

# **Laporan Tugas Besar 3**

## **IF2211 Strategi Algoritma**

### **2024/2025**

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



Disusun oleh :

Syahrizal Bani Khairan                  13523063

Muhammad Iqbal Haidar                  13523111

Ferdin Arsenarendra Purtadi                  13523117

**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung  
2025**

## Daftar Isi

|  |           |
|--|-----------|
| <b>I. Deskripsi Tugas</b>                        | <b>1</b>  |
| <b>II. Landasan Teori</b>                        | <b>2</b>  |
| 2.1 Knuth-Morris-Pratt                           | 2         |
| 2.2 Boyer-Moore                                  | 3         |
| 2.3 Aho-Corasick                                 | 4         |
| <b>III. Analisis Pemecahan Masalah</b>           | <b>5</b>  |
| 3.1 Langkah Pemecahan Masalah                    | 5         |
| 3.2 Proses Pemetaan Masalah ke Algoritma         | 6         |
| 3.3 Fitur Fungsionalitas dan Arsitektur Aplikasi | 8         |
| 3.4 Ilustrasi Kasus                              | 10        |
| <b>IV. Implementasi dan Pengujian</b>            | <b>11</b> |
| 4.1. Struktur File Program                       | 11        |
| 4.2. Modul Utama dan Fungsinya                   | 11        |
| 4.3. Tampilan Aplikasi                           | 34        |
| 4.4. Cara Menjalankan Aplikasi                   | 36        |
| 4.5. Hasil Pengujian                             | 36        |
| 4.6 Analisis Hasil Pengujian                     | 37        |
| <b>V. Penutup</b>                                | <b>38</b> |
| 5.1. Kesimpulan                                  | 38        |
| 5.2. Saran                                       | 38        |
| 5.3. Refleksi                                    | 38        |
| <b>Lampiran</b>                                  | <b>39</b> |
| <b>Daftar Pustaka</b>                            | <b>40</b> |

## 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.

## II. Landasan Teori

String matching merupakan teknik dalam ilmu komputer yang digunakan untuk menemukan kemunculan suatu pola atau kata kunci tertentu di dalam sebuah teks. Proses ini penting dalam berbagai aplikasi yang berkaitan dengan pengolahan data teks, seperti mesin pencari, sistem pemeriksa ejaan, analisis dokumen, dan sistem pelacakan pelamar kerja (ATS). Secara umum, pencocokan string melibatkan dua elemen utama, yaitu teks sebagai sumber utama pencarian, dan pola sebagai kata atau frasa yang ingin ditemukan dalam teks tersebut. Sebagai contoh, ketika seorang pengguna ingin mencari kandidat dengan kemampuan tertentu, seperti “python” atau “data analysis”, sistem akan mencocokkan kata-kata tersebut dengan isi dari dokumen CV pelamar.

Penerapan string matching dapat dilakukan melalui dua pendekatan utama, yakni pencocokan tepat (*exact matching*) dan pencocokan mendekati (*fuzzy matching*). Exact matching hanya mempertimbangkan kecocokan yang benar-benar identik antara pola dan bagian dari teks, sedangkan fuzzy matching tetap mempertimbangkan kemiripan meskipun terdapat perbedaan kecil, seperti kesalahan pengetikan atau variasi dalam penulisan. Untuk mengakomodasi kebutuhan ini, digunakan berbagai algoritma yang dirancang khusus untuk memproses teks secara efisien dan akurat, bahkan ketika ukuran teks sangat besar atau jumlah pola yang dicocokkan cukup banyak.

Dalam konteks sistem ATS, string matching dimanfaatkan untuk mencocokkan daftar kata kunci yang dimasukkan oleh pengguna terhadap konten CV yang telah dikonversi menjadi bentuk string. Dengan bantuan algoritma yang tepat, sistem dapat memproses pencarian dengan lebih cepat dan efektif, serta memberikan hasil yang relevan sesuai kebutuhan pencarian pengguna.

### 2.1 Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma string matching yang efisien karena menghindari perulangan pencocokan terhadap karakter teks yang sudah dibandingkan sebelumnya. KMP bekerja dengan membentuk sebuah *border function* (juga disebut prefix table) yang mencatat panjang prefiks terpanjang dari pola yang juga merupakan sufiks.

Karakteristik dari algoritma KMP meliputi:

- Kompleksitas waktu pencarian adalah  $O(n + m)$ , di mana  $n$  adalah panjang teks dan  $m$  adalah panjang pola, menjadikannya sangat efisien untuk pencocokan pola tunggal.
- Tidak memerlukan pergeseran mundur (backtracking) pada indeks teks.

- Stabil dan konsisten performanya, bahkan untuk pola yang memiliki banyak pengulangan atau teks yang panjang.
- Digunakan ketika pola yang dicari sering digunakan berulang kali pada banyak teks.

Langkah-langkah utama dari algoritma KMP dapat dijelaskan sebagai berikut:

### **1. Preprocessing pola**

Tahap awal dari algoritma adalah membentuk sebuah array prefix (juga dikenal sebagai border function atau failure function) dari pola. Array ini menyimpan panjang prefiks terpanjang dari bagian awal pola yang juga merupakan sufiks dari bagian sebelumnya. Tujuannya adalah agar saat terjadi ketidaksesuaian karakter antara teks dan pola, pencarian dapat dilanjutkan dari posisi yang tepat dalam pola, bukan dari awal.

### **2. Pencocokan terhadap teks**

Dengan memanfaatkan prefix array, proses pencocokan dilakukan secara linear. Setiap kali karakter dari teks dan pola cocok, indeks keduanya akan maju. Namun jika terjadi ketidakcocokan, KMP tidak akan mengulangi pencocokan dari awal pola, melainkan melompat ke indeks pola yang sesuai berdasarkan prefix array, dan tetap mempertahankan indeks teks saat ini.

### **3. Perekaman hasil**

Ketika seluruh karakter dalam pola berhasil dicocokkan dengan bagian dari teks, algoritma mencatat indeks posisi awal kecocokan di dalam teks, dan kemudian melanjutkan pencarian dari posisi setelah kecocokan.

## **2.2 Boyer-Moore**

Algoritma Boyer-Moore (BM) adalah salah satu algoritma pencocokan string tercepat dan paling banyak digunakan dalam praktik, terutama ketika teks yang dipindai sangat panjang dan polanya cukup besar. Keunggulan utama dari algoritma ini terletak pada kemampuannya untuk melakukan lompatan pencarian yang besar dalam teks sehingga tidak perlu memeriksa setiap karakter satu per satu..

Karakteristik dari algoritma Boyer-Moore meliputi:

- Kompleksitas waktu terbaiknya mendekati  $O(n/m)$  dalam banyak kasus nyata, menjadikannya lebih cepat dibandingkan algoritma lain terutama saat pencocokan jarang terjadi.
- Dilakukan dari kanan ke kiri pada pola, berbeda dengan algoritma lain seperti KMP yang memindai dari kiri ke kanan.
- Memiliki performa sangat baik ketika alfabet teks besar dan pola tidak terlalu pendek.

Langkah-langkah kerja algoritma Boyer-Moore dapat dijelaskan sebagai berikut:

**1. Preprocessing pola:**

Algoritma membentuk sebuah *last occurrence table* yang mencatat posisi terakhir dari setiap karakter dalam pola. Ketika terjadi ketidakcocokan saat pencocokan dari kanan ke kiri, tabel ini digunakan untuk menentukan berapa banyak pola dapat digeser ke kanan.

**2. Pencocokan dari kanan ke kiri:**

Proses pencocokan dimulai dari karakter paling kanan pola terhadap bagian teks yang sedang diperiksa. Jika cocok, maka pencocokan berlanjut ke kiri. Jika terjadi ketidaksesuaian, algoritma menggunakan informasi dari bad character table untuk menentukan seberapa jauh pola bisa digeser ke kanan tanpa melewatkannya kemungkinan kecocokan.

**3. Lompatan pencarian:**

Ketimbang maju satu per satu seperti algoritma lainnya, BM dapat melompati beberapa karakter sekaligus tergantung dari posisi karakter yang tidak cocok. Ini yang membuatnya sangat cepat dalam kasus umum, terutama jika teks tidak mengandung banyak karakter yang sama dengan pola.

### 2.3 Aho-Corasick

Algoritma Aho-Corasick merupakan algoritma pencocokan string yang dirancang khusus untuk menangani pencarian banyak pola (*multi-pattern matching*) secara bersamaan dalam sebuah teks. Algoritma ini sangat efisien karena hanya membutuhkan satu kali traversing terhadap teks, terlepas dari jumlah pola yang dicari. Hal ini berbeda dengan pendekatan konvensional yang mencocokkan setiap pola satu per satu. Dengan Aho-Corasick, seluruh daftar kata kunci dapat dicocokkan sekaligus dalam satu proses pemindaian, sehingga cocok untuk aplikasi yang memerlukan pencocokan banyak keyword seperti sistem Applicant Tracking System (ATS).

Karakteristik utama dari Aho-Corasick antara lain:

- Kompleksitas waktu:  $O(n + m + z)$ , di mana  $n$  adalah panjang teks,  $m$  adalah total panjang seluruh pola, dan  $z$  adalah jumlah total kecocokan yang ditemukan.
- Kemampuan *multi-pattern matching* menjadikannya lebih unggul dari KMP atau BM saat jumlah pola sangat banyak.
- Memanfaatkan struktur data Trie untuk menyimpan seluruh pola dan dilengkapi dengan failure link seperti pada automata, mirip konsep yang digunakan dalam algoritma KMP.
- Proses pencarian bersifat deterministik dan konsisten, tanpa perlu backtracking.

Langkah-langkah kerja dari algoritma Aho-Corasick dapat dijelaskan sebagai berikut:

**1. Membangun Trie:**

Pertama-tama, semua pola yang akan dicocokkan dimasukkan ke dalam struktur pohon Trie. Setiap simpul pada Trie mewakili karakter dalam pola, dan jalur dari akar ke sebuah simpul akhir mewakili sebuah pola lengkap.

## 2. Menambahkan Failure Links:

Setelah Trie terbentuk, langkah selanjutnya adalah membuat *failure link* untuk setiap simpul. Failure link ini berfungsi sebagai jalur alternatif ketika terjadi ketidaksesuaian karakter selama pemindaian teks. Jika karakter yang sedang diperiksa tidak ada di simpul saat ini, algoritma akan mengikuti failure link untuk mencari simpul lain yang mungkin cocok, mirip mekanisme lompatan pada KMP.

