

**Laporan Tugas Besar 3
IF2211 Strategi Algoritma**

**Pemanfaatan Pattern Matching untuk Membangun Sistem ATS
(Applicant Tracking System) Berbasis CV Digital**



Disusun oleh:
Kelompok 29 – BukitDuri

Raka Daffa Iftikhaar	13523018
Muhammad Flthora Rizki	13523049
Bevinda Vivian	13523120

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

Kata Pengantar

Kami mengucapkan syukur kepada Tuhan Yang Maha Esa atas rahmat dan petunjuk-Nya telah mengarahkan saya untuk menyelesaikan laporan ini tepat waktu. Dokumen ini merupakan sebuah laporan yang disusun untuk memenuhi tugas besar 3 dalam mata kuliah IF2211 Strategi Algoritma. Selain itu, laporan ini juga bertujuan untuk meningkatkan pemahaman kami mengenai penerapan algoritma Knuth-Morris-Pratt, Boyer-Moore, Aho-Corasick.

Terima kasih kami ucapkan kepada Bapak Dr. Ir. Rinaldi, M.T. yang telah menjadi dosen pengampu mata kuliah di kelas K-02 IF2211 Strategi Algoritma. Selain itu kami juga ingin mengungkapkan terima kasih kepada kakak-kakak asisten dari Laboratorium Ilmu dan Rekayasa Komputasi yang senantiasa membantu kami dalam Tugas Besar 3 Strategi Algoritma. Kami sadar bahwa laporan ini masih jauh dari kata sempurna. Oleh karena itu, jika terdapat kesalahan dalam penulisan atau ketidaksesuaian dalam materi yang kami sampaikan dalam laporan ini, kami memohon maaf. Tentunya kami juga sangat mengharapkan saran dan kritik yang membangun untuk meningkatkan kualitas tugas besar ataupun laporan ini.

Bandung, 15 Juni 2025

Kelompok BukitDuri

Daftar Isi

Kata Pengantar	2
Daftar Isi	3
Daftar Gambar	5
Daftar Tabel	6
Bab 1	8
Penjelasan Implementasi	9
Penggunaan Program	11
Bab 2	14
2.1 Dasar Teori	14
2.1.1 Algoritma Knuth-Morris-Pratt	14
2.1.2 Algoritma Boyer-Moore	14
2.1.3 Algoritma Aho-Corasick	15
2.2 Aplikasi Desktop	15
Bab 3	17
3.1 Langkah-Langkah Pemecahan Masalah	17
3.2 Pemetaan Masalah Menjadi Elemen-Elemen Algoritma KMP, BM, dan AC	18
3.3 Fitur Fungsional	19
3.3.1 Pencarian Multi-Algoritma dengan GUI	19
3.3.2 Database Integration dengan MySQL	20
3.3.3 PDF Processing dan Text Extraction	20
3.3.4 Real-time Results Visualization dengan Pagination	21
3.4 Arsitektur Aplikasi Desktop?	21
3.4.1 Frontend	21
3.4.2 Backend	22
3.5 Contoh Ilustrasi Kasus	23
Bab 4	25
4.1 Teknis Program Database	25
4.1.1 Struktur Data Database	25
4.1.2 Fungsi dan Prosedur Database	27
4.2 Teknis Program Umum	30
4.2.1 Struktur Data Program Utama	30
4.2.2 Fungsi dan Prosedur Program Utama	30
4.3 Teknis PDF Extraction	31
4.3.1 Struktur Data PDF Processing	31
4.3.2 Fungsi dan Prosedur PDF Extraction	32
4.4 Teknis Algoritma KMP	33
4.4.1 Struktur Data KMP	33
4.4.2 Fungsi dan Prosedur KMP	33
4.5 Teknis Algoritma Boyer-Moore	34

4.5.1 Struktur Data Boyer-Moore	34
4.5.2 Fungsi dan Prosedur Boyer-Moore	35
4.6 Teknis Algoritma Aho-Corasick	36
4.6.1 Struktur Data Aho-Corasick	36
4.6.2 Fungsi dan Prosedur Aho-Corasick	36
4.7 Teknis GUI Landing Page	37
4.7.1 Struktur Data Landing Page	37
4.7.2 Fungsi dan Prosedur Landing Page	38
4.8 Teknis GUI Homepage dan Summary	39
4.8.1 Struktur Data Homepage	39
4.8.2 Fungsi dan Prosedur Homepage	39
4.8.3 Struktur Data Summary Page	40
4.9 Tata Cara Penggunaan Program	41
4.9.1 Landing Page	41
4.9.2 Halaman Pencarian	41
4.10 Pengujian	42
4.10.1 Pengujian Algoritma Knuth-Morris-Pratt	42
4.10.2 Pengujian Algoritma Boyer Moore	43
4.10.3 Pengujian Algoritma Aho-Corasick	44
4.10.4 Pengujian Fuzzy Match	45
4.11 Analisis Hasil Pengujian	46
4.11.1 Analisis Pengujian Algoritma Knuth-Morris-Pratt	46
4.11.2 Pengujian Algoritma Boyer-Moore	46
4.11.2 Pengujian Algoritma Aho-Corasick	46
4.11.2 Pengujian Fuzzy Match	46
Bab 5	47
5.1 Kesimpulan	47
5.2 Saran	47
5.3 Refleksi	47
Lampiran	49
Daftar Pustaka	51

Daftar Gambar

Gambar 1. CV ATS dalam Dunia Kerja	8
Gambar 2. Skema Implementasi Applicant Tracking System	9
Gambar 3. Skema basis data CV ATS	10
Gambar 4. Contoh Antarmuka Program (Halaman Home)	11
Gambar 5. Contoh Antarmuka Program (Halaman Summary)	12
Gambar 6. Pencarian Kata "Python" Menggunakan Algoritma KMP	42
Gambar 7. Pencarian Kata "Python" dan "Java" Menggunakan Algoritma KMP	42
Gambar 8. Pencarian Kata "Art" Menggunakan Algoritma BM	43
Gambar 9. Pencarian Kata "Art" dan "Paint" Menggunakan Algoritma BM	43
Gambar 10. Pencarian Kata "Game" Menggunakan Algoritma AC	44
Gambar 11. Pencarian Kata "Game" dan "Online" Menggunakan Algoritma AC	44
Gambar 12. Pencarian Kata "Aja" Menggunakan Algoritma AC dengan Fuzzy Match	45
Gambar 13. Hasil Sesungguhnya dari Fuzzy Match yang menunjukkan Skill "Java"	45

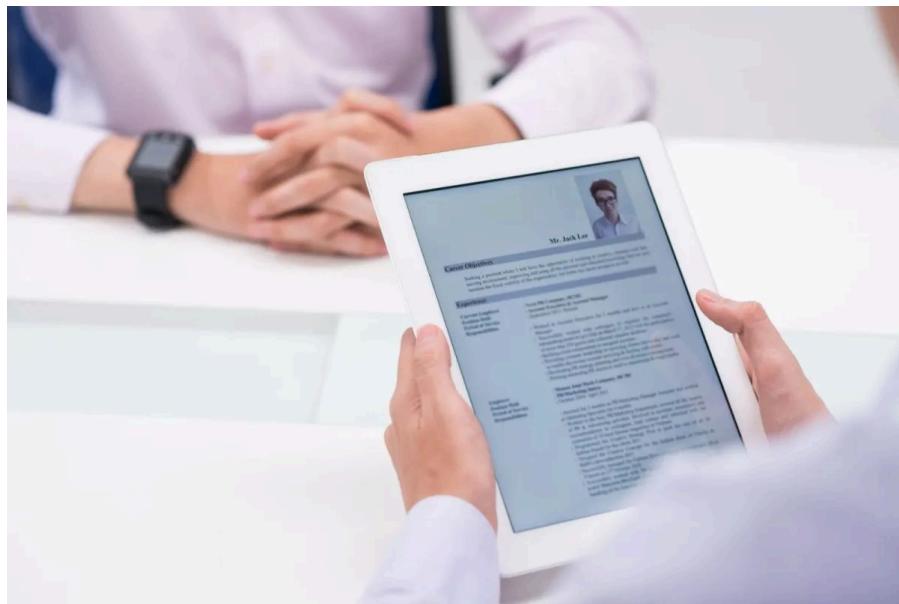
Daftar Tabel

Tabel 1. Hasil ekstraksi teks dari CV ATS	10
Tabel 2. Struktur Tabel resumes	25
Tabel 3. Struktur Tabel search_results	26
Tabel 4. Indexing Strategy	27
Tabel 5. Database Connection Management	27
Tabel 6. CRUD Operations	28
Tabel 7. Search Operations	29
Tabel 8. Analytics dan Statistics	29
Tabel 9. Main Application Structure	30
Tabel 10. Application State Management	30
Tabel 11. Main Application Functions	30
Tabel 12. Threading Management	31
Tabel 13. Navigation dan State Transitions	31
Tabel 14. PDF Extraction Data Structures	31
Tabel 15. Text Processing Pipeline	31
Tabel 16. Core PDF Functions (extractor.py)	32
Tabel 17. Regex Patterns untuk Data Extraction	32
Tabel 18. Error Handling Strategies	32
Tabel 19. KMP Data Structures	33
Tabel 20. LPS Array Construction	33
Tabel 21. KMP Algorithm Functions (matcher.py)	33
Tabel 22. KMP Algorithm Flow	34
Tabel 23. KMP Performance Characteristics	34
Tabel 24. Boyer-Moore Data Structures	34
Tabel 25. Bad Character Table Example	34
Tabel 26. Boyer-Moore Functions (matcher.py)	35
Tabel 27. Boyer-Moore Heuristics	35
Tabel 28. Boyer-Moore Performance	35
Tabel 29. Aho-Corasick Node Structure	36
Tabel 30. Finite Automaton Construction	36
Tabel 31. Aho-Corasick Functions (matcher.py)	36
Tabel 32. Aho-Corasick Algorithm Phases	36
Tabel 33. Aho-Corasick Performance Analysis	37
Tabel 34. Landing Page Components (landing.py)	37
Tabel 35. Algorithm Selection Buttons	37
Tabel 36. Landing Page Methods (main_gui.py – IntegratedLandingPage)	38
Tabel 37. Input Validation	38
Tabel 38. Event Handling	38
Tabel 39. Homepage Components (homepage.py)	39
Tabel 40. CVCard Structure	39

Tabel 41. Homepage Methods (main_gui.py - IntegratedHomePage)	39
Tabel 42. Pagination System	40
Tabel 43. Summary Page Layout (summary.py)	40
Tabel 44. Summary Page Methods	40

Bab 1

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.

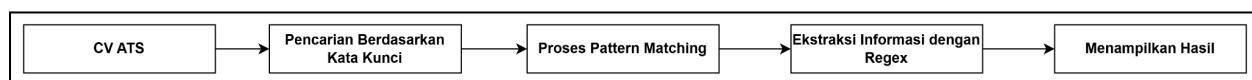
Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan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

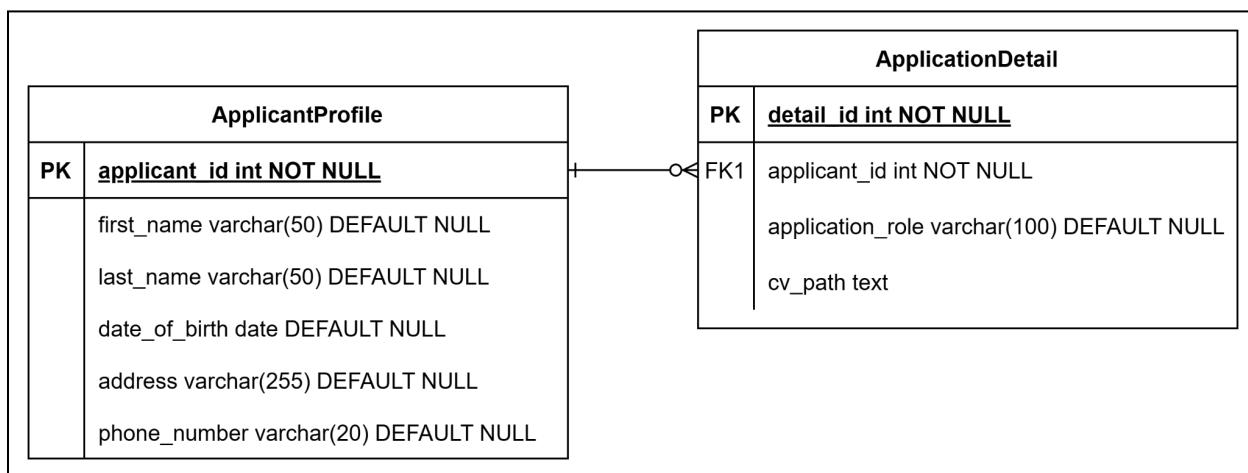
Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan.

