuzzy Time: 0.011s

Search CVs

Keywords

Search Keywords

asjkdksad

Search Options

Search Algorithm

Knuth-Morris-Pratt (KMP)

Max Results

10

Use Multiprocessing

Enable for faster search on multi-core systems

Search CVs

Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

No results found

No matching CVs found

Try different keywords or broaden your search criteria

No result found

Search Applicants

Search CVs

Keywords

Search Keywords

Assisted the Property Coordinator with daily tasks and worked on hotel computer programming systems Worked with HR department to control staffing and perform employee performance evaluations.

Search Options

Search Algorithm

Knuth-Morris-Pratt (KMP)

Max Results

10

Use Multiprocessing

Enable for faster search on multi-core systems

Search CVs

Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results 1 results • KMP • Total: 34.275s (Exact: 0.006s, Fuzzy: 34.208s)

1
Sous Chef
SOUS CHEF

★ 1.0

assisted the property coordinator with daily tasks and worked on hotel computer programming systems worked with hr department to

A highly- motivated, productive and customer-focused team player with strong communication, interpersonal, organizational, time management, analytical...

Search: Assisted the Property Coordinator with daily tasks and worked on hotel computer programming systems Worked with HR department to control staffing and perform employee performance evaluations.

Total Time: 34.275s (Process pdf + search)

Exact Time: 0.006s

Fuzzy Time: 34.208s

4.3.2 Menggunakan Boyer-Moore

Search Applicants

Search CVs

Keywords
Search Keywords
company

Search Options
Search Algorithm
Boyer-Moore (BM)

Max Results
10

☒ Use Multiprocessing
Enable for faster search on multi-core systems

Search CVs Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search: company

Total Time: 0.035s (Process pdf + search)

Exact Time: 0.004s

Fuzzy Time: 0.000s

Search Applicants

Search CVs

Keywords

Search Keywords

Search Options

Search Algorithm Boyer-Moore (BM)

Max Results 10

☒ Use Multiprocessing

Enable for faster search on multi-core systems

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

2 results • BM • Total: 0.057s (Exact: 0.005s, Fuzzy: 0.012s)

1

Career Focus
ENGINEERING LAB TECHNICIAN

★ 0.9

compay-company (0.9)

In seeking employment with Triumph Actuation Systems Inc. is to work in a professional atmosphere where I can utilize my

2

Sous Chef
SOUS CHEF

★ 0.9

compay-company (0.9)

A highly- motivated, productive and customer-focused team player with strong communication, interpersonal, organizational, time management, analytical...

Search: compay

Total Time: 0.057s (Process pdf + search)


Exact Time: 0.005s

Fuzzy Time: 0.012s

Search Applicants

Search CVs

Keywords

Search Keywords
 sadasdasdasdasd

Search Options

Search Algorithm

Boyer-Moore (BM)

Max Results

 10



Use Multiprocessing

 Enable for faster search on multi-core systems

 Search CVs

 Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

No results found



No matching CVs found

Try different keywords or broaden your search criteria

No result found

Search Applicants

Search CVs

Keywords

Search Keywords

Assisted the Property Coordinator with daily tasks and worked on hotel computer programming systems Worked with HR department to control staffing and perform employee performance evaluations.

Search Options

Search Algorithm

Boyer-Moore (BM)

Max Results

10



Use Multiprocessing



Enable for faster search on multi-core systems

Search CVs

Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

1 results • KMP • Total: 34.275s (Exact: 0.006s, Fuzzy: 34.208s)

1

Sous Chef

SOUS CHEF



1.0

assisted the property coordinator with daily tasks and worked on hotel computer programming systems worked with hr department to

A highly- motivated, productive and customer-focused team player with strong communication, interpersonal, organizational, time management, analytical...

Search: Assisted the Property Coordinator with daily tasks and worked on hotel computer programming systems Worked with HR department to control staffing and perform employee performance evaluations.

Total Time: 34.275s (Process pdf + search)

Exact Time: 0.006s

Fuzzy Time: 34.208s

4.3.3 Menggunakan Aho-Corasick

Search Applicants

Search CVs

Keywords

Search Keywords

company

Search Options

Search Algorithm

Aho-Corasick (AC)

Max Results

10

☒ Use Multiprocessing

Enable for faster search on multi-core systems

Search CVs Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

2 results • AC • Total: 0.053s (Exact: 0.001s, Fuzzy: 0.000s)

1

Sous Chef

SOUS CHEF

company (12)

A highly- motivated, productive and customer-focused team player with strong communication, interpersonal, organizational, time management, analytical...