## 3. Pemindaian Teks:

Setelah struktur selesai disiapkan, algoritma mulai memindai teks satu karakter demi satu. Setiap karakter akan menavigasi Trie sesuai dengan transisi yang ada. Jika ditemukan simpul yang memiliki *output* (daftar pola yang berakhir di simpul tersebut), maka semua pola tersebut dianggap cocok ditemukan di posisi teks saat ini.

## 4. Pencatatan Hasil:

Untuk setiap pola yang ditemukan, algoritma mencatat posisi awal kemunculannya dalam teks. Karena semua pola diperiksa dalam satu lintasan teks, hasil pencocokan menjadi sangat cepat dan efisien.

## III. Analisis Pemecahan Masalah

### 3.1 Langkah Pemecahan Masalah

Proses pemecahan masalah utama, yaitu menemukan kandidat yang relevan berdasarkan kata kunci dari database CV, dapat dipecah menjadi beberapa langkah logis yang diimplementasikan dalam aplikasi.

#### 1. Inisialisasi dan Pemuatan Data

Aplikasi dimulai dengan memuat data CV yang tersedia. Berdasarkan file src/database/cv\_database.py, sistem secara otomatis mendeteksi dan memuat semua file CV berformat .txt dari direktori data/txt/.

Setiap file teks ini dianggap sebagai representasi dari satu CV kandidat, dan kontennya disimpan dalam memori untuk proses pencarian. Nama file (misalnya, 10089434.txt) digunakan sebagai pengidentifikasi unik untuk setiap kandidat.

#### 2. Antarmuka Pengguna untuk Input

Pengguna (dalam hal ini, staf HR) disajikan dengan antarmuka grafis (GUI) yang dibangun menggunakan PyQt5, seperti yang terlihat pada file-file di dalam direktori src/gui/. Melalui src/gui/search\_page.py dan src/gui/input.py, pengguna dapat memasukkan kata kunci (keyword) yang ingin dicari (misalnya, "Python", "Data Analyst", "Bandung").

Pengguna juga diberikan pilihan untuk memilih algoritma pencocokan string yang akan digunakan, yaitu Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM), melalui komponen src/gui/algorithm\_choice.py.

### 3. Eksekusi Proses Pencarian:

Setelah pengguna menekan tombol "Search", aplikasi memicu fungsi pencarian yang diatur dalam src/models/[search.py](#). Fungsi ini mengambil kata kunci dan pilihan algoritma dari antarmuka.

Selanjutnya, fungsi tersebut akan mengiterasi setiap CV yang telah dimuat sebelumnya. Untuk setiap CV, algoritma yang dipilih (KMP dari src/lib/kmp.py atau BM dari src/lib/bm.py) dijalankan untuk menemukan kemunculan kata kunci.

### 4. Pengolahan dan Visualisasi Hasil

Hasil dari proses pencarian adalah daftar kandidat (diidentifikasi oleh nama file CV mereka) yang CV-nya mengandung kata kunci yang dicari. Hasil ini kemudian ditampilkan kepada pengguna melalui halaman hasil (src/gui/[result.py](#)).

Setiap kandidat yang cocok ditampilkan dalam format "kartu" (src/gui/applicant\_card.py), yang menampilkan informasi ringkas seperti ID kandidat dan mungkin cuplikan teks di mana kata kunci ditemukan. Pengguna kemudian dapat menindaklanjuti kandidat-kandidat yang relevan. Dari kartu tersebut terdapat tombol summary untuk melihat ringkasan kandidat dengan memanfaatkan regex.

## 3.2 Proses Pemetaan Masalah ke Algoritma

Inti dari aplikasi ini adalah penerapan algoritma string matching. Masalah bisnis "mencari keahlian di dalam CV" dapat dipetakan secara langsung ke elemen-elemen formal algoritma KMP, BM, dan Aho-corasick sebagai berikut:

- Teks (T): Dalam konteks ini, Teks adalah konten keseluruhan dari sebuah file CV. Setiap file .txt di dalam direktori data/txt/ dibaca sebagai satu string panjang yang menjadi target pencarian.
- Pola/pattern (P): Pola adalah kata kunci (keyword) yang dimasukkan oleh pengguna melalui kolom input pada antarmuka aplikasi.

### Pemetaan untuk Algoritma KMP (Knuth-Morris-Pratt)

#### 1. Preprocessing (Fase Persiapan)

Sebelum pencarian dimulai pada sebuah Teks (CV), algoritma KMP melakukan pra-pemrosesan pada Pola (kata kunci). Fungsi compute\_lps\_array dalam src/lib/kmp.py membangun sebuah Longest Proper Prefix which is also Suffix (LPS) array, atau sering disebut border array.

Array ini menyimpan informasi tentang struktur internal kata kunci, yang memungkinkan algoritma untuk menggeser Pola secara cerdas ketika terjadi ketidakcocokan, tanpa perlu membandingkan ulang karakter yang sudah pasti cocok.

## 2. Searching (Fase Pencarian)

Fungsi KMP\_search di src/lib/kmp.py melakukan iterasi pada Teks (konten CV) dan Pola (kata kunci) secara bersamaan. Ketika ketidakcocokan terdeteksi, nilai dari LPS array digunakan untuk menentukan seberapa jauh Pola bisa digeser ke kanan, sehingga proses pencarian menjadi sangat efisien dengan kompleksitas waktu  $O(n+m)$ , di mana n adalah panjang Teks dan m adalah panjang Pola.

## Pemetaan untuk Algoritma BM (Boyer-Moore)

### 1. Preprocessing (Fase Persiapan):

Seperti KMP, BM juga melakukan pra-pemrosesan pada Pola (kata kunci) untuk membuat tabel bantu. Dalam implementasi ini (src/lib/bm.py), fungsi build\_last\_function membangun sebuah tabel yang disebut last occurrence function. Tabel ini memetakan setiap karakter dalam alfabet ke posisi kemunculan terakhirnya di dalam Pola. Jika karakter tidak ada di Pola, nilainya adalah -1.

### 2. Searching (Fase Pencarian):

Fungsi BM\_search di src/lib/bm.py memiliki ciri khas membandingkan Pola dengan Teks dari kanan ke kiri. Ketika terjadi ketidakcocokan, last occurrence table digunakan. Algoritma melihat karakter di Teks yang menyebabkan ketidakcocokan dan menggunakan tabel last occurrence untuk melakukan pergeseran (lompatan) Pola ke kanan sejauh mungkin. Lompatan ini seringkali lebih besar daripada pergeseran satu karakter, membuat BM sangat cepat pada praktiknya, terutama untuk Pola yang panjang.

## Pemetaan untuk Algoritma Aho-Corasick

### 1. Preprocessing (Fase Persiapan)

Fase ini adalah inti dari Aho-Corasick, di mana semua kata kunci digabungkan menjadi satu struktur data tunggal yang disebut finite automaton (atau lebih spesifik, sebuah Trie dengan tautan tambahan).

**Pembangunan Trie:** Fungsi \_build\_tree dalam kelas Aho Corasick di file src/lib/aho\_corasick.py mengambil semua kata kunci dan menyusunnya ke dalam sebuah struktur pohon (Trie). Setiap node di pohon merepresentasikan sebuah prefiks dari satu atau lebih kata kunci. Node yang menandai akhir dari sebuah kata kunci akan ditandai secara khusus.

**Pembangunan Tautan Kegagalan (Failure Links):** Setelah Trie dibangun, fungsi \_build\_failure\_links membuat "jalan pintas" antar node. Failure link dari sebuah node menunjuk ke prefiks terpanjang dari string yang diwakili node tersebut, yang juga merupakan prefiks dari

kata kunci lain di dalam Trie. Tautan ini memungkinkan algoritma untuk melanjutkan pencarian dari posisi lain yang relevan ketika terjadi ketidakcocokan, tanpa harus memulai ulang dari awal teks.

## 2. Searching (Fase Pencarian)

Fungsi search di src/lib/aho\_corasick.py memproses Teks (konten CV) hanya satu kali dari awal hingga akhir. Untuk setiap karakter dalam Teks, algoritma bergerak mengikuti transisi yang sesuai di dalam automaton (Trie).

Jika mencapai node yang ditandai sebagai akhir dari sebuah kata kunci, algoritma akan mencatat temuan tersebut. Berkat failure links, ia juga akan memeriksa apakah node saat ini dapat dicapai dari node lain yang juga menandai akhir kata kunci (untuk kasus di mana satu kata kunci adalah substring dari kata kunci lain).

Ketika terjadi ketidakcocokan (tidak ada transisi untuk karakter saat ini), algoritma mengikuti failure link untuk beralih ke state berikutnya yang paling mungkin, menghindari pemindaiannya ulang teks.

## 3.3 Fitur Fungsionalitas dan Arsitektur Aplikasi

### Arsitektur Aplikasi

Arsitektur aplikasi ini mengadopsi pola yang mirip dengan Model-View-Controller (MVC) untuk memisahkan antara logika bisnis, data, dan antarmuka pengguna.

- Model: Lapisan ini bertanggung jawab atas data dan logika bisnis inti.
- Struktur Data: Didefinisikan dalam direktori src/models/, yang berisi kelas-kelas data seperti SearchParams, CVSummary, dan ApplicantMatchData untuk memastikan aliran data yang terstruktur di seluruh aplikasi.
- Akses Basis Data: Dikelola oleh src/database/cv\_database.py, yang berfungsi sebagai jembatan antara aplikasi dan basis data MySQL. Modul ini menangani semua operasi query untuk menyimpan dan mengambil profil aplikasi serta path CV.
- Logika Algoritma: Terletak di direktori src/lib/, di mana setiap algoritma (KMP, BM, Levenshtein, Regex) diimplementasikan dalam modulnya sendiri.
- View: Lapisan ini adalah representasi visual dari aplikasi yang dilihat oleh pengguna.
- Antarmuka Pengguna (GUI): Dibangun menggunakan framework PyQt6, semua komponen visual seperti jendela, tombol, dan kartu hasil didefinisikan dalam direktori src/gui/. Setiap bagian utama dari UI (misalnya, search\_page.py, summary\_page.py) dipisahkan menjadi kelasnya sendiri untuk modularitas.
- Controller: Bertindak sebagai pusat kendali yang menghubungkan Model dan View.
- Orkestrasi Utama: src/main\_window.py adalah kelas utama yang mengelola interaksi pengguna. Ia menerima sinyal dari komponen GUI (View), memanggil fungsi yang sesuai dari lapisan

Model (misalnya, memulai pencarian, mengambil data dari database), dan kemudian memperbarui View dengan hasil yang relevan.

