

**TUGAS BESAR 3**  
**STRATEGI ALGORITMA IF2211**  
**Pemanfaatan Pattern Matching untuk Membangun Sistem ATS**  
**(Applicant Tracking System) Berbasis CV Digital**



Kelompok 13  
Lo Siento

**Anggota Kelompok:**

Raudhah Yahya Kuddah	13122003
Felix Chandra	13523012
Ahmad Ibrahim	13523089

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

**2025**

## **DAFTAR ISI**

<b>BAB I</b>	
<b>DESKRIPSI TUGAS</b>	<b>3</b>
A. Dasar Teori	4
1. Penjelasan Algoritma Knuth-Morris-Pratt	4
2. Penjelasan Algoritma Boyer-Moore	4
3. Penjelasan Aho-Corasick	4
<b>BAB III</b>	
<b>ANALISIS PEMECAHAN MASALAH</b>	<b>7</b>
A. Langkah-langkah Pemecahan Masalah	7
B. Pemetaan Masalah ke Elemen-elemen Algoritma	7
C. Fitur Fungsional dan Arsitektur Aplikasi Web	8
D. Contoh ilustrasi kasus.	9
A. Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).	11
B. Penjelasan tata cara penggunaan program	13
C. Hasil pengujian	13
D. Analisis hasil pengujian.	18
A. Kesimpulan	20
B. Saran	20
C. Refleksi	21
<b>LAMPIRAN</b>	<b>22</b>
A. Checklist Penggerjaan	22
B. Tautan Repository Github	23
<b>DAFTAR PUSTAKA</b>	<b>24</b>

## BAB I

### DESKRIPSI TUGAS

**Applicant Tracking System (ATS)** merupakan solusi perangkat lunak berbasis pemrograman yang dirancang untuk mengotomatiskan proses penyaringan dan pencocokan informasi kandidat dalam rekrutmen kerja. Dalam sistem ini, setiap berkas lamaran – khususnya dalam bentuk Curriculum Vitae (CV) – akan diproses secara otomatis untuk mengekstrak informasi-informasi penting guna membantu perusahaan menemukan kandidat yang paling relevan secara efisien.

Pada Tugas Besar 3 ini, mahasiswa diminta untuk mengembangkan sistem deteksi informasi pelamar berbasis dokumen CV digital. Sistem yang dikembangkan harus mampu membaca *file CV*, mengenali informasi yang terkandung di dalamnya, dan mengaitkannya dengan identitas pelamar dalam basis data. Dengan demikian, sistem diharapkan dapat mengenali dan membentuk profil kandidat secara lengkap hanya dari satu dokumen CV, tanpa input manual tambahan.

Pendekatan *pattern matching* fokus pada pencocokan pola teks tertentu dalam sebuah dokumen untuk menemukan informasi yang relevan. Dalam konteks ini, mahasiswa akan mengimplementasikan dua algoritma pencarian string klasik, yaitu **Boyer-Moore** dan **Knuth-Morris-Pratt (KMP)**, yang dikenal memiliki performa tinggi dalam pencarian teks pada dokumen berukuran besar. Kedua algoritma ini akan digunakan untuk mendeteksi elemen-elemen penting pada CV, seperti frasa "pengalaman kerja", "pendidikan terakhir", atau "keahlian", dan kemudian mengambil konten yang terkait dengan pola tersebut.

Tantangan utama dalam pengembangan sistem ini terletak pada keragaman format CV yang dikirimkan oleh pelamar, yang sering kali tidak konsisten dan minim struktur yang baku. Selain itu, performa algoritma juga harus diperhatikan agar sistem tetap dapat memproses ribuan dokumen secara cepat dan akurat dalam waktu yang terbatas.

Laporan ini akan membahas bagaimana algoritma *pattern matching* dapat dimanfaatkan untuk mengekstrak informasi dari dokumen semi-terstruktur, mengevaluasi kelebihan dan kekurangan dari masing-masing algoritma, serta mengusulkan strategi terbaik untuk meningkatkan efektivitas sistem ATS secara keseluruhan. Dengan demikian, mahasiswa diharapkan mampu mengembangkan sistem yang tidak hanya efisien secara komputasional, tetapi juga adaptif terhadap variasi dokumen dunia nyata.

## BAB II

# LANDASAN TEORI

### A. Dasar Teori

#### 1. Penjelasan Algoritma Knuth-Morris-Pratt

Algoritma **Knuth-Morris-Pratt (KMP)** merupakan algoritma pencocokan string yang dirancang untuk menemukan kemunculan sebuah pattern di dalam teks dengan efisiensi tinggi. Keunggulan utama dari KMP terletak pada kemampuannya menghindari pencocokan ulang terhadap karakter yang sebelumnya sudah diperiksa. Hal ini dicapai melalui pembuatan tabel *partial match* (juga dikenal sebagai failure function), yang menyimpan informasi tentang panjang prefix yang cocok dalam pattern itu sendiri. Ketika terjadi mismatch, KMP menggunakan informasi dari tabel ini untuk melompat ke posisi berikutnya yang mungkin cocok tanpa harus memulai pencocokan dari awal. Dalam konteks sistem ATS ini, KMP digunakan untuk mencari keyword secara cepat dan efisien di dalam teks CV, terutama jika teks tersebut berukuran besar.

#### 2. Penjelasan Algoritma Boyer-Moore

Algoritma **Boyer-Moore (BM)** adalah salah satu algoritma pencarian string paling efisien dalam praktik nyata, terutama ketika digunakan untuk mencari pola dalam teks panjang. Keunggulan Boyer-Moore terletak pada pendekatannya yang memindai teks dari kanan ke kiri dan menggunakan dua heuristik utama: *bad character rule* dan *good suffix rule*. Kedua heuristik ini memungkinkan algoritma untuk melakukan lompatan lebih jauh saat terjadi mismatch, sehingga mempercepat proses pencarian. Dalam sistem ini, Boyer-Moore menjadi pilihan yang sangat cocok untuk mencari keyword secara exact match dalam teks CV yang panjang, serta memberikan alternatif performa terhadap KMP.

#### 3. Penjelasan Aho-Corasick

Algoritma **Aho-Corasick** dirancang untuk menangani *multi-pattern matching*, yaitu pencarian banyak pattern sekaligus dalam satu kali pemrosesan teks. Algoritma ini membangun struktur pohon tree dari semua pattern yang ingin dicari, kemudian menggabungkannya dengan automata yang memungkinkan pencarian efisien terhadap semua pola tersebut secara bersamaan. Dalam aplikasi ATS ini, Aho-Corasick sangat berguna ketika pengguna ingin mencari lebih dari satu keyword sekaligus, seperti "Java", "Python", dan "Machine Learning".

