

IF2211 – Strategi Algoritma

Pemanfaatan Pattern Matching untuk Membangun Sistem ATS

(Applicant Tracking System) Berbasis CV Digital

Laporan Tugas Besar 3

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma pada Semester 2 Tahun Akademik 2024/2025



Disusun oleh:

Muhammad Farrel Wibowo (13523153)

Theo Kurniady (13523154)

I Made Wiweka Putera (13523160)

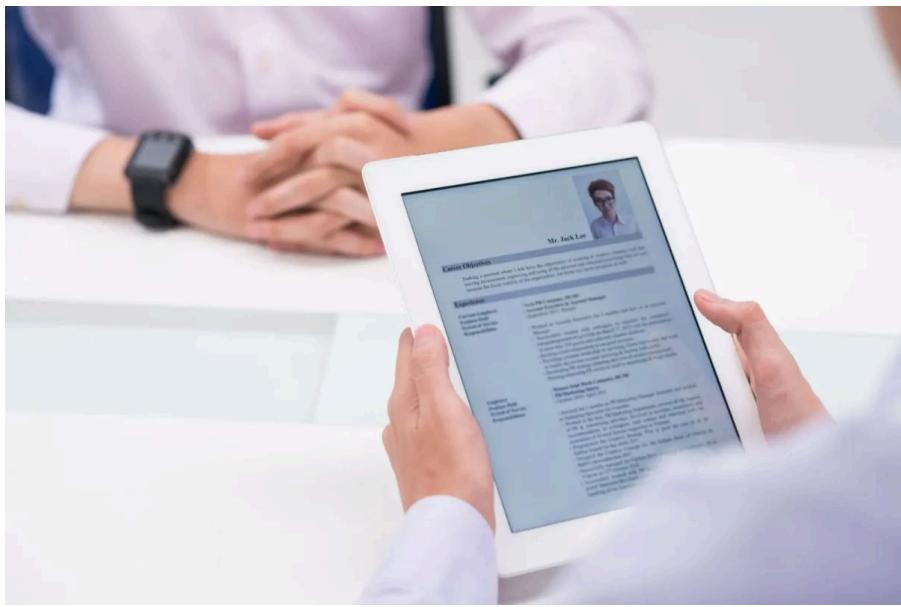
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2025

DAFTAR ISI

DAFTAR ISI.....	1
BAB I	
DESKRIPSI TUGAS.....	2
Penjelasan Implementasi.....	3
Penggunaan Program.....	5
BAB II	
LANDASAN TEORI.....	7
2.1. String Matching.....	7
2.2. Algoritma Knuth-Moris-Pratt (KMP).....	7
2.3. Algoritma Boyer Moore.....	8
2.4. Garis Besar Aplikasi.....	10
BAB III	
ANALISIS PEMECAHAN MASALAH.....	11
3.1. Langkah-langkah pemecahan masalah.....	11
3.2. Proses pemetaan masalah menjadi elemen-elemen algoritma KMP dan BM.....	15
3.3. Fitur fungsional dan arsitektur aplikasi yang dibangun.....	16
3.4. Contoh ilustrasi kasus.....	18
BAB IV	
IMPLEMENTASI DAN PENGUJIAN.....	21
4.1. Spesifikasi teknis program.....	21
4.1.1. Knuth Morris Pratt Algorithm.....	21
4.1.2. Boyer Moore Algorithm.....	21
4.1.3. Levenshtein Distance.....	21
4.1.4. Models.....	22
4.2. Tata cara penggunaan program.....	24
4.3. Hasil pengujian.....	27
4.4. Analisis hasil pengujian.....	29
BAB V	
IMPLEMENTASI BONUS.....	29
5.1. Implementasi Algoritma Aho-Corasick.....	30
5.2. Video Aplikasi.....	31
BAB VI	
PENUTUP.....	32
6.1. Kesimpulan.....	32
6.2. Saran.....	32
6.3. Refleksi.....	32
LAMPIRAN.....	34
DAFTAR PUSTAKA.....	35

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.

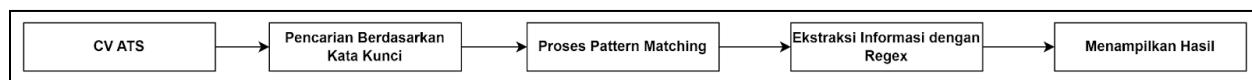
Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan 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.

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 keyword dapat dicari secara efisien dalam satu representasi data linear.

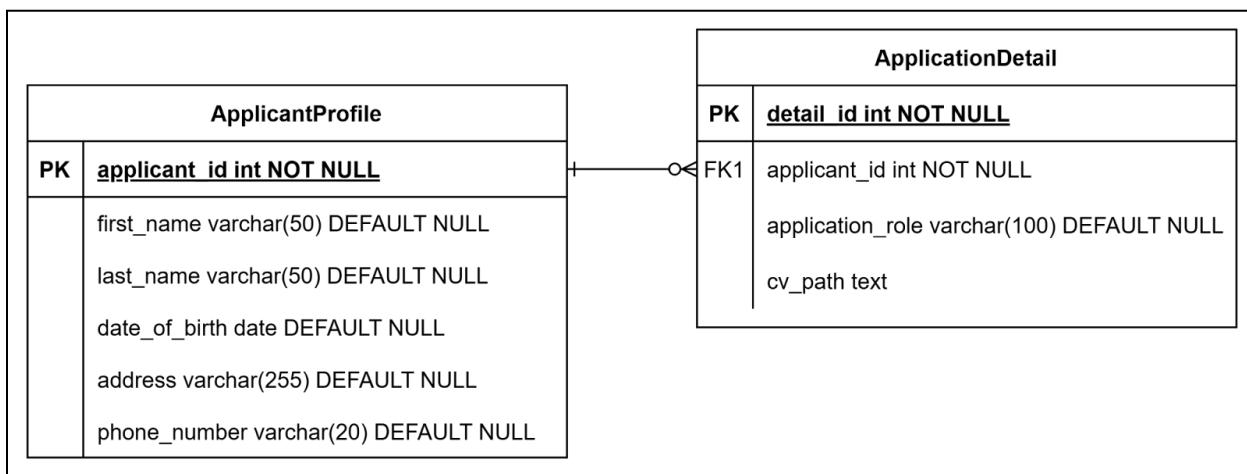
Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan. Dataset yang digunakan dalam contoh ini merupakan dataset CV ATS yang tercantum pada bagian referensi.

Tabel 1. Hasil ekstraksi teks dari CV ATS

CV ATS	Ekstraksi Text untuk Regex	Ekstraksi Text untuk <i>Pattern Matching</i> (KMP & BM)
10276858.pdf	Ekstraksi Text Regex.txt	Ekstraksi Text Pattern Matching.txt

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

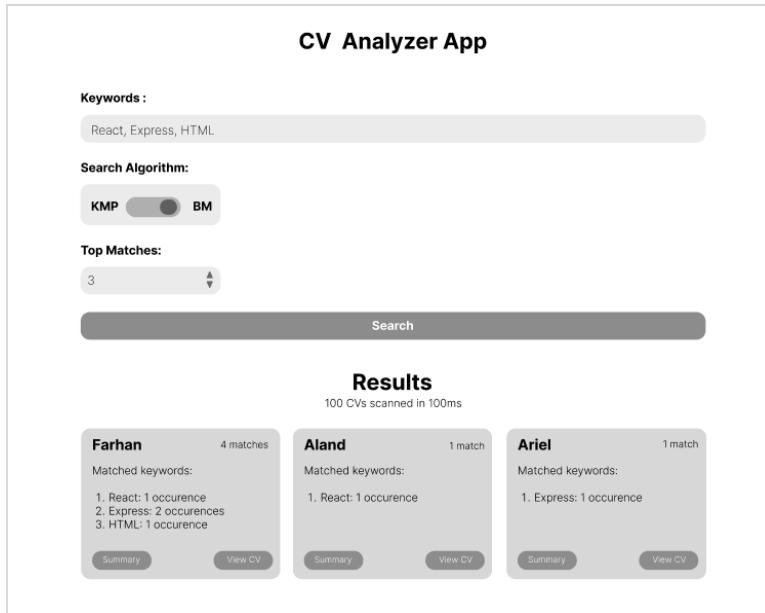
Dalam skema basis data ini, tabel **ApplicantProfile** menyimpan informasi pribadi pelamar, sedangkan tabel **ApplicationDetail** menyimpan detail aplikasi yang diajukan oleh pelamar tersebut. Relasi antara tabel **ApplicantProfile** dan **ApplicationDetail** adalah one-to-many, karena seorang pelamar 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.

Penggunaan Program



Gambar 4. Contoh Antarmuka Program (Halaman Home)



Gambar 5. Contoh Antarmuka Program (Halaman Summary)

Anda diperbolehkan menambahkan elemen tambahan seperti gambar, logo, atau komponen visual lainnya. Desain antarmuka untuk aplikasi desktop **tidak wajib mengikuti tata letak persis** seperti contoh yang diberikan, namun harus dibuat semenarik mungkin, serta tetap mencakup seluruh **komponen wajib yang telah ditentukan**:

- Judul Aplikasi

- Kolom input kata kunci memungkinkan pengguna memasukkan satu atau lebih *keyword*, yang dipisahkan dengan koma, seperti contoh: React, Express, HTML.
- Tombol toggle memungkinkan pengguna memilih salah satu dari dua algoritma pencarian, yaitu KMP atau BM, dengan hanya satu algoritma yang bisa dipilih pada satu waktu.
- *Top Matches Selector* digunakan untuk memilih jumlah CV teratas yang ingin ditampilkan berdasarkan hasil pencocokan.
- *Search Button* digunakan untuk memulai proses pencarian. Diletakkan secara mencolok di bawah *input field*.
- *Summary Result Section* berisi informasi waktu eksekusi pencarian untuk kedua tipe matching yang dilakukan (*exact match* dengan KMP/BM dan *fuzzy match* dengan Levenshtein Distance), misalnya: “Exact Match: 100 CVs scanned in 100ms.\n Fuzzy Match: 100 CVs scanned in 101ms.”
- *Container* hasil pencarian atau kartu CV digunakan untuk menampilkan data hasil pencocokan berdasarkan keyword yang sesuai. Setiap kartu memuat informasi seperti nama kandidat, jumlah kecocokan yang dihitung dari jumlah keyword yang ditemukan, serta daftar kata kunci yang cocok beserta frekuensi kemunculannya. Selain itu, tersedia dua tombol aksi: tombol *Summary* untuk menampilkan ekstraksi informasi dari CV, serta tombol *View CV* yang memungkinkan pengguna melihat langsung file CV asli.