## Fitur Fungsional

Aplikasi ini dilengkapi dengan serangkaian fitur yang dirancang untuk menyederhanakan dan mempercepat proses penyaringan kandidat:

### 1. Pencarian Cerdas Berbasis Teks:

- Exact Match: Pengguna dapat melakukan pencarian CV menggunakan kata kunci spesifik dengan memilih antara dua algoritma string matching yang efisien: Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM).
- Fuzzy Match: Jika pencarian exact tidak menemukan satupun kandidat, sistem secara otomatis melakukan pencarian fuzzy menggunakan Levenshtein Distance. Fitur ini sangat berguna untuk mengatasi kesalahan pengetikan (typo) pada kata kunci.

### 2. Ekstraksi Informasi Otomatis:

- Parsing PDF: Sistem mampu membaca dan mengekstrak seluruh konten teks dari dokumen CV berformat PDF.
- Ekstraksi Regex: Pada halaman ringkasan, informasi penting seperti keahlian, riwayat pekerjaan, dan pendidikan diekstrak secara otomatis dari teks CV menggunakan Regular Expression (Regex).

### 3. Antarmuka Pengguna yang Interaktif dan Informatif:

- Visualisasi Hasil: Hasil pencarian ditampilkan dalam bentuk kartu-kartu yang diurutkan berdasarkan relevansi (jumlah kata kunci yang ditemukan).
- Filter Hasil: Pengguna dapat membatasi jumlah hasil yang ditampilkan melalui opsi "Top Matches".
- Detail Kecocokan: Setiap kartu tidak hanya menampilkan nama dan jumlah total kecocokan, tetapi juga rincian kata kunci apa saja yang ditemukan beserta frekuensinya.
- Analisis Kinerja: Waktu eksekusi yang dibutuhkan untuk proses pencarian ditampilkan, memberikan gambaran tentang efisiensi algoritma.

### 4. Akses Mudah ke Detail Kandidat:

- Ringkasan Cerdas: Dengan satu klik pada tombol "Summary", pengguna dapat melihat halaman ringkasan yang menampilkan data diri aplikan dari basis data serta informasi penting yang telah diekstrak oleh Regex.
- Akses CV Asli: Tombol "View CV" memungkinkan rekruter untuk langsung membuka dan meninjau file PDF asli milik kandidat.

## 3.4 Ilustrasi Kasus

Untuk memberikan gambaran konkret tentang cara kerja aplikasi, berikut adalah ilustrasi kasus penggunaan oleh seorang Manajer HR.

Skenario: Seorang Manajer HR bernama Rina di sebuah perusahaan retail sedang mencari kandidat untuk posisi "HR Generalist". Kriteria utama yang ia cari adalah kandidat yang memiliki pengalaman dalam "rekrutmen", mengelola "payroll", dan menangani "employee relations".

## Langkah-langkah Penggunaan Aplikasi:

- Input Pencarian Awal: Rina membuka aplikasi ATS. Pada halaman utama, ia memasukkan kata kunci: recruitmen, payroll, employee relations di kolom "Keywords". Ia memilih algoritma KMP (Knuth-Morris-Pratt) dan menetapkan "Top Matches" sebanyak 5.
- Menerima Hasil Exact Match: Rina menekan tombol "Search". Dalam hitungan detik, aplikasi menampilkan 5 kandidat teratas dari kategori HR yang CV-nya mengandung kata kunci tersebut. Hasil diurutkan secara otomatis, dan seorang kandidat bernama "Kawaca Haryanto" muncul di posisi teratas. Kartu kandidat menunjukkan rincian kecocokan kata kunci dari CV-nya.
- Meninjau Ringkasan Kandidat: Tertarik dengan Kawaca, Rina mengklik tombol "Summary" pada kartunya. Halaman baru terbuka, menampilkan: Informasi Kontak: Nama lengkap, email, dan nomor telepon Kawaca yang diambil langsung dari basis data.
- Riwayat Pekerjaan: Jabatan terakhirnya sebagai "Staf HR" di perusahaan sebelumnya beserta periode kerja, yang diekstrak secara otomatis oleh Regex.
- Keahlian: Daftar keahlian relevan lainnya yang juga ditemukan di CV, seperti "HRIS" dan "manajemen kinerja".
- Verifikasi dengan CV Asli: Untuk memahami konteks pengalaman kandidat dalam menangani "employee relations", Rina mengklik tombol "View CV". File PDF asli milik Kawaca langsung terbuka, dan Rina dapat membaca detail tugas dan tanggung jawabnya pada pekerjaan sebelumnya.
- Kasus Pencarian dengan Typo (Fuzzy Match): Selanjutnya, Rina ingin mencari kandidat dengan keahlian spesifik dalam "kompensasi" (compensation), tetapi ia tidak sengaja salah mengetik menjadi kompansasi.
- Kecerdasan Fuzzy Match: Rina memasukkan "compansation" sebagai kata kunci dan menekan "Search". Karena tidak ada CV yang mengandung kata "compansation", pencarian exact tidak menghasilkan apa-apa. Sistem secara otomatis memicu fuzzy search dan menemukan beberapa kandidat yang memiliki kata "compensation" (ejaan yang benar) karena kemiripan katanya. Pada kartu hasil yang muncul, akan terdapat label seperti "compensation (~) : 1", yang memberitahu Rina bahwa hasil ini ditemukan melalui pencocokan fuzzy, membantunya menemukan kandidat yang relevan meskipun terjadi kesalahan pengetikan.

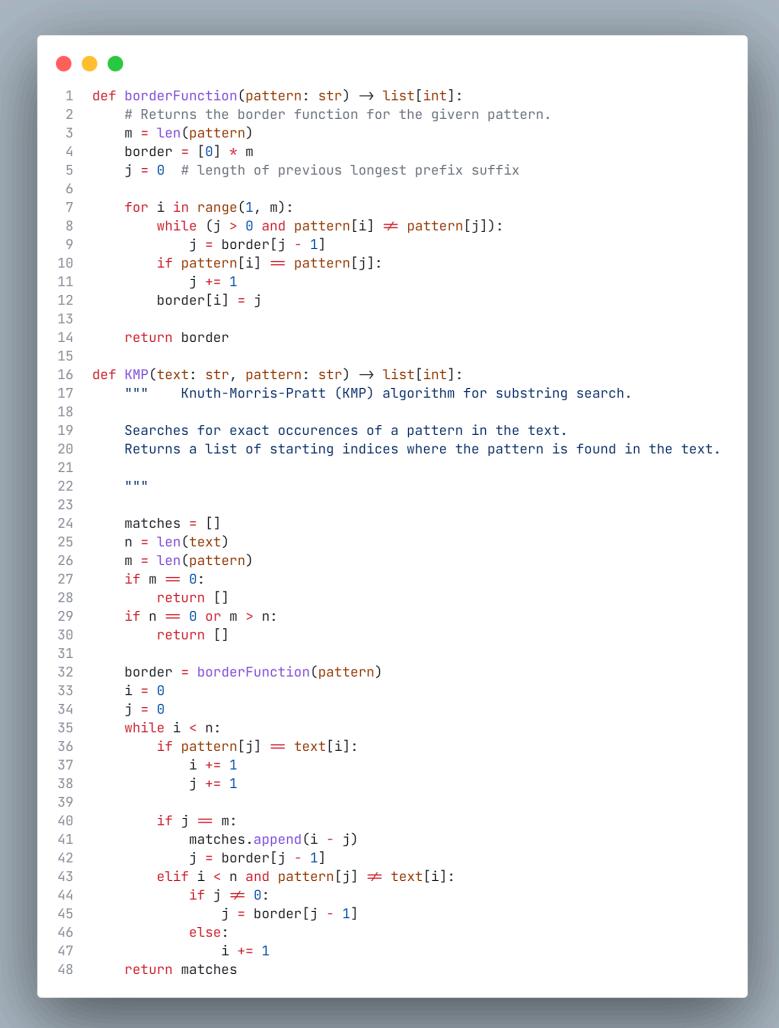
## IV. Implementasi dan Pengujian

### 4.1. Struktur File Program



```
    └── input.py
    └── result.py
    └── search_page.py
    └── summary_page.py
    └── wrap.py
    └── lib/
        └── aho_corasick.py
        └── bm.py
        └── kmp.py
        └── levenshtein.py
        └── regex.py
    └── models/
        └── search.py
    └── util/
        └── parser.py
    └── main_window.py
    └── main.py
    └── doc/
        └── HRDBawel.pdf
```

## 4.2. Modul Utama dan Fungsinya

| Screenshot   | Keterangan                 |
|--|----------------------------|
|  | Implementasi Algoritma KMP |

## Implementasi Algoritma BM

```
1 def last_occurrence(pattern: str) -> dict:
2     """
3         Membuat tabel 'bad character (last_occurrence)' menggunakan dictionary untuk mendukung semua karakter (Unicode).
4     """
5     last_char_table = {}
6     pattern_length = len(pattern)
7     for i in range(pattern_length):
8         last_char_table[pattern[i]] = i
9     return last_char_table
10
11 def BM(text: str, pattern: str) -> list[int]:
12     """
13         Fungsi pencarian string dengan algoritma Boyer-Moore yang sudah diperbaiki
14         untuk menangani semua jenis karakter (Unicode) dan mengembalikan semua indeks kemunculan.
15     """
16     n = len(text)
17     m = len(pattern)
18     matches = []
19
20     if m == 0 or n == 0 or m > n:
21         return matches
22
23     bad_char_table = last_occurrence(pattern)
24     shift = 0
25
26     while shift <= n - m:
27         j = m - 1
28
29         while j >= 0 and pattern[j] == text[shift + j]:
30             j -= 1
31
32         if j < 0:
33             matches.append(shift)
34             if shift + m < n:
35                 shift += m - bad_char_table.get(text[shift + m], -1)
36             else:
37                 shift += 1
38
39         else:
40             # Pattern tidak cocok, lakukan pergeseran berdasarkan 'bad character rule'
41             current_char_in_text = text[shift + j]
42             last_occurrence = bad_char_table.get(current_char_in_text, -1)
43             shift += max(1, j - last_occurrence)
44
45     return matches
```