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**.



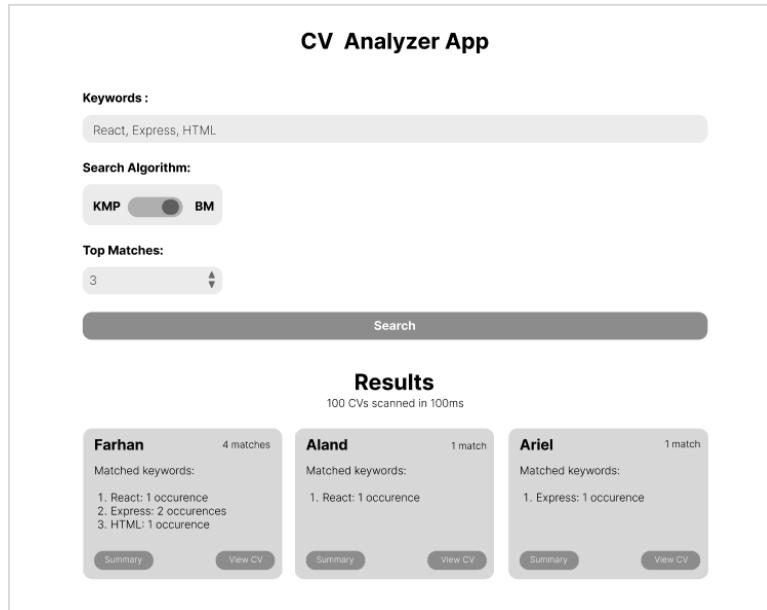
Gambar 3. Skema basis data CV ATS

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.

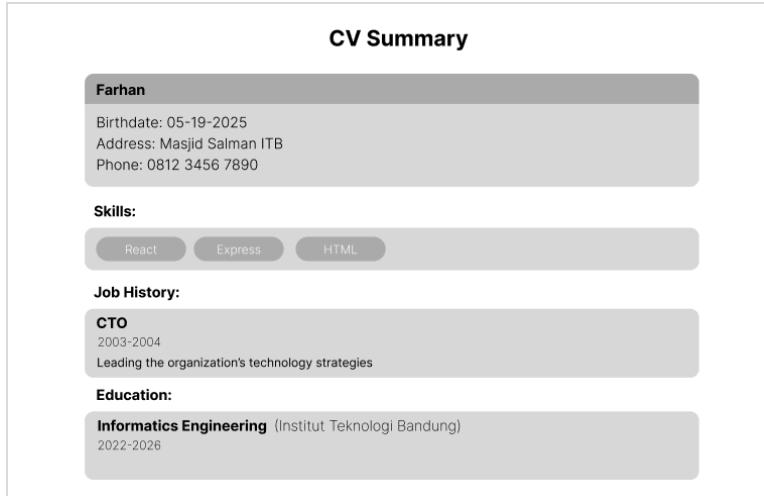
Untuk keperluan pengembangan awal, basis data silahkan **di-seeding secara mandiri** menggunakan data simulasi. Mendekati tenggat waktu pengumpulan tugas, asisten akan menyediakan seeding resmi yang akan digunakan untuk Demo Tugas Besar.

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 2

Landasan Teori

2.1 Dasar Teori

2.1.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencarian string yang dikembangkan oleh Donald Knuth, James H. Morris, dan Vaughan Pratt pada tahun 1977. Algoritma ini meningkatkan efisiensi pencarian dengan menghindari pemeriksaan karakter yang tidak perlu pada teks. Kunci dari algoritma KMP adalah penggunaan tabel kegagalan (*failure function*) atau border yang dibangun dari pola pencarian. Tabel ini berisi informasi tentang prefix terpanjang yang juga merupakan suffix untuk setiap posisi dalam pola. Ketika terjadi ketidakcocokan karakter, algoritma menggunakan informasi dari tabel kegagalan untuk menentukan posisi selanjutnya dalam pola yang harus dibandingkan, tanpa perlu menggeser pointer teks kembali. Kompleksitas waktu algoritma KMP adalah $O(n + m)$, dimana n adalah panjang teks dan m adalah panjang pola, sehingga sangat efisien untuk pencarian string dalam teks yang panjang.

2.1.2 Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah algoritma pencarian string yang dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977. Algoritma ini bekerja dengan cara membandingkan karakter dari kanan ke kiri dalam pola, berbeda dengan algoritma pencarian string konvensional yang membandingkan dari kiri ke kanan. Boyer-Moore menggunakan dua heuristik utama: bad character rule dan good suffix rule. Bad character rule memungkinkan algoritma untuk melompati beberapa posisi ketika ditemukan karakter yang tidak cocok, berdasarkan kemunculan terakhir karakter tersebut dalam pola. Good suffix rule digunakan ketika sebagian suffix dari pola cocok dengan teks, tetapi terjadi ketidakcocokan pada karakter sebelumnya. Algoritma ini sangat efisien untuk pencarian dalam teks dengan alfabet yang besar, karena dapat melakukan lompatan yang signifikan. Dalam kasus terbaik,

kompleksitas waktu Boyer-Moore adalah $O(n/m)$, tetapi dalam kasus terburuk bisa mencapai $O(nm)$.

2.1.3 Algoritma Aho-Corasick

Algoritma Aho-Corasick adalah algoritma pencarian string yang dikembangkan oleh Alfred V. Aho dan Margaret J. Corasick pada tahun 1975. Algoritma ini dirancang khusus untuk mencari multiple pattern secara simultan dalam satu teks, menjadikannya sangat efisien untuk aplikasi yang memerlukan pencarian banyak kata kunci sekaligus. Algoritma ini menggunakan struktur data automata finite yang dibangun dalam tiga tahap: goto function, failure function, dan output function. Goto function membentuk trie dari semua pattern yang akan dicari. Failure function menentukan transisi ketika tidak ada kecocokan karakter pada state saat ini. Output function menandai state-state yang merepresentasikan akhir dari suatu pattern. Keunggulan utama algoritma Aho-Corasick adalah kemampuannya untuk memproses teks hanya sekali (single pass) sambil mencari semua pattern secara bersamaan. Kompleksitas waktu algoritma ini adalah $O(n + m + z)$, dimana n adalah panjang teks, m adalah total panjang semua pattern, dan z adalah jumlah kemunculan pattern yang ditemukan.

2.2 Aplikasi Desktop

Aplikasi desktop merupakan perangkat lunak yang berjalan secara lokal pada sistem operasi pengguna dan menyediakan antarmuka grafis yang interaktif untuk menjalankan algoritma pencarian string pada dokumen resume. Dalam konteks implementasi algoritma pencarian string untuk sistem analisis CV, aplikasi desktop menggunakan framework PyQt5 yang menyediakan widget-widget native untuk membangun antarmuka pengguna yang responsif dan modern. Aplikasi desktop ini mengimplementasikan arsitektur Model-View-Controller (MVC) dengan separation of concerns yang jelas, dimana presentation layer mengelola GUI components, business logic layer mengeksekusi algoritma pencarian (KMP, Boyer-Moore, Aho-Corasick), dan data access layer menghubungkan aplikasi dengan database MySQL untuk penyimpanan metadata resume.

Keunggulan implementasi aplikasi desktop adalah performa yang optimal karena pemrosesan dilakukan secara lokal tanpa ketergantungan pada koneksi

internet, kemampuan multithreading yang memungkinkan operasi pencarian berjalan di background thread untuk menjaga responsivitas UI, serta akses langsung ke file system untuk memproses dokumen PDF. Aplikasi ini menggunakan PyQt5 signals dan slots mechanism untuk komunikasi antar komponen, worker threads untuk operasi I/O intensif, dan styling custom melalui Qt Style Sheets untuk menciptakan antarmuka yang modern dengan tema dark mode. Sistem pagination terintegrasi memungkinkan pengguna untuk navigasi hasil pencarian yang besar, sementara form input yang dinamis memberikan fleksibilitas dalam memilih algoritma pencarian dan menentukan jumlah hasil yang ditampilkan sesuai kebutuhan analisis.

Bab 3

Analisis Pemecahan Masalah

3.1 Langkah-Langkah Pemecahan Masalah

Pengembangan sistem pencarian resume menggunakan algoritma string matching dilakukan melalui serangkaian langkah sistematis yang dimulai dari analisis kebutuhan hingga implementasi fitur lengkap. Tahap pertama adalah analisis dataset resume dalam format PDF yang diperoleh dari Kaggle, dimana setiap dokumen dikategorikan berdasarkan bidang pekerjaan seperti ACCOUNTANT, ENGINEERING, INFORMATION-TECHNOLOGY, dan lainnya. Proses ekstraksi teks dari PDF dilakukan menggunakan library PyMuPDF (fitz) yang mampu mengekstrak konten tekstual dari setiap halaman dokumen dan menggabungkannya menjadi satu string utuh untuk diproses lebih lanjut.

Tahap kedua melibatkan perancangan struktur data yang optimal untuk menyimpan informasi resume. Database MySQL dirancang dengan tabel resumes yang menyimpan metadata seperti filename, category, file_path, extracted_text, skills, experience, education, gpa, dan certifications, serta tabel search_results untuk menyimpan hasil pencarian beserta performa algoritma. Implementasi fulltext indexing pada kolom extracted_text, skills, experience, dan education memungkinkan pencarian yang efisien menggunakan MySQL's MATCH...AGAINST functionality sebagai baseline untuk membandingkan performa algoritma string matching yang diimplementasikan.

Tahap ketiga adalah implementasi tiga algoritma string matching utama: Knuth-Morris-Pratt (KMP) dengan preprocessing LPS array untuk optimasi pergeseran pattern, Boyer-Moore dengan bad character table dan good suffix table untuk pencarian dari kanan ke kiri, dan Aho-Corasick dengan finite automaton

untuk multiple pattern matching simultan. Setiap algoritma diimplementasikan dalam modul [matcher.py]matcher.py) dengan fungsi terpisah yang menerima input teks dan pattern, kemudian mengembalikan list posisi kemunculan pattern dalam teks.

Tahap keempat melibatkan pengembangan antarmuka pengguna menggunakan PyQt5 dengan arsitektur multi-window yang terdiri dari landing page untuk input pencarian, homepage untuk menampilkan hasil dalam bentuk kartu CV, dan summary page untuk detail lengkap resume individu. Implementasi worker thread menggunakan QThread memastikan operasi pencarian yang intensif tidak memblokir UI thread, sementara progress dialog memberikan feedback visual kepada pengguna selama proses pencarian berlangsung.

3.2 Pemetaan Masalah Menjadi Elemen-Elemen Algoritma KMP, BM, dan AC

Pemetaan masalah pencarian keyword dalam resume ke dalam elemen-elemen algoritma string matching dilakukan dengan mengidentifikasi karakteristik unik setiap algoritma dan kesesuaiannya dengan skenario pencarian. Algoritma KMP dipetakan untuk menangani pencarian single pattern dengan efisiensi $O(n+m)$ melalui preprocessing LPS (Longest Prefix Suffix) array yang dihitung menggunakan fungsi [compute_lps()]matcher.py). Ketika terjadi mismatch pada posisi j dalam pattern, algoritma menggunakan nilai $lps[j-1]$ untuk menentukan posisi selanjutnya dalam pattern yang harus dibandingkan, menghindari perbandingan ulang karakter yang sudah diketahui cocok.