Dengan pendekatan ini, sistem tidak perlu menjalankan pencarian berulang kali untuk tiap keyword, melainkan cukup satu kali lintasan teks untuk menemukan semua pattern yang relevan.

#### 4. Penjelasan Aplikasi

Aplikasi Applicant Tracking System (ATS) yang dikembangkan dalam proyek ini dirancang sebagai solusi rekrutmen yang cerdas dan otomatis, dengan tujuan utama menyederhanakan dan mempercepat proses pencarian kandidat yang sesuai berdasarkan dokumen CV digital. Sistem ini hadir dalam bentuk antarmuka pengguna grafis (GUI) yang ramah dan mudah dioperasikan, memungkinkan tim rekrutmen untuk memasukkan satu atau lebih kata kunci yang berkaitan dengan kualifikasi atau keterampilan tertentu yang diinginkan, seperti bidang keahlian teknis, jabatan, atau riwayat pendidikan.

Setelah pengguna memasukkan kata kunci, aplikasi akan menjalankan proses pencarian terhadap seluruh CV yang telah dimuat dalam sistem. Pencocokan ini dapat dilakukan menggunakan algoritma yang dipilih pengguna, yakni Knuth-Morris-Pratt (KMP), Boyer-Moore, atau Aho-Corasick. Masing-masing algoritma menawarkan pendekatan efisien dalam menemukan pola teks secara tepat dalam dokumen, baik untuk satu kata kunci maupun banyak sekaligus. Kemampuan pengguna untuk memilih algoritma yang paling sesuai membuat sistem ini fleksibel dalam menyesuaikan kebutuhan pencarian di berbagai kondisi.

Jika tidak ditemukan kecocokan yang persis, sistem akan mengaktifkan mode pencarian fuzzy dengan menggunakan algoritma Levenshtein Distance. Pendekatan ini sangat berguna untuk tetap menemukan hasil yang relevan meskipun terdapat kesalahan penulisan pada kata kunci atau dalam isi CV. Dengan mempertimbangkan tingkat kemiripan antara teks dalam dokumen dan kata kunci, sistem memberikan alternatif hasil yang mendekati, bukan hanya yang eksak.

Setiap hasil pencarian disajikan dalam bentuk ringkasan informasi pelamar yang diperoleh melalui ekstraksi data dari teks CV. Informasi ini mencakup identitas pelamar yang dihubungkan dengan basis data internal, serta data yang diperoleh langsung dari CV seperti keahlian, pengalaman kerja, dan riwayat pendidikan. Proses ekstraksi dilakukan dengan menggunakan teknik Regular Expression (Regex) untuk mengenali pola-pola umum dalam struktur CV dan mengubahnya menjadi informasi yang mudah dibaca dan relevan bagi pengguna. Selain menampilkan ringkasan, sistem juga memberikan akses langsung ke

dokumen PDF CV asli, sehingga pengguna dapat membuka dan membaca dokumen lengkap secara langsung dari dalam aplikasi.

Aplikasi ini dirancang untuk mendukung efisiensi tinggi dalam pengolahan data pelamar secara otomatis dan akurat, menjadikannya sangat cocok digunakan dalam proses seleksi kandidat dalam jumlah besar. Dengan menggabungkan teknik pencocokan pola, pencarian berbasis kemiripan, dan ekstraksi data berbasis struktur teks, sistem ini mampu meningkatkan kecepatan dan ketepatan dalam menemukan kandidat yang paling sesuai dengan kebutuhan posisi yang ditawarkan oleh perusahaan

## BAB III

### ANALISIS PEMECAHAN MASALAH

#### A. Langkah-langkah Pemecahan Masalah

Pengembangan sistem Applicant Tracking System (ATS) ini diawali dengan analisis mendalam terhadap permasalahan dalam proses rekrutmen. Tahap awal melibatkan identifikasi tantangan yang dihadapi perusahaan, yaitu volume lamaran kerja yang tinggi serta format CV yang bervariasi, khususnya PDF yang tidak selalu terstruktur. Hal ini menyulitkan peninjauan manual oleh tim HR dan mengakibatkan proses yang tidak efisien.

Langkah selanjutnya adalah perancangan arsitektur sistem. Kami merancang alur kerja yang dimulai dengan konversi CV dari format PDF menjadi satu string panjang yang memuat seluruh teksnya. Representasi ini mempermudah pencocokan pola menggunakan algoritma string matching. Kemudian, algoritma seperti Knuth-Morris-Pratt (KMP), Boyer-Moore, dan Aho-Corasick diimplementasikan untuk melakukan pencarian keyword secara exact match. Apabila tidak ditemukan kecocokan, sistem secara konseptual akan menerapkan algoritma Levenshtein Distance untuk fuzzy matching, mempertimbangkan kemungkinan typo pada keyword. Selain itu, Regular Expression (Regex) digunakan untuk mengekstrak informasi penting dari teks CV. Seluruh data CV dan hasil pencarian juga terintegrasi dengan sistem database untuk penyimpanan dan pengelolaan.

#### B. Pemetaan Masalah ke Elemen-elemen Algoritma

Masalah	Algoritma	Deskripsi
Pencarian <i>Keyword</i> (Exact Match)	Knuth-Morris-Pratt (KMP), Boyer-Moore, Aho-Corasick	Mencocokkan <i>keyword</i> yang dimasukkan pengguna (pola) dengan <i>string</i> teks CV secara efisien. Pengguna dapat memilih algoritmanya.
Ekstraksi Informasi Spesifik dari CV	Regular Expression (Regex)	Mengidentifikasi dan mengekstrak informasi penting secara otomatis (misal: alamat <i>email</i> ,

		nomor telepon, riwayat pendidikan, keahlian) untuk ringkasan profil.
Pencarian <i>Fuzzy Match</i> (Konseptual)	Levenshtein Distance	Mencari CV yang paling mirip berdasarkan tingkat kemiripan kata, terutama untuk mengatasi kemungkinan kesalahan pengetikan ( <i>typo</i> ) pada <i>keyword</i> jika <i>exact match</i> gagal.

### C. Fitur Fungsional dan Arsitektur Aplikasi Web

Fitur Fungsional :

- Pencarian Jalur CV Berdasarkan Keyword: Pengguna dapat memasukkan satu atau lebih keyword untuk mencari CV yang relevan.
- Pemilihan Metode Pencarian Exact Match: Pengguna dapat memilih algoritma string matching yang akan digunakan (Knuth-Morris-Pratt, Boyer-Moore, atau Aho-Corasick).
- Pencarian Fuzzy Match (Konseptual): Fitur ini diaktifkan apabila tidak ditemukan exact match untuk suatu keyword, mencari kemiripan menggunakan algoritma Levenshtein Distance.
- Perhitungan Waktu Pencarian: Sistem menampilkan waktu eksekusi untuk proses pencarian, memberikan gambaran kinerja.
- Ringkasan Informasi CV: Menampilkan detail penting yang diekstrak dari CV (seperti kontak, keahlian, riwayat pendidikan).
- Visualisasi/Tampilan CV Asli: Pengguna dapat melihat file PDF CV secara keseluruhan langsung dari aplikasi.
- Manajemen File CV: Fitur untuk mengunggah dan mengelola daftar file PDF CV yang akan diproses.
- Integrasi Database: Menyimpan data pelamar, CV, dan hasil pencarian untuk pengelolaan yang efisien.