## Implementasi Algoritma Levenshtein

```
1 def levenshtein_distance(s1: str, s2: str) -> int:
2     """
3         Menghitung Levenshtein Distance antara dua string.
4     """
5     Jarak ini adalah jumlah minimum operasi edit (penyisipan,
6     penghapusan, atau substitusi) yang dibutuhkan untuk mengubah
7     string s1 menjadi s2.
8     """
9     m, n = len(s1), len(s2)
10
11     # Inisialisasi matriks DP (Dynamic Programming)
12     # dp[i][j] akan menjadi jarak antara i karakter pertama s1
13     # dan j karakter pertama s2.
14     dp = [[0] * (n + 1) for _ in range(m + 1)]
15
16     # Inisialisasi baris dan kolom pertama
17     # Jarak dari string kosong ke string lain adalah panjang string itu sendiri
18     for i in range(m + 1):
19         dp[i][0] = i
20     for j in range(n + 1):
21         dp[0][j] = j
22
23     # Mengisi matriks DP
24     for i in range(1, m + 1):
25         for j in range(1, n + 1):
26             cost = 0 if s1[i - 1] == s2[j - 1] else 1 # Biaya substitusi
27             dp[i][j] = min(dp[i - 1][j] + 1,           # Deletion
28                            dp[i][j - 1] + 1,          # Insertion
29                            dp[i - 1][j - 1] + cost) # Substitution
30
31     return dp[m][n]
```

## Implementasi Regex untuk mencari summary

```
1 # Mengembalikan string panjang, "" jika tidak memiliki summary (ada sekitar 70/600 cv yang gapunya summary)
2 def extractSummary(pathfile, keywordsSummary=keywordsSumSection, keywordsSection=keywordsSection):
3     text = pdfToString(pathfile)
4     patternSummary = re.compile(createRegex(keywordsSummary))
5     patternSection = re.compile(createRegex(keywordsSection))
6
7     startIdx = -1
8     endIdx = -1
9     inSection = False
10
11    lines = text.split('\n')
12    for i, line in enumerate(lines):
13        if patternSummary.match(line) and not inSection:
14            startIdx = i + 1
15            inSection = True
16        elif patternSection.match(line) and inSection:
17            endIdx = i - 1
18            break
19
20    result = " ".join(lines[startIdx:endIdx + 1])
21    return result
22
23 # Mengembalikan list[WorkExperienceEntry], harusnya tinggal di print setiap element dari listnya
24 def extractJob(pathfile, keywordsJob=keywordsJobSection, keywordsSection=keywordsSection):
25     text = pdfToString(pathfile)
26     patternJob = re.compile(createRegex(keywordsJob))
27     patternSection = re.compile(createRegex(keywordsSection))
28     patternTitle = r'^((?:\b[A-Z][a-zA-Z]*\b|\b(?:at|to|in|of|for)\b)|[\//,&C"\\-])\s+*$'
29     patternYear = r'\b(19\d{2})|20\d{2}\b'
30
31     jobs = []
32     jobsAsEntry = []
33     inSection = False
34
35     lines = text.split('\n')
36     for line in lines:
37         cleanedLine = line.strip().replace('\u200b', '')
38         if not cleanedLine:
39             continue
40
41         if inSection:
42             if patternSection.match(cleanedLine):
43                 break
44
45         isTitle = re.fullmatch(patternTitle, cleanedLine)
46         containYear = re.search(patternYear, cleanedLine)
47
48         if isTitle or containYear:
49             jobs.append(cleanedLine)
50             entry = WorkExperienceEntry(
51                 position=cleanedLine,
52                 company="",
53                 start_date="",
54                 end_date="",
55                 description="")
56         )
57         jobsAsEntry.append(entry)
58
59     elif patternJob.match(cleanedLine):
60         inSection = True
61
62     return jobsAsEntry
63
64 # Mengembalikan list[EducationEntry], harusnya tinggal di print setiap element dari listnya
65 def extractEdu(pathfile, keywordsEdu=keywordsEduSection, keywordsSection=keywordsSection):
66     text = pdfToString(pathfile)
67     patternEdu = re.compile(createRegex(keywordsEdu))
68     patternSection = re.compile(createRegex(keywordsSection))
69
70     edus = []
71     edusAsEntry = []
72     inSection = False
73
74     lines = text.split('\n')
75     for line in lines:
76         cleanedLine = line.strip().replace('\u200b', '')
77         if not cleanedLine:
78             continue
79
80         if inSection:
81             if patternSection.match(cleanedLine):
82                 break
83             edus.append(cleanedLine)
84             entry = EducationEntry(
85                 institution=cleanedLine,
86                 program="",
87                 start_date="",
88                 end_date="")
89         )
90         edusAsEntry.append(entry)
91
92     elif patternEdu.match(cleanedLine):
93         inSection = True
94
95     return edusAsEntry
96
97 # Mengembalikan list[str], harusnya tinggal di print setiap element dari listnya
98 def extractSkill(pathfile, keywordsSkill=keywordsSkillSection, keywordsSection=keywordsSection):
99     text = pdfToString(pathfile)
100    patternSkill = re.compile(createRegex(keywordsSkill))
101    patternSection = re.compile(createRegex(keywordsSection))
102
103    skills = []
104    inSection = False
105
106    lines = text.split('\n')
107    for line in lines:
108        cleanedLine = line.strip().replace('\u200b', '')
109        if not cleanedLine:
110            continue
```

## Implementasi Algoritma Aho Corasick

```
● ● ●
1 def build_trie(patterns: list[str]) → tuple[list[dict], list[list[str]]]:
2     """
3         Membangun struktur Trie dari daftar pola (patterns).
4         Struktur ini juga dikenal sebagai 'goto function' dalam Aho-Corasick.
5
6         Returns:
7             goto (list[dict]): Transisi state. goto[state][char] → next_state.
8             output (list[list[str]]): Daftar pola yang berakhir di setiap state.
9         """
10    # Inisialisasi root node (state 0)
11    goto = [{}: 0] # List of dictionaries, index is the state
12    output = [[]] # List of lists, index is the state
13    new_state = 0
14
15    for pattern in patterns:
16        state = 0
17        for char in pattern:
18            if char not in goto[state]:
19                new_state += 1
20                goto.append({})
21                output.append([])
22                goto[state][char] = new_state
23            state = goto[state][char]
24            output[state].append(pattern)
25
26    return goto, output
27
28 def build_failure_links(goto: list[dict], output: list[list[str]]) → list[int]:
29     """
30         Membangun 'failure links' untuk setiap state di Trie.
31         Failure link menunjuk ke state lain yang mewakili suffix terpanjang
32         dari state saat ini yang juga merupakan prefix dari pola lain.
33
34         Returns:
35             list[int]: Array failure, di mana failure[state] adalah state tujuan.
36         """
37     failure = [0] * len(goto)
38     queue = deque()
39
40     # Inisialisasi failure links untuk semua state di level 1
41     for char in goto[0]:
42         state = goto[0][char]
43         if state ≠ 0:
44             queue.append(state)
45
46     # Proses sisa state menggunakan Breadth-First Search (BFS)
47     while queue:
48         state = queue.popleft()
49         for char, next_state in goto[state].items():
50             queue.append(next_state)
51
52         # Tentukan failure link untuk next_state
53         f = failure[state]
54         while char not in goto[f] and f ≠ 0:
55             f = failure[f]
56
57         failure[next_state] = goto[f].get(char, 0)
58
59         # Gabungkan output dari failure link ke state saat ini
60         # Ini penting untuk menemukan pola yang merupakan suffix dari pola lain
61         # (misalnya, menemukan "he" saat kita menemukan "she")
62         output[next_state].extend(output[failure[next_state]])
63
64     return failure
65
66 def aho_corasick(text: str, patterns: list[str]) → dict[str, list[int]]:
67     """
68         Algoritma Aho-Corasick untuk mencari semua kemunculan dari beberapa pola
69         dalam sebuah teks secara efisien.
70
71         Args:
72             text (str): Teks untuk dicari.
73             patterns (list[str]): Daftar pola (kata kunci) yang ingin dicari.
74
75         Returns:
76             dict[str, list[int]]: Sebuah dictionary di mana key adalah pola yang
77             ditemukan dan value adalah daftar indeks awal kemunculannya.
78         """
79     if not patterns or not text:
80         return {}
81
82     goto, output = build_trie(patterns)
83     failure = build_failure_links(goto, output)
84
85     matches = {pattern: [] for pattern in patterns}
86     state = 0
87
88     for i, char in enumerate(text):
89         while char not in goto[state] and state ≠ 0:
90             state = failure[state]
91
92         state = goto[state].get(char, 0)
93
94         # Jika ada output di state ini, berarti ada pola yang cocok
95         if output[state]:
96             for pattern in output[state]:
97                 # Posisi akhir adalah i, posisi awal adalah i - len(pattern) + 1
98                 start_index = i - len(pattern) + 1
99                 matches[pattern].append(start_index)
100
101     return matches
```

## Struktur data yang digunakan

```
● ● ●
1 @dataclass
2 class ApplicantProfile:
3     applicant_id: int
4     first_name: str
5     last_name: str
6     date_of_birth: datetime
7     address: str
8     phone_number: str
9
10 @dataclass
11 class ApplicationDetail:
12     detail_id: int
13     applicant_id: int
14     application_role: str
15     cv_path: str
16
17 """ Search data """
18 class SearchAlgorithm(Enum):
19     KMP = "KMP"
20     BM = "BM"
21     AHO_CORASICK = "Aho-Corasick"
22
23 @dataclass
24 class ApplicantMatchData:
25     detail_id: int
26     name: str
27     match_count: int
28     matched_keywords: dict[str, int]
29     fuzzy_matched_keywords: dict[str, int] = None # For fuzzy search, if applicable
30
31 @dataclass
32 class SearchResult:
33     applicants: list[ApplicantMatchData]
34     cvs_scanned: int
35     runtime: float # In milliseconds
36     fuzzy_runtime: float = 0.0 # In milliseconds, for fuzzy search if applicable
37
38 @dataclass
39 classSearchParams:
40     keywords: list[str]
41     algorithm: SearchAlgorithm
42     top_matches: int
43
44 # Summary data
45 @dataclass
46 class EducationEntry:
47     institution: str
48     program: str
49     start_date: str
50     end_date: str
51
52 @dataclass
53 class WorkExperienceEntry:
54     position: str
55     company: str
56     start_date: str
57     end_date: str
58     description: str
59
60 @dataclass
61 class CVSummary:
62     name: str
63     birthdate: datetime
64     address: str
65     contacts: list[str] # Phone, emails, etc
66     description: str
67
68     skills: list[str]
69     education: list[EducationEntry]
70     work_experience: list[WorkExperienceEntry]
71
72 """CV Extraction data"""
73 @dataclass
74 class CVSummaryExtraction:
75     # Extracted at runtime
76     description: str
77     skills: list[str]
78     education: list[EducationEntry]
79     work_experience: list[WorkExperienceEntry]
```