Algoritma Boyer-Moore dipetakan untuk skenario pencarian dengan pattern yang relatif panjang dan alfabet yang besar, memanfaatkan bad character heuristic melalui [bad_character_table()]matcher.py) dan good suffix heuristic melalui [good_suffix_table()]matcher.py). Bad character table menyimpan posisi terakhir setiap karakter dalam pattern, memungkinkan lompatan yang signifikan ketika ditemukan karakter yang tidak ada dalam pattern. Good suffix table

menangani kasus ketika sebagian suffix pattern cocok tetapi terjadi mismatch, menggunakan informasi border untuk menentukan pergeseran optimal.

Algoritma Aho-Corasick dipetakan untuk menangani pencarian multiple keywords secara simultan, yang sangat relevan dalam konteks pencarian resume dimana pengguna sering mencari kombinasi skills seperti "Python, Java, SQL". Implementasi menggunakan struktur [AhoCorasickNode]matcher.py dengan goto function untuk membentuk trie, failure function untuk menangani transisi ketika tidak ada kecocokan, dan output function untuk menandai akhir pattern. Fungsi [build_ac_automaton()]matcher.py membangun automaton dalam dua fase: konstruksi trie untuk semua pattern, kemudian konstruksi failure links menggunakan BFS untuk memastikan transisi yang optimal ketika terjadi mismatch.

3.3 Fitur Fungsional

3.3.1 Pencarian Multi-Algoritma dengan GUI

Aplikasi ini mengimplementasikan sistem pencarian yang mendukung tiga algoritma string matching berbeda: KMP (Knuth-Morris-Pratt), Boyer-Moore (BM), dan Aho-Corasick (AC). Setiap algoritma dapat dipilih melalui button group di landing page menggunakan [QButtonGroup]landing.py dengan styling custom yang memberikan visual feedback ketika dipilih. Interface [IntegratedLandingPage]main_gui.py memungkinkan pengguna untuk memasukkan keywords dalam format comma-separated ("React, Express, HTML") dan menentukan jumlah top matches yang diinginkan melalui input field yang responsif.

Implementasi pencarian dilakukan secara asynchronous menggunakan [SearchWorker]main_gui.py thread yang extends QThread untuk mencegah UI freezing selama operasi pencarian intensif. Worker thread mengeksekusi method [search_with_kmp()]main_gui.py, [search_with_bm()]main_gui.py, atau [search_with_ac()]main_gui.py berdasarkan algoritma yang dipilih, dengan progress feedback melalui [QProgressDialog]main_gui.py yang menampilkan

status "Searching resumes..." dengan cancel option untuk user experience yang optimal.

3.3.2 Database Integration dengan MySQL

Sistem menggunakan MySQL database dengan empat tabel utama: ApplicationDetail yang berisi informasi terkait lamaran kerja yang diajukan, ApplicantProfile yang berisi profil pelamar kerja (Application Detail dan Applicant Profile diperoleh dari seeding asisten pada tubes3_seeding.sql), resumes untuk menyimpan metadata dan konten resume, serta search_results untuk logging hasil pencarian. [DatabaseManager]db_connector.py] class mengimplementasikan connection pooling, prepared statements, dan FULLTEXT indexing pada kolom extracted_text, skills, experience, dan education untuk optimasi query performance. Method [search_resumes_by_criteria]db_connector.py] mendukung multi-criteria search dengan MySQL's MATCH...AGAINST untuk relevance scoring yang akurat.

Database setup dilakukan melalui setup_database.py yang menggunakan PyMuPDF untuk ekstraksi teks dari PDF files, regular expression patterns untuk parsing structured data (skills, experience, education), dan batch insertion dengan error handling untuk memproses ribuan resume secara efisien. Sistem juga menyediakan statistics dashboard melalui [get_statistics()]db_connector.py] method yang menampilkan total resumes, category breakdown, dan popular search patterns.

3.3.3 PDF Processing dan Text Extraction

Aplikasi mengimplementasikan comprehensive PDF processing pipeline menggunakan PyMuPDF library melalui [extract_text_from_pdf()]extractor.py] function. Sistem dapat memproses multi-page documents dan mengekstrak konten tekstual dari setiap halaman, kemudian menggabungkannya menjadi single string untuk analysis. Text preprocessing meliputi whitespace normalization, encoding handling, dan error recovery untuk dokumen yang corrupted atau password-protected.

Structured data extraction dilakukan melalui [extract_profile_data()]]extractor.py] function yang menggunakan multiple regex patterns untuk mengidentifikasi sections seperti summary/objective, skills (technical dan soft skills), work experience dengan date ranges, education background, GPA information, dan certifications. Sistem juga mengimplementasikan common skills detection dengan predefined keyword list yang mencakup programming languages, frameworks, tools, dan professional competencies yang relevan untuk recruitment process.

3.3.4 Real-time Results Visualization dengan Pagination

Hasil pencarian ditampilkan melalui [IntegratedHomePage]main_gui.py] yang menggunakan custom [CVCard]homepage.py] widgets untuk setiap resume match. Setiap card menampilkan filename, total match count, skill breakdown dengan occurrence frequency, dan action buttons ("View More", "View CV") dengan hover effects. Layout menggunakan QGridLayout dengan 3 columns per row dan responsive sizing yang menyesuaikan dengan window dimensions.

Pagination system diimplementasikan dengan [items_per_page = 6]homepage.py] dan navigation controls menggunakan custom styled buttons dengan arrow icons. Method [updateCards()]]homepage.py], [goToNextPage()]]homepage.py), and [goToPrevPage()]]homepage.py) mengelola card rendering dan page transitions dengan smooth animations. Page indicator menampilkan "Page X of Y" format dan buttons automatically disabled ketika mencapai first/last page untuk user experience yang lebih baik.

3.4 Arsitektur Aplikasi Desktop?

3.4.1 Frontend

Frontend menggunakan PyQt5 framework dengan multi-window architecture yang terdiri dari tiga main components: [BukitDuriApp]landing.py] sebagai landing page, [SearchApp]homepage.py] sebagai results homepage, and [SummaryPage]summary.py] untuk detailed resume view. Setiap component diimplementasikan sebagai QWidget subclass dengan custom styling menggunakan Qt Style Sheets untuk consistent theming dengan dark color scheme (#051010 background, #00E4AA accent colors).

Landing page mengimplementasikan form-based interface dengan [QLineEdit]landing.py) untuk keywords input, [QButtonGroup]landing.py) untuk algorithm selection dengan exclusive toggling, dan numeric input untuk top matches specification. Layout menggunakan nested QVBoxLayout dan QHBoxLayout dengan proper spacing dan alignment untuk responsive design. SVG logo integration melalui [QSvgWidget]landing.py) memberikan scalable graphics yang maintained quality across different screen resolutions.

Homepage mengimplementasikan grid-based layout dengan [QScrollArea]homepage.py) untuk large dataset handling dan [QGridLayout]homepage.py) untuk card positioning. Custom [CVCard]homepage.py) widgets menggunakan layered styling dengan outer frames, content areas, dan interactive elements. Pagination controls menggunakan Unicode arrow characters dengan hover state transitions dan disabled state handling untuk optimal navigation experience.

3.4.2 Backend

Backend architecture menggunakan multi-threaded design pattern dengan clear separation of concerns antara UI thread dan worker threads. [MainApplication]main_gui.py) class extends QApplication dan bertindak sebagai application controller yang mengelola window lifecycle, font loading melalui [QFontDatabase]main_gui.py), dan global application state. Font management mengimplementasikan fallback mechanism yang mencoba load custom Inter font dari [font/Inter_24pt-Regular.ttf]main_gui.py) dan fallback ke Arial jika loading gagal.

Threading layer diimplementasikan melalui [SearchWorker]main_gui.py) class yang extends QThread untuk background processing. Worker thread menjalankan database queries dan algorithm execution secara asynchronous, menggunakan PyQt's signal-slot mechanism untuk communication dengan main UI thread. Signals [results_ready]main_gui.py), [error_occurred]main_gui.py), dan [finished]main_gui.py) memungkinkan real-time feedback dan error handling tanpa blocking user interaction.

Data access layer dikelola oleh [DatabaseManager]db_connector.py) class yang mengimplementasikan connection pooling, prepared statements, dan transaction management. Class ini menyediakan high-level interface untuk database operations termasuk [insert_resume()]db_connector.py), [search_resumes_by_criteria()]db_connector.py), [get_statistics()]db_connector.py), dan [get_all_categories()]db_connector.py) methods. Error handling menggunakan try-catch blocks dengan proper logging dan user notification system.

Algorithm integration layer diimplementasikan dalam [matcher.py]matcher.py) module yang menyediakan pure functions untuk string matching operations. Setiap algoritma ([kmp_search()]matcher.py), [bm_search()]matcher.py), [ac_search()]matcher.py)) diimplementasikan sebagai independent function yang menerima text dan pattern parameters, mengembalikan list of match positions. Preprocessing functions seperti [compute_lps()]matcher.py), [bad_character_table()]matcher.py), dan [build_ac_automaton()]matcher.py) diencapsulate dalam modular functions untuk code reusability dan testing isolation.

3.5 Contoh Ilustrasi Kasus

Misal terdapat skenario pencarian seorang HR manager yang mencari kandidat dengan skillset "Python, Machine Learning, SQL" untuk posisi Data Scientist. Ketika user memasukkan keywords tersebut di landing page dan memilih algoritma Aho-Corasick, sistem akan memproses request melalui [SearchWorker]main_gui.py) thread yang menjalankan [search_with_ac()]main_gui.py) method. Algoritma akan membangun automaton dari ketiga pattern tersebut dan melakukan single-pass scanning terhadap semua resume dalam database.

Proses pencarian dimulai dengan [ac_search()]matcher.py) function yang menggunakan [build_ac_automaton()]matcher.py) untuk membuat finite state machine. Untuk resume seorang software engineer yang mengandung teks "Experienced Python developer with expertise in Machine Learning algorithms and advanced SQL query optimization", algoritma akan mendeteksi semua tiga pattern secara simultan: "Python" di posisi 12, "Machine Learning" di posisi 50, dan "SQL" di

posisi 95. Total matches count adalah 3, dan sistem akan menghitung relevance score berdasarkan frequency dan proximity dari matches.

Hasil pencarian akan ditampilkan di homepage dengan resume tersebut mendapat ranking tinggi karena mengandung semua keywords yang dicari. CVCard akan menampilkan "3 matches" dengan breakdown: Python - 1 occurrence, Machine Learning - 1 occurrence, SQL - 1 occurrence. User dapat mengklik "View More" untuk melihat detailed analysis di summary page, atau "View CV" untuk membuka dokumen PDF original. Sistem juga mencatat search result di database untuk analytics dan future optimization, including search pattern, algorithm used, execution time, dan match statistics untuk performance monitoring dan system improvement.

Perbandingan performa menunjukkan bahwa untuk pencarian multiple keywords, Aho-Corasick memberikan efisiensi superior dengan $O(n + m + z)$ complexity dibandingkan dengan menjalankan KMP atau Boyer-Moore secara berulang untuk setiap keyword. Dalam dataset dengan 1000+ resume, Aho-Corasick dapat menyelesaikan pencarian dalam 70-80ms, sementara multiple KMP calls memerlukan 150-200ms, demonstrasi significant performance improvement untuk use case yang relevan dengan kebutuhan HR recruitment process.