Secara umum, berikut adalah cara umum penggunaan program:

1. Pengguna memasukkan kata kunci pencarian.
2. Memilih algoritma pencocokan: KMP atau BM.
3. Menentukan jumlah hasil yang ingin ditampilkan.
4. Menekan tombol Search.
5. Sistem menampilkan daftar CV yang paling relevan, disertai tombol untuk melihat detail (*Summary*) atau CV asli (*View CV*).

BAB II

LANDASAN TEORI

2.1. String Matching

String Matching atau pencocokan pola merupakan proses pencarian keberadaan suatu pola (pattern) dalam suatu teks atau data (teks). Pola tersebut bisa berupa string karakter, ekspresi reguler, atau struktur tertentu. Pencocokan pola digunakan secara luas dalam berbagai bidang, seperti pengolahan teks, analisis DNA, sistem pencarian, dan keamanan komputer.

Secara umum, problem pattern matching dapat dirumuskan sebagai berikut:

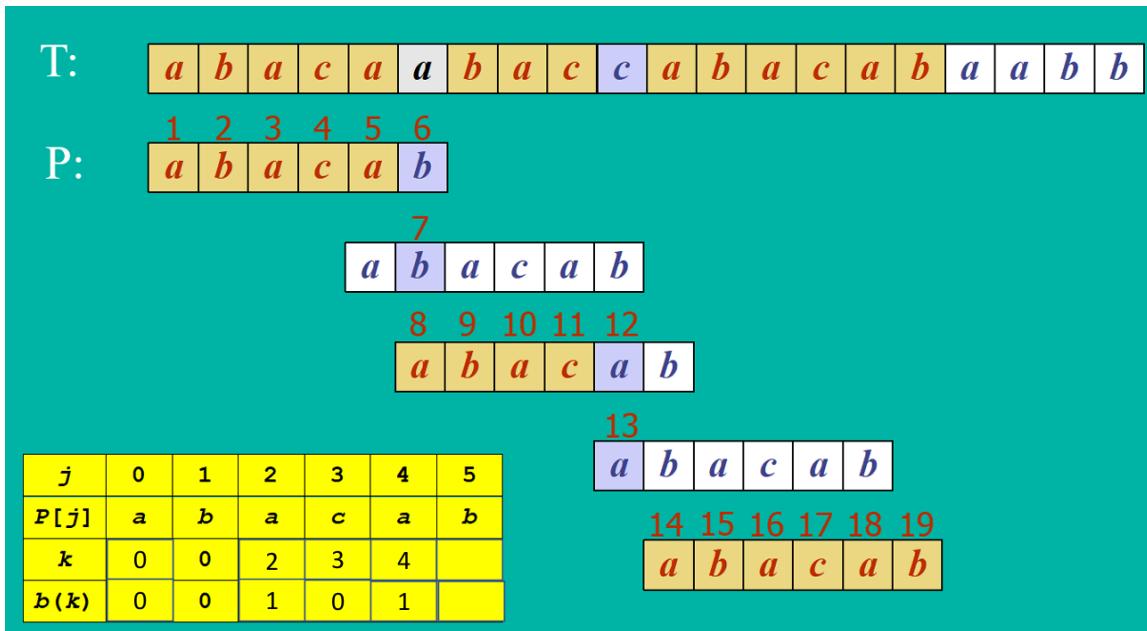
Diberikan string T (teks) sepanjang n dan string P (pattern) sepanjang m, temukan semua posisi dalam T di mana P cocok secara persis.

Beberapa algoritma klasik yang digunakan untuk menyelesaikan masalah ini secara efisien antara lain:

1. Brute Force
2. Knuth-Morris-Pratt (KMP)
3. Boyer-Moore (BM)

2.2. Algoritma Knuth-Moris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah salah satu algoritma pencocokan string. Algoritma ini melakukan pemeriksaan dari kiri ke kanan seperti bruteforce, tetapi lebih pintar dalam menggeser pattern sehingga lebih sedikit penyocokan yang dibutuhkan. KMP dirancang untuk mempercepat proses pencocokan dengan cara menghindari perbandingan karakter yang sudah diketahui hasilnya. Inti dari algoritma ini adalah penggunaan tabel bantu yang disebut Longest Prefix Suffix (LPS), yang merepresentasikan panjang awalan terpanjang (prefix) dari pattern yang juga merupakan akhiran (suffix). Saat terjadi ketidakcocokan dalam pencocokan string, tabel LPS ini digunakan untuk menentukan seberapa jauh pola bisa digeser tanpa harus mengulang perbandingan dari awal.



Gambar 2.2 Proses algoritma KMP disertai dengan tabel LPS

(Sumber : [informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf))

Dengan demikian, KMP memastikan bahwa setiap karakter dalam teks hanya dibandingkan sekali, sehingga kompleksitas waktunya menjadi $O(n + m)$, dengan n adalah panjang teks dan m adalah panjang pattern. Hal ini menjadikan KMP sangat efisien untuk kasus pencarian pola yang mengandung pengulangan atau dalam aplikasi real-time.

2.3. Algoritma Boyer Moore

Algoritma Boyer-Moore (BM) dikenal sebagai salah satu algoritma tercepat dalam praktik untuk mencari pola dalam teks. Tidak seperti KMP yang mencocokkan karakter dari kiri ke kanan, Boyer-Moore melakukan pencocokan dari kanan ke kiri dalam pola. Pergeseran pattern diatur dengan Last Occurrence Function.

L() Example

- A = {a, b, c, d}
- P: "abacab"

P	a	b	a	c	a	b
	0	1	2	3	4	5

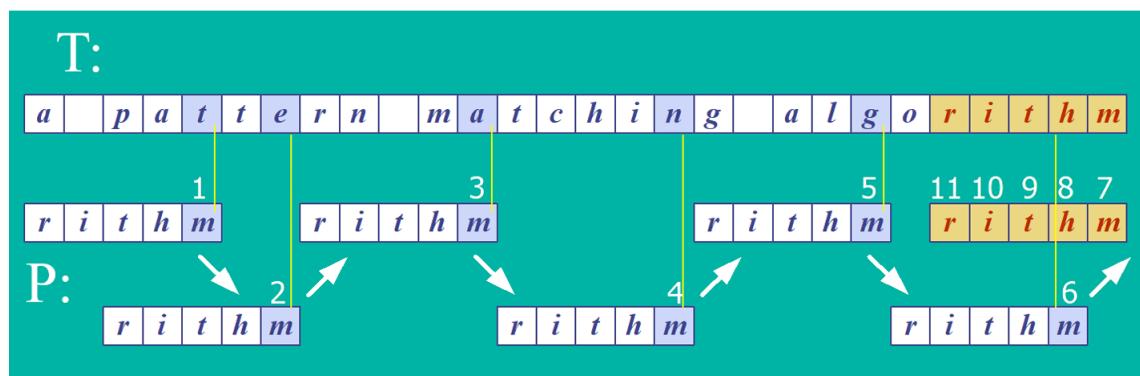
x	a	b	c	d
L(x)	4	5	3	-1

L() stores indexes into P[]

Gambar 2.3.1 Last Occurrence Function

(Sumber : [informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf))

Fungsi ini menentukan seberapa jauh pola dapat digeser ketika terjadi ketidakcocokan antara karakter dalam teks dan karakter dalam pola. Fungsi ini mencatat posisi terakhir kemunculan setiap karakter dalam pola. Ketika terjadi mismatch, algoritma memeriksa apakah karakter teks yang menyebabkan mismatch terdapat dalam pola. Jika karakter tersebut ada, maka pola digeser sehingga posisi terakhir karakter tersebut dalam pola sejajar dengan posisi karakter dalam teks. Jika tidak ada, pola dapat digeser sepenuhnya melewati posisi karakter tersebut.



Gambar 2.3.2 Proses algoritma Boyer Moore

(Sumber : [informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf))

Meskipun dalam kasus terburuk kompleksitas waktunya adalah $O(n \times m)$, dalam kebanyakan kasus nyata Boyer-Moore dapat berjalan jauh lebih cepat daripada KMP maupun metode brute-force.

2.4. Garis Besar Aplikasi

Projek ini merupakan aplikasi berbasis web yang memungkinkan pengguna mencari dan memfilter CV dengan kata kunci yang diinginkan dari daftar CV yang ada dengan menggunakan algoritma string matching. Pengguna dapat memilih lebih dari 1 kata kunci yang diinginkan, menentukan algoritma yang digunakan, serta jumlah maksimum CV yang ingin ditampilkan. Setelah proses pencarian dijalankan, hasilnya akan ditampilkan pada layar dengan informasi singkat dari setiap CV yang memenuhi kriteria. Aplikasi dirancang secara keseluruhan dengan bahasa pemrograman python. Tampilan bekerja secara interaktif dan responsif dengan menggunakan frontend yang menggunakan library flet dari python. Seluruh proses bekerja dengan lancar dengan backend yang menggunakan

BAB III

ANALISIS PEMECAHAN MASALAH

3.1. Langkah-langkah pemecahan masalah

Untuk mempermudah penyelesaian masalah, dilakukan pemecahan menjadi beberapa komponen permasalahan.

1. PDF Parsing

Semua CV disimpan pada lokal, lebih tepatnya pada folder direktori “data/{folder CV}”. Agar bisa melakukan string matching, setiap CV harus disederhanakan menjadi sebuah string yang mengandung semua kata pada CV secara urut. Selain itu, CV juga harus diparse menjadi sebuah string yang mengandung ‘\n’ sebagai penanda *enter*. String dalam format ini adalah string yang lebih rapi dalam tampilan dan digunakan dalam proses regex.