## Parser file ekstensi .pdf

```
1 def pdf_to_string(pdf_path):
2     doc = fitz.open(pdf_path)
3     all_text = []
4     for page in doc:
5         text = page.get_text().replace("\n", " ").replace("\r", " ").lower()
6         all_text.append(text)
7     return " ".join(all_text)
8
9 def pdf_to_text(pdf_path, txt_output_path):
10    doc = fitz.open(pdf_path)
11    all_text = ""
12    for page in doc:
13        all_text += page.get_text().replace("\n", " ").replace("\r", " ").lower()
14    with open(txt_output_path, "w", encoding="utf-8") as f:
15        f.write(all_text)
16
17 def extract_all_pdffs(input_dir, output_dir):
18     os.makedirs(output_dir, exist_ok=True) # Pastikan folder output ada
19
20     for filename in os.listdir(input_dir):
21         if filename.lower().endswith(".pdf"):
22             input_path = os.path.join(input_dir, filename)
23             output_filename = os.path.splitext(filename)[0] + ".txt"
24             output_path = os.path.join(output_dir, output_filename)
25
26             pdf_to_text(input_path, output_path)
27
```

## """ SQL Queries """

```
CREATE_APPLICANT_PROFILE = '''
CREATE TABLE IF NOT EXISTS ApplicantProfile (
    applicant_id INT AUTO_INCREMENT NOT NULL
PRIMARY KEY,
    first_name VARCHAR(50) DEFAULT NULL,
    last_name VARCHAR(50) DEFAULT NULL,
    date_of_birth DATE DEFAULT NULL,
    address VARCHAR(255) DEFAULT NULL,
    phone_number VARCHAR(20) DEFAULT NULL
);
'''

CREATE_APPLICATION_DETAIL = '''
CREATE TABLE IF NOT EXISTS ApplicationDetail (
    detail_id INT AUTO_INCREMENT NOT NULL
PRIMARY KEY,
    applicant_id INT NOT NULL,
    application_role VARCHAR(100) DEFAULT NULL,
    cv_path TEXT,
    FOREIGN KEY (applicant_id) REFERENCES
ApplicantProfile(applicant_id) ON DELETE CASCADE
);
```

## Implementasi database

```

"""
INSERT_NEW_APPLICANT_PROFILE = """
    INSERT INTO ApplicantProfile (first_name,
last_name, date_of_birth, address, phone_number)
    VALUES (%s, %s, %s, %s, %s);
"""

INSERT_NEW_APPLICATION_DETAIL = """
    INSERT INTO ApplicationDetail (applicant_id,
application_role, cv_path)
    VALUES (%s, %s, %s);
"""

SELECT_CV_PATH = """
    SELECT cv_path FROM ApplicationDetail WHERE
detail_id = %s;
"""

SELECT_ALL_APPLICATION_DETAIL = """
    SELECT * FROM ApplicationDetail;
"""

SELECT_APPLICATION_DETAIL = """
    SELECT * FROM ApplicationDetail WHERE detail_id
= %s;
"""

SELECT_APPLICANT_PROFILE = """
    SELECT * FROM ApplicantProfile WHERE
applicant_id = %s;
"""

try:
    import mysql.connector
    MYSQL_AVAILABLE = True
except ImportError:
    print("MySQL connector not available. Database
features will be disabled.")
    MYSQL_AVAILABLE = False

```

```

class CVDatabase:
    def __init__(self):
        self.db_name = "ats_cv_hrdawel"
        self.connection = None

        if not MYSQL_AVAILABLE:
            print("MySQL not available, using
file-based mode")
            return

        # Check if MySQL port is open first
        print("Checking if MySQL server is
running...")
        if not self._is_mysql_running():
            print("MySQL server is not running or
not accessible on localhost:3306")
            return

        try:
            print("MySQL server detected.
Attempting connection...")
            self.connect()

            if self.connection is None:
                print("Failed to establish database
connection")
                return

            cursor = self.connection.cursor()
            cursor.execute("SHOW DATABASES LIKE
%s", (self.db_name,))
            result = cursor.fetchone()
            cursor.close()
            if result is None:
                print(f"Database {self.db_name}
does not exist. Initializing...")
                self.init_database()
            else:
                cursor = self.connection.cursor()
                cursor.execute(f"USE

```

```

{self.db_name}")

        print(f"Database '{self.db_name}'"
ready.")

    except Exception as err:
        print(f"Database initialization error:
{err}")
        print("Continuing in file-based
mode...")
        self.connection = None
    finally:
        if cursor:
            cursor.close()

def _is_mysql_running(self):
    """Check if MySQL server is running on
localhost:3306"""
    try:
        sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        sock.settimeout(2) # 2 second timeout
        result = sock.connect_ex(('localhost',
3306))
        sock.close()
        return result == 0
    except Exception as e:
        print(f"Error checking MySQL port:
{e}")
        return False

def init_database(self):
    Faker.seed(42) # For reproducibility

    cursor = self.connection.cursor()
    cursor.execute(f"CREATE DATABASE
{self.db_name};")
    cursor.execute(f"USE {self.db_name};")
    cursor.execute(CREATE_APPLICANT_PROFILE)
    cursor.execute(CREATE_APPLICATION_DETAIL)
    self.connection.commit()

```

```

        cursor.close()

        data_cursor = self.connection.cursor()
        faker = Faker('id_ID')
        # Insert dummy data ApplocantProfile
        for _ in range(20):

            first_name = faker.first_name()
            last_name = faker.last_name()
            date_of_birth =
faker.date_of_birth(minimum_age=18, maximum_age=60)
            address = faker.address().replace('\n',
', ')
            phone_number = faker.phone_number()

data_cursor.execute(INSERT_NEW_APPLICANT_PROFILE,
                   (first_name, last_name,
date_of_birth, address, phone_number)
)
            status = data_cursor.rowcount
            if status > 0:
                print(f"Inserted dummy applicant:
{first_name} {last_name}, DOB: {date_of_birth},
Address: {address}, Phone: {phone_number}")
            self.connection.commit()
            data_cursor.close()

    def seed_database(self,
relative_data_directory, role=""):
        import random
        random.seed(42) # For reproducibility
        if role == "":
            role = "Unknown"
        cursor = self.connection.cursor()
        applicant_ids =
self.get_all_applicant_profiles_id()
        for file in
os.listdir(os.path.join(relative_data_directory)):
            if file.endswith('.pdf'):
                cv_path =

```

```

os.path.join(relative_data_directory, file)
    applicant_id =
random.choice(applicant_ids)
        print(f"Seeding application detail
for file: {file} with applicant_id: {applicant_id}
and role: {role}")

cursor.execute(INSERT_NEW_APPLICATION_DETAIL,
(applicant_id, role, cv_path))
    self.connection.commit()
    cursor.close()

def _connect_with_timeout(self):
    """Internal method to create MySQL
connection with proper timeout handling"""
    try:
        connection = mysql.connector.connect(
            host='localhost',
            user='root',
            password='admin',
            connect_timeout=3,    # Reduced
timeout
            autocommit=True,
            use_pure=True,    # Force pure Python
implementation
            sql_mode='',
            charset='utf8mb4',
            collation='utf8mb4_unicode_ci'
        )
        return connection
    except Exception as e:
        raise e

def connect(self):
    if not MYSQL_AVAILABLE:
        return

    try:
        print("Attempting MySQL connection
with:")
        print("- Host: localhost")

```

```

        print("- User: root")
        print("- Password: (empty) ")
        print("- Timeout: 3 seconds")
        print("- Using pure Python
implementation")

        # Use ThreadPoolExecutor to enforce
timeout
        with ThreadPoolExecutor(max_workers=1)
as executor:
            future =
        executor.submit(self._connect_with_timeout)
            try:
                self.connection =
        future.result(timeout=5) # 5 second total timeout
                print("✓ MySQL connection
successful")
            except FutureTimeoutError:
                print("✗ MySQL connection
timed out after 5 seconds")
                self.connection = None
            return
        except Exception as e:
            raise e

        except mysql.connector.Error as err:
            print(f"✗ MySQL connection failed:
{err}")
            if err(errno ==
mysql.connector.errorcode.ER_ACCESS_DENIED_ERROR:
                print(" → Check
username/password")
            elif err(errno ==
mysql.connector.errorcode.ER_BAD_DB_ERROR:
                print(" → Database does not
exist")
            else:
                print(f" → Error code:
{err(errno}")
            self.connection = None

```

```

        except Exception as err:
            print(f"X Unexpected error during
MySQL connection: {err}")
            self.connection = None

    def close(self):
        if self.connection and
self.connection.is_connected():
            self.connection.close()
            print("Database connection closed.")

    def get_cv_path(self, detail_id: int) -> str | None:
        if not self.connection:
            print("Database connection is not
established.")
            return None
        cursor = self.connection.cursor()
        cursor = self.connection.cursor()
        cursor.execute(SELECT_CV_PATH,
(detail_id,))
        result = cursor.fetchone()
        return result[0] if result else None

    def get_all_application_details(self) ->
list[ApplicationDetail]:
        if not self.connection:
            print("Database connection is not
established.")
            return []
        cursor = self.connection.cursor()

        cursor.execute(SELECT_ALL_APPLICATION_DETAIL)
        results = cursor.fetchall()
        application_details = []
        for row in results:

            application_details.append(ApplicationDetail(
                detail_id=row[0],
                applicant_id=row[1],
                application_role=row[2],