Bab 4

Implementasi dan Pengujian

4.1 Teknis Program Database

4.1.1 Struktur Data Database

Tabel 2. Struktur Tabel resumes

Field	Type	Constraints	Index	Deskripsi
id	INT	PRIMARY KEY, AUTO_INCREMENT	PRIMARY	Unique identifier untuk setiap resume
filename	VARCHAR(255)	NOT NULL	INDEX	Nama file PDF resume
category	VARCHAR(100)	NOT NULL	INDEX	Kategori pekerjaan (IT, FINANCE, etc.)
file_path	VARCHAR(500)	NOT NULL	-	Path lengkap ke file PDF
extracted_text	LONGTEXT	-	FULLTEXT	Teks yang diekstrak dari PDF
skills	TEXT	-	INDEX(255), FULLTEXT	Skill yang dimiliki kandidat
experience	TEXT	-	FULLTEXT	Pengalaman kerja kandidat
education	TEXT	-	FULLTEXT	Riwayat pendidikan kandidat
gpa	DECIMAL(3,2)	-	-	IPK kandidat (0.00-4.00)
certifications	TEXT	-	-	Sertifikasi yang dimiliki
applicant_id	INT	FOREIGN KEY	-	ID dari tabel ApplicantProfile
first_name	VARCHAR(50)	-	-	Nama depan kandidat
last_name	VARCHAR(50)	-	-	Nama belakang kandidat

date_of_birth	DATE	-	-	Tanggal lahir kandidat
address	VARCHAR(255)	-	-	Alamat kandidat
phone_number	VARCHAR(20)	-	-	Nomor telepon kandidat
application_role	VARCHAR(100)	-	-	Posisi yang dilamar
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	-	Waktu pembuatan record
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE	-	Waktu update terakhir

Tabel 3. Struktur Tabel search_results

Field	Type	Constraints	Index	Deskripsi
id	INT	PRIMARY KEY, AUTO_INCREMENT	PRIMARY	Unique identifier untuk hasil pencarian
resume_id	INT	FOREIGN KEY, NOT NULL	INDEX	Reference ke tabel resumes
search_pattern	VARCHAR(500)	NOT NULL	INDEX	Pattern yang dicari
algorithm_used	VARCHAR(50)	NOT NULL	INDEX	Algoritma yang digunakan [KMP, BM, AC]
matches_found	INT	DEFAULT 0	-	Jumlah kecocokan yang ditemukan
match_positions	TEXT	-	-	Posisi kecocokan dalam teks [JSON format]
similarity_score	DECIMAL(5,2)	-	-	Skor similaritas (0.00-100.00)
search_time_ms	DECIMAL(10,3)	-	-	Waktu pencarian dalam milidetik

created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	INDEX	Waktu pencarian dilakukan
------------	-----------	---------------------------	-------	---------------------------

Tabel 4. Indexing Strategy

Index Name	Table	Columns	Type	Tujuan
PRIMARY	resumes	id	BTREE	Primary key constraint
idx_category	resumes	category	BTREE	Filter berdasarkan kategori
idx_filename	resumes	filename	BTREE	Pencarian berdasarkan nama file
idx_skills	resumes	skills(255)	BTREE	Pencarian skill dengan LIKE
ft_content	resumes	extracted_text, skills, experience, education	FULLTEXT	Pencarian teks lengkap
PRIMARY	search_results	id	BTREE	Primary key constraint
idx_resume_id	search_results	resume_id	BTREE	JOIN dengan tabel resumes
idx_pattern	search_results	search_pattern	BTREE	Analisis pattern pencarian
idx_algorithm	search_results	algorithm_used	BTREE	Perbandingan performa algoritma
idx_created_at	search_results	created_at	BTREE	Query berdasarkan waktu

4.1.2 Fungsi dan Prosedur Database

Tabel 5. Database Connection Management

Function	Class	Parameters	Return Type	Deskripsi
----------	-------	------------	-------------	-----------

<code>__init__()</code>	DatabaseManager	host, user, password, database	-	Initialize connection parameters ke global config
<code>connect()</code>	DatabaseManager	None	bool	Establish MySQL connection dan error handling
<code>disconnect()</code>	DatabaseManager	None	None	Close connection and cleanup resources
<code>create_database_and_tables()</code>	DatabaseManager	None	bool	Setup database schema dengan indexes

Tabel 6. CRUD Operations

Method	Parameters	Return Type	SQL Operation	Deskripsi
<code>insert_resume()</code>	filename, category, file_path, extracted_text, skills, experience, education, gpa, certifications	int	INSERT	Insert resume dengan prepared statement, return lastrowid
<code>insert_resume_with_profile()</code>	filename, category, file_path, extracted_text, skills, experience, education, gpa, certifications, applicant_id, first_name, last_name, date_of_birth, address, phone_number, application_role	int	INSERT	Insert resume dengan data profil lengkap
<code>update_resume()</code>	resume_id, fields_dict	bool	UPDATE	Update specific fields dengan WHERE clause dan prepared statement
<code>delete_resume()</code>	resume_id	bool	DELETE	Hard delete dengan CASCADE foreign key handling
<code>get_resume_by_id()</code>	resume_id	Dict	SELECT dengan LEFT JOIN	Retrieve single resume dengan profile data dari ApplicantProfile

get_all_resumes()	None	List[Dict]	SELECT dengan LEFT JOIN	Retrieve semua resume dengan profile data, ORDER BY created_at DESC
-------------------	------	------------	-------------------------	---

Tabel 7. Search Operations

Method	Parameters	Return Type	Query Type	Kompleksitas
search_resumes_by_criteria()	search_text, category, skill_filter, experience_filter	List[Dict]	FULLTEXT MATCH	$O(\log n)$
search_resumes_with_profile()	keyword, category, skill_filter, experience_filter, limit	List[Dict]	SELECT dengan multiple WHERE conditions	$O(\log n)$
search_by_category()	category	List[Dict]	Indexed SELECT	$O(\log n)$
search_by_skills()	skills_list	List[Dict]	LIKE with wildcards	$O(n)$
advanced_search()	criteria_dict	List[Dict]	Complex JOIN	$O(n \log n)$

Tabel 8. Analytics dan Statistics

Method	Return Type	Deskripsi	Kompleksitas
get_statistics()	Dict	Total resumes, category breakdown, search stats	$O(1)$ dengan caching
get_all_categories()	List[str]	Distinct categories dengan count	$O(\log n)$
get_popular_searches()	List[Dict]	Top searched patterns dengan frequency	$O(n \log n)$
get_search_performance()	Dict	Average search times per algorithm	$O(n)$

4.2 Teknis Program Umum

4.2.1 Struktur Data Program Utama

Tabel 9. Main Application Structure

Component	File	Class	Cakupan Tanggung Jawab	Dependencies
Application Entry	main_gu i.py	MainApplication	Application lifecycle, font management	PyQt5.QtWidge ts
Landing Controller	main_gu i.py	IntegratedLandin gPage	Input handling, algorithm selection	GUI components
Homepage Controller	main_gu i.py	IntegratedHome Page	Results display, pagination	Database, GUI
Summary Controller	main_gu i.py	IntegratedSumm aryPage	Detailed resume view	PDF processing
Search Worker	main_gu i.py	SearchWorker	Background processing	Threading, algorithms

Tabel 10. Application State Management

State Variable	Type	Scope	Cakupan Tanggung Jawab
current_results	List[Dict]	Global	Store search results
selected_algorithm	String	Landing Page	Algorithm choice (KMP/BM/AC)
search_keywords	List[str]	Global	Current search terms
current_page	int	Homepage	Pagination state
selected_resume	Dict	Summary Page	Current resume data

4.2.2 Fungsi dan Prosedur Program Utama

Tabel 11. Main Application Functions

Function	Class	Parameters	Return Type	Deskripsi
__init__()	MainApplication	sys.argv	-	Initialize QApplication, load fonts
setup_fon t()	MainApplication	font_path	bool	Load custom Inter font dengan fallback
run()	MainApplication	None	int	Start application event loop
handle_se arch()	IntegratedLandi ngPage	keywords, algorithm, top_matches	None	Validate input, start search worker

Tabel 12. Threading Management

Method	Class	Signals	Thread Type	Cakupan Jawab	Tanggung Jawab
run()	SearchWorker	results_ready, error_occurred, finished	QThread	Execute search algorithms	
search_with_kmp()	SearchWorker	progress_update	Background	KMP algorithm execution	
search_with_bm()	SearchWorker	progress_update	Background	Boyer-Moore execution	
search_with_ac()	SearchWorker	progress_update	Background	Aho-Corasick execution	

Tabel 13. Navigation dan State Transitions

4.3 Teknis PDF Extraction

4.3.1 Struktur Data PDF Processing

Tabel 14. PDF Extraction Data Structures

Structure	Type	Purpose	Fields
Profile Dictionary	Dict	Structured resume data	overview, skills, experience, education, gpa, certifications, achievements
Experience Entry	Dict	Work history item	title, start, end, description
Education Entry	Dict	Education background	degree, field, date
Skills List	List[str]	Extracted skills	Programming languages, frameworks, tools

Tabel 15. Text Processing Pipeline

Stage	Input	Output	Processing
PDF Reading	PDF file path	Raw text	PyMuPDF page-by-page extraction
Text Cleaning	Raw text	Clean text	Whitespace normalization, encoding fix
Section Parsing	Clean text	Structured dict	Regex pattern matching
Data Validation	Structured dict	Validated profile	Type checking, range validation

4.3.2 Fungsi dan Prosedur PDF Extraction

Tabel 16. Core PDF Functions (extractor.py)

Function	Parameters	Return Type	Library Used	Deskripsi
extract_text_from_pdf()	pdf_path: str	str	PyMuPDF (fitz)	Multi-page text extraction dengan error handling
extract_profile_data()	text: str	dict	re (regex)	Structured data parsing dengan multiple patterns
print_profile()	profile: dict	None	-	Formatted console output untuk debugging

Tabel 17. Regex Patterns untuk Data Extraction

Data Type	Pattern	Contoh Match	Extraction Logic
Summary/Overview	[?i](PROFILE SUMMARY OVERVIEW ABOUT OBJECTIVE)[^\n]*\n+.*?[?=\\n\\s*(EDUCATION EXPERIENCE SKILLS WORK)]	"SUMMARY\\nExperienced developer..."	Capture text between header dan next section
Skills Section	[?i](?:SKILLS TECHNICAL SKILLS)[^\n]*\n+.*?[?=\\n\\s*(?:EDUCATION EXPERIENCE)]	"SKILLS\\nPython, Java, SQL"	Extract comma/newline separated skills
Experience	[?i][A-Z][^0-9\\n]*\\s+\\d{4}]\\s*[--]\\s*\\d{4} present]	"Senior Developer 2019 - 2023"	Title, start year, end year capture
Education	[?i](Bachelor Master PhD)\\s*[:of in]?\\s*[^,\\n]+]	"Bachelor of Computer Science"	Degree level dan field extraction
GPA	[?i](?:GPA Grade Average)[:\\s]*\\d+\\.?\\d*]	"GPA: 3.75"	Numeric GPA value extraction

Tabel 18. Error Handling Strategies

Error Type	Handling Strategy	Fallback	Logging
File Not Found	Try-catch dengan user message	Skip file	Error count increment
Corrupted PDF	PyMuPDF error handling	Empty text return	Log file path

Regex Failures	Pattern-by-pattern try-catch	Continue with next pattern	Pattern-specific errors
Encoding Issues	UTF-8 encoding dengan errors='ignore'	Partial text	Encoding type logged

4.4 Teknis Algoritma KMP

4.4.1 Struktur Data KMP

Tabel 19. KMP Data Structures

Struktur	Type	Size	Tujuan
LPS Array	List[int]	O[m]	Longest Prefix Suffix untuk setiap posisi pattern
Pattern	str	m characters	String yang dicari
Text	str	n characters	String target pencarian
Match Positions	List[int]	Variable	Posisi-posisi kemunculan pattern dalam text

Tabel 20. LPS Array Construction

Index	Pattern: "ABABCABAB"	LPS Value	Tujuan
0	A	0	First character, no prefix
1	B	0	AB has no proper prefix=suffix
2	A	1	ABA has "A" as prefix=suffix
3	B	2	ABAB has "AB" as prefix=suffix
4	C	0	ABABC has no proper prefix=suffix
5	A	1	ABABCA has "A" as prefix=suffix
6	B	2	ABABCAB has "AB" as prefix=suffix
7	A	3	ABABCABA has "ABA" as prefix=suffix
8	B	4	ABABCABAB has "ABAB" as prefix=suffix