2

Career Focus

ENGINEERING LAB TECHNICIAN

company (6)

in seeking employment with Triumph Actuation Systems Inc. is to work in a professional atmosphere where I can utilize my

Search: company

Total Time: 0.053s (Process pdf + search)

Exact Time: 0.001s

Fuzzy Time: 0.000s

Tugas Besar 3 IF2211 Strategi Algoritma

45

Search Applicants

Search CVs

Keywords

Search Keywords

Q compay

Search Options

Search Algorithm

Aho-Corasick (AC)

Max Results

10

☒ Use Multiprocessing

Enable for faster search on multi-core systems

Q Search CVs X Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search: compay

Total Time: 0.069s (Process pdf + search)

Exact Time: 0.007s

Fuzzy Time: 0.011s

Search Applicants

Search CVs

Keywords

Search Keywords

Q asdasdavasv

Search Options

Search Algorithm

Aho-Corasick (AC)

Max Results

10

☒ Use Multiprocessing

Enable for faster search on multi-core systems


Q Search CVs X Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

No results found



No matching CVs found

Try different keywords or broaden your search criteria

No result found

Search CVs

Keywords

Search Keywords

Assisted the Property Coordinator with daily tasks and worked on hotel computer programming systems Worked with HR department to control staffing and perform employee performance evaluations.

Search Options

Search Algorithm

Aho-Corasick (AC)

Max Results

10

Use Multiprocessing

Enable for faster search on multi-core systems

Search CVs

X Clear

Algorithm Information

- KMP: Good for single pattern matching
- Boyer-Moore: Efficient for longer patterns
- Aho-Corasick: Best for multiple patterns (Bonus)

Search Results

1 results • AC • Total: 25.440s (Exact: 0.001s, Fuzzy: 25.406s)

1

Sous Chef

SOUS CHEF

★
1.0

assisted the property coordinator with daily tasks and worked on hotel computer programming systems worked with hr department to

A highly- motivated, productive and customer-focused team player with strong communication, interpersonal, organizational, time management, analytical...

Fuzzy Time: 25.406s

Waktu Eksekusi Pencarian

Waktu eksekusi pencarian mencakup pemrosesan PDF dan proses pencarian. Waktu total bervariasi tergantung pada kompleksitas keyword dan algoritma yang dipilih.

Pencarian "company": Ketiga algoritma (KMP, Boyer-Moore, Aho-Corasick) memberikan waktu eksekusi yang relatif cepat untuk pencarian exact matching "company". Waktu fuzzy matching adalah 0 detik, yang berarti keyword ditemukan secara persis dan tidak diperlukan pencarian kabur.

Pencarian "compay": Untuk kesalahan ketik "compay", waktu eksekusi meningkat karena sistem harus melakukan fuzzy matching untuk menemukan kandidat yang mirip. Ketiga algoritma menunjukkan peningkatan waktu, dengan Boyer-Moore yang sedikit lebih cepat dibandingkan KMP dan Aho-Corasick.

Pencarian kalimat panjang: Untuk keyword yang sangat panjang ("Assisted the Property Coordinator..."), waktu eksekusi meningkat drastis. Ini disebabkan oleh kompleksitas pencarian fuzzy matching untuk kalimat panjang. Algoritma Aho-Corasick menunjukkan waktu eksekusi yang paling cepat dibandingkan KMP dan Boyer-Moore dalam kasus ini.

Perbandingan Algoritma

Knuth-Morris-Pratt (KMP): KMP menunjukkan kinerja yang baik untuk pencarian exact matching tetapi sedikit lebih lambat untuk pencarian fuzzy matching dengan kalimat panjang.

Boyer-Moore: Boyer-Moore menunjukkan kinerja yang baik untuk pencarian exact matching dan sedikit lebih cepat daripada KMP untuk pencarian fuzzy matching sederhana dengan kesalahan ketik. Namun, untuk kalimat panjang, kinerjanya mirip dengan KMP.

Aho-Corasick: Aho-Corasick menunjukkan kinerja terbaik untuk pencarian exact matching dan pencarian kalimat panjang. Ini menunjukkan efisiensinya dalam menangani pencarian dengan banyak keyword atau pola yang kompleks.