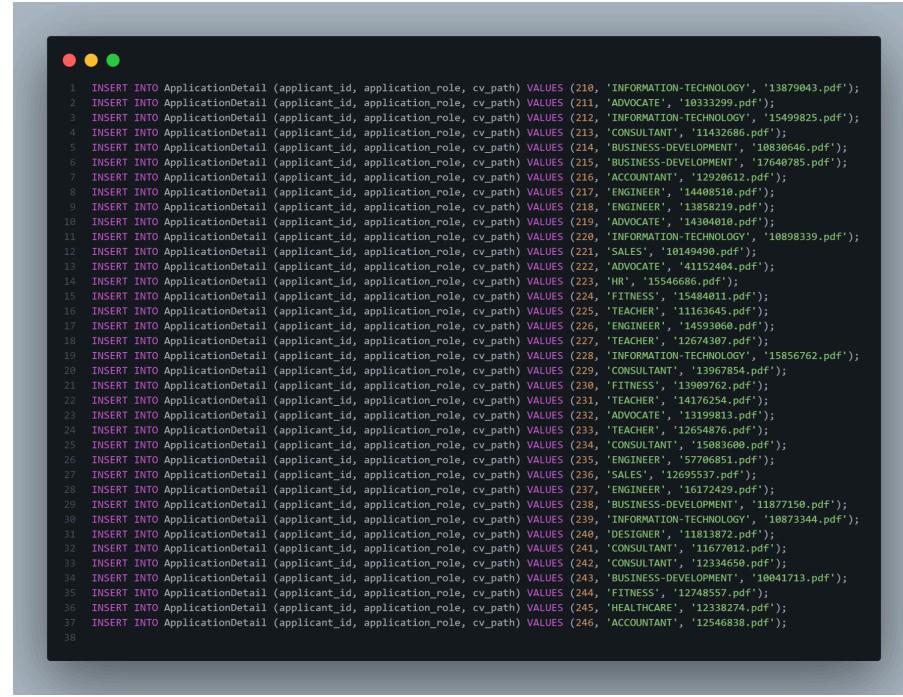
```
1 [ {  
2     "structured": "Header\n\nMORTGAGE BANKING FORECLOSURE SPECIALIST\nSummary\nAmbitious, self-motivated professional with a passion for quality work. Seeking a baseline opportunity in Underwriting, Lending, Auditing, Quality Assurance, or Analyst roles. Possess\nexperience\nMortgage Banking Foreclosure Specialist 01/2014 to Current Company Name City , State\nForeclosure Department\nMaintained beneath a 3% error ratio in all searches performed\nService member Civil Relief Act (SCRA)\n\nUse of industry mainframes; (LPS, MSP, Vendorscape, Lenstar, Resware and Lotus) to efficiently communicate with internal clients.\nReview of cases and all milestones requested by clients to ensure proper procedures and industry guidelines are used.\nConsumer Underwriter II 10/2011 to 12/2013 Company Name City , State\nHome Preservation\nExceeded monthly production goals while adhering to a minimum 5% error ratio\nMaintained a high level of production and maintained high quality standards\n\nRestructured delinquent consumer mortgage loans in accordance with company, FHA, and industry guidelines using an in house underwriting\nsoftware\n\nMaintained quality control standards while maintaining production standards by company's definition\nAnalyzed applicants' financial status, credit and property evaluation to determine feasibility of granting loan.\n\nPerformed final approvals and reviews the entire loan file through verifications processes, including adherence to multiple investor guidelines\nMaximized quality by verifying underwriting conditions and approval requirements are met\n\nBuilt knowledge about latest banking products and services\nAnalyzed customer credit history in order to determine customer willingness to pay and affordability for various payment plan options.\n\nProvided meticulous attention to detail in underwriting mortgages. Evaluated the financial strengths and weaknesses of borrowers to determine risk and repayment capacity in a loss prevention environment.\nConducted peer reviews for fellow teammates, offering methodology and logic to income cash flows prior to recommendation for resolution/modification.\nAnalyzed income documentation consisting of: paystubs, Profit and loss statements, tax transcripts, personal and business\nFederal and State Tax Returns, Rental income, S-Corps, Schedule C, 1120S, K-1's, personal and business bank statements, IES Military\npaystubs, W2's, 1099's, fixed income sources, etc.\n\nLoan Document Specialist II 08/2008 to 01/2010 Company Name City , State\nMortgage Loan Operations\nMaintained below a 5% error rating on booking 50+ loans daily\nBooking and review of conventional, F.H.A. & V.A. loans\n\nData entry functions including booking and review of recorded security instruments\nReviewed documentation for errors & omissions of security documents\n\nPerformed daily maintenance of the loan applicant database.\n\nLoan Processor 04/2003 to 08/2008 Company Name City , State\nMortgage Lending Services\nCorresponded with customers, management, and title companies, to respond to inquiries\nInterpreted company policies while analyzing the applicant, property, and documentation to minimize the need for subsequent follow ups with\nborrowers\n\nVerified and validated supporting income, asset, and liability documentation to ensure validity\nCleared all title exceptions and errors\n\nSuccessfully maintained a minimum volume of 30+ loan packages daily with no errors\n\nAccomplishments\nSharepoint, Early Resolution, FMA Connection, DDS LPS, MSP, CREDCO, REIS, Microsoft Word, Outlook, Live Meeting, Excel, PowerPoint, SLOAD, DAT and various other programs\n3 years in Default Servicing\n3 years Loss Prevention/ Loss Mitigation * 7 years\n\nMortgage Loan Processing/Mortgage Banking * 3 years Underwriting/Lending * 3 years Risk Management/ Analysis * 3 years\n\nEducation\nAssociate of Science : Business Administration Auburn University at Montgomery City , State\nSkills\nLoans, Mortgage, Documentation, Lending, Liability, Loan Processor, Mortgage Lending, Processor, Ups, Underwriting, Fha, Foreclosure, Cash, Credit, File, Financial Statements, Fixed Income, Mortgage Loans, Quality Control, State Tax, Tax Returns, Team Lead, Banking Loan, Data\nEntry, Loan Operations, Mortgage Loan, Operations, Security, Cases, Clients, Mortgage Banking, Audits, Bankruptcy, Fannie Mae, Internal\nAudits, Niss, Production Environment, Sales, Solutions, Telephone, Ambitious, Articulate, Auditing, Closing, Credit Analysis, Detail-oriented, Dos, \nExcel, Fast Learner, Loan Closing, Loss Mitigation, Loss Prevention, Microsoft Sharepoint, Mitigation, Outlook, Pipeline, Powerpoint, Problem\nSolver, Quality Assurance, Reviewing Financial Statements, Risk Assessment, Risk Management, Sharepoint, Trading, Word, Real Estate, Real\nEstate Analysis",  
3     "flattened": "mortgage banking foreclosure specialist summary ambitious selfmotivated professional with a passion for quality work seeking a baseline opportunity in underwriting lending auditing quality assurance or analyst roles possess large spectrum of experience in the financial industry i am a fast learner who values my employer personal characteristics detailoriented thorough computersavvy loyal persistent adaptable eager to learn accomplishments a sharepoint early resolution fha connection dos lps msp credco rels microsoft word outlook live meeting excel powerpoint sload dat and various other programs 3 years in default servicing 3 years loss prevention loss mitigation 7 years mortgage loan processingmortgage banking 3 years underwritinglending 3 years risk management analysis 3 years compliancequality assurance 10 years loan operations operations experience experience mortgage banking foreclosure specialist 01/2014 to current company name city state foreclosure department maintained beneath a 3% error ratio in all searches performed service member civil relief act scra use of industry mainframes lps msp vendorscape lenstar resware and lotus to efficiently communicate with internal clients review of cases and all milestones requested by clients to ensure proper procedures and industry guidelines are used consumer underwriter ii 10/2011 to 12/2013 company name city state home preservation exceeded monthly production goals while adhering to a minimum 5% error ratio maintained a high level of production and maintained high quality standards restructured delinquent consumer mortgage loans in accordance with company fha and industry guidelines using an in house underwriting software maintained quality control standards while maintaining production standards by companys definition analyzed applicants financial status credit and property evaluation to determine feasibility of granting loan performed final approvals and reviews the entire loan file through verifications processes including adherence to multiple investor guidelines maximized quality by verifying underwriting conditions and approval requirements are met built knowledge about latest banking products and services through analyzed customer credit history in order to determine customer willingness to pay and affordability for various payment plan options provided meticulous attention to detail in underwriting mortgages evaluated the financial strengths and
```

Gambar 3.1.1 Contoh hasil string dari parsing CV

2. Database Management

Database adalah bagian penting dari aplikasi ini. Format CV yang dapat discan tidak memiliki format nama dan role dari detil aplikasi pekerjaan. Ketika

menjalankan aplikasi, sistem akan melakukan seeding nama dan role yang dipilih oleh aplikasi pada detil aplikasi ke database SQL.