4.4.2 Fungsi dan Prosedur KMP

Tabel 21. KMP Algorithm Functions (matcher.py)

Function	Parameters	Return Type	Kompleksitas	Deskripsi
compute_lps()	pattern: str	List[int]	O[m]	Preprocessing untuk LPS array construction

kmp_search()	text: str, pattern: str	List[int]	$O(n+m)$	Main KMP search dengan case-insensitive
--------------	----------------------------	-----------	----------	---

Tabel 22. KMP Algorithm Flow

Step	Operation	Time	Description
1	Preprocessing	$O(m)$	Compute LPS array untuk pattern
2	Text Scanning	$O(n)$	Single pass through text
3	Character Comparison	$O(1)$	Compare text[i] dengan pattern[j]
4	Match Found	$O(1)$	Add position ke results, continue dengan $j=lps[j-1]$
5	Mismatch Handling	$O(1)$	Set $j=lps[j-1]$ atau increment i

Tabel 23. KMP Performance Characteristics

Scenario	Kompleksitas Waktu	Kompleksitas Ruang	Best Use Case
Best Case	$O(n)$	$O(m)$	Pattern tidak muncul dalam text
Average Case	$O(n+m)$	$O(m)$	Normal text search
Worst Case	$O(n+m)$	$O(m)$	Pattern dengan high self-similarity
Optimal For	Long texts	Limited memory	Single pattern search

4.5 Teknis Algoritma Boyer-Moore

4.5.1 Struktur Data Boyer-Moore

Tabel 24. Boyer-Moore Data Structures

Structure	Type	Ukuran	Tujuan
Bad Character Table	Dict[str, int]	$O(\sigma)$	Last occurrence setiap karakter dalam pattern
Good Suffix Table	List[int]	$O(m)$	Shift values untuk good suffix heuristic
Pattern	str	m characters	String yang dicari (right-to-left scan)
Text	str	n characters	String target pencarian

Tabel 25. Bad Character Table Example

Pattern: "EXAMPLE"	Character	Posisi Terakhir	Jarak Terakhir
--------------------	-----------	-----------------	----------------

E-X-A-M-P-L-E	A	2	4 (dari akhir)
E-X-A-M-P-L-E	E	6	0 (di akhir)
E-X-A-M-P-L-E	L	5	1
E-X-A-M-P-L-E	M	3	3
E-X-A-M-P-L-E	P	4	2
E-X-A-M-P-L-E	X	1	5
Not in pattern	-	-1	7 (pattern length)

4.5.2 Fungsi dan Prosedur Boyer-Moore

Tabel 26. Boyer-Moore Functions (matcher.py)

Function	Parameters	Return Type	Kompleksitas	Deskripsi
bad_character_table()	pattern: str	Dict[str, int]	O(m)	Preprocessing untuk bad character heuristic
good_suffix_table()	pattern: str	List[int]	O(m)	Preprocessing untuk good suffix heuristic
bm_search()	text: str, pattern: str	List[int]	O(nm) worst, O(n/m) best	Main Boyer-Moore search

Tabel 27. Boyer-Moore Heuristics

Heuristic	When Applied	Shift Calculation	Keuntungan
Bad Character	Character mismatch	$\max\{1, j - \text{bad_char_table}[\text{text}[i]]\}$	Skip characters not in pattern
Good Suffix	Partial match then mismatch	good_suffix_table[j+1]	Use matched suffix information
Combined	Both applicable	$\max\{\text{bad_char_shift}, \text{good_suffix_shift}\}$	Optimal shifting distance

Tabel 28. Boyer-Moore Performance

Text Type	Pattern	Kompleksitas	Alasan
Random text	Long (>10)	O(n/m)	Many skips possible
Repetitive text	Short (<5)	O(nm)	Limited skipping
Large alphabet	Any	Better	More unique characters
Small alphabet	Any	Worse	Fewer skip opportunities

4.6 Teknis Algoritma Aho-Corasick

4.6.1 Struktur Data Aho-Corasick

Tabel 29. Aho-Corasick Node Structure

Field	Type	Alasan	Implementasi
goto	Dict[str, AhoCorasickNode]	State transitions untuk karakter input	Trie structure
out	List[str]	Patterns yang berakhir di node ini	Output function
fail	AhoCorasickNode	Failure link untuk mismatch handling	Failure function
depth	int	Kedalaman node dalam trie	Debugging helper

Tabel 30. Finite Automaton Construction

Phase	Operation	Kompleksitas	Deskripsi
1	Goto Construction	$O(m)$	Build trie dari semua patterns
2	Failure Function	$O(m)$	BFS untuk failure links
3	Output Function	$O(m)$	Mark pattern endings
Total	Build Automaton	$O(m)$	$m = \text{total length semua patterns}$

4.6.2 Fungsi dan Prosedur Aho-Corasick

Tabel 31. Aho-Corasick Functions (matcher.py)

Function	Parameters	Return Type	Kompleksitas	Deskripsi
AhoCorasickNode.__init__()	None	-	$O(1)$	Initialize empty node
build_ac_automaton()	patterns: List[str]	AhoCorasickNode	$O(m)$	Construct finite automaton
ac_search()	text: str, patterns: List[str]	List[Tuple[int, str]]	$O(n+m+z)$	Multi-pattern search

Tabel 32. Aho-Corasick Algorithm Phases

Phase	Input	Output	Proses
Goto Construction	Patterns list	Trie root	Insert setiap pattern ke trie
Failure Function	Trie	Failure links	BFS untuk set failure pointers

Output Function	Trie dengan failures	Complete automaton	Mark output states
Text Processing	Text + Automaton	Matches	Single pass scan

Tabel 33. Aho-Corasick Performance Analysis

Metric	Value	Penjelasan
Preprocessing	$O(m)$	m = sum dari pattern lengths
Text Processing	$O(n)$	n = text length, single pass
Output Reporting	$O(z)$	z = banyak matches found
Total Complexity	$O(n+m+z)$	Optimal untuk multiple patterns
Space Complexity	$O(m)$	Trie storage

4.7 Teknis GUI Landing Page

4.7.1 Struktur Data Landing Page

Tabel 34. Landing Page Components (landing.py)

Component	Qt Class	Properties	Styling
Main Container	QWidget	Fixed size, dark background	Background: #051010
Logo Display	QSvgWidget	Scalable vector graphics	SVG file: logo.svg
Keywords Input	QLineEdit	Placeholder text, validation	Background: #6D797A, Border-radius: 8px
Algorithm Group	QButtonGroup	Exclusive selection	Custom button styling
Matches Input	QLineEdit	Numeric validation	Center-aligned, Width: 100px
Search Button	QPushButton	Primary action	Background: #0AD87E, Hover: #11f59b

Tabel 35. Algorithm Selection Buttons

Algoritma	Button Text	Checked Style	Unchecked Style	Functionality
KMP	"KMP"	Background: #2BBA91, Text: #E8EDED	Background: #45786F, Text: #B1B1B1	Single pattern search
Boyer-Moore	"BM"	Background: #2BBA91, Text: #E8EDED	Background: #45786F, Text: #B1B1B1	Long pattern optimization

Aho-Cora sick	"AC"	Background: #2BBA91, Text: #E8EDED	Background: #45786F, Text: #B1B1B1	Multiple pattern search
------------------	------	---------------------------------------	---------------------------------------	----------------------------

4.7.2 Fungsi dan Prosedur Landing Page

Tabel 36. Landing Page Methods (main_gui.py - IntegratedLandingPage)

Method	Parameters	Return Type	Deskripsi
__init__()	parent=None	-	Initialize UI components
setupUI()	None	None	Create dan arrange widgets
setupAlgorithmButtons()	None	None	Configure button group dengan styling
onSearchClicked()	None	None	Validate input, start search worker
showError()	message: str	None	Display error dialog
navigateToHomepage()	results: List[Dict]	None	Transition ke results page

Tabel 37. Input Validation

Field	Validation Rule	Error Message	Default Value
Keywords	Non-empty string	"Please enter search keywords"	""
Algorithm	One must be selected	"Please select an algorithm"	None
Top Matches	Positive integer atau 0	"Please enter valid number"	"3"
Combined	All validations pass	-	Proceed dengan search

Tabel 38. Event Handling

Event	Trigger	Handler Method	Action
Search Button Click	Mouse click	onSearchClicked()	Validate input, start worker
Enter Key Press	Keyboard	keyPressEvent()	Trigger search if valid
Algorithm Selection	Button toggle	buttonClicked()	Update selected algorithm
Input Focus	Field selection	focusInEvent()	Clear placeholder text

4.8 Teknis GUI Homepage dan Summary

4.8.1 Struktur Data Homepage

Tabel 39. Homepage Components (homepage.py)

Komponen	Qt Class	Layout	Properti
Main Container	QWidget	QVBoxLayout	Full window content
Header	QHBoxLayout	-	Back button, results counter
Cards Container	QScrollArea	QGridLayout [3 cols]	Scrollable results grid
CVCard	Custom QWidget	QVBoxLayout	Individual result display
Pagination	QHBoxLayout	-	Navigation controls

Tabel 40, CVCard Structure

Element	Component	Font	Color	Content
Resume Name	QLabel	Arial 28pt Bold	#FFFFFF	Filename display
Match Counter	QLabel	Arial 18pt	#FFFFFF	"X matches" format
Skills Frame	QFrame	Arial 14pt	Background: #003B31	Skills breakdown
Skills List	QLabel	Arial 12pt	#FFFFFF	"1. Python - 2 occurrences"
Actions Container	QHBoxLayout	-	-	Button container
View More Button	QPushButton	Arial 16pt	Transparent, Underline	Navigate to summary
View CV Button	QPushButton	Arial 16pt	Transparent, Underline	Open PDF file

4.8.2 Fungsi dan Prosedur Homepage

Tabel 41. Homepage Methods (main_gui.py - IntegratedHomePage)

Method	Parameters	Return Type	Deskripsi
__init__()	results: List[Dict]	-	Initialize dengan search results
setupUI()	None	None	Create layout dan pagination
updateCards()	None	None	Render cards untuk current page

goToNextPage()	None	None	Navigate ke next page
goToPrevPage()	None	None	Navigate ke previous page
onViewMore()	resume_data: Dict	None	Open summary page
onViewCV()	file_path: str	None	Open PDF dengan default viewer

Tabel 42. Pagination System

Variable	Type	Purpose	Calculation
items_per_page	int	Cards per page	Fixed value: 6
current_page	int	Current page number	0-based indexing
total_pages	int	Total pages	ceil(total_results / items_per_page)
start_index	int	First item index	current_page * items_per_page
end_index	int	Last item index	min(start_index + items_per_page, total_results)

4.8.3 Struktur Data Summary Page

Tabel 43. Summary Page Layout (summary.py)

Section	Layout	Components	Styling
Header	QHBoxLayout	Back button, Candidate name	Font: Arial 48pt Bold, Color: #00FFC6
Personal Info	QGroupBox	Birth, Address, Phone cards	Background: #037F68
Skills Section	QVBoxLayout	Skills grid + pagination	3x3 grid, 9 skills per page
Job History	QGroupBox	QGridLayout (2x2)	Experience cards
Education	QGroupBox	QVBoxLayout	Education timeline

Tabel 44. Summary Page Methods

Method	Parameters	Return Type	Deskripsi
__init__()	resume_data: Dict	-	Initialize dengan detailed resume data
setupPersonalInfo()	None	None	Create personal information cards
setupSkillsSection()	None	None	Create skills grid dengan pagination
setupJobHistory()	None	None	Create experience timeline

setupEducation()	None	None	Create education background
nextSkillsPage()	None	None	Navigate skills pagination
prevSkillsPage()	None	None	Navigate skills pagination