BAB V: Kesimpulan, Saran, dan Refleksi

5.1. Kesimpulan

Tugas Besar Pembuatan sistem ATS (Applicants Tracking System) berhasil diimplementasikan sesuai dengan tujuan yang telah ditetapkan. Program mampu mengelola, menguraikan, dan mencari CV dari pelamar sesuai dengan kriteria yang diinginkan. Beberapa kesimpulan yang dapat ditarik dari implementasi meliputi:

- Implementasi Algoritma Pencocokan *String*

Program mengimplementasikan tiga algoritma pencocokan string, yaitu algoritma Knuth-Morris-Pratt, algoritma Boyer-Moore, dan algoritma Aho-Corasick. Tiap algoritma memiliki kelebihan dan kekurangannya sendiri. Algoritma Knuth-Morris-Pratt cocok untuk pencocokan pola tunggal. Algoritma Boyer-Moore efisien untuk pola yang panjang. Algoritma Aho-Corasick digunakan untuk mencari lebih dari satu pola.

- Pencocokan *Exact Matching* dan *Fuzzy Matching*

Pencocokan dilakukan dalam dua tahap, yaitu tahap *exact matching* dan tahap *fuzzy matching*. Pada tahap *exact matching*, mesin pencari akan mencari instans-instans yang cocok persis dengan *keyword* yang dimasukkan. Setelah itu, program akan melakukan *fuzzy matching* yang memanfaatkan *Levenshtein Distance* untuk memberikan toleransi pada kesalahan ketik dan variasi kata.

- Pemrosesan CV

Sistem memproses file CV dalam format PDF, mengekstrak teks mentah, dan menguraikannya menjadi informasi yang lebih terstruktur menggunakan kombinasi dari pustaka *pdfplumber* dan *regex*.

- Peningkatan Performa

Program memanfaatkan *multiprocessing* untuk mempercepat proses pencarian pada kumpulan data yang besar, di mana tugas pencarian didistribusikan ke beberapa prosesor.

- Keamanan Data

Lapisan keamanan dasar telah diterapkan untuk mengenkripsi data pribadi dari pelamar. Hasil enkripsi tersebut dapat di-reverse (*reversible*) untuk mendapat kembali data aslinya.

Secara keseluruhan, aplikasi ini menunjukkan penerapan praktis dari algoritma pencocokan *string* dan aplikasi dari Regex dalam kasus penggunaan nyata.

5.2. Saran

Meskipun program telah memenuhi spesifikasi utama, terdapat beberapa daerah yang masih dapat diperbaiki atau dikembangkan lebih lanjut. Daerah-daerah tersebut meliputi:

- Optimasi Performa

Walaupun *multiprocessing* telah diimplementasikan, proses pemuatan dan penguraian data di awal masih bisa menyebabkan *bottleneck*. Pemanfaatan mekanisme *caching* agar tidak perlu melakukan ekstraksi dari PDF berulang kali.

- Penggunaan Data CV yang lebih ATS-friendly

Data CV yang digunakan tidak semuanya ATS-friendly. Akibatnya, ada bagian teks yang tidak terbaca dengan benar, sehingga hasilnya tidak sesuai dengan apa yang diinginkan.

- Peningkatan Akurasi Ekstraksi CV

Ekstraksi fitur dari CV berbasis regex cukup efektif namun pada kasus nyata cukup rentan terhadap kegagalan jika format CV tidak standar. Sistem dapat menggunakan model Natural Language Processing untuk mengidentifikasi entitas dengan akurasi yang lebih tinggi. Ekstraksi data teks dari gambar pun tidak dapat dilakukan karena *Optical Character Recognition* tidak diimplementasikan.

5.3. Refleksi

- Tantangan Utama

Tantangan terbesar adalah menangani keragaman format CV yang tidak terstruktur. Merancang pola regex yang dapat berfungsi secara andal untuk berbagai CV terbukti sangat sulit.

- Pembelajaran yang Diperoleh

- Keterbatasan Regex

Walaupun regex cukup efektif dalam pemrosesan teks, implementasi ini menunjukkan batasan-batasan yang muncul dari pendekatan ini. Di depannya, penggunaan pendekatan lain layak dipertimbangkan.

- Penerapan Teori ke Praktik
Tugas ini menunjukkan bagaimana cara mengimplementasi algoritma-algoritma yang dipelajari ke dalam proyek yang nyata.
- Kelebihan dan Kekurangan Algoritma Pencocokan *String*
Tiap algoritma yang diimplementasikan memiliki kelebihan dan kekurangannya sendiri. Tidak ada algoritma yang paling bagus untuk seluruh kasus.

LAMPIRAN

Tautan Repository: [Tubes3_chisli](#)

Link Video: [Tugas Besar Stima 3](#)

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

DAFTAR PUSTAKA

- Munir, R. 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)), diakses 13 Juni 2025
- Khodra, M. L. 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)), diakses 13 Juni 2025