Arsitektur Aplikasi Web :

- Antarmuka Pengguna (UI): Dibangun menggunakan pustaka PyQt6, yang menyediakan kerangka kerja untuk aplikasi desktop berbasis Graphical User Interface (GUI).
- Logika Backend/Algoritma: Diimplementasikan dalam Python, mencakup algoritma string matching, pemrosesan PDF, ekstraksi informasi menggunakan Regular Expression, dan interaksi dengan database MySQL.

**D. Contoh ilustrasi kasus.**

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

Misalkan seorang *recruiter* sedang mencari kandidat untuk posisi "Software Engineer" yang memiliki keahlian "Python", "React", dan "SQL". Rekruter akan memasukkan keyword tersebut ke dalam kolom input pada antarmuka aplikasi, dipisahkan dengan koma. Selanjutnya, rekruter akan memilih algoritma pencarian yang diinginkan, misalnya "Aho-Corasick", melalui opsi yang tersedia. Rekruter juga dapat menentukan jumlah CV teratas yang ingin ditampilkan, misalnya "5". Setelah semua kriteria dimasukkan, rekruter akan menekan tombol "Search" untuk memulai proses pencarian.

Sistem akan memindai teks dari semua CV yang ada (baik dari *database* maupun yang baru diunggah). Setiap file CV telah dikonversi menjadi string panjang yang memuat seluruh teksnya. Setelah proses pencarian exact match selesai, sistem akan menampilkan waktu eksekusi pencarian tersebut. Jika ada keyword yang tidak ditemukan secara exact match, sistem akan melakukan fuzzy matching secara konseptual dan menampilkan waktu pencarian fuzzy match juga.

Hasil pencarian akan ditampilkan dalam bentuk tampilan beberapa kartu CV. Setiap kartu memuat informasi seperti nama kandidat, jumlah kecocokan keyword yang ditemukan, serta daftar keyword yang cocok beserta frekuensi kemunculannya. Contohnya, kartu untuk kandidat "Farhan" mungkin menampilkan "3 keyword cocok" dan di bawahnya daftar keyword yang cocok beserta frekuensinya (misalnya: "- Python: 2 kemunculan"). Dari hasil ini, rekruter dapat mengklik tombol "Summary" untuk melihat ekstraksi informasi dari CV seperti identitas, keahlian, pengalaman kerja, dan riwayat pendidikan yang didapatkan dari Regex. Atau, mereka dapat menekan tombol "View CV" untuk melihat langsung file PDF CV asli kandidat.

### E. Penjelasan dynamic threshold untuk fuzzy matching

Dalam implementasi bagian fuzzy matching untuk menghindari typo pada penulisan CV, kami tidak menggunakan threshold levenshtein distance yang konstan melainkan menggunakan threshold dinamis. Hal ini dikarenakan setiap keyword memiliki panjang yang berbeda-beda sehingga thresholdnya juga seharusnya berbeda. Untuk keyword yang sangat pendek (3 karakter atau kurang), tidak ada kesalahan yang diizinkan sama sekali (threshold=0). Seiring bertambahnya panjang kata, toleransi kesalahan juga meningkat; kata dengan 4-5 karakter boleh memiliki satu kesalahan (threshold=1), kata dengan 6-9 karakter diizinkan dua kesalahan (threshold=2), dan kata yang lebih panjang dari 9 karakter dapat memiliki hingga tiga kesalahan (threshold=3). Pendekatan ini secara efektif menyeimbangkan akurasi, dengan menjadi lebih ketat pada kata-kata pendek di mana satu kesalahan kecil dapat mengubah makna secara signifikan, dan lebih longgar pada kata-kata yang lebih panjang di mana beberapa kesalahan ketik lebih mungkin terjadi dan tidak terlalu ambigu. Sebagai contoh apabila kita menggunakan threshold yang konstan (misal 2), saat kita mencari keyword C++, CV dengan keyword C# juga ikut masuk kedalam hasil pencarian karena levenshtein distance nya masih termasuk dalam threshold. Sedangkan dengan kata kata yang sangat panjang seperti heksakosioiheksekontaheksafobia, maka jika terdapat typo sebanyak 3 karakter, CV tidak termasuk kedalam hasil pencarian dikarenakan threshold yang terlalu rendah. Jadi sebelum melakukan fuzzy matching, program akan memanggil fungsi calculate\_dynamic\_threshold(keyword: str) -> int untuk menentukan threshold masing-masing keyword.

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **A. Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).**

Struktur data:

Struktur data	Kegunaan
PDFProcessor	Kelas PDFProcessor menyediakan fungsionalitas untuk mengekstraksi teks dari dokumen PDF menggunakan pustaka PyMuPDF, dengan kapabilitas untuk menghasilkan teks dalam format standar dan format yang diproses (huruf kecil, tanpa jeda baris) untuk analisis lebih lanjut.
MainWindow	Kelas MainWindow adalah fondasi utama dari antarmuka pengguna grafis (GUI) aplikasi. Kelas ini menjadi elemen tingkat paling bawah yang menjadi tempat elemen-elemen lain diletakkan.
FlowLayout	Kelas ini mengatur widget secara berurutan dalam baris, secara otomatis memindahkan widget ke baris berikutnya jika tidak ada cukup ruang pada baris saat ini, mirip dengan bagaimana teks mengalir dalam paragraf.
CVCard	Kelas CVCard adalah widget tampilan yang berfungsi sebagai "kartu" individual untuk setiap pelamar. Setiap kartu menampilkan nama pelamar, jumlah kata kunci yang cocok dari CV-nya, dan daftar detail kata kunci yang cocok tersebut. Kartu ini juga menyediakan tombol untuk melihat ringkasan CV dalam dialog terpisah dan tombol untuk membuka file CV asli.
SummaryDialog	Kelas SummaryDialog adalah dialog pop-up yang menampilkan ringkasan informasi seorang pelamar. Dialog ini menggabungkan data pribadi pelamar yang mungkin berasal dari database dengan informasi tambahan yang diekstraksi langsung dari teks CV, seperti alamat email, nomor telepon, keahlian, dan riwayat pendidikan, menyajikannya dalam format yang mudah dibaca.
KeywordTag	Kelas KeywordTag adalah widget kustom yang merepresentasikan sebuah "tag" kata kunci. Setiap tag menampilkan teks kata kunci dan sebuah tombol kecil untuk menghapusnya. Ketika tombolhapus diklik, tag ini akan memancarkan sinyal, memberitahukan bagian lain dari aplikasi bahwa kata kunci tersebut perlu dihapus.