```

```

        cv_path=row[3]
    ))
    return application_details

    def get_application_detail(self, detail_id: int) -> ApplicationDetail | None:
        if not self.connection:
            print("Database connection is not established.")
            return None
        cursor = self.connection.cursor()
        cursor.execute(SELECT_APPLICATION_DETAIL,
        (detail_id,))
        result = cursor.fetchone()
        if result:
            return ApplicationDetail(
                detail_id=result[0],
                applicant_id=result[1],
                application_role=result[2],
                cv_path=result[3]
            )
        return None

    def get_applicant_profile(self, applicant_id: int) -> ApplicantProfile | None:
        if not self.connection:
            print("Database connection is not established.")
            return None
        cursor = self.connection.cursor()
        cursor.execute(SELECT_APPLICANT_PROFILE,
        (applicant_id,))
        result = cursor.fetchone()
        if result:
            return ApplicantProfile(
                applicant_id=result[0],
                first_name=result[1],
                last_name=result[2],
                date_of_birth=result[3],
                address=result[4],
                phone_number=result[5]
            )

```

```

        )
    return None

    def get_all_applicant_profiles_id(self) ->
list[int]:
        if not self.connection:
            print("Database connection is not
established.")
            return []
        cursor = self.connection.cursor()
        cursor.execute("SELECT applicant_id FROM
ApplicantProfile")
        results = cursor.fetchall()
        cursor.close()
        return [row[0] for row in results if row[0]
is not None]

```

```

from PyQt6.QtWidgets import QMainWindow,
QStackedWidget
from PyQt6.QtGui import QDesktopServices
from PyQt6.QtCore import QUrl
import time
import os
import re
from datetime import datetime
from gui.search_page import SearchPage
from gui.summary_page import SummaryPage
from models.search importSearchParams,
ApplicantMatchData, ApplicationDetail,
SearchResult, SearchAlgorithm, WorkExperienceEntry,
EducationEntry, CVSummary

from lib.kmp import KMP
from lib.bm import BM
from lib.aho_corasick import aho_corasick
from lib.levenshtein import levenshtein_distance
from lib.regex import extractEdu, extractJob,
extractSkill, extractSummary
from database.cv_database import CVDatabase
from util.parser import pdf_to_string

```

Implementasi GUI main window

```

algorithm_map = {
    SearchAlgorithm.KMP: KMP,
    SearchAlgorithm.BM: BM,
    SearchAlgorithm.AHO_CORASICK: aho_corasick,
}

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Application Tracking
System")
        self.setFixedSize(640, 400)

        self.stack = QStackedWidget()
        self.setCentralWidget(self.stack)

        # Initialize database with error handling
        try:
            print("Initializing database...")
            self.db = CVDatabase()
            print("Database initialized
successfully")
        except Exception as e:
            print(f"Database initialization failed:
{e}")
            self.db = None

        print("Creating search page...")
        self.search_page = SearchPage()
        print("Creating summary page...")
        self.summary_page = SummaryPage()

        self.stack.addWidget(self.search_page)
        self.stack.addWidget(self.summary_page)

        # Connect signals
        print("Connecting signals...")

self.search_page.search_initiate.connect(self.searc
h)

```

```

self.search_page.view_summary.connect(self.summary)

self.search_page.view_cv.connect(self.view_cv)

self.summary_page.return_from_summary.connect(self.
return_to_search)

    def show_search_page(self):
        """Method untuk kembali ke halaman
search"""

self.stack.setCurrentWidget(self.search_page)

def search(self, search_params: SearchParams):
    search_results =
    SearchResult(applicants=[], cvs_scanned=0,
runtime=0)
    applications =
    self.db.get_all_application_details()

        # Compute a mapping of detail_id to
ApplicantMatchData
        app_matches =
    self.exact_search(search_params, search_results,
applications)

    each_keyword_found = False
    for keyword in search_params.keywords:
        for match in app_matches.values():
            if keyword in
match.matched_keywords:
                each_keyword_found = True
                break

            if not each_keyword_found or
len(search_results.applicants) <
search_params.top_matches:
                # Compute additional matches using
fuzzy search

```

```

        self.fuzzy_search(search_params,
search_results, applications, app_matches)

        # Aggregate app_matches into SearchResult
        for detail_id, match_data in
app_matches.items():
            applicant_id = -1
            for app in applications:
                if app.detail_id == detail_id:
                    applicant_id = app.applicant_id
                    break
            profile =
self.db.get_applicant_profile(applicant_id)
            if profile:
                match_data.name =
f"{profile.first_name} {profile.last_name}"

            search_results.applicants.append(match_data)

            # Sort the results by match count in
descending order
            search_results.applicants.sort(key=lambda
x: x.match_count, reverse=True)

self.search_page.show_results(search_results)

    def exact_search(self, search_params:
SearchParams, search_results: SearchResult,
applications: list[ApplicationDetail]) -> dict[int,
ApplicantMatchData]:
        print(f"Performing exact search with
parameters: {search_params}")

        search_function =
algorithm_map.get(search_params.algorithm)

        final_results: dict[int,
ApplicantMatchData] = {}
        start_time = time.time()
        for app in applications:
            cv_text =

```

```

pdf_to_string(app.cv_path).lower() if app.cv_path
else ""

    exact_matches: dict[str, int] = {}
        if search_params.algorithm ==

SearchAlgorithm.AHO_CORASICK:
            matches = aho_corasick(cv_text,
search_params.keywords)

            exact_matches = {k: len(v) for k, v
in matches.items() if v}
        else:
            for keyword in
search_params.keywords:
                keyword_lower =
keyword.strip().lower()
                if not keyword_lower: continue

                occurrences =
search_function(cv_text, keyword_lower)
                if occurrences:
                    exact_matches[keyword] =
len(occurrences)

                if exact_matches:
                    final_results[app.detail_id] =

ApplicantMatchData(
            detail_id=app.detail_id,
            name="", # Filled later
            match_count =
sum(exact_matches.values()),
            matched_keywords=exact_matches,
        )
        end_time = time.time()

        search_results.cvs_scanned =
len(applications)
        search_results.runtime = (end_time -
start_time) * 1000
        return final_results

    def fuzzy_search(self, search_params:
SearchParams, search_results: SearchResult,

```

```

applications: list[ApplicationDetail],
previous_matches: dict[int, ApplicantMatchData]) ->
None:
    print(f"Performing fuzzy search with
parameters: {search_params}")
    SIMILARITY_THRESHOLD = 80.0

    start_time = time.time()
    for app in applications:
        cv_text =
pdf_to_string(app.cv_path).lower() if app.cv_path
else ""
        fuzzy_matches: dict[str, int] = {}
        for keyword in search_params.keywords:
            keyword_lower =
keyword.strip().lower()
            if not keyword_lower: continue

            words_in_cv =
set(re.findall(r'[a-z]+', cv_text))
            for word_in_cv in words_in_cv:
                len_max =
max(len(keyword_lower), len(word_in_cv))
                if len_max > 0:
                    distance =
levenshtein_distance(keyword_lower, word_in_cv)
                    similarity = (1 - (distance
/ len_max)) * 100
                    if SIMILARITY_THRESHOLD <=
similarity < 100:
                        fuzzy_key =
f"{word_in_cv} (~)"

fuzzy_matches[fuzzy_key] =
fuzzy_matches.get(fuzzy_key, 0) + 1

            if fuzzy_matches:
                detail_id = app.detail_id
                if detail_id in previous_matches:
                    match_data =
previous_matches[detail_id]

```

```

        match_data.match_count +=
sum(fuzzy_matches.values())

match_data.fuzzy_matched_keywords = fuzzy_matches
    else:
        match_data =
ApplicantMatchData(
            detail_id=detail_id,
            name="", # Filled later

match_count=sum(fuzzy_matches.values()),
            matched_keywords={},

fuzzy_matched_keywords=fuzzy_matches
)
previous_matches[detail_id] =
match_data
end_time = time.time()

search_results.cvs_scanned =
len(applications)
search_results.fuzzy_runtime = (end_time -
start_time) * 1000

def view_cv(self, detail_id: str):
"""
    Finds the PDF file path from the database
using the detail_id
    and opens it with the default system PDF
viewer.
"""
    print(f"Attempting to view CV for applicant
ID: {detail_id}")

    # Get the relative path of the PDF from the
database
    pdf_relative_path =
self.db.get_cv_path(detail_id)

    if pdf_relative_path:
        # Construct the absolute path. This

```

```

assumes the script is run from the project root.

    # A more robust way might be needed if
    # the execution path changes.

    base_dir =
os.path.abspath(os.path.dirname(__file__) + "/..")
    pdf_full_path = os.path.join(base_dir,
pdf_relative_path)

    print(f"Found PDF at: {pdf_full_path}")

    if os.path.exists(pdf_full_path):
        # Open the PDF file using the
        # system's default application
        url =
QUrl.fromLocalFile(pdf_full_path)
        QDesktopServices.openUrl(url)
    else:
        print(f"Error: PDF file not found
at path: {pdf_full_path}")
    else:
        print(f"Error: Could not find a CV path
for ID: {detail_id}")

    def summary(self, detail_id: id):
        app_detail =
self.db.get_application_detail(detail_id)
        app_profile =
self.db.get_applicant_profile(app_detail.applicant_
id)
        cv_summary = CVSummary(
            name = app_profile.first_name + " " +
app_profile.last_name,
            birthdate = app_profile.date_of_birth,
            address = app_profile.address,
            contacts = [app_profile.phone_number],
            description = "",
            skills = [],
            education = [],
            work_experience = []
        )