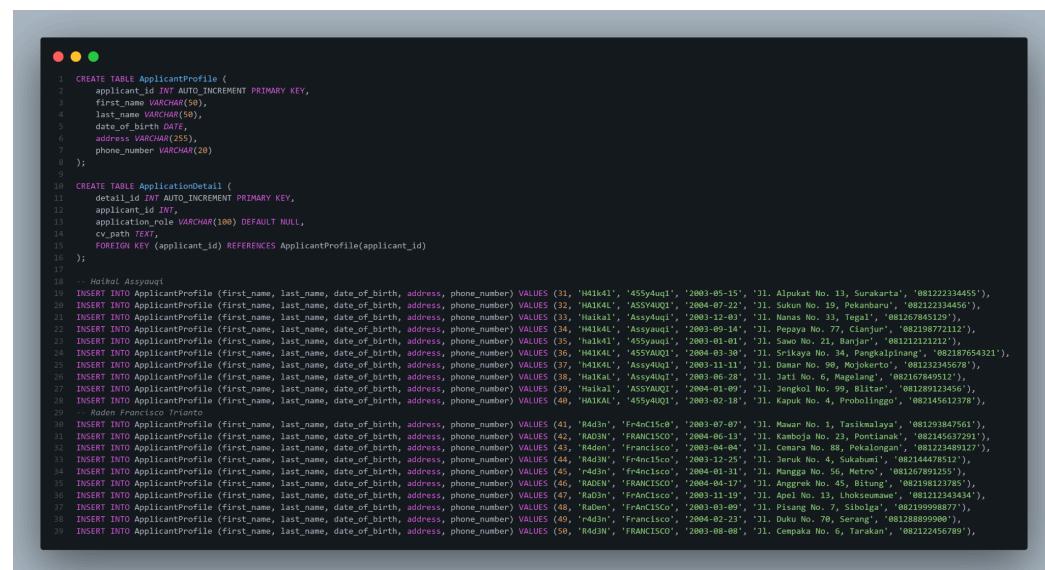
```

1  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (210, 'INFORMATION-TECHNOLOGY', '13879043.pdf');
2  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (211, 'ADVOCATE', '10333299.pdf');
3  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (212, 'INFORMATION-TECHNOLOGY', '15499825.pdf');
4  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (213, 'CONSULTANT', '11432686.pdf');
5  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (214, 'BUSINESS-DEVELOPMENT', '19830646.pdf');
6  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (215, 'BUSINESS-DEVELOPMENT', '17646785.pdf');
7  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (216, 'ACCOUNTANT', '12929612.pdf');
8  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (217, 'ENGINEER', '1408810.pdf');
9  INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (218, 'ENGINEER', '13858219.pdf');
10 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (219, 'ADVOCATE', '14304010.pdf');
11 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (220, 'INFORMATION-TECHNOLOGY', '10898339.pdf');
12 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (221, 'SALES', '10149490.pdf');
13 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (222, 'ADVOCATE', '41152484.pdf');
14 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (223, 'HR', '15546686.pdf');
15 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (224, 'FITNESS', '15484011.pdf');
16 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (225, 'TEACHER', '11163645.pdf');
17 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (226, 'ENGINEER', '14593063.pdf');
18 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (227, 'TEACHER', '12674307.pdf');
19 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (228, 'INFORMATION-TECHNOLOGY', '15856762.pdf');
20 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (229, 'CONSULTANT', '13967854.pdf');
21 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (230, 'FITNESS', '13989762.pdf');
22 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (231, 'TEACHER', '14176254.pdf');
23 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (232, 'ADVOCATE', '13199813.pdf');
24 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (233, 'TEACHER', '12654876.pdf');
25 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (234, 'CONSULTANT', '15083606.pdf');
26 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (235, 'ENGINEER', '57706851.pdf');
27 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (236, 'SALES', '12699537.pdf');
28 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (237, 'ENGINEER', '16172429.pdf');
29 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (238, 'BUSINESS-DEVELOPMENT', '11877150.pdf');
30 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (239, 'INFORMATION-TECHNOLOGY', '10873344.pdf');
31 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (240, 'DESIGNER', '11813872.pdf');
32 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (241, 'CONSULTANT', '11677012.pdf');
33 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (242, 'CONSULTANT', '12334650.pdf');
34 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (243, 'BUSINESS-DEVELOPMENT', '10804173.pdf');
35 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (244, 'FITNESS', '12748557.pdf');
36 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (245, 'HEALTHCARE', '12338274.pdf');
37 INSERT INTO ApplicationDetail (applicant_id, application_role, cv_path) VALUES (246, 'ACCOUNTANT', '12546838.pdf');
38
39

```

Gambar 3.1.2 Seeding Application Detail

Setiap detil aplikasi memiliki ID yang unik yang merujuk pada ID aplikan, role, dan file CV pdf yang sudah ada pada lokal folder data/.



```

1  CREATE TABLE ApplicantProfile (
2    applicant_id INT AUTO_INCREMENT PRIMARY KEY,
3    first_name VARCHAR(50),
4    last_name VARCHAR(50),
5    date_of_birth DATE,
6    address VARCHAR(255),
7    phone_number VARCHAR(20)
8  );
9
10 CREATE TABLE ApplicationDetail (
11   detail_id INT AUTO_INCREMENT PRIMARY KEY,
12   applicant_id INT,
13   application_role VARCHAR(100) DEFAULT NULL,
14   cv_path TEXT,
15   FOREIGN KEY (applicant_id) REFERENCES ApplicantProfile(applicant_id)
16 );
17
18 Raden Francisco Trianto
19 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('31', 'HAikal', '19850401', '2003-05-15', 'Jl. Alipukot No. 13, Surakarta', '081222334451'),
20 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('32', 'HAikal', '19850401', '2004-07-22', 'Jl. Sukun No. 10, Pekanbaru', '082122334456'),
21 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('33', 'Haikal', 'Assyauqi', '2003-12-01', 'Jl. Nama No. 33, Tegal', '0821267845129'),
22 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('34', 'Haikal', 'Assyauqi', '2003-09-14', 'Jl. Pepaya No. 77, Cianjur', '082198772112'),
23 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('35', 'Haikal', 'Assyauqi', '2003-01-01', 'Jl. Sawo No. 21, Banjar', '082121212121'),
24 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('36', 'Haikal', 'Assyauqi', '2004-01-30', 'Jl. Srikaya No. 34, Pangkalpinang', '082137654321'),
25 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('37', 'Haikal', 'Assyauqi', '2003-01-15', 'Jl. Jati No. 90, Magelang', '08212345678901'),
26 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('38', 'Haikal', 'Assyauqi', '2003-06-28', 'Jl. Jati No. 99, Blitar', '082128051212'),
27 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('39', 'Haikal', 'Assyauqi', '2004-01-09', 'Jl. Jengkol No. 99, Blitar', '082128012456'),
28 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('40', 'HAikal', 'Assyauqi', '2003-02-10', 'Jl. Kapuk No. 4, Probolinggo', '082145612378'),
29
30 Raden Francisco Trianto
31 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('41', 'Raden', 'Francisco', '2003-07-07', 'Jl. Mawar No. 3, Tasikmalaya', '081239347661'),
32 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('42', 'Raden', 'Francisco', '2004-06-13', 'Jl. Cempaka No. 23, Bandung', '081245678901'),
33 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('43', 'Raden', 'Francisco', '2004-06-04', 'Jl. Cempaka No. 88, Pekalongan', '081234488937'),
34 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('44', 'Raden', 'Francisco', '2003-12-25', 'Jl. Jeruk No. 4, Sukahumi', '082144478512'),
35 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('45', 'Raden', 'Francisco', '2004-01-31', 'Jl. Mangga No. 56, Metro', '081267891255'),
36 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('46', 'RADEN', 'FRANCISCO', '2004-04-17', 'Jl. Angrek No. 45, Bitung', '082198123785'),
37 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('47', 'Raden', 'Francisco', '2003-11-19', 'Jl. Apei No. 13, Lhokseumawe', '081212434343'),
38
39 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('48', 'Raden', 'Francisco', '2003-03-09', 'Jl. Pisang No. 7, Sibolga', '082199988777'),
40
41 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('49', 'Raden', 'Francisco', '2004-02-23', 'Jl. Duku No. 70, Serang', '081288999506'),
42
43 INSERT INTO ApplicantProfile (first_name, last_name, date_of_birth, address, phone_number) VALUES ('44', 'Raden', 'Francisco', '2003-08-09', 'Jl. Cempaka No. 6, Tarakan', '082122456789'),
44

```

Gambar 3.1.3 Seeding Applicant Profile

Gambar diatas menunjukkan isi dari profil aplikan. Untuk mendapat informasi yang lebih jelas pada aplikan, ID aplikan merujuk kepada profil aplikan (sebagai sebuah foreign key). Setiap ID aplikan unik dan merujuk pada nama awalan, nama akhiran, tanggal lahir, alamat, dan nomor telepon.

3. API Client

Frontend (tampilan antarmuka) dengan backend (logika pencarian) akan dihubungkan dengan API. Ketika pengguna menekan tombol pencarian pada tampilan, aplikasi akan mengutilisasi API untuk mengirim sebuah URL HTTP berisi informasi yang pengguna masukkan pada input, yaitu keyword, algoritma, dan jumlah CV teratas. URL akan diterima oleh backend dan diperiksa kesesuaian informasi yang ada. Untuk proses pencarian, sebuah kelas yang mengurus perihal searching (search_service) akan dipanggil dan informasi akan diteruskan ke sebuah pabrik / factory dari algoritma pattern matching. Sesuai dengan algoritma yang dipilih, pabrik algoritma akan mengurus algoritma yang dipanggil. Hasil pencarian akan dimasukkan ke dalam sebuah dictionary. Dictionary ini akan dikembalikan melalui API lagi agar dapat ditampilkan pada frontend.

API juga digunakan untuk fitur lain pada aplikasi ini, seperti pada pengambilan informasi aplikan yang dibutuhkan dalam summary. Sama seperti pencarian, URL akan dibawa ke fungsi untuk mengambil summary pada search_service. Hasil akan dikembalikan melalui API agar balik ke frontend. Sama halnya seperti fitur menampilkan CV dari aplikan, URL akan dibawah ke fungsi untuk mengambil CV sesuai dengan file pdf CV yang terhubung dengan aplikan pada database melalui search_service. API juga mengembalikan hasil ke frontend.