Fungsi / Prosedur:

Fungsi / Prosedur	Kegunaan
kmp_search	Fungsi kmp_search mengimplementasikan algoritma pencarian string Knuth-Morris-Pratt (KMP) untuk menemukan semua kemunculan suatu pola dalam sebuah teks.
boyer_moore_search	Fungsi boyer_moore_search mengimplementasikan algoritma pencarian string Boyer-Moore untuk menemukan semua kemunculan suatu pola dalam teks.
aho_corasick_search	Fungsi aho_corasick_search menerapkan algoritma pencarian string Aho-Corasick, yang dirancang untuk secara efisien menemukan multiple patterns dalam satu teks. Algoritma ini bekerja dengan membangun struktur trie dari pola-pola yang diberikan dan menambahkan "failure links" untuk navigasi yang cepat.
levenstein_distance	Fungsi levenstein_distance menghitung jarak Levenshtein antara dua string. Metrik ini mengukur jumlah minimum operasi (penyisipan, penghapusan, atau substitusi satu karakter) yang diperlukan untuk mengubah satu string menjadi string lainnya, berguna untuk menentukan kesamaan antar dua urutan.
find_most_similar	Fungsi find_most_similar memanfaatkan levenstein_distance untuk menemukan kata-kata dalam sebuah teks yang paling mirip dengan suatu kata kunci yang diberikan. Fungsi ini mengidentifikasi kata-kata yang memiliki jarak Levenshtein di bawah ambang batas (threshold) tertentu, cocok untuk fitur pencarian fuzzy atau koreksi ejaan.
regex_search	Fungsi regex_search melakukan pencarian ekspresi reguler (regex) pada teks yang diberikan, mengembalikan daftar semua kecocokan. Untuk setiap kecocokan yang ditemukan, fungsi ini menyediakan detail seperti posisi awal dan akhir, teks yang cocok, serta konteks di sekitar kecocokan tersebut.
calculate_dynamic_threshold(keyword)	Menentukan threshold Levenshtein secara dinamis berdasarkan panjang kata kunci. Kata yang lebih panjang diizinkan memiliki lebih banyak toleransi kesalahan ketik.
extract_email_addresses(text)	Mengekstrak semua alamat email yang ditemukan dalam

t)	teks CV menggunakan pola Regex.
extract_phone_numbers(text)	Mengekstrak semua nomor telepon dari teks CV menggunakan beberapa pola Regex yang umum.
extract_education_info(text)	Mencari dan mengekstrak informasi yang berkaitan dengan riwayat pendidikan, seperti nama universitas, gelar, atau IPK.
extract_skills_keywords(text, skills_list)	Mendeteksi keahlian (skills) dari teks CV dengan mencocokkannya dengan daftar keahlian yang telah ditentukan.

## B. Penjelasan tata cara penggunaan program

Untuk menjalankan aplikasi ini, pengguna terlebih dahulu perlu menyiapkan lingkungan basis data dengan menjalankan perintah docker-compose up -d pada terminal. Langkah ini akan menginisialisasi layanan basis data MySQL dan phpMyAdmin sesuai konfigurasi yang tersedia dalam file docker-compose.yml. Setelah layanan database aktif, aplikasi utama dapat dijalankan melalui perintah uv run src/main.py, yang akan membuka antarmuka pengguna PyQt6 untuk memulai proses analisis CV.

Sebelum menjalankan aplikasi, pastikan semua prasyarat telah terpenuhi, yaitu Python versi 3.12 atau lebih baru, Docker beserta Docker Compose, serta manajer paket uv yang dapat diinstal melalui skrip resmi dari situs uv. Setelah repositori dikloning, pengguna dapat masuk ke direktori proyek dan melakukan sinkronisasi dependensi dengan perintah uv sync. Selanjutnya, pengguna perlu menyalin file .env.example menjadi .env dan menyesuaikan konfigurasi database yang dibutuhkan.

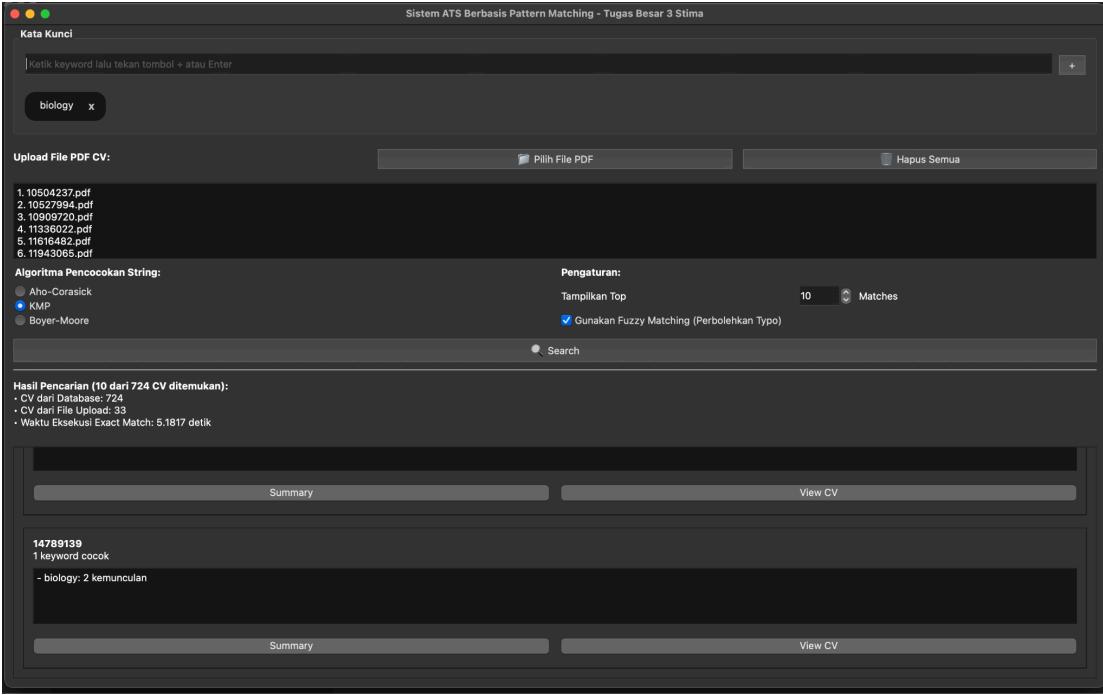
Setelah aplikasi berjalan, pengguna dapat mengunggah file CV dalam format PDF yang kemudian akan diproses menggunakan pustaka PyMuPDF untuk mengekstraksi teks. Aplikasi ini memungkinkan pencarian kata kunci secara exact match menggunakan algoritma KMP dan Boyer-Moore, serta pencarian fuzzy menggunakan Levenshtein Distance. Informasi seperti alamat email, nomor telepon, riwayat pendidikan, dan keterampilan dapat diambil secara otomatis melalui Regex. Semua data hasil parsing dan pencarian akan disimpan di basis data MySQL, yang dapat diakses dan dikelola melalui phpMyAdmin di alamat <http://localhost:8081>.