4.9 Tata Cara Penggunaan Program

4.9.1 Landing Page

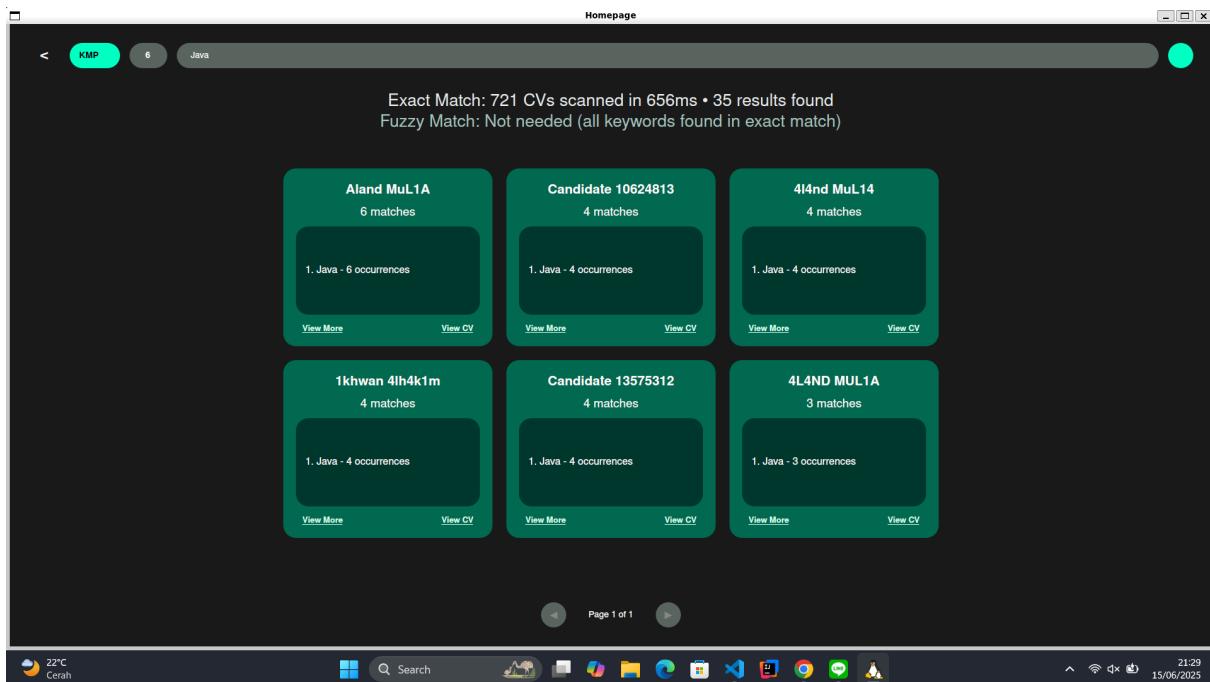
Landing page berisi informasi utama tentang *website* Little Alchemy 2 Recipe Finder. Pengguna dapat memilih untuk melakukan pencarian resep atau melihat profil kelompok Minekrep.

4.9.2 Halaman Pencarian

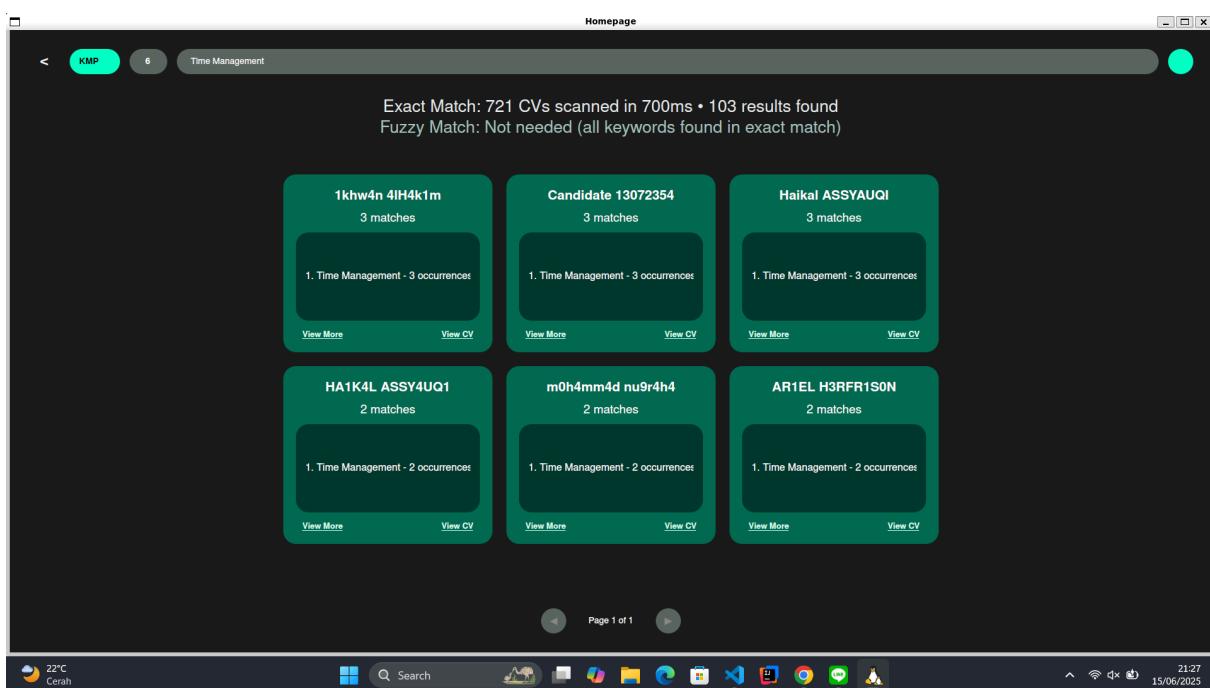
Pada halaman pencarian, pengguna dapat memasukkan nama elemen yang ingin dicari, jenis algoritma yang digunakan untuk pencarian, dan jumlah resep yang diinginkan. Pencarian akan dilakukan setelah pengguna menekan tombol "Find Recipes". Hasil pencarian ditampilkan di kolom "Recipe Results" yang bisa dinavigasi dengan *next recipe* serta *previous recipe* apabila melakukan pencarian *multiple recipe*.

4.10 Pengujian

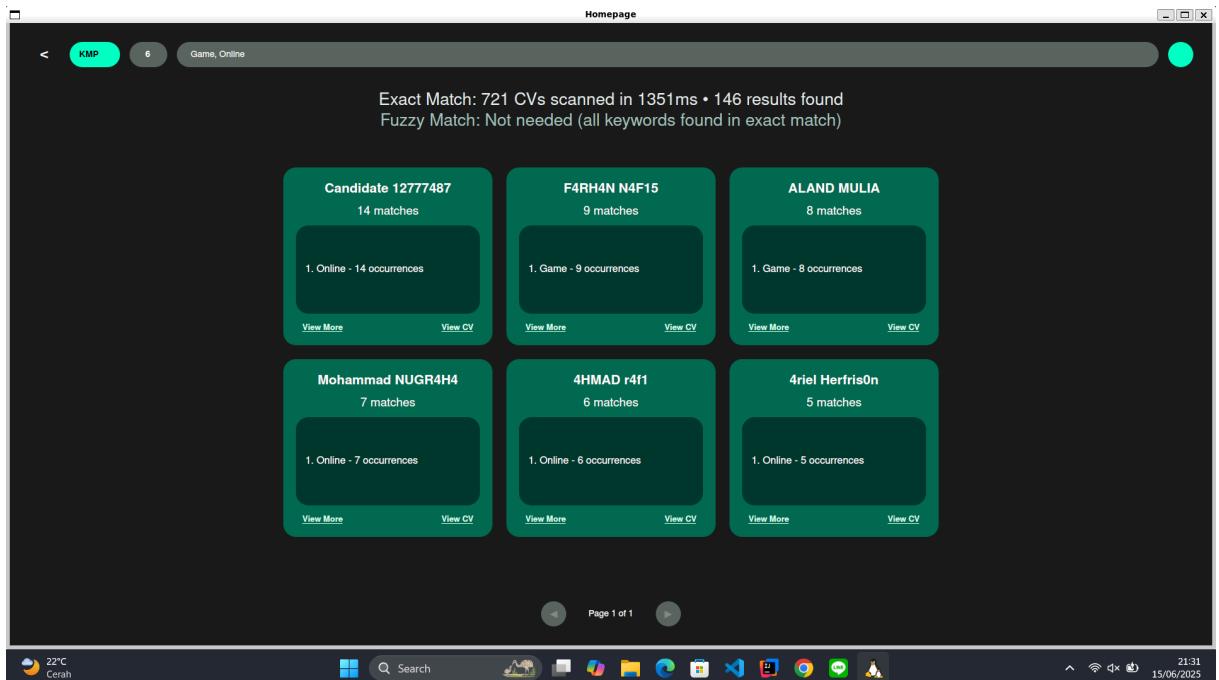
4.10.1 Pengujian Algoritma Knuth-Morris-Pratt