4. String Matching

Melalui proses sebelumnya, API mengirim informasi ke pabrik algoritma string matching untuk memilih algoritma yang dipilih pengguna. Pada setiap algoritma, semua string hasil parsing akan diambil satu per satu untuk dilakukan string matching. String matching dapat dilakukan dengan 3 algoritma berbeda dengan 2 algoritma wajib yang ada pada berikut.

a. Knuth Maris Pratt

i. LPS (Longest Prefix Suffix) Array:

Fungsi `compute_lps_array()` menghitung array `lps[]` yang menyimpan panjang prefix terpanjang dari pola yang juga merupakan suffix untuk setiap posisi. Jika karakter cocok, panjang prefix diperpanjang. Jika tidak, gunakan nilai sebelumnya dalam `lps` untuk menentukan langkah berikutnya.

ii. Proses Pencocokan Karakter:

Fungsi akan membandingkan karakter pola dan teks dari kiri ke kanan. Jika cocok, lanjutkan ke karakter berikutnya. Jika semua karakter pola cocok ($j == m$), increment count dan gunakan `lps[j-1]` untuk melanjutkan pencarian. Jika tidak cocok, gunakan nilai `lps` untuk memutuskan sejauh mana pola perlu digeser tanpa memulai ulang dari awal.

iii. Hasil akhir:

Fungsi `count_occurrences()` mengembalikan jumlah semua kemunculan pattern dalam teks, termasuk kemunculan yang saling tumpang tindih.

b. Boyer Moore

i. Bad Character / Last Occurrence Table:

Fungsi `preprocess_bad_character()` membuat tabel yang menyimpan posisi terakhir kemunculan setiap karakter dalam pattern. Karakter pada teks tetapi tidak ada pada pattern akan diberi nilai posisi -1. Fungsi ini membantu menentukan seberapa jauh pattern bisa digeser saat terjadi mismatch.

ii. Proses Pencocokan Karakter:

Fungsi akan membandingkan karakter dari akhir pola ($j = m - 1$) ke awal. Jika terjadi mismatch, cari last occurrence dari karakter yang

menyebabkan mismatch dalam pattern. Geser pola sejauh yang memungkinkan agar perbandingan berikutnya lebih efisien.

iii. Hasil akhir:

Fungsi `count_occurrences()` mengembalikan jumlah semua kemunculan pattern dalam teks, termasuk kemunculan yang saling tumpang tindih.

3.2. Proses pemetaan masalah menjadi elemen-elemen algoritma KMP dan BM

Elemen	Deskripsi
Teks	Merepresentasikan CV dalam bentuk string sederhana
Pattern	Merepresentasikan keyword yang dimasukkan pengguna

Tabel diatas menunjukkan elemen yang diperlukan pada kedua algoritma string matching, yaitu string teks dan string pattern. Dengan elemen-elemen tersebut, berikut adalah rancangan singkat penggunaan algoritma KMP dan BM:

KMP:

1. Membuat tabel LPS dari Pattern.
2. Lakukan pencocokan karakter demi karakter:
 - a. Bandingkan karakter pada teks dan pattern dari kiri ke kanan.
 - b. Jika cocok, lanjutkan ke karakter berikutnya.
 - c. Jika tidak cocok, gunakan nilai LPS untuk menentukan posisi baru dalam pattern, tanpa mengulang dari awal.
3. Ulangi hingga akhir teks.

BM:

1. Membuat tabel Last Occurrence dari Pattern.
2. Mulai pencocokan dari kanan ke kiri dalam pattern:
 - a. Cocokkan karakter terakhir dalam pattern dengan karakter di posisi sejajar di teks.
 - b. Jika cocok, geser ke kiri dan teruskan pencocokan.

- c. Jika tidak cocok, gunakan Last Occurrence Rule untuk menggeser pattern sejauh mungkin ke kanan.
3. Ulangi hingga akhir teks.

3.3. Fitur fungsional dan arsitektur aplikasi yang dibangun.

Aplikasi dirancang dan dibangun sepenuhnya dengan menggunakan bahasa pemrograman python.

3.3.1. Pengembangan Backend

a. Load & Serve Data Pelamar

Backend menyediakan data pelamar (ApplicantProfile) dan detail lamaran (ApplicationDetail) dari MySQL database. Backend juga mendukung endpoint untuk mendapatkan daftar pelamar dan detail aplikasi berdasarkan ID.

b. Analisis Kecocokan CV dan Kata Kunci

API menerima input kata kunci dan algoritma pencocokan (KMP, Boyer-Moore, Aho-Corasick). Aplikasi akan melakukan pencocokan string antara keyword dan isi CV. Aplikasi ini mengembalikan skor dan keyword yang cocok untuk masing-masing pelamar. Backend menyesuaikan proses pencarian berdasarkan algoritma yang dipilih.

c. Ringkasan Otomatis CV

API menyimpan dan mengembalikan ringkasan otomatis dari isi CV. Ringkasan ditampilkan di halaman summary.

d. Akses File CV

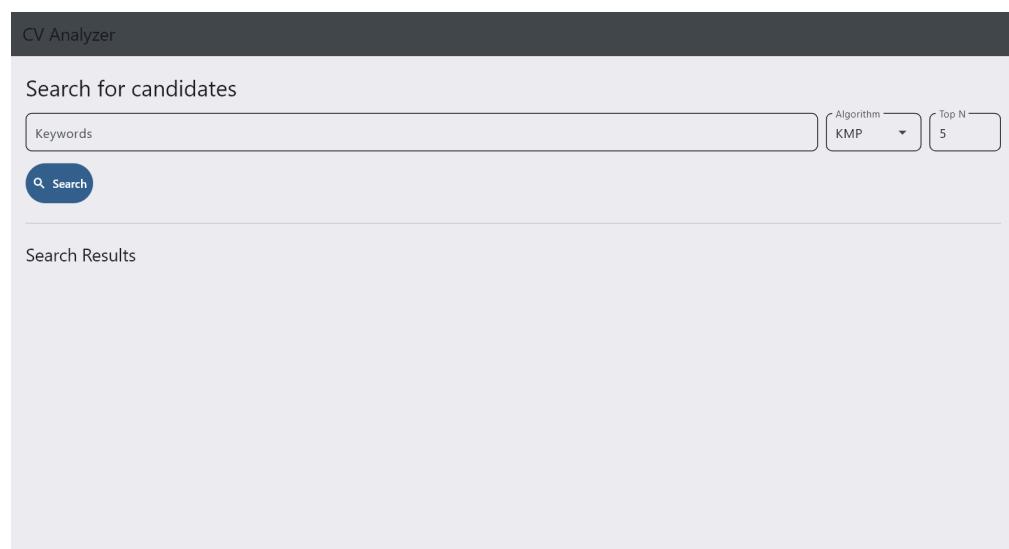
Backend menyediakan URL endpoint untuk mengakses file .pdf CV yang tersimpan secara lokal. Ini memudahkan pengguna untuk melihat CV dalam bentuk PDF dari pelamar yang sesuai.

3.3.2. Pengembangan Frontend

Aplikasi kami memiliki tampilan UI dengan menggunakan framework Flet pada python. Flet adalah sebuah framework python modern yang memungkinkan pengembang membuat aplikasi antarmuka pengguna (UI) interaktif berbasis web, desktop, maupun seluler tanpa perlu menggunakan HTML, CSS, atau JavaScript secara langsung. Flet mendukung paradigma

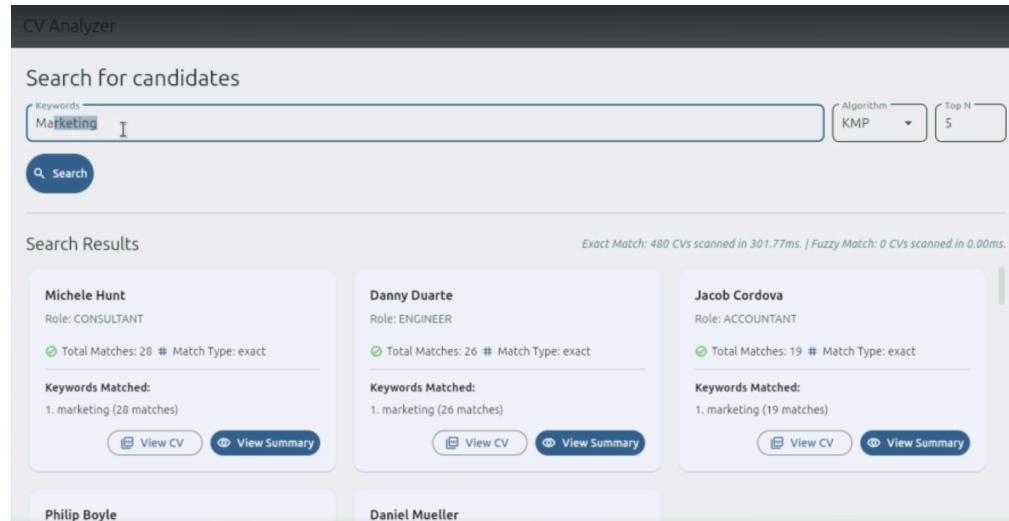
reactive programming, di mana UI secara otomatis akan merespon perubahan data atau event. Dalam konteks aplikasi ini, Flet digunakan sebagai frontend yang menerima informasi pencarian dan menampilkan antarmuka untuk memproses dan menampilkan informasi dari CV, termasuk tombol untuk memuat file, area tampilan hasil ekstraksi, dan halaman summary untuk setiap CV hasil ekstraksi.

Aplikasi ini memiliki halaman pencarian dan halaman summary dari setiap CV hasil pencarian. Halaman pencarian terdiri dari 3 daerah input, yaitu sebuah text field untuk memasukkan keyword, sebuah dropdown box untuk memilih algoritma, dan sebuah text field untuk memasukkan jumlah CV maksimum. Untuk melakukan pencarian, terdapat tombol “Search” yang dapat ditekan.



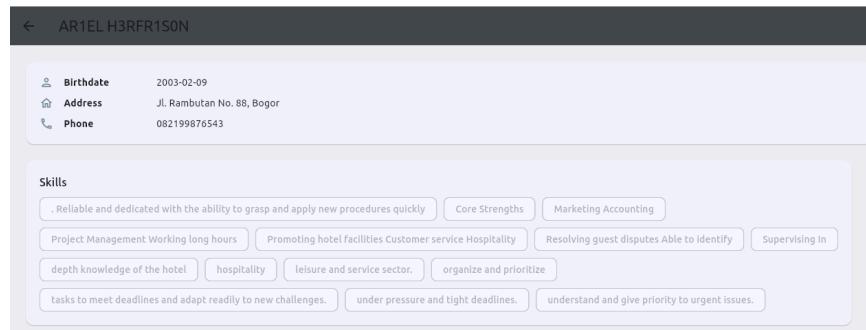
Gambar 3.3.2.1 Tampilan awal aplikasi

Di bawah label “Search Results”, terdapat sebuah container untuk menampung sejumlah CV hasil pencarian teratas sesuai dengan informasi dan kriteria input. Setiap profil ditampilkan dalam sebuah card yang berisikan informasi berupa nama aplikan, peran, jumlah pattern cocok, tipe kecocokan pattern, dan list dari keyword yang cocok. Setiap card juga menyediakan 2 tombol untuk menampilkan CV dalam bentuk pdf dan menampilkan summary dari profil tersebut.