## C. Hasil pengujian

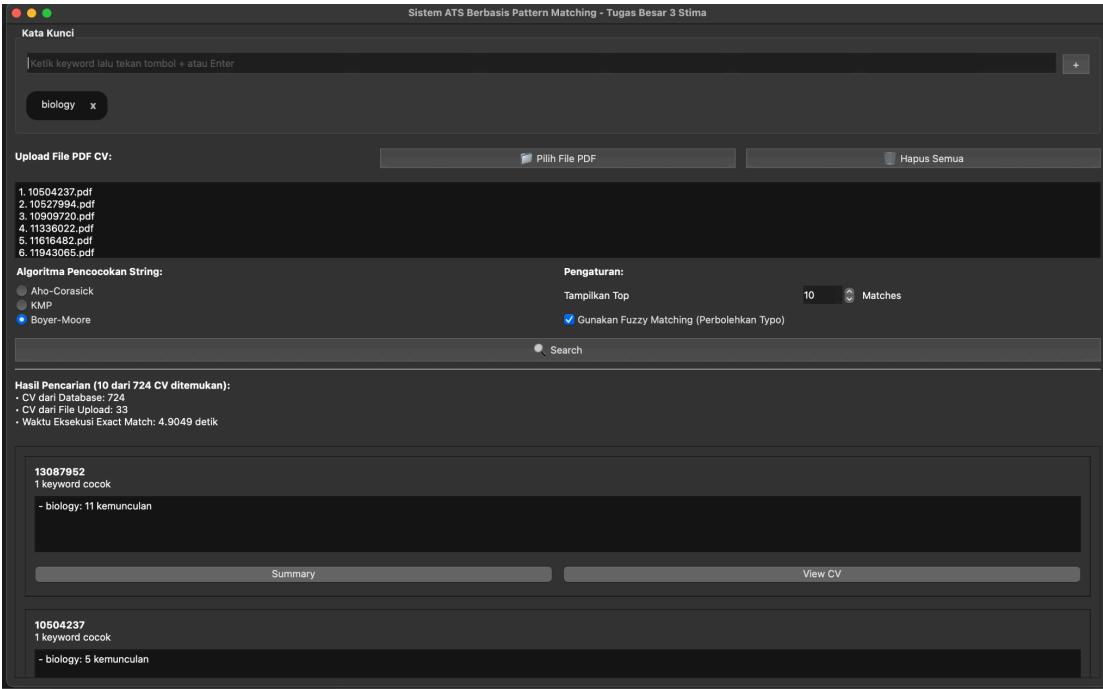
**Seluruh pengujian dilakukan dengan menggunakan 724 data CV**