```

```

        cv_summary.description =
extractSummary(app_detail.cv_path)
        cv_summary.skills =
extractSkill(app_detail.cv_path)
        cv_summary.education =
extractEdu(app_detail.cv_path)
        cv_summary.work_experience =
extractJob(app_detail.cv_path)

        self.summary_page.set_summary(detail_id,
cv_summary)

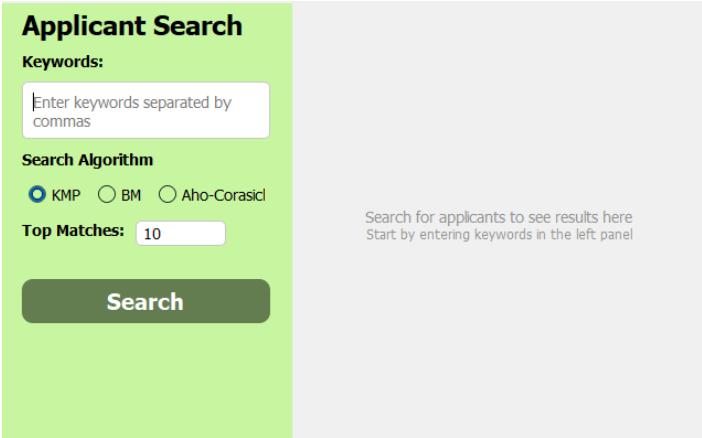
self.stack.setCurrentWidget(self.summary_page)
    # Extracted CV summary

def return_to_search(self):

self.stack.setCurrentWidget(self.search_page)
    self.search_page.setFocus()

```

#### 4.3. Tampilan Aplikasi

| Gambar  | Keterangan    |
|---|---------------|
|  | Halaman utama |

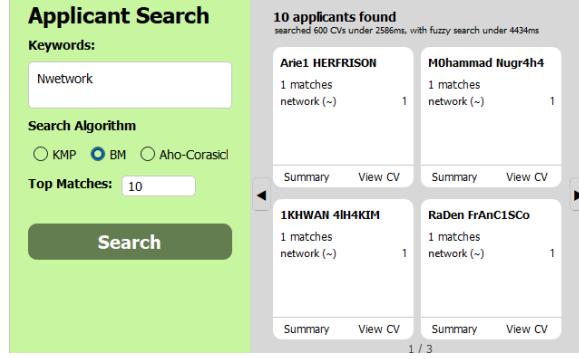
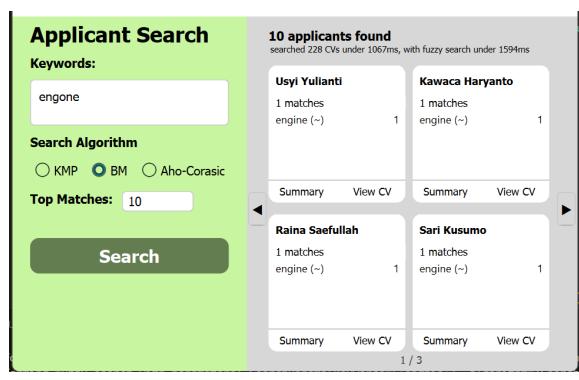
|   |   |  |  |                         |                         |                         |                         |  |  |                         |                         |                         |                         |                         |                         |                   |
|---|---|--|--|-------------------------|-------------------------|-------------------------|-------------------------|--|--|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------|
| <h2>Applicant Search</h2> <p><b>Keywords:</b></p> <input type="text" value="SQL"/> <p><b>Search Algorithm</b></p> <p><input type="radio"/> KMP <input checked="" type="radio"/> BM <input type="radio"/> Aho-Corasick</p> <p><b>Top Matches:</b> 10</p> <p><b>Search</b></p>  | <p><b>10 applicants found</b><br/>searched 600 CVs under 2781ms</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tbody> <tr> <td style="width: 50%;"> <b>RaDen FrAnC1SCo</b><br/>           14 matches<br/>           SQL 14         </td> <td style="width: 50%;"> <b>RaDen FrAnC1SCo</b><br/>           14 matches<br/>           SQL 14         </td> </tr> <tr> <td style="text-align: center;"><a href="#">Summary</a></td> <td style="text-align: center;"><a href="#">View CV</a></td> <td style="text-align: center;"><a href="#">Summary</a></td> <td style="text-align: center;"><a href="#">View CV</a></td> </tr> <tr> <td style="text-align: center;"><b>RaDen FrAnC1SCo</b><br/>           14 matches<br/>           SQL 14</td> <td style="text-align: center;"><b>RaDen FrAnC1SCo</b><br/>           14 matches<br/>           SQL 14</td> <td style="text-align: center;"><a href="#">Summary</a></td> <td style="text-align: center;"><a href="#">View CV</a></td> </tr> <tr> <td style="text-align: center;"><a href="#">Summary</a></td> <td style="text-align: center;"><a href="#">View CV</a></td> <td style="text-align: center;"><a href="#">Summary</a></td> <td style="text-align: center;"><a href="#">View CV</a></td> </tr> </tbody> </table> <p style="text-align: center;">1 / 3</p> | <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14 | <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14 | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> | <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14 | <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14 | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> | Halaman pencarian |
| <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14  | <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14  |  |  |                         |                         |                         |                         |  |  |                         |                         |                         |                         |                         |                         |                   |
| <a href="#">Summary</a>   | <a href="#">View CV</a>   | <a href="#">Summary</a>                        | <a href="#">View CV</a>                        |                         |                         |                         |                         |  |  |                         |                         |                         |                         |                         |                         |                   |
| <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14  | <b>RaDen FrAnC1SCo</b><br>14 matches<br>SQL 14  | <a href="#">Summary</a>                        | <a href="#">View CV</a>                        |                         |                         |                         |                         |  |  |                         |                         |                         |                         |                         |                         |                   |
| <a href="#">Summary</a>   | <a href="#">View CV</a>   | <a href="#">Summary</a>                        | <a href="#">View CV</a>                        |                         |                         |                         |                         |  |  |                         |                         |                         |                         |                         |                         |                   |
| <p><b>RaDen FrAnC1SCo</b></p> <p>Date of Birth: 09 March 2003<br/>           Address: Jl. Pisang No. 7, Sibolga<br/>           Contact: 082199998877</p> <p>SUMMARY Applying technology and workflow solutions to business challenges is exciting for me because I love to learn and apply new lessons and approaches to support and enhance the organization to achieve its goals and mission.</p> <p><b>Skills</b></p> <ul style="list-style-type: none"> <li>C# Oracle SQL Oracle</li> <li>Information Management SQL Server SQL*Plus</li> <li>TFS SharePoint SharePoint Designer</li> <li>Database Design Database Administration Data M</li> <li>ASP.NET Team Leadership IIS</li> <li>PL/SQL ADO.NET Tactical Planning</li> <li>Application Development Web Applications IT St</li> <li>Microsoft SQL Server Software Development Agi</li> <li>Requirements Analysis SDLC XML</li> <li>Information Technology SQL Software Project M</li> <li>Project Management Analysis Business Intelligen</li> <li>Leadership</li> </ul> | <p><b>Job History</b></p> <p><b>Director of Information Technology</b><br/>           • 11/2012 to 08/2015 Company Name</p> <p>I had all the hardware and software refreshed as well as bringing the development group up to sql server, tfs, and VS 2012.</p> <p><b>Team Leader</b> 05/2005 to 11/2012<br/>           • Company Name</p> <p>Chief Information Officer 07/2000 to 02/2005 Company Name</p> <p><b>Director of Applications Development</b><br/>           • 02/1996 to 06/2000 Company Name</p> <p>I lead the Year 2000.</p> <p><b>Education</b></p> <p><b>Education</b></p> <p>Master of Business Administration (MBA) : Management Information Systems,<br/>           • General Delta Mu Delta Suffolk University - Sawyer School of</p> <p>Management City Management Information Systems, General Delta Mu Delta</p> <p>Bachelor of Arts (BA) : Economics GPA: Omicron Delta Epsilon Economics Omicron Delta Epsilon</p>  | Halaman summary                                |  |                         |                         |                         |                         |  |  |                         |                         |                         |                         |                         |                         |                   |

#### 4.4. Cara Menjalankan Aplikasi

1. uv run src/main.py -seed data/{folder}
2. uv run src/main.py
3. mysql -u root -p ats\_cv\_hrdbawel < tubes3\_seeding.sql  
(Opsional untuk inject seeding)