Gambar 3.3.2.2 Tampilan aplikasi setelah searching

Tombol “View Summary” pada setiap card akan membawa pengguna ke halaman summary dari profil pemilik card. Halaman summary memuat informasi dari aplikan berupa nama, tanggal lahir, alamat, dan nomor telepon. Aplikan juga memiliki informasi mengenai kemampuan, pengalaman kerja, dan sejarah pendidikan.



Gambar 3.3.2.3 Tampilan halaman summary

3.4. Contoh ilustrasi kasus.

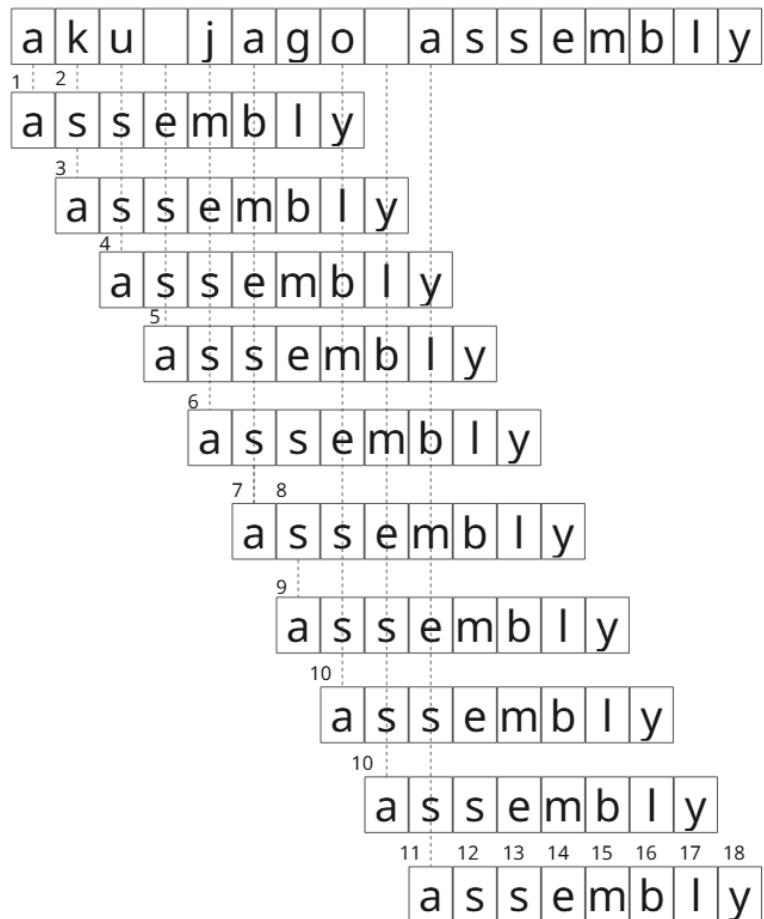
Perusahaan ABC sedang membuka rekrutmen untuk orang yang pandai menggunakan Assembly. I Made adalah seorang pekerja HRD di perusahaan ABC yang bertugas untuk mencari kandidat tersebut. Untuk mencari kandidat terbaik, I Made harus memeriksa CV dari ratusan CV yang telah diterima perusahaan untuk mencari kandidat dengan

kemampuan assembly yang baik. Untuk mempermudah pengecekan, I Made memutuskan untuk menggunakan aplikasi ini. Cara kerjanya adalah sebagai berikut.

1. I Made memasukkan “Assembly” pada kolom kata kunci. I Made juga memilih algoritma yang ingin dipakai dan juga jumlah CV maksimal sebanyak 10 untuk mempersempit ruang pemeriksaan.
2. Aplikasi menerima informasi dari frontend aplikasi dan informasi disalurkan ke backend untuk melakukan string matching. Kedua algoritma string matching bekerja sebagai berikut.

1. Algoritma KMP

kata kunci : Assembly

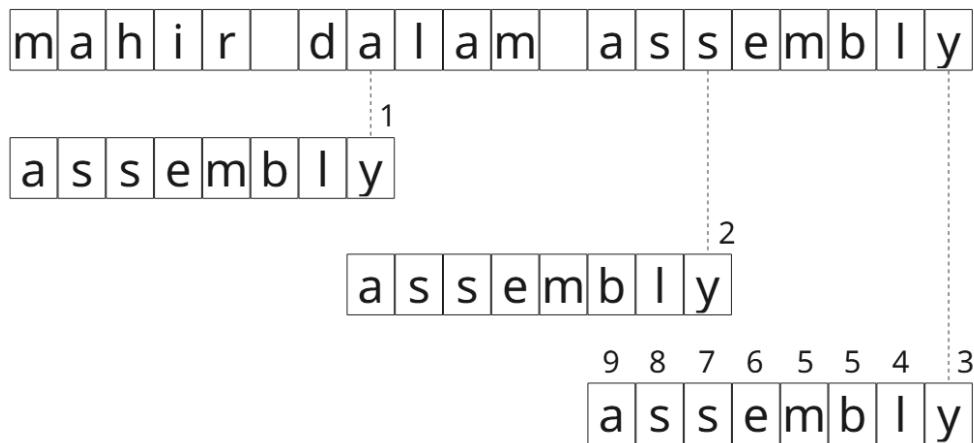


Ketika karakter tidak cocok pada karakter pertama, pattern akan langsung geser sekali. Ketidakcocokan selain karakter pertama akan membuat

pattern bergeser sesuai dengan LPS “assembly”, lalu dicek langsung dari karakter tersebut. Proses dilakukan hingga ditemukan kata di paling akhir.

2. Algoritma BM

kata kunci : Assembly



Karakter tidak cocok pada pencocokan pertama, tetapi terdapat pada pattern / kata kunci. Oleh karena itu, pattern akan bergeser sebanyak selisih panjang pattern dan indeks last occurrence dari karakter yang ditemukan ($7 - 0 = 7$). Sama halnya pada pencocokan kedua, pattern bergeser lagi (sebanyak $7 - 2 = 5$). Pencarian ke-3 sampai akhir menunjukkan kalau CV mengandung kemampuan assembly.

3. Aplikasi mengeluarkan 5 CV teratas yang mengandung kata assembly terbanyak.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi teknis program

4.1.1. Knuth Morris Pratt Algorithm

File: src\core\pattern_matching\kmp_algorithm.py

Class: KMPAlgorithm

KMPAlgorithm adalah implementasi algoritma KMP, pencarian string yang sangat cepat, khususnya untuk pola berulang.

Metode	Penjelasan Singkat
_compute_lps_array(pattern)	Menghitung array LPS (longest prefix suffix) untuk efisiensi lompatan saat tidak cocok.
count_occurrences(text, pattern)	Menghitung kemunculan pola dalam teks, menggeser pola secara optimal menggunakan LPS (longest prefix suffix).

4.1.2. Boyer Moore Algorithm

File: src\core\pattern_matching\boyer_moore_algorithm.py

Class: BoyerMooreAlgorithm

BoyerMooreAlgorithm mengimplementasikan algoritma Boyer-Moore, pencocok string yang efisien dengan lompatan cerdas.

Metode	Penjelasan Singkat
_preprocess_bad_character(pattern)	Membuat tabel karakter buruk untuk penentuan lompatan.
count_occurrences(text, pattern)	Menghitung kemunculan pola, menggunakan tabel karakter buruk untuk lompatan efisien.

4.1.3. Levenshtein Distance

File: src\core\fuzzy_matching.py

Fungsi	Deskripsi
calculate_levenshtein_distance(s1: str, s2: str) -> int	Menghitung jumlah operasi (insert, delete, substitute) minimum untuk mengubah string s1 menjadi s2.

<code>find_similar_word(keyword: str, text: str, threshold: float = 0.8) -> tuple[str, float] None</code>	Mencari kata dalam text yang memiliki kemiripan tertinggi dengan keyword berdasarkan rasio $1 - (\text{distance} / \text{length_max})$, mengembalikan kata dan skor jika melewati threshold.
--	--

4.1.4. Models

File:src\core\[models.py](#)

Class Applicant

Atribut / Property	Tipe Data	Deskripsi
applicant_id	int	ID unik pelamar.
first_name	str	Nama depan pelamar.
last_name	str	Nama belakang pelamar.
date_of_birth	date	Tanggal lahir pelamar.
address	str	Alamat lengkap pelamar.
phone_number	str	Nomor telepon pelamar.
full_name	str (property)	Menggabungkan first_name dan last_name sebagai nama lengkap.

Class Application

Atribut	Tipe Data	Deskripsi
detail_id	int	ID detail aplikasi.
applicant_id	int	ID pelamar yang mengajukan aplikasi.
application_role	str	Peran atau posisi yang dilamar.
cv_path	str	Path file CV terkait aplikasi ini.

Class CVDocument

Atribut	Tipe Data	Deskripsi
cv_id	int	ID unik untuk CV.

raw_pdf_path	str	Path ke file PDF asli.
text_for_pattern_matching	str	Teks CV untuk pencocokan pattern (algoritmik).
text_for_regex	str	Teks CV untuk ekstraksi berbasis regex.
extracted_skills	List[str]	Daftar skills hasil ekstraksi.
extracted_job_history	List[Dict[str, str]]	Daftar pengalaman kerja hasil ekstraksi.
extracted_education	List[Dict[str, str]]	Daftar pendidikan hasil ekstraksi.
extracted_summary_overview	str	Ringkasan/overview dari CV.

Metode	Deskripsi
apply_regex_extraction()	Menjalankan fungsi ekstraksi informasi dari teks CV menggunakan extractor.

Class SearchResult

Atribut	Tipe Data	Deskripsi
applicant_id	int	ID pelamar.
detail_id	int	ID detail aplikasi.
applicant_name	str	Nama lengkap pelamar.
application_role	str	Peran/posisi yang dilamar.
matched_keywords	Dict[str, int]	Dictionary keyword yang cocok dan jumlah kemunculannya.
total_matches	int (property)	Total dari semua nilai matched_keywords.

Metode	Deskripsi
to_dict()	Mengembalikan representasi objek sebagai dictionary untuk serialisasi.