### 1. Pengujian 1 : keyword “biology” dengan KMP

# IF2211 Strategi Algoritma

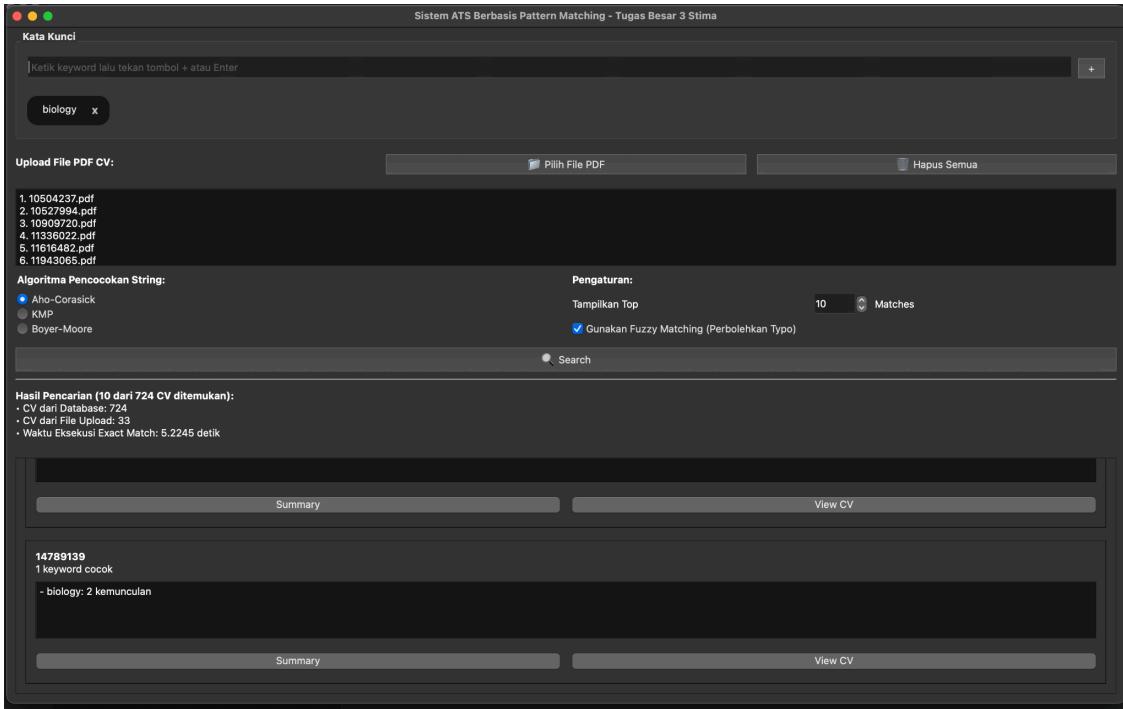


## 2. Pengujian 2 : keyword “biology” dengan Boyer - Moore

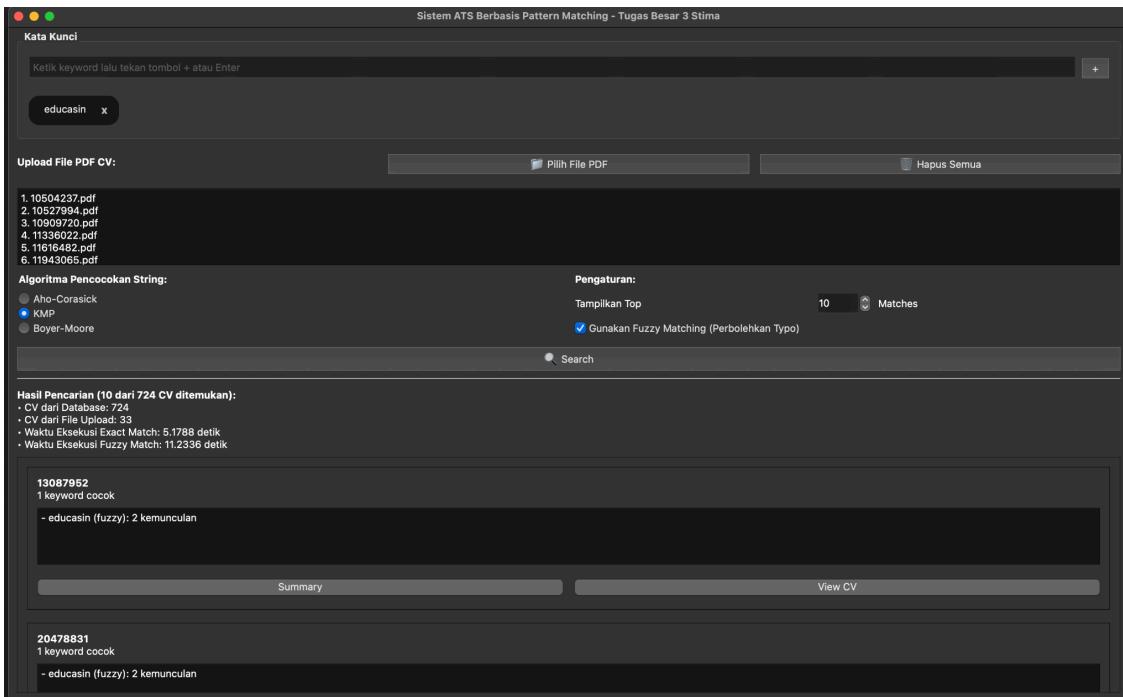


## 3. Pengujian 3 : keyword “biology” dengan Aho-Corasik

## IF2211 Strategi Algoritma

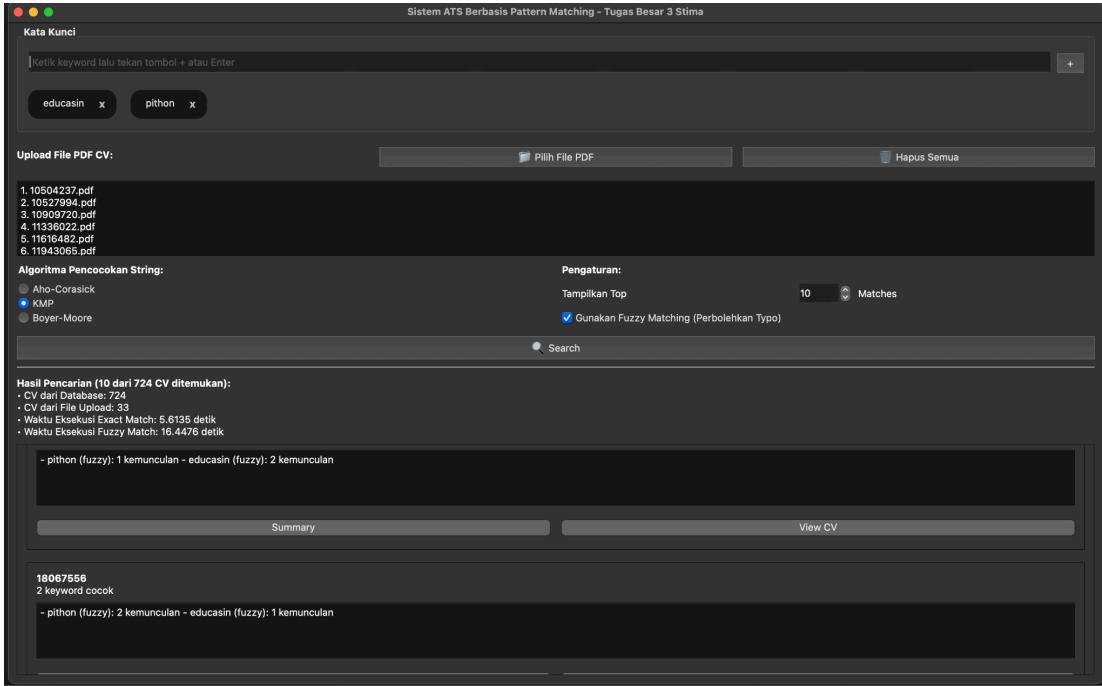


### 4. Pengujian 4 : keyword typo “educasin” dengan KMP

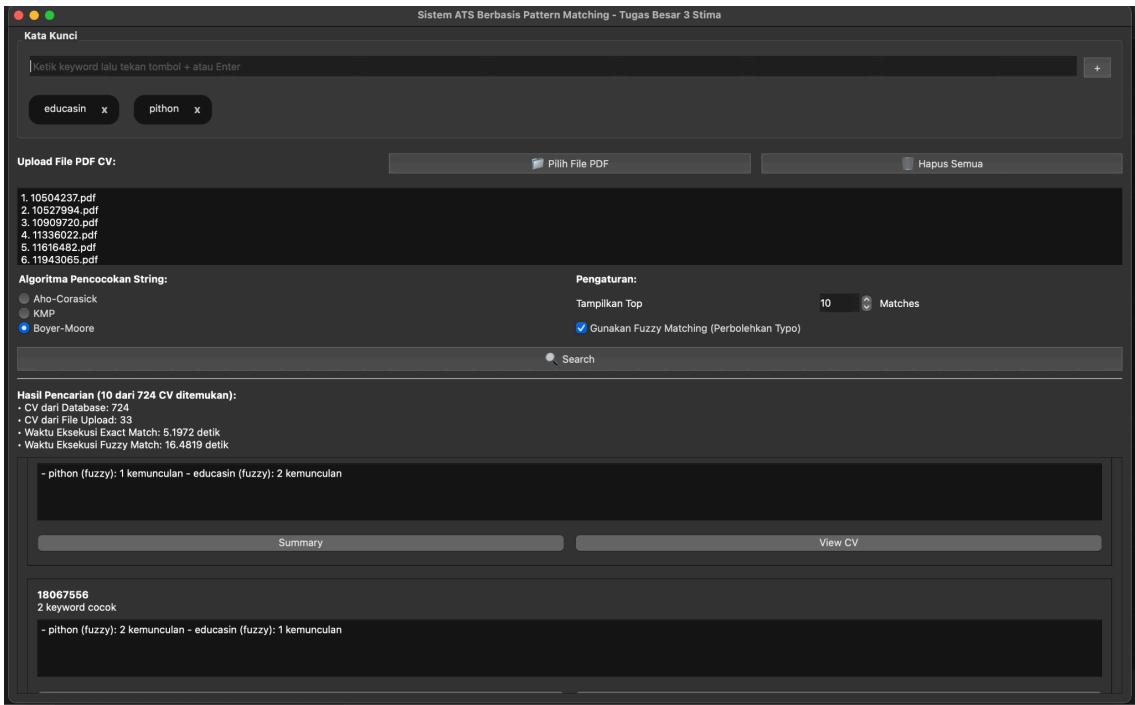


### 5. Pengujian 5 : keyword typo “educasin” dan “python” dengan KMP

## IF2211 Strategi Algoritma

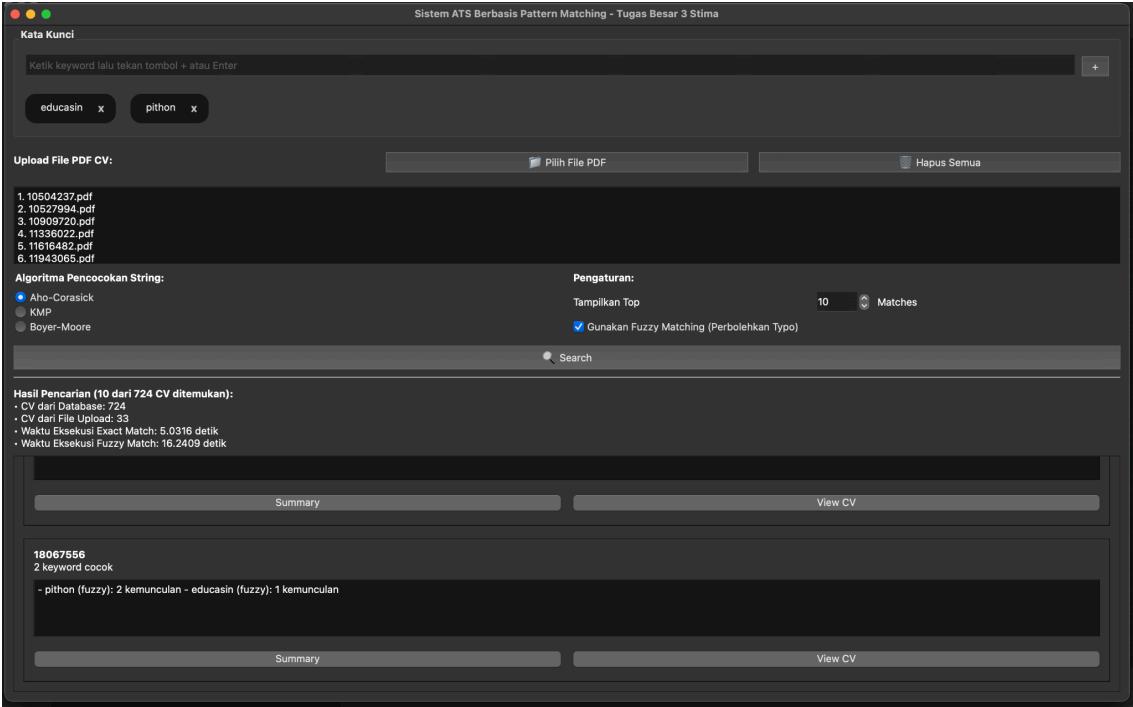


### 6. Pengujian 6 : keyword typo “educasin” dan “pithon” dengan Boyer-Moore

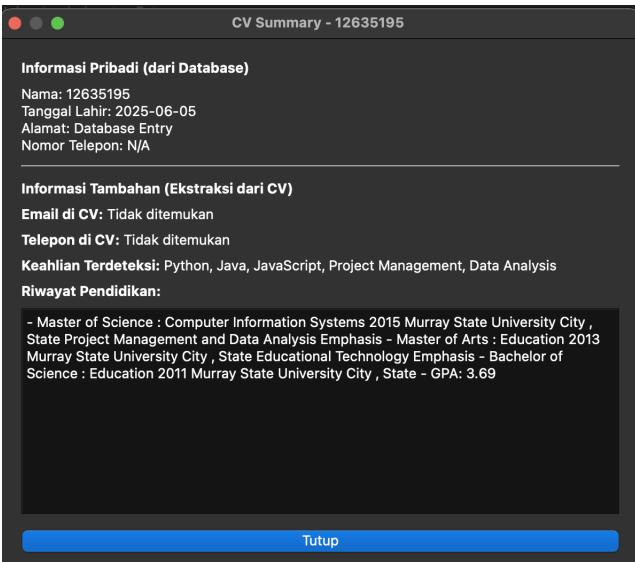


### 7. Pengujian 7 : keyword typo “educasin” dan “pithon” dengan Aho-Corasick

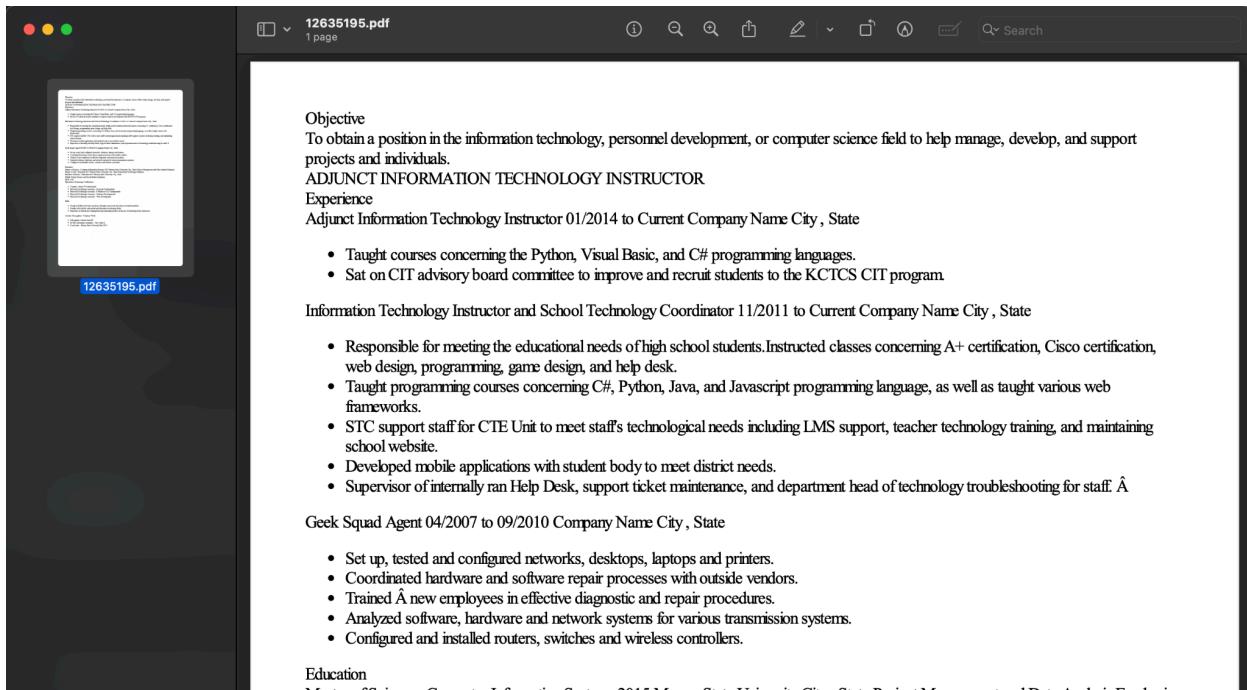
## IF2211 Strategi Algoritma



### 8. Tampilan Summary CV



### 9. Tampilan View CV



#### D. Analisis hasil pengujian.

Berdasarkan analisa hasil pengujian diatas dapat dilihat bahwa, untuk pencarian beberapa pola secara bersamaan, algoritma Aho-Corasick melakukan pencarian tercepat (Pengujian 7) dibandingkan dengan algoritma KMP (Pengujian 5) dan juga Boyer-Moore (Pengujian 6) dengan keyword yang sama. Hal ini karena Aho-Corasick dirancang untuk memproses seluruh daftar kata kunci hanya dalam satu kali penelusuran teks, membuatnya jauh lebih efisien dibandingkan harus menjalankan algoritma lain secara berulang-ulang.

Sebaliknya, untuk pencarian satu keyword saja, Tampak bahwa algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) adalah pilihan yang lebih cepat. Meskipun keduanya dianggap memiliki efisiensi tinggi, Boyer-Moore seringkali lebih cepat dalam praktiknya karena kemampuannya untuk melompati sebagian besar teks (Jika keyword merupakan teks panjang) melainkan KMP lebih cepat saat keyword memiliki banyak perulangan seperti kode biner (01010100). Menggunakan Aho-Corasick untuk satu pola akan menjadi tidak efisien karena kompleksitasnya dalam membangun struktur data awal tidak sebanding dengan tugas yang sederhana.