#### 4.5. Hasil Pengujian

| No   | Pencarian  | Hasil   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
|--|--|---|---|--|-------------------------|-------------------------|-------------------------|-------------------------|--|---|-------------------------|-------------------------|-------------------------|-------------------------|
| 1.   | ClienT seRVer mOdeL apPLicatiOnS   | <p><b>Applicant Search</b><br/>Keywords: ClienT seRVer mOdeL apPLicatiOnS</p> <p>Search Algorithm<br/> <input type="radio"/> KMP   <input checked="" type="radio"/> BM   <input type="radio"/> Aho-Corasick</p> <p>Top Matches: 10</p> <p><b>Search</b></p> <p><b>4 applicants found</b><br/>searched 600 CVs under 2572ms</p> <table border="1"> <tr> <td>RaDen FrAnC1SCo<br/>1 matches<br/>ClienT seRVer ...</td> <td>RaDen FrAnC1SCo<br/>1 matches<br/>ClienT seRVer ...</td> </tr> <tr> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> </tr> </table> <table border="1"> <tr> <td>RaDen FrAnC1SCo<br/>1 matches<br/>ClienT seRVer ...</td> <td>RaDen FrAnC1SCo<br/>1 matches<br/>ClienT seRVer ...</td> </tr> <tr> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> </tr> </table> <p>1 / 1</p>   | RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ... | RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...                                    | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> | RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...                                  | RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...           | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> |
| RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...                                  | RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...                                    |   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| <a href="#">Summary</a>  | <a href="#">View CV</a>  | <a href="#">Summary</a>   | <a href="#">View CV</a>                           |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...                                  | RaDen FrAnC1SCo<br>1 matches<br>ClienT seRVer ...                                    |   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| <a href="#">Summary</a>  | <a href="#">View CV</a>  | <a href="#">Summary</a>   | <a href="#">View CV</a>                           |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| 2.   | SQL  | <p><b>Applicant Search</b><br/>Keywords: SQL</p> <p>Search Algorithm<br/> <input type="radio"/> KMP   <input checked="" type="radio"/> BM   <input type="radio"/> Aho-Corasick</p> <p>Top Matches: 10</p> <p><b>Search</b></p> <p><b>10 applicants found</b><br/>searched 600 CVs under 2725ms</p> <table border="1"> <tr> <td>RaDen FrAnC1SCo<br/>14 matches<br/>SQL</td> <td>RaDen FrAnC1SCo<br/>14 matches<br/>SQL</td> </tr> <tr> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> </tr> </table> <table border="1"> <tr> <td>RaDen FrAnC1SCo<br/>14 matches<br/>SQL</td> <td>RaDen FrAnC1SCo<br/>14 matches<br/>SQL</td> </tr> <tr> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> </tr> </table> <p>1 / 3</p>   | RaDen FrAnC1SCo<br>14 matches<br>SQL              | RaDen FrAnC1SCo<br>14 matches<br>SQL   | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> | RaDen FrAnC1SCo<br>14 matches<br>SQL   | RaDen FrAnC1SCo<br>14 matches<br>SQL                        | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> |
| RaDen FrAnC1SCo<br>14 matches<br>SQL   | RaDen FrAnC1SCo<br>14 matches<br>SQL   |   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| <a href="#">Summary</a>  | <a href="#">View CV</a>  | <a href="#">Summary</a>   | <a href="#">View CV</a>                           |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| RaDen FrAnC1SCo<br>14 matches<br>SQL   | RaDen FrAnC1SCo<br>14 matches<br>SQL   |   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| <a href="#">Summary</a>  | <a href="#">View CV</a>  | <a href="#">Summary</a>   | <a href="#">View CV</a>                           |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| 3.   | recruitmen, payroll, employee relations  | <p><b>Applicant Search</b><br/>Keywords: recruitmen, payroll, employee relations</p> <p>Search Algorithm<br/> <input type="radio"/> KMP   <input type="radio"/> BM   <input checked="" type="radio"/> Aho-Corasick</p> <p>Top Matches: 5</p> <p><b>Search</b></p> <p><b>5 applicants found</b><br/>searched 110 CVs under 633ms</p> <table border="1"> <tr> <td>Warsita Prasasta<br/>29 matches<br/>payroll</td> <td>Warsita Prasasta<br/>24 matches<br/>recruitmen 16<br/>payroll 3<br/>employee relations 5</td> </tr> <tr> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> </tr> </table> <table border="1"> <tr> <td>Jinawi Maulana<br/>23 matches<br/>recruitmen 4<br/>payroll 17<br/>employee relations 2</td> <td>Galuh Anggraini<br/>21 matches<br/>recruitmen 7<br/>payroll 14</td> </tr> <tr> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> <td><a href="#">Summary</a></td><td><a href="#">View CV</a></td> </tr> </table> <p>1 / 2</p> | Warsita Prasasta<br>29 matches<br>payroll         | Warsita Prasasta<br>24 matches<br>recruitmen 16<br>payroll 3<br>employee relations 5 | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> | Jinawi Maulana<br>23 matches<br>recruitmen 4<br>payroll 17<br>employee relations 2 | Galuh Anggraini<br>21 matches<br>recruitmen 7<br>payroll 14 | <a href="#">Summary</a> | <a href="#">View CV</a> | <a href="#">Summary</a> | <a href="#">View CV</a> |
| Warsita Prasasta<br>29 matches<br>payroll  | Warsita Prasasta<br>24 matches<br>recruitmen 16<br>payroll 3<br>employee relations 5 |   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| <a href="#">Summary</a>  | <a href="#">View CV</a>  | <a href="#">Summary</a>   | <a href="#">View CV</a>                           |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| Jinawi Maulana<br>23 matches<br>recruitmen 4<br>payroll 17<br>employee relations 2 | Galuh Anggraini<br>21 matches<br>recruitmen 7<br>payroll 14                          |   |   |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |
| <a href="#">Summary</a>  | <a href="#">View CV</a>  | <a href="#">Summary</a>   | <a href="#">View CV</a>                           |  |                         |                         |                         |                         |  |   |                         |                         |                         |                         |

|    |          |   |
|----|----------|---|
| 4. | Nwetwork |  <p>The screenshot shows the 'Applicant Search' interface. The search bar contains 'Nwetwork'. The search algorithm selected is BM. The top matches are displayed, and the 'Search' button is visible. To the right, a grid of search results is shown with columns for name, matches, and network (~). The results include Arie1 HERFRISON, Mohammad Nugr4h4, IKHWAN 4IH4KIM, and RaDen FrAnC1SCO.</p> |
| 5. | engone   |  <p>The screenshot shows the 'Applicant Search' interface. The search bar contains 'engone'. The search algorithm selected is BM. The top matches are displayed, and the 'Search' button is visible. To the right, a grid of search results is shown with columns for name, matches, and engine (~). The results include Usyi Yulianti, Kawaca Haryanto, Raina Saefullah, and Sari Kusumo.</p>          |

#### 4.6 Analisis Hasil Pengujian

Dalam pengujian aplikasi ini, dilakukan lima skenario pencarian dengan pendekatan algoritma yang berbeda: Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Aho-Corasick (AC).

Pengujian pertama menggunakan KMP dengan query "Client seRVer mOdeL apPLicatiOnS". Algoritma ini mampu menemukan hasil meskipun terdapat perbedaan kapitalisasi karena sistem telah normalisasi input dan teks (lowercase). KMP menunjukkan efisiensi dalam pola panjang tunggal karena mampu memanfaatkan prefix table tanpa perlu melakukan backtracking. Hasil pencarian menunjukkan bahwa keywords yang digunakan mampu secara exact dicocokkan dengan yang ada di dalam String Teks pada setiap CV.

Pengujian kedua hingga kelima menggunakan kombinasi Boyer-Moore dan Aho-Corasick. Pencarian "SQL" berhasil cepat dengan BM karena kata yang pendek dan spesifik sangat cocok dengan strategi lompatan Boyer-Moore dan karena variasi karakter yang beragam sehingga algoritma ini efisien untuk digunakan. Pencarian "recruitmen, payroll, employee relations" menggunakan Aho-Corasick efektif karena bisa mencari banyak kata sekaligus dalam satu kali traversal teks, membuktikan keunggulannya untuk multi-pattern search. Untuk kata "Nwetwork" dan "engone" yang mengandung typo, pencarian tetap berhasil berkat fuzzy search yang diimplementasikan dengan algoritma levenshtein distance. Ini menunjukkan bahwa pencarian tetap mampu bekerja asal typo-nya tidak terlalu "jauh" dari isi data sebenarnya.

## **V. Penutup**

### **5.1. Kesimpulan**

Dalam tugas besar ini, telah dibangun sebuah sistem pencarian informasi berbasis teks untuk mengekstrak dan menganalisis konten dari berkas Curriculum Vitae (CV) dalam format PDF. Proses dimulai dengan ekstraksi teks mentah dari file PDF, yang kemudian diproses menjadi bentuk string panjang yang seragam tanpa baris baru dan seluruh huruf dalam format huruf kecil agar sesuai dengan kebutuhan algoritma pencarian string. Dengan memanfaatkan pendekatan struktur umum CV seperti penggunaan header dan bullet points, sistem ini mampu mengenali dan memisahkan bagian-bagian penting seperti keterampilan (skills), pengalaman kerja, dan pendidikan secara otomatis, terutama dengan bantuan ekspresi reguler (regex) untuk mendeteksi pola-pola umum dalam teks CV.

Dalam tahap pencarian, sistem menggunakan algoritma-algoritma klasik seperti Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Aho-Corasick untuk melakukan pencocokan string secara efisien. Masing-masing algoritma memiliki kekuatan tersendiri tergantung pada konteks dan pola pencarian, sehingga pengguna dapat memilih algoritma yang paling sesuai dengan kebutuhan analisis. Gabungan antara teknik praproses teks yang cermat dan penggunaan algoritma pencarian yang optimal menjadikan sistem ini mampu bekerja secara cepat dan akurat dalam menemukan informasi relevan dari berbagai format dan variasi isi CV.

### **5.2. Saran**

Sebagai saran pengembangan, sistem ini dapat ditingkatkan dengan menambahkan mekanisme evaluasi akurasi hasil ekstraksi dan pencarian, misalnya dengan membandingkan hasil sistem terhadap anotasi manual sebagai ground truth. Selain itu, pendekatan berbasis pembelajaran statistik atau pemrosesan bahasa alami (NLP) dapat dipertimbangkan untuk mengenali konteks dan semantik dari isi CV, sehingga sistem menjadi lebih fleksibel dalam mengenali informasi penting meskipun tidak secara eksplisit menggunakan kata kunci yang telah ditentukan.

### **5.3. Refleksi**

Melalui penggeraan tugas besar ini, kami mendapatkan pemahaman yang lebih mendalam mengenai bagaimana algoritma pencocokan string seperti KMP, Boyer-Moore, Aho-Corasick, dan regex dapat diterapkan secara langsung pada kasus nyata, khususnya dalam pengolahan dan pencarian informasi pada dokumen teks seperti CV. Proses implementasi mulai dari ekstraksi data mentah hingga pencarian informasi relevan mengajarkan pentingnya tahap pra proses data serta pemilihan algoritma yang tepat sesuai kebutuhan. Selain itu, tantangan yang muncul dari variasi format CV juga memberikan wawasan bahwa fleksibilitas dan ketelitian dalam desain sistem sangat penting untuk mencapai hasil yang akurat dan efisien.

## Lampiran

Pranala repository Github : [https://github.com/Ferdin-Arsenic/Tubes3\\_HRDBawel.git](https://github.com/Ferdin-Arsenic/Tubes3_HRDBawel.git)

### Checklist Tugas Besar 3

| 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 dengan menggunakan Regular Expression (Regex).      | v  |       |
| 4. | Algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) dapat menemukan kata kunci dengan benar. | v  |       |
| 5. | Algoritma Levenshtein Distance dapat mengukur kemiripan kata kunci dengan benar.                 | v  |       |
| 6. | Aplikasi dapat menampilkan summary CV applicant.   | v  |       |
| 7. | Aplikasi dapat menampilkan CV applicant secara keseluruhan.                                      | v  |       |
| 8. | Membuat laporan sesuai dengan spesifikasi.   | v  |       |
| 9. | Membuat bonus enkripsi data profil applicant.  |    | v     |
| 10 | Membuat bonus algoritma Aho-Corasick.  | v  |       |
| 11 | Membuat bonus video dan diunggah pada Youtube.   |    | v     |

## **Daftar Pustaka**

Munir, Rinaldi. 2025. *Pencocokan String(String/Pattern Matching)*. [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf) (Diakses pada 15 Juni 2025).

Munir, Rinaldi. 2025. *String Matching dengan Regular Expression*. [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-(2025).pdf) (Diakses pada 15 Juni 2025).