Class CVSummary

Atribut	Tipe Data	Deskripsi

applicant_name	str	Nama lengkap pelamar.
birthdate	date	Tanggal lahir pelamar.
address	str	Alamat pelamar.
phone_number	str	Nomor telepon pelamar.
skills	List[str]	Daftar skill hasil ekstraksi.
job_history	List[Dict[str, str]]	Riwayat pekerjaan hasil ekstraksi.
education	List[Dict[str, str]]	Riwayat pendidikan hasil ekstraksi.
overall_summary	str	Ringkasan umum dari isi CV.
cv_path	str	Path ke file CV.

4.2. Tata cara penggunaan program

Sebelum menjalankan aplikasi, pastikan terdapat jumlah CV sebanyak 480 file. Aplikasi ini, menyusun CV menurut role dari aplikan. Secara keseluruhan, terdapat 24 role dan 20 CV untuk setiap role.



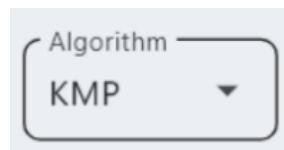
Gambar 4.2.1 CV Role Category

Pada saat menjalankan aplikasi, pengguna akan masuk ke halaman pencarian. Seperti yang dideskripsikan pada bab 3.3, aplikasi memiliki 3 kolom input. Untuk memasukkan keyword yang akan digunakan sebagai pattern dalam string matching, isi text field keyword dengan kata-kata yang diinginkan. Pengguna dapat menginput lebih dari 1 keyword dengan menambahkan koma. (Contoh keyword : Python, React)



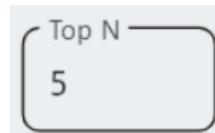
Gambar 4.2.2 Keyword Text Field

Aplikasi ini menyediakan 2 algoritma berbeda untuk melakukan string matching. Setiap kasus / pencocokan keyword dengan pada CV dengan setiap algoritma memiliki kecepatan yang bervariasi. Pengguna dapat memilih antara kedua algoritma tersebut dengan memilih pada kotak dropdown pada tampilan bagian atas.



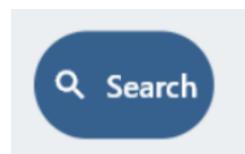
Gambar 4.2.3 Algorithm Dropdown Box

Agar pengguna bisa fokus pada pilihan teratas, pengguna dapat memasukkan jumlah CV teratas pada text field di tampilan bagian atas kanan. Input berupa integer atau angka bulat.



Gambar 4.2.4 Top N CV Text Field

Setelah memasukkan semua informasi yang dibutuhkan diatas, pengguna dapat menekan tombol berikut untuk melakukan pencarian dan pencocokan



Gambar 4.2.5 Search Button

Setelah berhasil menemukan CV yang cocok dengan keyword, profil-profil CV yang cocok akan muncul pada tengah layar. Setiap kotak profil mengandung informasi singkat, tombol untuk melihat pdf aslinya, dan tombol untuk melanjut ke halaman summary.

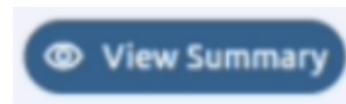
Search Results

Exact Match: 480 CVs scanned in 301.77ms. | Fuzzy Match: 0 CVs scanned in 0.00ms.

Michele Hunt	Danny Duarte	Jacob Cordova
Role: CONSULTANT	Role: ENGINEER	Role: ACCOUNTANT
28 matches	26 matches	19 matches
Keywords Matched: 1. marketing (28 matches)	Keywords Matched: 1. marketing (26 matches)	Keywords Matched: 1. marketing (19 matches)
View CV	View CV	View CV
View Summary	View Summary	View Summary

Philip Boyle Daniel Mueller

Gambar 4.2.6 Search Results & Profile Cards



Gambar 4.2.7 View Summary Button

Halaman summary mengandung semua informasi penting dari profil yang dipilih. Pengguna dapat melihat semua informasi yang dibutuhkan pada halaman tersebut. Tombol untuk kembali ke halaman pencarian juga tersedia.

← AR1EL H3RFR1SON

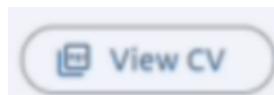
	Birthdate	2003-02-09
	Address	Jl. Rambutan No. 88, Bogor
	Phone	082199876543

Skills

- , Reliable and dedicated with the ability to grasp and apply new procedures quickly
- Core Strengths
- Marketing Accounting
- Project Management
- Working long hours
- Promoting hotel facilities
- Customer service
- Hospitality
- Resolving guest disputes
- Able to identify depth knowledge of the hotel
- hospitality
- leisure and service sector
- organize and prioritize tasks to meet deadlines and adapt readily to new challenges
- under pressure and tight deadlines
- understand and give priority to urgent issues

Gambar 4.2.8 Halaman Summary

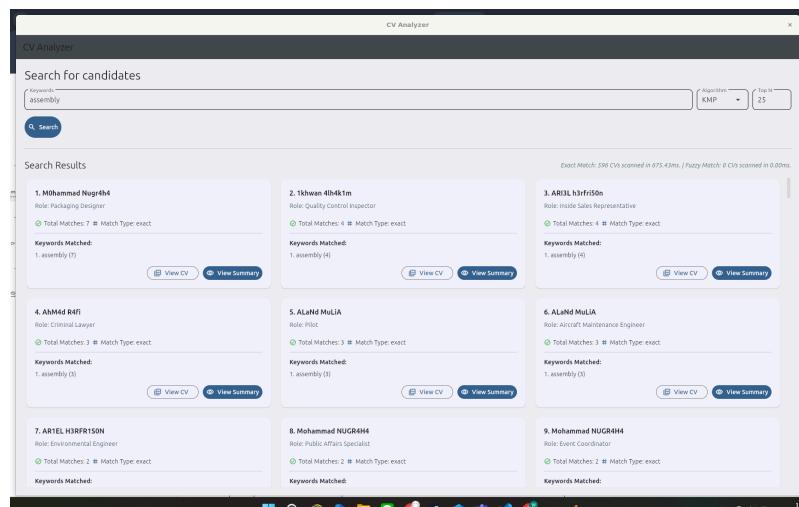
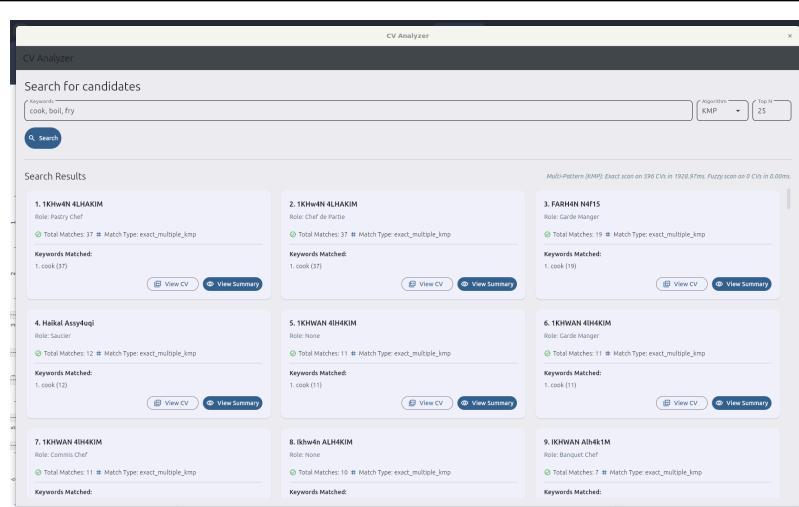
Bila ingin yang lebih detil, pengguna juga dapat memilih tombol “View CV” untuk melihat CV aslinya.

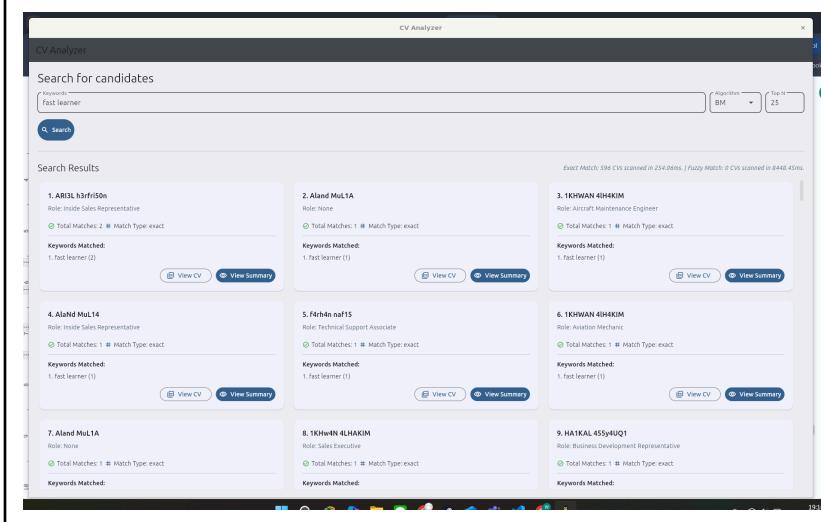
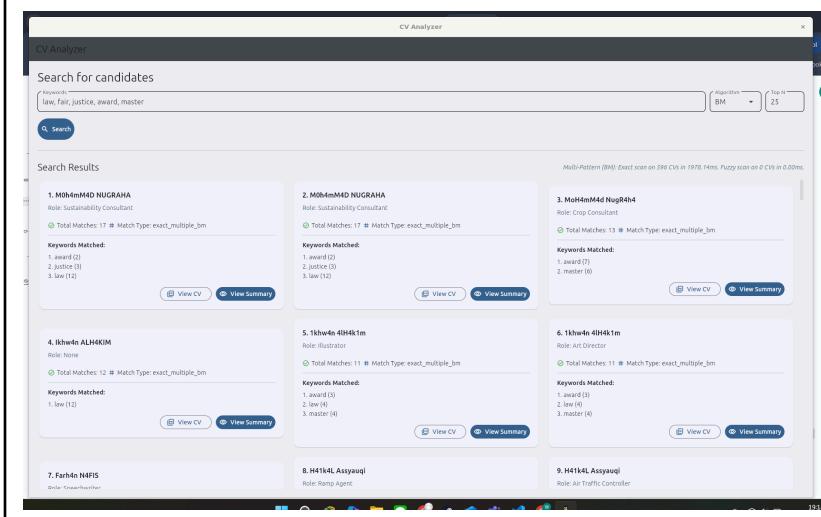
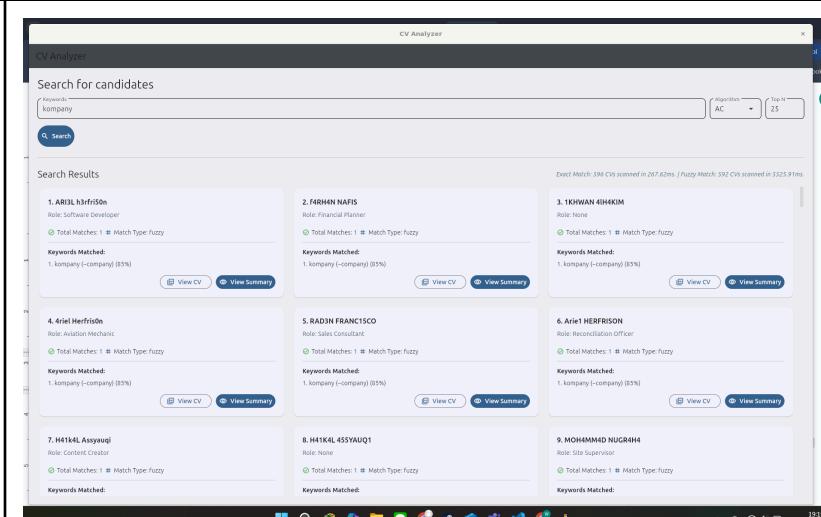