## **BAB V**

# **KESIMPULAN, SARAN, DAN REFLEKSI**

### **A. Kesimpulan**

Proyek Tugas Besar 3 ini berhasil mengimplementasikan sebuah sistem *Applicant Tracking System* (ATS) yang fungsional dengan memanfaatkan algoritma *pattern matching* untuk otomatisasi proses penyaringan *Curriculum Vitae* (CV) digital. Sistem yang dibangun mampu melakukan ekstraksi teks dari dokumen PDF, kemudian melakukan pencarian kata kunci secara efisien menggunakan tiga algoritma berbeda: Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Aho-Corasick.

Berdasarkan hasil pengujian terhadap 724 data CV, dapat disimpulkan bahwa:

1. Algoritma Aho-Corasick menunjukkan performa paling unggul untuk pencarian *multi-pattern* (banyak kata kunci sekaligus), dengan waktu eksekusi yang jauh lebih cepat dibandingkan KMP dan Boyer-Moore yang dijalankan secara iteratif.
2. Untuk pencarian *single-pattern* (satu kata kunci), algoritma Boyer-Moore dan KMP terbukti lebih efisien. Boyer-Moore cenderung lebih cepat dalam praktiknya karena mekanisme lompatan karakternya, menjadikannya pilihan ideal untuk pencarian kata kunci tunggal pada teks yang panjang.
3. Sistem berhasil mengekstrak informasi penting dari CV seperti kontak, pengalaman, dan pendidikan menggunakan Regular Expression (Regex), serta menyajikannya dalam format ringkas yang mudah dibaca.
4. Fitur pencarian *fuzzy* menggunakan Levenshtein Distance berhasil diimplementasikan untuk menangani kesalahan pengetikan (*typo*) pada kata kunci, sehingga meningkatkan relevansi hasil pencarian.

Secara keseluruhan, sistem ATS ini membuktikan bahwa penerapan algoritma strategi yang tepat dapat secara signifikan meningkatkan efisiensi dan akurasi dalam proses rekrutmen awal.

### **B. Saran**

Untuk pengembangan lebih lanjut, beberapa hal dapat dipertimbangkan sebagai perbaikan:

Implementasi Penuh Fuzzy Matching pada Konten CV: Saat ini, *fuzzy matching* hanya diterapkan pada kata kunci input. Pengembangan selanjutnya dapat mengaplikasikan Levenshtein Distance atau algoritma serupa untuk menganalisis kemiripan konten di dalam CV, sehingga dapat menemukan kandidat yang relevan meskipun menggunakan terminologi yang sedikit berbeda (misalnya, "manajer" vs. "manager").

Peningkatan Akurasi Regex: Pola Regex dapat diperluas dan divalidasi dengan dataset CV yang lebih beragam untuk meningkatkan akurasi ekstraksi informasi, terutama dalam menangani format CV yang tidak terstruktur atau sangat kreatif.

### C. Refleksi

Pengerjaan Tugas Besar 3 ini memberikan pengalaman yang sangat berharga dalam menerapkan konsep teoritis strategi algoritma ke dalam sebuah aplikasi dunia nyata. Tantangan terbesar yang kami hadapi adalah mengatasi keragaman format CV yang tidak memiliki standar baku.

Melalui proyek ini, kami, Kelompok 13 "Lo Siento", belajar bahwa pemilihan algoritma yang tepat bergantung pada kasus penggunaan spesifik—seperti perbedaan efisiensi antara Aho-Corasick untuk pencarian banyak pola dan Boyer-Moore untuk satu pola. Proses *debugging* dan pengujian juga menyadarkan kami akan pentingnya analisis kompleksitas dan performa dalam pengembangan perangkat lunak. Proyek ini menjadi jembatan antara teori di kelas dengan solusi praktis yang relevan dengan kebutuhan industri, khususnya di bidang rekrutmen dan sumber daya manusia.

## LAMPIRAN

### A. Checklist Penggerjaan

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

11	Membuat bonus video dan diunggah pada Youtube.			✓
----	------------------------------------------------	--	--	---

**B. Tautan Repository Github**

[https://github.com/aibrahim185/Tubes3\\_lo-siento](https://github.com/aibrahim185/Tubes3_lo-siento)

## **DAFTAR PUSTAKA**

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

[https://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)

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Modul-Praktikum-NLP-Regex.pdf>