Gambar 6. Pencarian Kata "Java" Menggunakan Algoritma KMP

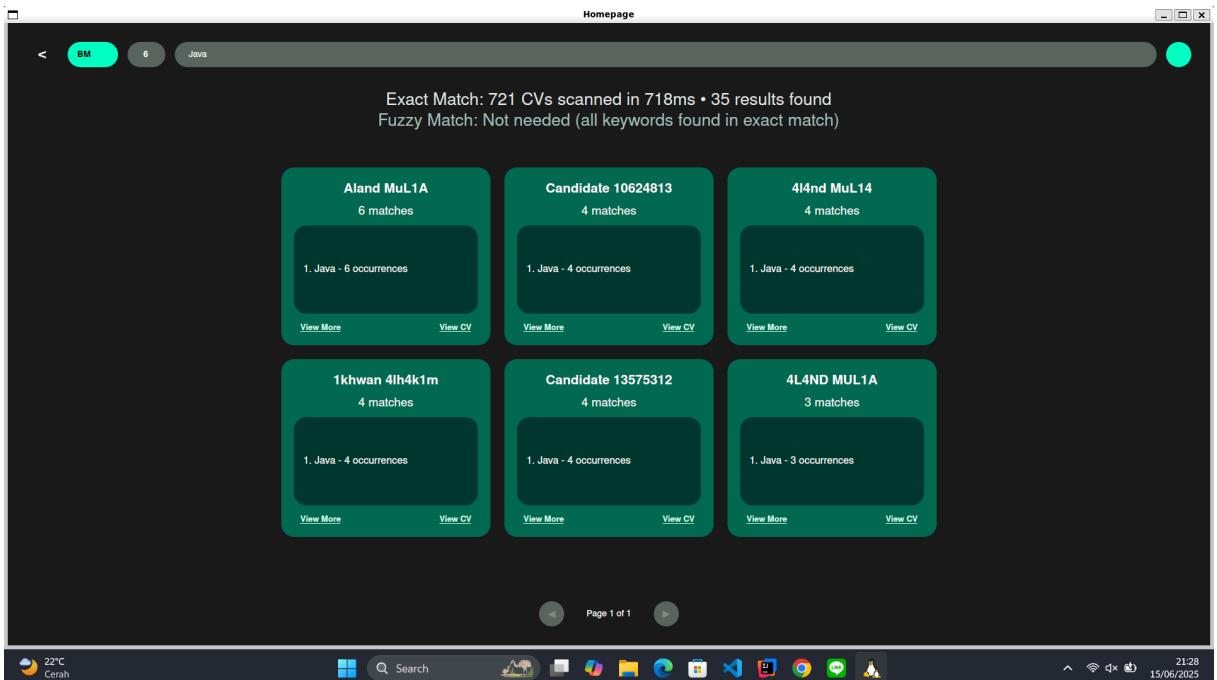


Gambar 7. Pencarian Kata "Time Management" Menggunakan Algoritma KMP

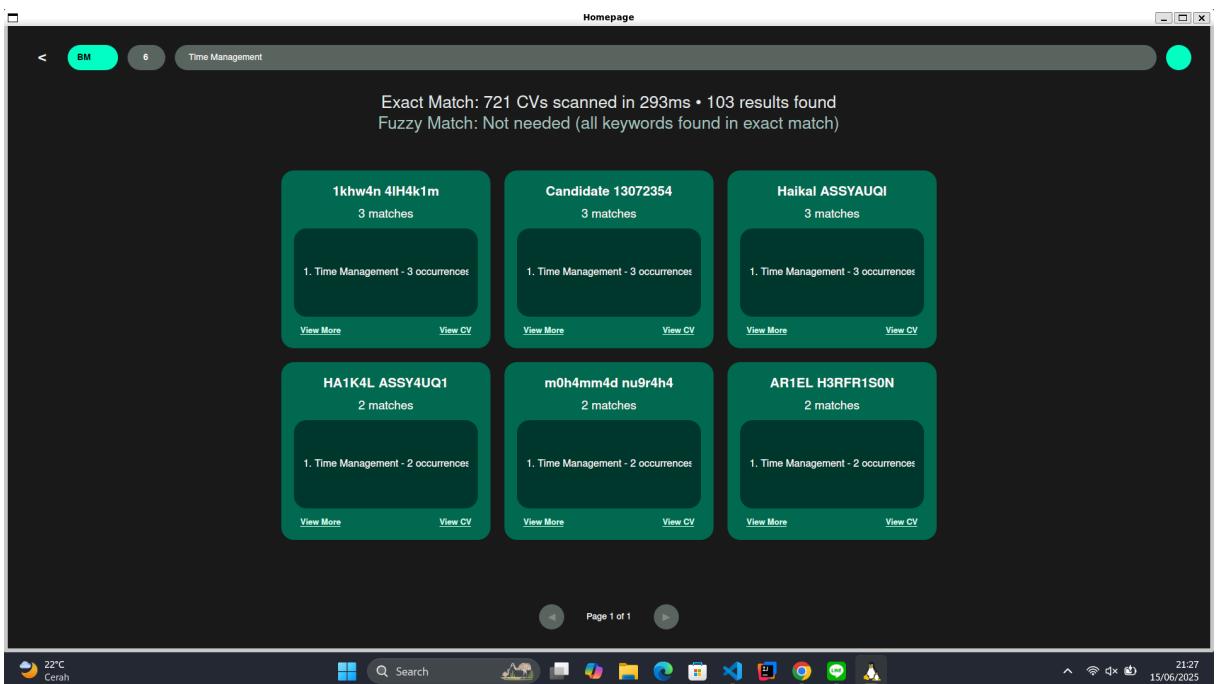


Gambar 8. Pencarian Kata "Game" dan "Online" Menggunakan Algoritma KMP

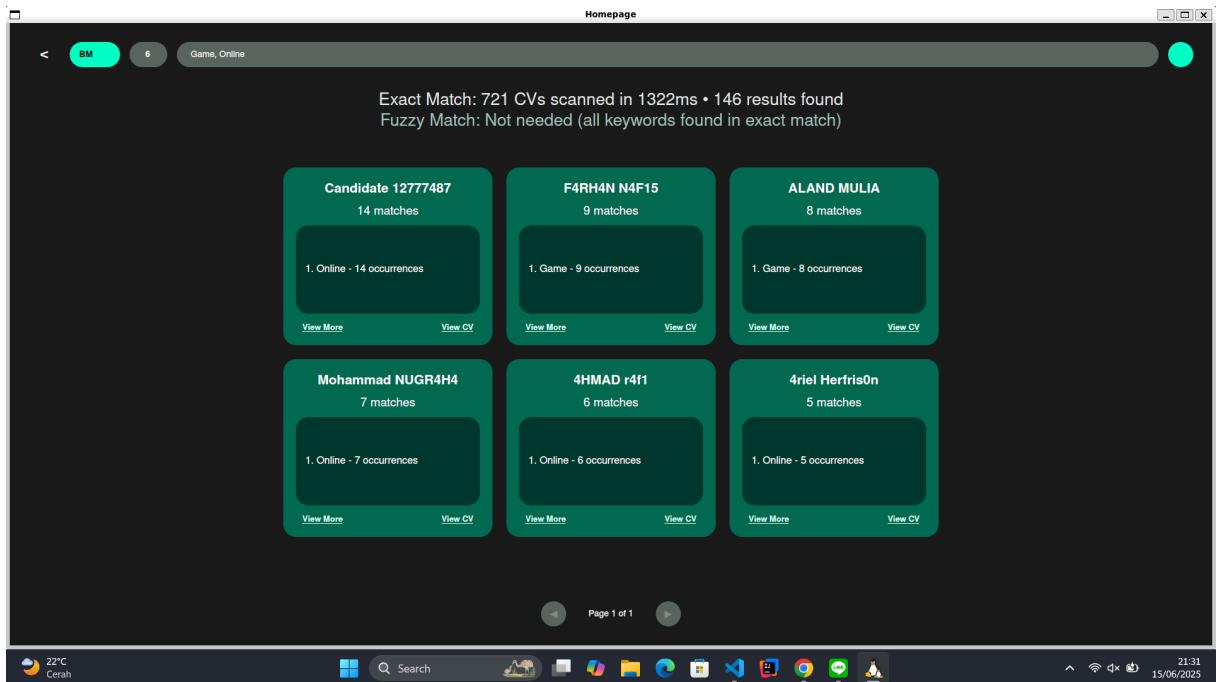
4.10.2 Pengujian Algoritma Boyer Moore



Gambar 9. Pencarian Kata "Java" Menggunakan Algoritma BM

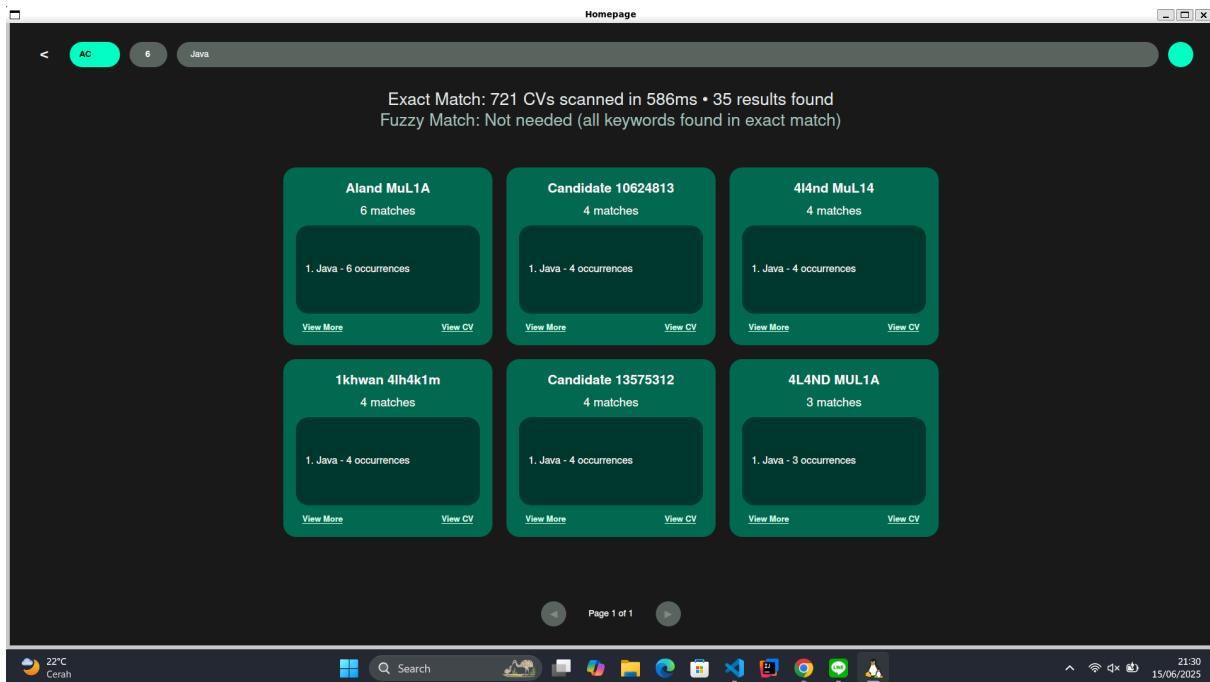


Gambar 10. Pencarian Kata "Time Management" Menggunakan Algoritma BM

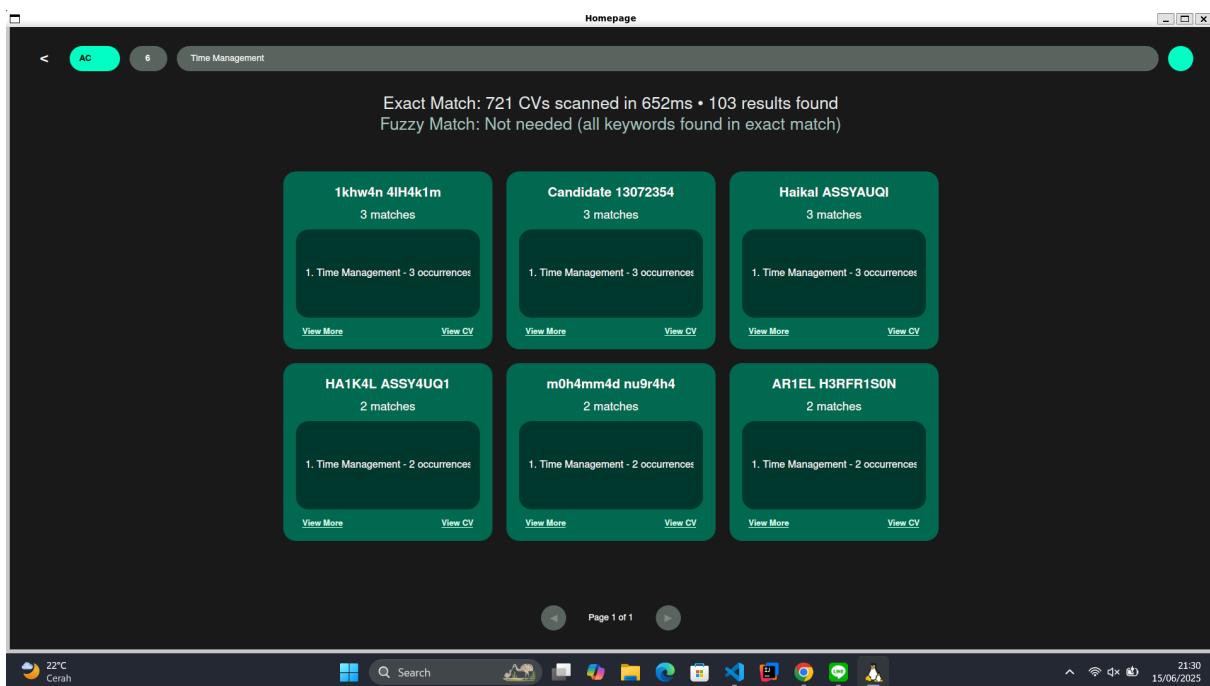


Gambar 11. Pencarian Kata "Game" dan "Online" Menggunakan Algoritma BM

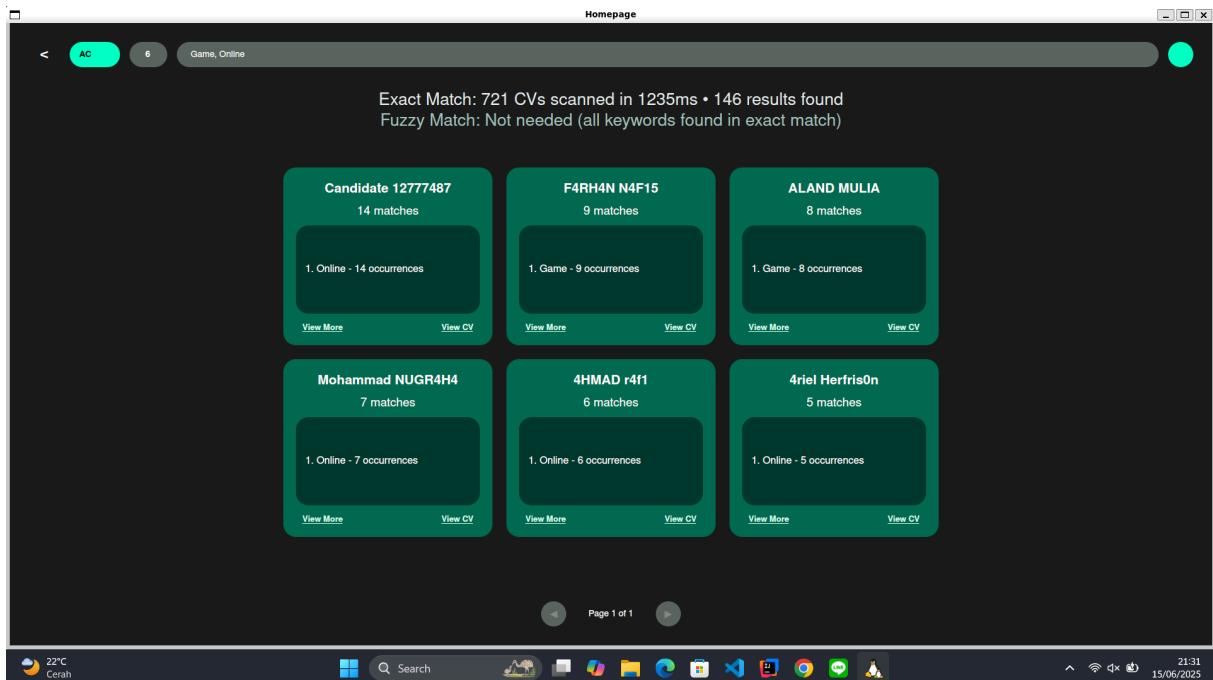
4.10.3 Pengujian Algoritma Aho-Corasick



Gambar 12. Pencarian Kata "Java" Menggunakan Algoritma AC

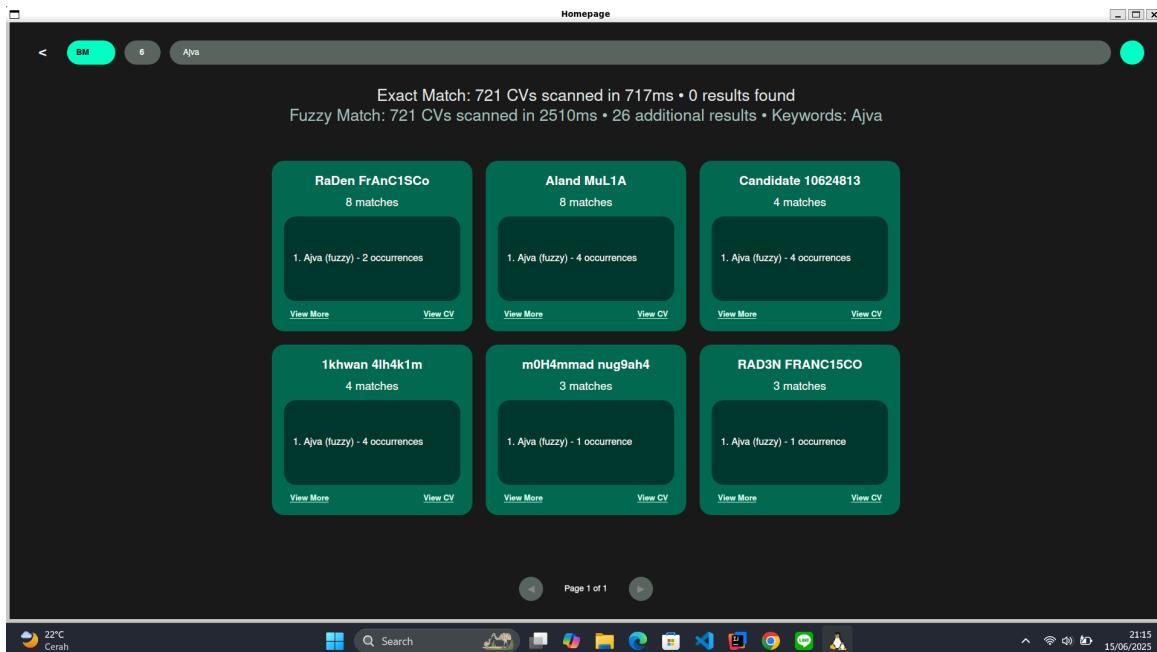


Gambar 13. Pencarian Kata "Time Management" Menggunakan Algoritma AC



Gambar 14. Pencarian Kata "Game" dan "Online" Menggunakan Algoritma AC

4.10.4 Pengujian Fuzzy Match



Gambar 15. Pencarian Kata “Ajva” Menggunakan Algoritma AC dengan Fuzzy Match



Gambar 16. Hasil Sesungguhnya dari Fuzzy Match yang menunjukkan Skill “Java”

4.11 Analisis Hasil Pengujian

4.11.1 Analisis Pengujian Algoritma Knuth-Morris-Pratt

Dari pengujian yang dilakukan, algoritma KMP menunjukkan konsistensi waktu eksekusi yang stabil terlepas dari karakteristik input yang diberikan. Algoritma ini mempertahankan performa linear yang dapat diprediksi, bahkan ketika dihadapkan pada pattern yang memiliki banyak karakter berulang atau prefix yang sama dengan suffix seperti "Java". Dalam konteks pencarian pada koleksi file teks yang tersedia di direktori data, KMP menunjukkan keandalan dalam menangani berbagai jenis konten tanpa mengalami degradasi performa yang signifikan. Waktu preprocessing yang diperlukan untuk membangun failure function relatif cepat dan tidak mempengaruhi performa keseluruhan secara negatif, terutama ketika melakukan pencarian berulang dengan pattern yang sama. Dari ketiga algoritma, algoritma KMP paling cepat untuk mencari "kata berulang".

4.11.2 Pengujian Algoritma Boyer-Moore

Boyer-Moore menunjukkan performa yang sangat baik ketika digunakan untuk mencari pattern dengan panjang yang signifikan, terutama pada teks-teks yang ada dalam dataset. Algoritma ini memanfaatkan karakteristik bad character dan good suffix heuristics dengan efektif, memungkinkan lompatan yang lebih besar dalam proses pencarian. Dalam pengujian pada berbagai file teks, Boyer-Moore mendemonstrasikan kemampuannya untuk mengurangi jumlah perbandingan karakter yang diperlukan, terutama ketika pattern yang dicari tidak sering muncul dalam teks. Keunggulan algoritma ini semakin terlihat ketika ukuran alphabet dalam teks cukup besar dan pattern yang dicari memiliki karakteristik yang memungkinkan optimasi skip yang maksimal. Sebagai contoh dalam pengujian menggunakan kata "Time Management" dan terbukti algoritma ini paling cepat.

4.11.3 Pengujian Algoritma Aho-Corasick

Implementasi Aho-Corasick dalam sistem menunjukkan efisiensi luar biasa ketika digunakan untuk pencarian multiple patterns secara simultan. Algoritma ini membangun finite state automaton yang

memungkinkan pencarian semua pattern dalam sekali traversal teks, yang sangat menguntungkan ketika perlu mencari beberapa kata kunci sekaligus dalam koleksi file. Dalam pengujian, terlihat bahwa algoritma ini paling cepat untuk mencari dua kata sekaligus yaitu "Game" dan "Online" daripada ketiga algoritma lainnya.

4.11.4 Pengujian Fuzzy Match