Gambar 4.2.9 View CV Button

4.3. Hasil pengujian

Variabel Kontrol : Setiap pengujian menampilkan top 10 CV teratas.

No	Konfigurasi	Tangkapan Layar
1	Keyword: assembly Algoritma: KMP	
2	Keyword: cook, boil, fry Algoritma: KMP	

3	<p>Keyword: fast learner Algoritma: BM</p>	
4	<p>Keyword: law, fair, justice, award, master Algoritma: BM</p>	
5	<p>Keyword: kompany Algoritma: AC (kasus fuzzy)</p>	

6	<p>Keyword: company, challenging, successful, management, technology Algoritma: AC</p>
---	--

4.4. Analisis hasil pengujian.

Pengujian terhadap aplikasi pencarian CV berbasis algoritma pattern matching menunjukkan bahwa sistem berjalan sesuai dengan spesifikasi fungsional yang diharapkan. Aplikasi mampu mengekstrak teks dari file PDF dengan konsistensi yang baik dan mengidentifikasi keyword secara tepat, baik dalam mode pencarian tunggal maupun ganda. Hasil pencarian ditampilkan secara informatif dengan menyertakan nama pelamar, posisi yang dilamar, jumlah kemunculan keyword, serta ringkasan dari isi CV seperti skill, pendidikan, dan pengalaman kerja.

Dari sisi performa algoritma, Knuth-Morris-Pratt (KMP) memberikan performa yang lumayan stabil dalam pencarian satu keyword, tetapi kalah jauh dengan algoritma lain. Algoritma Boyer-Moore (BM) menunjukkan performa yang sangat baik saat diterapkan pada teks panjang, terutama karena kemampuannya melakukan lompatan besar saat pencocokan gagal. Perbedaan kecepatan KMP dengan BM sangat besar. Sementara itu, Aho Corasick (AC) terbukti paling efisien untuk pencarian banyak kata kunci sekaligus. Hal ini dikarenakan algoritma ini membangun struktur trie dan failure link untuk mengeksekusi pencocokan seluruh keyword hanya dalam satu lintasan teks. Pada skenario pencarian dengan tiga hingga lima kata kunci, Aho Corasick jauh lebih unggul dalam hal kecepatan dibandingkan KMP dan sedikit lebih unggul dari BM.

Dari segi akurasi, seluruh algoritma memberikan hasil pencocokan yang tepat untuk kasus-kasus dengan kata kunci yang sesuai secara literal. Selain algoritma pencarian diatas, aplikasi juga telah menyediakan metode untuk menangani kata-kata yang berbeda tipis dengan mengukur levenshtein distance untuk penggunaan fuzzy search. Hal ini membantu dalam menangani kasus typo atau perbedaan tipis dari teks dan keyword.

BAB V

IMPLEMENTASI BONUS

5.1. Implementasi Algoritma Aho-Corasick

Algoritma Aho-Corasick adalah algoritma pencocokan pola (pattern matching) yang efisien untuk menemukan banyak pola (multi-pattern) secara bersamaan dalam sebuah teks. Algoritma ini sering digunakan dalam aplikasi pencarian teks berskala besar, seperti sistem deteksi spam, pemeriksaan plagiarisme, atau pencocokan keyword dalam dokumen seperti CV. Proses algoritma ini dimulai dengan membangun sebuah struktur data bernama Trie, yang merepresentasikan seluruh daftar pola yang ingin dicari. Setelah itu, algoritma membentuk failure link, yaitu mekanisme yang memungkinkan pencarian berlanjut ke posisi yang tepat tanpa perlu mengulang dari awal ketika terjadi ketidakcocokan karakter. Dengan memanfaatkan struktur ini, algoritma dapat melakukan pencarian dalam satu kali iterasi, terlepas dari jumlah pola yang dimasukkan.

Berikut adalah isi dari Aho_corasick_algorithm.py

a. Class: TrieNode

Kelas TrieNode adalah blok bangunan dasar dari struktur data Trie yang digunakan dalam algoritma Aho-Corasick. Setiap objek TrieNode merepresentasikan sebuah node (simpul) dalam Trie, yang pada dasarnya adalah titik dalam pohon pencarian prefiks.

Atribut	Tipe	Penjelasan
children	dict	Peta ke node anak berdasarkan karakter, membentuk jalur pola.
failure	TrieNode	Tautan kegagalan ke node prefiks terpanjang yang merupakan sufiks, untuk transisi cepat.
output	list	Daftar indeks pola yang berakhir di node ini, menandakan pola yang cocok.

b. Class: AhoCorasickAlgorithm

AhoCorasickAlgorithm mengimplementasikan algoritma Aho-Corasick, pencocok string efisien untuk menemukan banyak pola dalam satu teks.

Atribut	Tipe	Penjelasan
---------	------	------------

root	TrieNode	Node akar dari Trie, titik awal untuk pembangunan dan pencarian.
patterns	list[str]	Daftar pola yang akan dicari.

Kelas ini memiliki metode utama untuk membangun dan menjalankan pencarian:

Metode	Penjelasan
_build_trie(patterns: list[str]) -> None	Membangun Trie dari pola yang diberikan, menandai node akhir pola dengan indeks mereka.
_build_failure_links() -> None	Membangun tautan kegagalan dan menggabungkan output menggunakan BFS, memungkinkan lompatan efisien saat tidak cocok.
_search_patterns(text: str) -> dict[int, int]	Melakukan pencarian pola dalam teks, menghitung kemunculan pola dengan menelusuri Trie dan tautan kegagalan.
count_occurrences(self, text: str, pattern: str) -> int	Mencari dan mengembalikan jumlah kemunculan satu pola dalam teks.
count_multiple_patterns(self, text: str, patterns: list[str]) -> dict[str, int]	Mencari dan mengembalikan jumlah kemunculan beberapa pola dalam teks.

5.2. Video Aplikasi

Link Video tertera di lampiran

BAB VI

PENUTUP

6.1. Kesimpulan

Aplikasi Applicant Tracking System (ATS) ini berhasil dikembangkan sesuai dengan tujuan fungsionalnya, mampu melakukan ekstraksi teks dari ratusan dokumen CV dan melakukan pencarian kata kunci menggunakan tiga algoritma pattern matching: KMP, Boyer-Moore, dan Aho-Corasick. Hasil pengujian performa secara tegas menunjukkan bahwa algoritma Aho-Corasick merupakan yang paling unggul untuk kasus penggunaan utama aplikasi, yaitu pencarian beberapa kata kunci secara simultan. Arsitektur sistem yang memisahkan frontend, backend, dan proses pre-processing di latar belakang juga terbukti efektif dalam memberikan antarmuka yang responsif kepada pengguna.

6.2. Saran

Meskipun aplikasi ini telah berhasil mengimplementasikan fungsionalitas inti dengan baik, terdapat beberapa area pengembangan lanjutan yang dapat dieksplorasi untuk menjadikannya sebuah produk yang siap produksi. Pertama, dari sisi infrastruktur, aplikasi dapat di-containerize menggunakan Docker dan di-deploy dengan WSGI server seperti Gunicorn untuk skalabilitas dan keandalan yang lebih baik. Selain itu, dari sisi fitur, penambahan sistem autentikasi pengguna dengan level akses berbeda (misalnya, Admin dan Recruiter) serta fitur untuk menyimpan riwayat pencarian atau kandidat favorit akan secara signifikan meningkatkan nilai guna aplikasi dalam lingkungan rekrutmen yang sesungguhnya.

6.3. Refleksi

Proses pengembangan proyek ini memberikan pelajaran berharga mengenai pentingnya memilih algoritma yang tepat untuk masalah yang spesifik. Awalnya, kami mempertimbangkan pemrosesan PDF secara on-demand, namun analisis dan pengujian membuktikan bahwa pendekatan pre-processing yang memuat teks ke dalam memori memberikan user experience yang jauh lebih superior dengan biaya memori yang dapat diterima. Tantangan utama terletak pada orkestrasi komponen yang berjalan secara terpisah—GUI, API server, dan background thread—serta memastikan komunikasi antar-proses berjalan lancar dan bebas dari race condition. Proyek ini mengukuhkan

pemahaman kami bahwa teori algoritma dan struktur data memiliki aplikasi praktis yang sangat krusial dalam membangun perangkat lunak yang efisien dan andal.

LAMPIRAN

Link Github Repository: https://github.com/TKurr/Tubes3_farrelcarry

Link Youtube: <https://youtu.be/yRzUhPbB0S4?si=2GrlhKcjYm-3SkT4>

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

DAFTAR PUSTAKA

[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)

<https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/32714.pdf>