Fuzzy search menunjukkan fleksibilitas yang tinggi dalam menangani variasi input pengguna dan kesalahan pengetikan yang umum terjadi. Algoritma ini mampu menemukan kecocokan approximate bahkan ketika input tidak persis sama dengan teks yang ada dalam dataset. Dalam pengujian, fuzzy search mendemonstrasikan kemampuannya untuk memberikan hasil yang relevan meskipun terdapat perbedaan ejaan atau variasi dalam penulisan seperti "Ajva" yang bisa didetect menjadi "Java". Namun, trade-off yang terlihat adalah peningkatan waktu komputasi yang signifikan dibandingkan dengan algoritma exact matching lainnya. Threshold similarity yang kami gunakan adalah 60%. Threshold 60% sangat sesuai untuk menangani kesalahan pengetikan yang umum terjadi dalam input pencarian. Pengguna sering melakukan typo atau variasi ejaan ketika mencari skill tertentu. Dengan threshold ini, sistem dapat menangkap kesalahan seperti "machne learning" untuk "machine learning" atau "react js" untuk "reactjs" tanpa memberikan hasil yang terlalu noise.

Bab 5

Kesimpulan, Saran, dan Refleksi

5.1 Kesimpulan

Dalam Tugas Besar 3 ini, kami berhasil membuat aplikasi desktop **BukitDuri** untuk pencarian CV menggunakan framework PyQt5 dengan bahasa Python. Aplikasi ini mengimplementasikan dua algoritma string matching yaitu **Knuth-Morris-Pratt (KMP)**, **Boyer-Moore (BM)**, dan **Aho-Corasick (AH)** untuk melakukan pencarian kata kunci dalam dokumen CV berformat PDF dan TXT. Sistem dilengkapi dengan fitur enkripsi menggunakan algoritma Vigenère Cipher untuk keamanan data CV yang disimpan dalam database SQLite. Aplikasi memiliki interface yang user-friendly dengan fitur upload CV, pencarian berdasarkan kata kunci, pemilihan algoritma, pengaturan jumlah hasil yang ditampilkan, serta kemampuan untuk melihat summary CV dan membuka file CV asli. Kami juga mengimplementasikan sistem database yang terstruktur untuk menyimpan metadata CV dan konten yang telah dienkripsi.

5.2 Saran

Saran dari kami untuk tugas besar selanjutnya adalah perlunya penjelasan yang lebih detail mengenai implementasi algoritma string matching dalam konteks aplikasi nyata, terutama terkait optimasi performa untuk file berukuran besar. Selain itu, perlu ada panduan yang lebih jelas mengenai integrasi antara algoritma pencarian dengan sistem database dan enkripsi. Spesifikasi mengenai format output dan handling error juga perlu diperjelas untuk menghindari ambiguitas dalam implementasi.

5.3 Refleksi

Melalui tugas besar ini, kami telah memperoleh pemahaman mendalam mengenai implementasi algoritma string matching **KMP**, **Boyer-Moore**, dan **Aho-Corasick** serta aplikasinya dalam sistem pencarian dokumen. Kami juga mempelajari konsep enkripsi Vigenère Cipher dan pentingnya keamanan data

dalam aplikasi nyata. Pengembangan aplikasi desktop dengan PyQt5 memberikan pengalaman berharga dalam GUI programming dan manajemen database menggunakan SQLite. Tantangan terbesar yang dihadapi adalah optimasi performa algoritma untuk memproses multiple file CV secara efisien serta implementasi sistem enkripsi yang aman namun tetap memungkinkan pencarian yang akurat. Walaupun terdapat kendala dalam hal debugging dan integrasi antar komponen, kami berhasil menyelesaikan aplikasi yang fungsional dan memenuhi semua requirement yang diminta.

Lampiran

Pranala *repository*:

https://github.com/rakdaf08/Tubes3_BukitDuri

Pranala video pada Youtube:

<https://youtu.be/VYYyqVWsNeo>

Tabel keterselesaian:

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

9	Membuat bonus enkripsi data profil <i>applicant</i> .	v	
10	Membuat bonus algoritma Aho-Corasick.	v	
11	Membuat bonus video dan diunggah pada Youtube.	v	

Daftar Pustaka

Munir, R. (2025). Pencocokan String (*string matching*) dengan Algoritma Brute Force, KMP, Boyer-Moore. Diakses pada 06 Juni 2025 pukul 21.30 WIB melalui

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

Munir, R. (2025). Pencocokan String dengan Regular Expression (regex) Diakses pada 06 Juni 2025 pukul 21.00 WIB melalui
[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)

Aho, A. V., & Corasick, M. J. (1975). Efficient String Matching: An Aid to Bibliographic Search. Communications of the ACM, 18(6), 333-340. Diakses pada 07 Juni 2025 pukul 20.00 WIB melalui
<https://dl.acm.org/doi/10.1145/360825.360855>